Prediction and Analysis of Mixed Flow Properties of Turbofan Engines with Different Afterburning Configurations Using Artificial Intelligence Models

A project presented to the
Faculty of the Department of Aerospace Engineering
San José State University

In partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

by

Nicholas S. Riabtchenko

May 2026

approved by

Dr. Yawo Ezunkpe
Faculty Advisor



© 2026
Nicholas S. Riabtchenko
ALL RIGHTS RESERVED

Abstract

This report develops a physics-guided machine learning model for predicting the performance characteristics and internal flow properties of mixed-flow afterburning turbofan engines across a wide operating envelope. A unified parametric cycle formulation is established for both ideal and real component behavior, with explicit station numbering, flow assumptions, and mixer relations defining the forward mapping from design inputs to performance metrics. The core model architecture is a binary-gated mixture of experts forward surrogate network that learns the multi-dimensional input to output mapping while remaining anchored to parametric cycle physics through constrained training and consistent thermodynamic relations.

Acknowledgments

I would like to thank my parents for their support, as well as the faculty of the San Jose State University Aerospace Engineering department for their guidance, especially Dr. Yawo Ezunkpe.

Contents

Abstract i

Acknowledgments ii

Symbols x

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Literature Review . . . 1
  - 1.2.1 Turbofan Augmentation Studies . . . 2
  - 1.2.2 Mixed Flow Turbofan Modeling Studies . . . 3
  - 1.2.3 Adaptive Cycle Engine Studies . . . 4
  - 1.2.4 Machine Learning Prediction Studies . . . 6
  - 1.2.5 Learning-Based Thrust Control Studies . . . 7
- 1.3 Project Proposal . . . 8
- 1.4 Methodology . . . 9

2 Engine Overview 10
- 2.1 Mixed Flow Afterburning Turbofan Engine . . . 10
- 2.2 Fan and Bypass . . . 11
- 2.3 Compressor . . . 12
- 2.4 Combustor . . . 12
- 2.5 Turbine . . . 13
- 2.6 Mixer . . . 14
- 2.7 Afterburner . . . 15
- 2.8 Variable Geometry Nozzle . . . 16

3 Parametric Cycle Analysis 18
- 3.1 Fundamental Theory and Notation . . . 18
  - 3.1.1 Brayton Cycle Overview . . . 18
  - 3.1.2 Station Numbering and Flow Assumptions . . . 20
  - 3.1.3 Dimensionless Quantities and Performance Metrics . . . 21
- 3.2 Ideal Mixed-Flow Afterburning Turbofan Model . . . 24
  - 3.2.1 Inputs and Outputs . . . 24
  - 3.2.2 Summary of Ideal Equations . . . 25
- 3.3 Real Mixed-Flow Afterburning Turbofan Model . . . 28

3.3.1 Inputs and Outputs . . . 28
3.3.2 Summary of Real-Cycle Relations . . . 29

4 Machine Learning Architecture/Integration 37
4.1 Machine Learning Fundamentals . . . 37
4.1.1 Surrogate Modeling Objective . . . 38
4.1.2 Multi-Layer Perceptron Formulation . . . 39
4.1.3 Supervised Regression Formulation . . . 40
4.1.4 Loss Functions and Output Weighting . . . 41
4.1.5 Input and Output Normalization . . . 43
4.2 Physics-Guided Learning Approach . . . 44
4.2.1 Parametric Cycle Analysis as the Reference Model . . . 45
4.2.2 Physics-Guided Versus Physics-Informed Learning . . . 45
4.2.3 Physical Constraints in the Training Workflow . . . 46
4.2.4 Internal State Prediction and Diagnostic Outputs . . . 47
4.3 Sampling and Dataset Construction . . . 48
4.3.1 High-Dimensional Sampling Requirements . . . 49
4.3.2 Latin Hypercube Sampling . . . 49
4.3.3 Sobol Sampling and Dataset Sharding . . . 50
4.3.4 Input Bounds and Operating Envelope Definition . . . 51
4.3.5 Training, Validation, and Test Splitting . . . 53
4.4 Feasibility Filtering . . . 54
4.4.1 Thermodynamic Feasibility Requirements . . . 54
4.4.2 Fuel-Flow and Bypass-Ratio Constraints . . . 56
4.4.3 Mixer and Nozzle Feasibility Requirements . . . 59
4.4.4 Efficiency and Performance Bounds . . . 62
4.5 Wet and Dry Mixture-of-Experts Strategy . . . 65
4.5.1 Afterburning Mode Separation . . . 66
4.5.2 Binary Expert-Gating Formulation . . . 66
4.5.3 Connection to Models A, B, and C . . . 68
4.6 Chapter Summary . . . 69

5 Model Training and Validation 71
5.1 Model Development Overview . . . 71
5.1.1 Staged Development Strategy . . . 71
5.1.2 Relationship Between Models A, B, and C . . . 72
5.2 Common Neural Network Implementation . . . 73
5.2.1 Input and Output Data Preparation . . . 74
5.2.2 Training, Validation, and Test Data Splitting . . . 76
5.2.3 Network Initialization and Forward Propagation . . . 76
5.2.4 Backpropagation and Parameter Updates . . . 78
5.2.5 Training Loss and Validation Loss Monitoring . . . 80
5.2.6 Model Storage and Prediction Workflow . . . 81
5.3 Model-Specific Training and Baseline Validation . . . 83
5.3.1 Model A: Fixed-Property Ideal-Cycle Surrogate . . . 83

5.3.2 Model B: Variable-Thermochemistry Ideal-Cycle Surrogate 89
5.3.3 Model C: Real-Cycle Surrogate 94
5.4 Chapter Summary 97

6 Results and Analysis **98**
6.1 Results and Validation Overview 98
6.1.1 Purpose of Final Validation Studies 99
6.1.2 Evaluation Metrics 99
6.1.3 Distinction Between Statistical and Physical Validation 100
6.2 Parametric Sweep Validation Method 101
6.2.1 Anchor-Point Sweep Procedure 103
6.2.2 Single-Variable Perturbation Method 104
6.2.3 Reference PCA and Neural Network Comparison 105
6.3 Dry-Mode Parametric Sweep Results 105
6.3.1 Mach Number Sweep 106
6.3.2 Compressor Pressure Ratio Sweep 109
6.3.3 Fan Pressure Ratio Sweep 112
6.3.4 Burner Temperature Ratio Sweep 115
6.3.5 Mixer Pressure Recovery Sweep 118
6.3.6 Dry-Mode Sweep Summary 119
6.4 Afterburning-Mode Parametric Sweep Results 120
6.4.1 Mach Number Sweep 121
6.4.2 Compressor Pressure Ratio Sweep 123
6.4.3 Fan Pressure Ratio Sweep 126
6.4.4 Burner Temperature Ratio Sweep 129
6.4.5 Afterburner Temperature Ratio Sweep 132
6.4.6 Afterburning-Mode Sweep Summary 135
6.5 Extrapolation and Edge-Envelope Behavior 136
6.5.1 Model A Extrapolation Behavior 136
6.5.2 Model B Extrapolation Behavior 145
6.5.3 Model C Extrapolation Behavior 154
6.5.4 Extrapolation Synthesis and Observed Failure Modes 163
6.6 Runtime Benchmarking 165
6.6.1 Model A Runtime Benchmark 165
6.6.2 Model B Runtime Benchmark 166
6.6.3 Model C Runtime Benchmark 167
6.6.4 Implications for Large-Scale Design Studies 169
6.7 Model Limitations 169
6.7.1 Limitations of the Parametric Cycle Reference Model 170
6.7.2 Limitations of the Neural-Network Surrogate 171
6.7.3 Limitations of Binary Wet-Dry Gating 173
6.7.4 Limitations in Extrapolation 173
6.7.5 Unmodeled Combustion and Component Effects 174
6.8 Practical and Economic Implications 175
6.8.1 Reduced Cost of Repeated Cycle Evaluation 176

6.8.2 Use in Preliminary Engine Design . . . 176
6.8.3 Use in Optimization and Control Studies . . . 177
6.8.4 Scaling to Larger Surrogate Models . . . 178
6.9 Recommended Model Improvements . . . 179
6.9.1 Neural Gating Function . . . 179
6.9.2 Hyperparameter Optimization . . . 180
6.9.3 Expanded Dataset Generation . . . 181
6.9.4 Boundary-Refined Sampling . . . 182
6.9.5 Feasibility Classification . . . 182
6.9.6 Uncertainty Quantification . . . 183
6.10 Chapter Summary . . . 184

7 Conclusions and Future Work **186**
7.1 Conclusion . . . 186
7.1.1 Project Objective Revisited . . . 187
7.1.2 Summary of Technical Approach . . . 187
7.1.3 Summary of Machine Learning Framework . . . 188
7.2 Major Accomplishments . . . 189
7.2.1 Parametric Cycle Analysis Implementation . . . 189
7.2.2 Dataset Generation and Feasibility Filtering . . . 190
7.2.3 Development of Models A, B, and C . . . 191
7.2.4 Mixture-of-Experts Surrogate Architecture . . . 192
7.2.5 Validation Through Parametric Sweep Studies . . . 192
7.2.6 Runtime Acceleration Demonstration . . . 193
7.3 Principal Findings . . . 194
7.3.1 Effectiveness of Neural Surrogates for Cycle Prediction . . . 194
7.3.2 Importance of Wet and Dry Expert Separation . . . 195
7.3.3 Importance of Physics-Guided Dataset Construction . . . 196
7.3.4 Interpolation Capability and Extrapolation Limitations . . . 197
7.3.5 Computational Value of the Surrogate Model . . . 198
7.4 Recommended Future Work . . . 199
7.4.1 Improved Neural Gating Architecture . . . 200
7.4.2 Physics-Guided Loss Function Refinement . . . 201
7.4.3 Hyperparameter and Architecture Optimization . . . 202
7.4.4 Expanded and Boundary-Refined Datasets . . . 202
7.4.5 Component Map Integration . . . 203
7.4.6 Uncertainty Quantification and Feasibility Classification . . . 204
7.4.7 Inverse Model and Closed-Loop Control . . . 205
7.5 Final Remarks . . . 206

References **208**

Appendices **211**
Appendix A: Code Availability . . . 211

List of Figures

2.1 Pratt & Whitney F100 engine [14] 10
2.2 Schematic of mixed flow afterburning turbofan engine [15] 11
2.3 Schematic of axial compressor [15] 12
2.4 Schematic of combustion chamber cross section [15] 13
2.5 Schematic of axial turbine [15] 14
2.6 Schematic of ideal flow mixer [15] 15
2.7 Typical afterburner schematic [16] 16
2.8 Schematic of variable geometry nozzle [15] 17

3.1 T-s diagram of Brayton cycle [15] 19

5.1 Model A dry expert training and validation loss history 84
5.2 Model A wet expert training and validation loss history 84
5.3 Model A dry expert test-set parity plots 85
5.4 Model A wet expert test-set parity plots 86
5.5 Model A dry expert test-set residual plots 86
5.6 Model A wet expert test-set residual plots 87
5.7 Model A dry expert test-set consistency checks 88
5.8 Model A wet expert test-set consistency checks 88
5.9 Model B dry expert training and validation loss history 89
5.10 Model B wet expert training and validation loss history 90
5.11 Model B dry expert test-set parity plots 91
5.12 Model B wet expert test-set parity plots 91
5.13 Model B dry expert test-set residual plots 92
5.14 Model B wet expert test-set residual plots 93
5.15 Model B dry expert test-set consistency checks 93
5.16 Model B wet expert test-set consistency checks 94
5.17 Model C dry expert training and validation loss history 95
5.18 Model C wet expert training and validation loss history 95
5.19 Model C combined dry and wet test-set parity and error-distribution plots . 96

6.1 Model A dry-mode parametric sweep over freestream Mach number 107
6.2 Model B dry-mode parametric sweep over freestream Mach number 108
6.3 Model C dry-mode parametric sweep over freestream Mach number 109
6.4 Model A dry-mode parametric sweep over compressor pressure ratio 110
6.5 Model B dry-mode parametric sweep over compressor pressure ratio 111
6.6 Model C dry-mode parametric sweep over compressor pressure ratio 112

6.7 Model A dry-mode parametric sweep over fan pressure ratio 113
6.8 Model B dry-mode parametric sweep over fan pressure ratio 114
6.9 Model C dry-mode parametric sweep over fan pressure ratio 115
6.10 Model A dry-mode parametric sweep over burner temperature ratio 116
6.11 Model B dry-mode parametric sweep over burner temperature ratio 117
6.12 Model C dry-mode parametric sweep over burner exit total temperature . . . 118
6.13 Model C dry-mode parametric sweep over maximum mixer pressure recovery 119
6.14 Model A afterburning-mode parametric sweep over freestream Mach number 121
6.15 Model B afterburning-mode parametric sweep over freestream Mach number 122
6.16 Model C afterburning-mode parametric sweep over freestream Mach number 123
6.17 Model A afterburning-mode parametric sweep over compressor pressure ratio 124
6.18 Model B afterburning-mode parametric sweep over compressor pressure ratio 125
6.19 Model C afterburning-mode parametric sweep over compressor pressure ratio 126
6.20 Model A afterburning-mode parametric sweep over fan pressure ratio 127
6.21 Model B afterburning-mode parametric sweep over fan pressure ratio 128
6.22 Model C afterburning-mode parametric sweep over fan pressure ratio 129
6.23 Model A afterburning-mode parametric sweep over burner temperature ratio 130
6.24 Model B afterburning-mode parametric sweep over burner temperature ratio 131
6.25 Model C afterburning-mode parametric sweep over burner exit total temperature 132
6.26 Model A afterburning-mode parametric sweep over afterburner temperature ratio 133
6.27 Model B afterburning-mode parametric sweep over afterburner temperature ratio 134
6.28 Model C afterburning-mode parametric sweep over afterburner exit total temperature 135
6.29 Model A dry-mode extrapolation sweep over freestream Mach number 137
6.30 Model A dry-mode extrapolation sweep over compressor pressure ratio . . . 138
6.31 Model A dry-mode extrapolation sweep over fan pressure ratio 139
6.32 Model A dry-mode extrapolation sweep over burner temperature ratio 140
6.33 Model A afterburning-mode extrapolation sweep over freestream Mach number 141
6.34 Model A afterburning-mode extrapolation sweep over compressor pressure ratio 142
6.35 Model A afterburning-mode extrapolation sweep over fan pressure ratio . . . 143
6.36 Model A afterburning-mode extrapolation sweep over burner temperature ratio 144
6.37 Model A afterburning-mode extrapolation sweep over afterburner temperature ratio 145
6.38 Model B dry-mode extrapolation sweep over freestream Mach number 146
6.39 Model B dry-mode extrapolation sweep over compressor pressure ratio . . . 147
6.40 Model B dry-mode extrapolation sweep over fan pressure ratio 148
6.41 Model B dry-mode extrapolation sweep over burner temperature ratio 149
6.42 Model B afterburning-mode extrapolation sweep over freestream Mach number 150
6.43 Model B afterburning-mode extrapolation sweep over compressor pressure ratio 151
6.44 Model B afterburning-mode extrapolation sweep over fan pressure ratio . . . 152
6.45 Model B afterburning-mode extrapolation sweep over burner temperature ratio 153
6.46 Model B afterburning-mode extrapolation sweep over afterburner temperature ratio 154

6.47 Model C dry-mode extrapolation sweep over freestream Mach number 155
6.48 Model C dry-mode extrapolation sweep over compressor pressure ratio . . . 156
6.49 Model C dry-mode extrapolation sweep over fan pressure ratio 157
6.50 Model C dry-mode extrapolation sweep over burner exit total temperature . 158
6.51 Model C afterburning-mode extrapolation sweep over freestream Mach number 159
6.52 Model C afterburning-mode extrapolation sweep over compressor pressure ratio 160
6.53 Model C afterburning-mode extrapolation sweep over fan pressure ratio . . . 161
6.54 Model C afterburning-mode extrapolation sweep over burner exit total temperature 162
6.55 Model C afterburning-mode extrapolation sweep over afterburner exit total temperature 163
6.56 Model A dry and afterburning runtime scaling 166
6.57 Model B dry and afterburning runtime scaling 167
6.58 Model C dry and afterburning runtime scaling 168

Symbols

Symbol	Description	Units
T_i	Static temperature at station i	K
$T_{t,i}$	Total temperature at station i	K
P_i	Static pressure at station i	Pa
p_i	Static pressure at station i	Pa
$P_{t,i}$	Total pressure at station i	Pa
$p_{t,i}$	Total pressure at station i	Pa
h_i	Static specific enthalpy at station i	J/kg
$h_{t,i}$	Total specific enthalpy at station i	J/kg
M_i	Mach number at station i	–
V_i	Flow velocity at station i	m/s
ρ_i	Density at station i	$\mathrm{kg/m^3}$
A_i	Cross-sectional flow area at station i	$\mathrm{m^2}$
M_0	Freestream Mach number	–
T_0	Freestream static temperature	K
P_0	Freestream static pressure	Pa
p_0	Freestream static pressure	Pa
T_{t0}	Freestream total temperature	K
P_{t0}	Freestream total pressure	Pa
p_{t0}	Freestream total pressure	Pa
a_0	Freestream speed of sound	m/s
V_0	Freestream flight speed	m/s
h_0	Freestream static specific enthalpy	J/kg
γ	Ratio of specific heats	–
γ_c	Compressor-side ratio of specific heats	–
γ_t	Turbine-side ratio of specific heats	–
γ_{AB}	Afterburner-gas ratio of specific heats	–
γ_{6A}	Mixed-stream ratio of specific heats	–
γ_9	Nozzle-exit ratio of specific heats	–
c_p	Specific heat at constant pressure	$\mathrm{J/(kg\,K)}$
$c_{p,c}$	Compressor-side specific heat at constant pressure	$\mathrm{J/(kg\,K)}$
$c_{p,t}$	Turbine-side specific heat at constant pressure	$\mathrm{J/(kg\,K)}$
$c_{p,\mathrm{AB}}$	Afterburner-gas specific heat at constant pressure	$\mathrm{J/(kg\,K)}$

Symbol	Description	Units
$c_{p,6A}$	Mixed-stream specific heat at constant pressure	J/(kg K)
$c_{p,9}$	Nozzle-exit specific heat at constant pressure	J/(kg K)
R	Gas constant of the working fluid	J/(kg K)
R_c	Compressor-side gas constant	J/(kg K)
R_t	Turbine-side gas constant	J/(kg K)
R_{AB}	Afterburner gas constant	J/(kg K)
R_{6A}	Mixed-stream gas constant	J/(kg K)
R_9	Nozzle-exit gas constant	J/(kg K)
h_{PR}	Fuel heating value per unit mass of fuel	J/kg
g_0	Standard gravitational acceleration	$\mathrm{m/s^2}$
g_c	Unit-conversion constant	–
$\dot{m}_0$	Freestream mass flow rate entering the engine	kg/s
$\dot{m}_f$	Total fuel mass flow rate	kg/s
α	Engine bypass ratio	–
α'	Effective bypass ratio entering the mixer	–
f	Main-burner fuel-air ratio	–
f_{AB}	Afterburner fuel-air ratio	–
f_0	Total fuel-air ratio per unit freestream air	–
F	Uninstalled thrust	N
$F/\dot{m}_0$	Specific thrust	N s/kg
S	Thrust specific fuel consumption (TSFC)	kg/(N s)
η_T	Thermal efficiency	–
η_P	Propulsive efficiency	–
η_O	Overall efficiency	–
$\eta_{P,\mathrm{phys}}$	Momentum-based propulsive efficiency	–
$\eta_{O,\mathrm{phys}}$	Momentum-based overall efficiency	–
w_c	Compressor specific work	J/kg
w_t	Turbine specific work	J/kg
w_{net}	Net cycle specific work	J/kg
q_{in}	Specific heat addition to the cycle	J/kg
q_{out}	Specific heat rejection from the cycle	J/kg
K	Kinetic-energy term used in efficiency checks	–
τ_r	Ram total-temperature ratio	–
τ_d	Inlet total-temperature ratio	–
τ_f	Fan total-temperature ratio	–
τ_c	Compressor total-temperature ratio	–
τ_λ	Main-burner exit total-temperature ratio	–
τ_b	Main combustor total-temperature ratio	–
τ_t	Turbine total-temperature ratio	–
τ_{16}	Bypass-stream total-temperature ratio	–
τ_M	Mixer total-temperature ratio	–
τ_{AB}	Afterburner total-temperature ratio	–
$\tau_{\lambda\mathrm{AB}}$	Afterburner exit total-temperature ratio	–
τ_n	Nozzle total-temperature ratio	–

Symbol	Description	Units
π_r	Ram total-pressure ratio	–
π_d	Inlet total-pressure ratio	–
$\pi_{d,\max}$	Maximum inlet total-pressure recovery	–
π_f	Fan total-pressure ratio	–
π_c	Compressor total-pressure ratio	–
π_b	Main combustor total-pressure ratio	–
π_t	Turbine total-pressure ratio	–
π_{16}	Bypass-stream total-pressure ratio	–
π_M	Mixer total-pressure ratio	–
$\pi_{M,\text{ideal}}$	Ideal mixer total-pressure ratio	–
$\pi_{M,\max}$	Maximum mixer pressure recovery	–
π_{AB}	Afterburner total-pressure ratio	–
π_n	Nozzle total-pressure ratio	–
e_c	Compressor polytropic efficiency	–
e_f	Fan polytropic efficiency	–
e_t	Turbine polytropic efficiency	–
η_d	Inlet total-pressure recovery efficiency	–
η_f	Fan isentropic efficiency	–
η_c	Compressor isentropic efficiency	–
η_b	Main-burner efficiency	–
η_t	Turbine isentropic efficiency	–
η_{AB}	Afterburner efficiency	–
η_m	Mechanical efficiency	–
η_n	Nozzle efficiency	–
A_6	Core-stream flow area at mixer inlet	m^2
A_{16}	Bypass-stream flow area at mixer inlet	m^2
A_{6A}	Constant-area mixer duct area	m^2
A_{16}/A_6	Bypass-to-core mixer inlet area ratio	–
T_{t6}	Core-stream total temperature at mixer inlet	K
T_{t16}	Bypass-stream total temperature at mixer inlet	K
p_{t6}	Core-stream total pressure at mixer inlet	Pa
p_{t16}	Bypass-stream total pressure at mixer inlet	Pa
T_{t6A}	Mixed-stream total temperature at mixer exit	K
p_{t6A}, P_{t6A}	Mixed-stream total pressure at mixer exit	Pa
u_6	Core-stream axial velocity at mixer inlet	m/s
u_{16}	Bypass-stream axial velocity at mixer inlet	m/s
u_{6A}	Mixed-stream axial velocity at mixer exit	m/s
ϕ	Partial mixing function	–
ϕ_6	Partial mixing function for the core stream	–
ϕ_{16}	Partial mixing function for the bypass stream	–
Φ	Composite mixing function	–
MFP	Mass-flow parameter	$\text{s}\sqrt{\text{K}}/\text{m}$

Symbol	Description	Units
T_{t7}	Afterburner exit total temperature	K
$T_{t7,\mathrm{cmd}}$	Commanded afterburner exit total temperature	K
p_{t7}	Afterburner exit total pressure	Pa
T_9	Nozzle exit static temperature	K
p_9, P_9	Nozzle exit static pressure	Pa
M_9	Nozzle exit Mach number	–
A_9	Nozzle exit area	m^2
A_t	Nozzle throat area	m^2
A_9/A_t	Nozzle exit-to-throat area ratio	–
$\vec{\boldsymbol{x}}$	Generic input vector	mixed SI
$\vec{\boldsymbol{y}}$	Generic output or target vector	mixed SI
$\widehat{\vec{\boldsymbol{y}}}$	Neural-network predicted output vector	mixed SI
$\vec{\boldsymbol{r}}_i$	Prediction residual for sample i	mixed SI
$\vec{\boldsymbol{x}}_{\mathrm{ideal}}$	Ideal-cycle input vector	mixed SI
$\vec{\boldsymbol{y}}_{\mathrm{ideal}}$	Ideal-cycle output vector	mixed SI
$\vec{\boldsymbol{x}}_{\mathrm{real}}$	Real-cycle input vector	mixed SI
$\vec{\boldsymbol{y}}_{\mathrm{real}}$	Real-cycle output vector	mixed SI
$\vec{\boldsymbol{y}}_{\mathrm{perf}}$	Performance-output subvector	mixed SI
$\vec{\boldsymbol{y}}_{\mathrm{state}}$	Internal-state output subvector	mixed SI
$\vec{\boldsymbol{x}}_{A,dry}$	Model A dry-expert input vector	mixed SI
$\vec{\boldsymbol{x}}_{A,wet}$	Model A afterburning-expert input vector	mixed SI
$\vec{\boldsymbol{x}}_{B,dry}$	Model B dry-expert input vector	mixed SI
$\vec{\boldsymbol{x}}_{B,wet}$	Model B afterburning-expert input vector	mixed SI
$\vec{\boldsymbol{x}}_C$	Model C real-cycle input vector	mixed SI
$\vec{\boldsymbol{x}}_{\mathrm{gas}}$	Model C gas-property input group	mixed SI
$\vec{\boldsymbol{x}}_{\mathrm{loss}}$	Model C pressure-loss input group	mixed SI
$\vec{\boldsymbol{x}}_{\mathrm{eff}}$	Model C efficiency input group	mixed SI
$\vec{\boldsymbol{x}}_{\mathrm{op}}$	Model C operating-condition input group	mixed SI
$\vec{\boldsymbol{y}}_{C,p}$	Model C primary-output vector	mixed SI
$\vec{\boldsymbol{y}}_{C,a}$	Model C auxiliary-output vector	mixed SI
X	Training input matrix	mixed SI
Y	Training target matrix	mixed SI
Y_C	Complete Model C target matrix	mixed SI
$Y_{C,p}$	Model C primary-output target matrix	mixed SI
$Y_{C,a}$	Model C auxiliary-output target matrix	mixed SI
$\mathcal{F}_{\mathrm{PCA}}$	Parametric cycle analysis mapping	–
$\mathcal{F}_{\mathrm{NN}}$	Neural-network surrogate mapping	–
$\mathcal{F}_{dry}$	Dry expert surrogate mapping	–
$\mathcal{F}_{wet}$	Afterburning expert surrogate mapping	–
$\mathcal{F}_{\mathrm{ideal}}$	Ideal-cycle neural-network expert mapping	–
$\mathcal{G}(\vec{\boldsymbol{x}})$	Learned continuous gate for future work	–
m_{AB}	Binary afterburner mode flag	–
$\boldsymbol{\theta}$	Trainable neural-network parameters	–
$\boldsymbol{\theta}^*$	Trained parameter set	–

Symbol	Description	Units
$\boldsymbol{\theta}_{dry}$	Dry-expert parameter set	–
$\boldsymbol{\theta}_{wet}$	Afterburning-expert parameter set	–
$\boldsymbol{\theta}_T$	Model C shared-trunk parameter set	–
$\boldsymbol{\theta}_p$	Model C primary-head parameter set	–
$\boldsymbol{\theta}_a$	Model C auxiliary-head parameter set	–
L	Number of neural-network layers	–
l	Neural-network layer index	–
$\boldsymbol{W}^{(l)}$	Weight matrix of layer l	–
$\boldsymbol{b}^{(l)}$	Bias vector of layer l	–
$\boldsymbol{z}^{(l)}$	Pre-activation vector of layer l	–
$\boldsymbol{a}^{(l)}$	Post-activation vector of layer l	–
$g_{\text{act}}(\cdot)$	Neural-network activation function	–
$\boldsymbol{a}^{(r)}$	Residual feature vector	–
$\boldsymbol{\psi}_C$	Model C learned shared feature vector	–
$\mathcal{T}$	Model C shared trunk mapping	–
$\mathcal{H}_p$	Model C primary-output head	–
$\mathcal{H}_a$	Model C auxiliary-output head	–
$\mathcal{D}$	Supervised dataset of input-output pairs	–
$\mathcal{D}_{\text{train}}$	Training dataset	–
$\mathcal{D}_{\text{val}}$	Validation dataset	–
$\mathcal{D}_{\text{test}}$	Test dataset	–
$\mathcal{D}_{\text{accepted}}$	Accepted dataset after filtering	–
$\mathcal{D}_{dry}$	Dry-mode dataset	–
$\mathcal{D}_{wet}$	Afterburning-mode dataset	–
$\mathcal{D}^{(s)}_{\text{accepted}}$	Accepted subset from shard s	–
Ω	Sampled input domain	–
Ω_{phys}	Physically acceptable output set	–
N	Number of samples	–
N_{train}	Number of training samples	–
N_{val}	Number of validation samples	–
N_{test}	Number of test samples	–
N_{grid}	Number of samples in a full grid	–
N_s	Number of candidates in shard s	–
N_{shard}	Total number of generated shards	–
d	Number of sampled input dimensions	–
n_x	Number of input variables	–
n_y	Number of output variables	–
i	Sample index	–
j	Input or output component index	–
k	Generic index or optimizer iteration index	–
$x_{j,\min}$	Lower bound of input variable j	varies
$x_{j,\max}$	Upper bound of input variable j	varies
$\xi_{i,j}$	Normalized sampling coordinate	–
$\vec{\boldsymbol{\xi}}_i$	Normalized Sobol candidate vector	–

Symbol	Description	Units
$\ell_{i,j}$	Permuted Latin-hypercube interval index	–
$\nu_{i,j}$	Random Latin-hypercube interval offset	–
$\mathcal{S}_{\xi}^{(s)}$	Normalized candidate set for shard s	–
χ	Dimensionless parametric-sweep coordinate	–
$\tilde{x}_{i,j}$	Normalized input value	–
$\tilde{y}_{i,j}$	Normalized output value	–
$\widehat{\tilde{y}}_{i,j}$	Normalized predicted output value	–
$\mu_{x,j}$	Training-set mean of input variable j	varies
$\sigma_{x,j}$	Training-set standard deviation of input variable j	varies
$\mu_{y,j}$	Training-set mean of output variable j	varies
$\sigma_{y,j}$	Training-set standard deviation of output variable j	varies
ϵ	Small numerical stabilization constant	–
w_j	Loss weight for output variable j	–
$\mathcal{L}$	Generic training loss function	–
$\mathcal{L}_{\mathrm{MSE}}$	Mean-squared-error loss	–
$\mathcal{L}_{W}$	Weighted normalized mean-squared-error loss	–
$\mathcal{L}_{\mathrm{data}}$	Data-mismatch loss term	–
$\mathcal{L}_{\mathrm{residual}}$	Physics-residual loss term	–
$\mathcal{L}_{\mathrm{informed}}$	Conceptual physics-informed loss	–
$\mathcal{L}_{\mathrm{guided}}$	Physics-guided supervised loss	–
ω_{phys}	Physics-residual weighting coefficient	–
$\mathcal{L}_b$	Mini-batch loss	–
$\mathcal{L}_{\mathrm{train}}^{(n)}$	Training loss at epoch n	–
$\mathcal{L}_{\mathrm{val}}^{(n)}$	Validation loss at epoch n	–
$\mathcal{L}_{dry}$	Dry-expert training loss	–
$\mathcal{L}_{wet}$	Afterburning-expert training loss	–
N_b	Mini-batch size	–
B_n	Number of mini-batches in epoch n	–
n	Epoch index	–
$\boldsymbol{g}_b$	Mini-batch gradient vector	–
$\boldsymbol{g}_k$	Gradient vector at optimizer iteration k	–
$\boldsymbol{m}_k$	Adam first-moment estimate	–
$\boldsymbol{v}_k$	Adam second-moment estimate	–
$\widehat{\boldsymbol{m}}_k$	Bias-corrected Adam first-moment estimate	–
$\widehat{\boldsymbol{v}}_k$	Bias-corrected Adam second-moment estimate	–
β_1	Adam first-moment decay coefficient	–
β_2	Adam second-moment decay coefficient	–
ζ_k	Optimizer step size at iteration k	–
ζ_n	Scheduled optimizer step size at epoch n	–
ζ_0	Initial optimizer step size	–
$\zeta_{\min}$	Minimum optimizer step size	–
δ_{lr}	Learning-rate decay factor	–
E_d	Learning-rate decay interval in epochs	–

Symbol	Description	Units
$\widehat{y}_i$	Predicted scalar output value	varies
y_i	Reference scalar output value	varies
r_i	Scalar residual in validation studies	varies
RMSE	Root-mean-square error	varies
MAE	Mean absolute error	varies
E_i^{rel}	Relative error for evaluation point i	–
t_{PCA}	Runtime of deterministic cycle solver	s
t_{NN}	Runtime of neural-network surrogate	s
$\mathcal{R}_{\mathrm{speed}}$	Runtime speedup factor	–

Chapter 1: Introduction

1.1 Motivation

Turbofan engines are widely used in modern aircraft, with differing configurations serving in aircraft ranging from airlines to modern fighter jets. Turbofan engines have the advantage of producing high thrust with improved fuel efficiency by combining flow from the turbojet core and the bypass, but this comes with a decrease in top speed. The use of an afterburner and a low bypass is a common method to increase the operating speed into supersonic regimes, with modern fighter jets such as the F-22 Raptor utilizing afterburning turbofan engines. Mixed flow afterburning turbofan engines are becoming increasingly relevant to the functionality of modern fighter aircraft, responsible for efficient cruise regimes while retaining the capability of rapid acceleration commonly seen in afterburning systems. This project focuses on the analysis of such mixed flow configurations, with the goal of enabling adaptive afterburner behavior. By simulating a wide range of heat addition levels and afterburner configurations over a wide range of operating conditions and training a machine learning (ML) model to predict the resulting performance and mixed flow properties, this project aims to demonstrate the feasibility of real time engine prediction and optimization for next-generation propulsion systems.

1.2 Literature Review

Mixed flow afterburning turbofan engines are inherently complex systems, and a literature review is necessary to determine the parameters that drive performance as well as the feasibility of machine learning models for prediction of the dynamics. Afterburners add heat while incurring pressure loss while variable-area nozzles set back-pressure that becomes coupled to fuel addition. Literature review is conducted to further understand how afterburners and variable geometry nozzles operate. Further literature review is conducted for an improved understanding of mixed flow turbofan engines and similar adaptive cycle engines. Additional literature review is conducted on the use of machine learning models for engine analysis and the relative success of this methodology for engine performance assessment. Lastly, literature review is conducted on the potential application of twin surrogate models for active thrust control of such engines with efficiency in mind. The literature review motivates the

use of a physics-guided forward surrogate model for the prediction and analysis of mixed flow afterburning turbofan engine.

1.2.1 Turbofan Augmentation Studies

Afterburner augmentation adds heat downstream of the turbine to raise exhaust stagnation temperature and consequently increase jet velocity and thrust at the expense of specific fuel consumption. In subsonic ducts this is a Rayleigh-type heating process: total temperature rises while total pressure falls due to friction and mixing losses, so hardware should maximize heat addition while limiting pressure loss. Marshall and Canuel [1] outline the fundamental elements of a turbofan afterburner: diffuser, flameholders (often V-gutters or bluff bodies), fuel spray bars or manifolds, a combustion liner with dilution/film holes, igniters, and a variable-geometry convergent-divergent nozzle. The study explains the central trade: higher flameholder blockage and stronger recirculation stabilize combustion, but add a pressure drop that lowers net thrust. They also emphasize that the variable nozzle is not merely a thrust device; it is a matching element that sets back-pressure on the turbine/fan system, protecting compressor stability and enabling augmented operation across altitude and throttle.

Altitude testing by Johnsen and Cullom [2] demonstrates how those differing configurations behave under flight-representative conditions. Using a turbofan fitted with several afterburner configurations and a hydrogen heater to simulate elevated turbine discharge temperature, they show that higher inlet temperature to the afterburner enlarges the stable operating envelope, reduces ignition delay, and improves augmentation efficiency. The study found this occurs because energy is consumed to raise reactants to ignition and reaction-rate-dominated regimes. Across configurations, thrust increment and TSFC penalty are sensitive to flameholder/liner design and the nozzle schedule: designs that improved mixing and heat release uniformity tended to yield larger, more repeatable thrust gains, provided liner pressure losses were kept in check. Equally important, the tests highlight the system nature of augmentation; nozzle area changes alter engine matching and should be coordinated with afterburner fuel flow to avoid fan/turbine mismatch at altitude.

A separate NASA paper [3] investigation into the low-frequency combustion instability in a turbofan afterburner at altitude isolates the operating and geometric factors that precipitate "rumble"-type oscillations. Johnsen and Cullom identify a feedback loop between unsteady heat release and the system's acoustic compliance/inertance, and show that equivalence ratio, fuel distribution among spray bars, flameholder spacing and blockage, liner perforation/damping characteristics, and ambient pressure collectively tune the susceptibility to large-amplitude, low-frequency modes. Altitude reduces aerodynamic damping and shifts acoustic boundary conditions, making some configurations that are benign on the ground become unstable aloft. Practically, this yields clear constraints: avoid fuel schedules and flameholder/liner combinations that couple strongly to the dominant acoustic length scales; stage fuel to desensitize heat-release phasing; and use nozzle scheduling to keep afterburner operation within stable pressure and temperature bounds. Due to the complexity of combustion instability interactions the engine model will assume the ideal case, however for real application combustion instability has to be factored into the analysis.

Taken together, these sources establish the baseline physics and constraints that govern augmented operation in turbofan architectures. Marshall and Canuel provide the definitive hardware/control map and the core performance tradeoffs (thrust gain versus pressure loss and fuel penalty), while Johnsen and Cullom supply altitude-validated evidence on how configuration and scheduling affect ignition reliability, augmentation efficiency, and stability boundaries. For a mixed-flow afterburning turbofan, the implications are direct: the afterburner fuel flow and variable nozzle area should be coordinated properly to maintain efficient function and desired thrust levels.

1.2.2 Mixed Flow Turbofan Modeling Studies

Mixed-flow turbofans route the fan and core streams through a mixer to a common augmentor and nozzle, trading the hardware simplicity of separate streams for better propulsive efficiency and controllability across a wider range of flight conditions. By mixing before expansion, the engine presents a single back-pressure to the upstream turbomachinery, making the coupling between afterburner fuel flow rate and nozzle area more straightforward. Conceptually, the mixer enforces a single exit static pressure while conserving mass, momentum, and total enthalpy; losses are captured with a mixer pressure-loss coefficient in their analysis and a mixing effectiveness parameter that accounts for incomplete thermal and velocity uniformity. Downstream, an afterburner raises total temperature, and a variable-geometry nozzle sets the expansion ratio and thus the exhaust Mach number and thrust. These features make the architecture an attractive basis for high-speed concepts where high control fidelity of engine functionality is desired for a wide flight envelope.

Piccirillo, Gregorio, Fusaro, Ferretto, and Viola developed a conceptual-design model of a mixed-flow turbofan tailored to sustainable supersonic applications and, crucially, organize the station-by-station thermodynamic relationships so they can be parameterized cleanly for early-phase trade studies [4]. Their formulation uses component maps and polytropic efficiencies for the fan and core compressors/turbines, couples the streams through a steady one-dimensional mixer block with specified pressure losses, and propagates the mixed state through an augmentor/nozzle module that handles choking and expansion. The model exposes the primary variable design criteria: the bypass ratio, fan pressure ratio, overall pressure ratio, turbine inlet temperature, mixer loss and effectiveness, afterburner temperature rise, and nozzle area. The model from the study returns the performance metrics needed at conceptual level such as net thrust, specific thrust, and thrust specific fuel consumption. Because the variables are explicit and the losses are lumped into a small set of coefficients, the framework supports large parametric sweeps without burying results in detailed combustor or turbomachinery CFD, yet in theory may still be slower than a single neural network run.

For this project, that structure maps directly to the inputs and outputs of a physics-guided surrogate, while highlighting some parametric cycle analysis equations for the simplified physics. The parameter vector for the training dataset naturally follows similar organization typical of parametric cycle analysis: core and fan pressure ratios and efficiencies, bypass ratio, mixer loss and effectiveness, augmentor heat addition, and nozzle area (or effective expansion ratio). The forward model in this project also targets similar conceptual metrics

(thrust, specific thrust, TSFC) augmented by internal states needed for feasibility checks (such as mixed-stream total pressure and temperature entering the augmentor, nozzle throat Mach indication for choking logic). Because the formulation in this study already treats the mixer afterburner and nozzle sequence as a compact block with explicit coupling to upstream matching, it provides a clear template for embedding physical limits and scheduling logic (fuel flow and nozzle area) into the surrogate model. In short, their mixed-flow engine representation supplies an example of the physics and parameters to create a fast surrogate backed by parametric cycle analysis while keeping the model small enough for potential control-oriented learning later on.

1.2.3 Adaptive Cycle Engine Studies

Adaptive cycle engine research frames how variable geometry and flow management reshape engine performance across widely separated operating points. The study by Chen, Deng, Zhang, and Tang [5] discusses architectures that modulate bypass, fan/core work split, and nozzle scheduling to preserve efficiency while unlocking high specific thrust at demand, establishing the evaluation metrics used across modern studies. A different study by Zou and co-authors [6] show how to couple a one-dimensional cycle with three-dimensional variable-geometry turbine models in a loop, illustrating how geometry schedules propagate upstream as matching constraints. Conceptually this is similar to the kind of coupling a mixed-flow augmentor and variable nozzle will impose on fan and turbine maps. On the thermodynamic side a study by Chen, Wang, Chen, Cai, and Zhang [7] shows that cooling-air management needs to be considered as a design variable if multiple bypass flows are considered because it redistributes irreversibilities and shifts exergy efficiency and infrared signature in tandem. This adds potential for adding efficiency and IR observability proxies to a fundamentally parametric cycle based model. Finally, a study by Aygun [8] and colleagues framed adaptive cycle design as a multi objective optimization problem over mission segments, balancing thrust targets against fuel use and constraints. This provides an interesting potential structure for setting objectives and penalties when training a fast surrogate model for predicting mixed-flow afterburning turbofan behavior.

A study by Chen, Deng, Zhang, and Tang [5] focuses on how variable geometry and flow management affect performance across mission segments, highlighting a common set of parameters that map directly onto a mixed flow afterburner. Architectures in this study are framed around modulating bypass and work split (valved flowpaths, variable splitter scheduling), adjusting effective expansion through variable area fan and core nozzles, and retiming component operating points via variable stator vanes, bleed, and turbine cooling control. Together these parameters trade specific thrust against fuel economy while preserving surge margin and staying within thermal limits. The study emphasizes mode scheduling and the need to evaluate designs over mission profiles rather than point conditions, using thrust, thrust specific fuel consumption, operability margins, turbine inlet temperatures, and sometimes exergy derived efficiency as criteria. The study documents the prevailing use of one dimensional cycle models augmented by calibrated loss models with augmentation where effects from geometry variance feed back into cycle. The key takeaway for a mixed flow

afterburning turbofan is the importance of coordinated scheduling for variable components and systems: nozzle area defines back-pressure and component parameter matching, fuel addition in the combustion chamber as well as the afterburner defines thermal states, and both have to be optimized across altitude and Mach to maintain stability and efficiency while meeting thrust targets, which directly translates to the model design in this project.

A study by Zou and co-authors [6] demonstrates a multi-fidelity approach to adaptive cycle engines that couples a one-dimensional cycle similar to previous papers with a three-dimensional variable geometry turbine model. This effectively means the changing geometry schedules propagate upstream as matching constraints. In this study the 3D turbine resolves how vane and rotor settings reshape incidence, loss, and effective flow capacity, and those effects are fed back into the 1D solver. An iterative co-simulation enforces shaft power balance and pressure-ratio compatibility while the geometry is varied over the flight conditions. The study looks at vane angle laws, bleed and cooling settings, and nozzle area coordination and shows how they alter compressor operation and surge margin through back-pressure. The result is a cycle that can "feel" actuator schedules in a physically meaningful way, capturing off-design penalties and recoveries tied to real hardware motion. In a mixed-flow afterburner context, the same pattern can be applied: nozzle-area laws and mixer and liner loss parameters can be treated as geometry dependent parameters that feed back into the fan and turbine rather than as fixed coefficients. This however can be complex and for this project may be considered based on time.

A separate study by Chen and others [7] treat cooling air management as a first-class design variable and track its impact on propulsion performance, thermodynamic and exergy efficiencies, and infrared observables across operating conditions. Cooling flows reshape where irreversibilities occur by raising mixing losses and pressure drops in some stations while lowering part temperatures and enabling higher firing temperatures. The authors show how these redistributions shift exergy destruction among compressor discharge, turbine stages, and hot-section mixing. Because afterbody and plume temperatures respond to both total temperature rise and the manner in which cooling air is injected, the same settings that improve durability can increase or decrease infrared intensity depending on the balance between core temperature, dilution effectiveness, and expansion ratio. The paper links parameter choices to performance outcomes, making clear that objectives like thrust specific fuel consumption and infrared intensity cannot be optimized independently. The study uses a parametric cycle level backbone augmented with calibrated loss models and signature proxies, showing how a limited set of state variables can drive both efficiency and observability metrics. This provides a template for adding exergy-based figures of merit and IR proxies to models without resorting to full CFD or radiation solvers. For mixed flow afterburners, adopting cooling-flow fractions or effective film-cooling parameters as explicit inputs adds to potential optimization of thrust and TSFC within a parametric cycle framework.

Another study by Aygun, Cilgin, Ekmekci, and Turan [8] approached adaptive-cycle engine design as a multi-objective problem evaluated over mission segments rather than isolated points. The study treats competing goals (meeting thrust targets while minimizing fuel use) as simultaneous objectives and utilizes hard constraints (surge margin, temperature

limits, choking and back-pressure compatibility). Decision variables are scheduled rather than static, reflecting how adaptive architectures work in practice; nozzle area laws, flow-split/bypass control, and other geometry or bleed and cooling settings are varied across flight conditions to keep components matched. A cycle analysis backbone computes state and performance for each schedule realization, and solutions are compared to find where small fuel savings demand large thrust compromises or the other way around. The study finds that a variable nozzle schedule that improves high-demand thrust can degrade part-power efficiency unless it is coordinated with the rest of the flowpath. The researchers avoid designs that look favorable at a single condition but underperform across the mission profile. For a mixed-flow afterburning engine, the same structure is directly applicable; balance thrust tracking and efficiency, enforce feasibility, and compare multiple parameters rather than single settings. This means evaluating afterburner fuel flow and variable-nozzle area over Mach altitude and flow rates together, and scoring candidates by thrust error and thrust specific fuel consumption alongside constraint penalties. The Aygun formulation also motivates using weighting during data generation and model training so the model learns the right priorities, not just pointwise accuracy for thrust control. Because trade study efficiency hinges on fast evaluations, their approach provides a template for pairing a parametric cycle based surrogate with a multi-objective optimization. The study's constraint handling suggests a clean way to encode feasibility into loss terms through soft penalties for approaching surge or thermal limits, which can be useful for limiting model output to realistic values and possibly for guiding an engine controller model for a mixed-flow afterburning turbofan.

1.2.4 Machine Learning Prediction Studies

A study by Kim, Kim, and Son [9] demonstrated that data-driven surrogates can capture engine behavior with sufficient fidelity and speed to support system-level studies and control development. Their approach pairs a parametric cycle analysis structure with learned mappings, allowing the simulator to track thrust and thermodynamic states through throttle changes without the computational cost of full component-map dynamics at every step. The model is trained on representative operating trajectories so that rate-dependent effects and actuator lags are preserved at the system level, and it is validated by comparing predicted time histories of thrust and state variables against a higher-fidelity reference. Their model keeps the inputs and outputs aligned with physical quantities such as fuel flow commands, spool speeds, temperatures, pressures. This allows the model to be embedded alongside controllers without additional translation layers. The result is a predictor that is light enough for rapid use but anchored in the same variables used for cycle matching and operability checks. These factors make their model a practical template for fast forward models in propulsion applications.

Another study by Luan, Li, Cao, and Gu [10] focuses on the "digital twin" use case and shows how surrogate models can monitor engine conditions by mapping measured data to residuals that find deviations from nominal behavior. The study uses a physics-aware baseline to define the state and performance expected under current conditions, then a compact surrogate estimates corrections for degradation, sensor drift, or minor off-nominal engine

operation. By structuring the surrogate around tangible parameters such as efficiencies, flow capacities, or loss coefficients, their method supports fast condition inference. Their approach also emphasizes data curation and feature selection so that the surrogate remains accurate across operating regimes rather than overfitting to a small slice of the operational envelope. The surrogate model made by the researchers reconciles commanded schedules and measured responses, functioning as a lightweight estimator of the underlying cause when the system leaves normal expected operation.

A study by Wang, Ji, Chen, Chen, Wang, and Shi [11] presents a neural network approach that predicts infrared output from an axisymmetric turbofan exhaust using standard parametric cycle inputs. Their model maps engine and exhaust properties such as mixed-stream total temperature upstream of the nozzle, mass flow, nozzle pressure ratio and effective expansion ratio, area distribution along the nozzle, and plume mixing parameters to infrared heat signature estimates. By learning the relationships that traditional radiation solvers capture, the network delivers several orders of magnitude speedup while retaining accuracy. The formulation keeps the predictors physically interpretable using thermodynamic states and geometric changes. This makes it straightforward to couple model predictions with actuator schedules like afterburner fuel flow and nozzle area. For a mixed flow afterburner, this implies a direct pathway from cycle states and nozzle scheduling to fast IR estimates. This demonstrates the power of properly trained neural network models for quick prediction of complex behavior in a similar problem.

For this project, these studies motivate a forward surrogate that outputs physically meaningful states alongside thrust and TSFC to regularize training and support feasibility checks. Adopting Kim's alignment of inputs and outputs with cycle variables could keep the model reusable for a control model, while Luan's digital-twin framing suggests adding health residuals or effective-parameter estimates as extra heads so the surrogate can detect and penalize unrealistic output states, with Wang's model demonstrating potential for extending the surrogate to further engine properties. Together, these studies provide considerations for a control-ready engine dynamics predictor that not only computes thrust and TSFC use but also monitors output feasibility.

1.2.5 Learning-Based Thrust Control Studies

Another application of neural network models for mixed flow afterburning turbofan engine dynamics is the possibility of utilizing a similar method to develop a physics-informed controller. A study by Zhu, Pan, Zhou, and Huang [12] shows the potential of neural network driven direct thrust control where fuel flow and variable geometry is coordinated to achieve thrust targets while respecting operability limits. The researchers' reinforcement/deep-learning approach effectively trains an approximate inverse model of the engine dynamics from simulation data, using thrust objectives augmented by penalties on fuel use, actuator saturation, and rate limits for usability on a real engine. The controller model takes spool speeds, temperatures, pressures, and nozzle area as input and outputs parameter changes to meet targets which makes integration with existing systems possible. By training across a spread of operating conditions and injecting model uncertainty, their method improves

generalization, reducing reliance on gain-based classical loops at the edges of the operational envelope. Safety is handled with constraint layers or barrier terms so surge margin, temperature ceilings, and nozzle choking boundaries are not violated. The result is a model that delivers fast thrust convergence without excessive fuel transients and that can be integrated with conventional schedules to smooth operational mode transitions. For a mixed-flow afterburning engine, the same structure can in theory be applied: An inverse model could alter afterburner fuel flow and nozzle area to meet thrust commands while keeping and back-pressure in operable regions for the added possibility of control.

A similar study by Feng, Hao, Wang, Nie, and Song [13] focuses on the afterburner state specifically and uses a machine learning framework to optimize thrust through the adjustment fuel into the afterburner alongside nozzle area changes. The general idea is to embed a physics-informed model that approximates the afterburner dynamics from available measurements, then uses that state to compute control moves with respect to thrust and efficiency objectives under stability and temperature constraints. Because nozzle back-pressure couples strongly to afterburner performance, the method treats nozzle area and fuel rate as a joint decision, updating both simultaneously as the deduced state evolves with altitude, Mach, and throttle. Training focuses on consistency between measured responses and model-implied states, so the controller remains efficient and interpretable. Constraint handling is built into the deduction layer and objective, which helps avoid low frequency combustion instability regions and protects heat limits. This let the researchers create a thrust optimization control model that achieves desired augmentation levels with less fuel and fewer oscillations than naive open-loop scheduling similarly to the previous study discussed. For a mixed-flow afterburning turbofan, this study provides a framework for an inverse controller with the underlying physics embedded in the model, making the possibility of adaptive engine cycles possible.

1.3 Project Proposal

To analyze and simulate the mixed flow properties of afterburning turbofan engines as well as obtain performance characteristics, this project will utilize MATLAB modeling alongside machine learning to model the performance and flow characteristics of varying mixed flow afterburning turbofan engine configurations. Simulating and analyzing various configurations of ideal and real mixed flow afterburning turbofan engines over a wide range of operating conditions allows for a large dataset for analysis as well as for training the ML model guided by parametric cycle analysis. The trained model will serve as a tool for evaluating performance trends, enabling real time exploration of adaptive afterburner strategies. This work aims to demonstrate the feasibility of using ML to support data-driven design and analysis with potential control of advanced propulsion systems.

1.4 Methodology

To create an accurate model for mixed flow turbofan engines, characterization of standard fuels, bypass ratios, mixing methods, and afterburner geometries will be identified through literature review as necessary to define an initial set of configurations used for primary simulation. These configurations will be modeled using MATLAB based simulation tools to obtain the thermal and fluid dynamic effects of heat addition post mixing to identify optimal nozzle conditions and obtain engine performance characteristics such as thrust specific fuel consumption, exhaust flow properties and engine efficiency. Incremental variation on the set of configurations will be introduced to obtain a larger dataset for training the machine learning model. The results of this parametric cycle analysis dataset will be utilized to train a neural network using MATLAB's deep learning toolbox, enabling rapid prediction of flow properties and engine performance across a wide range of configurations. The machine learning model will be validated against simulation results to assess its capabilities for data extrapolation and generalization on configurations outside of the training dataset. Through iterative simulation and neural network training as well as fine tuning, a ML model for predicting mixed flow properties and engine performance in differing afterburning configurations will be obtained, serving as a scalable and time efficient alternative to traditional methods.

Chapter 2: Engine Overview

2.1 Mixed Flow Afterburning Turbofan Engine

Modern fighter aircraft typically use low bypass ratio, mixed-flow afterburning turbofan engines to provide the high thrust and acceptable specific fuel consumption required for both subsonic cruise and sustained supersonic flight. One example is the Pratt & Whitney F100 engine which can be found on the F-15 and F-16, shown in Fig. 2.1. In this configuration, a small amount of the inlet air is routed through a high pressure turbojet core to generate power to drive the fan. The bypass flow, driven by the fan, is used primarily to increase mass flow rate and thrust while reducing exhaust jet velocity when operating with the afterburner off. Downstream of the turbines, the core and bypass streams are mixed and can be further heated in an afterburner before expansion through a variable geometry exhaust nozzle. This engine design allows the turbofan to operate efficiently in cruise while still providing large increases in thrust when the afterburner is engaged.

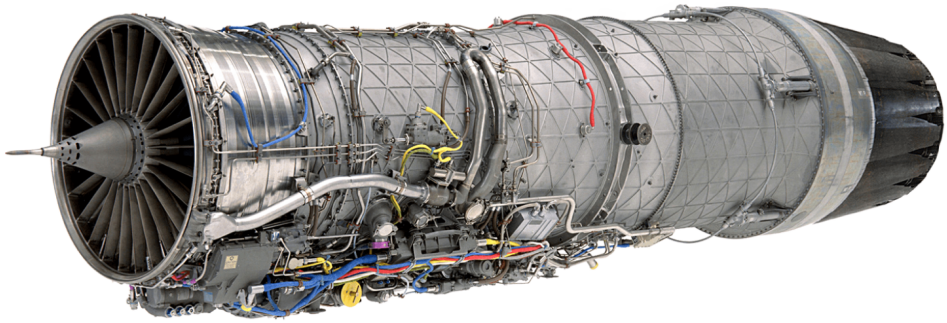


Figure 2.1 – Pratt & Whitney F100 engine [14]

Fundamentally the mixed-flow afterburning turbofan comprises an inlet, a multi-stage fan, a core compressor, a main combustor, turbines, a mixer, an afterburner, and a variable-area nozzle, as illustrated in Fig. 2.2. Ambient air enters through the inlet and is guided to the fan, where it is compressed and split into a bypass stream and a core stream. The core stream passes through the high pressure compressor and combustor, where fuel is added and combusted to raise the stagnation temperature before expansion through the turbines, which extract work to drive the compressor and fan. The turbine exhaust then mixes with the cooler

bypass air in a mixing section, forming a single flow stream that feeds the afterburner. When higher thrust is needed, additional fuel is injected and burned in this mixed stream which substantially increases exhaust temperature before the products are expanded through the variable geometry nozzle to produce thrust. The following subsections provide an overview of each of these components in more detail, with emphasis on the features relevant to parametric cycle analysis and ML model development.

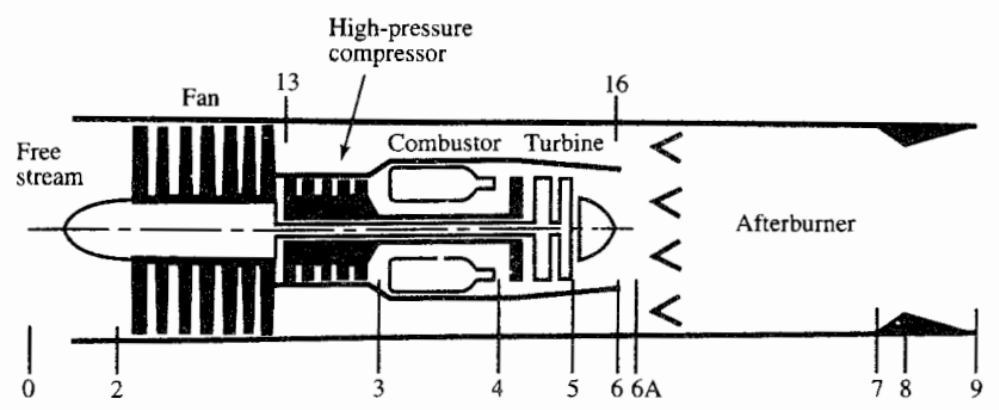


Figure 2.2 – Schematic of mixed flow afterburning turbofan engine [15]

2.2 Fan and Bypass

The fan is the first component encountered by the inlet air and is similar to a low pressure compressor operating on the full engine mass flow. Its role is to raise the stagnation pressure of the incoming air with relatively small temperature rise. This increases the mass flow and provides a significant fraction of the total engine thrust especially when afterburners are disengaged. Downstream of the fan, the flow is typically divided into a core stream and a bypass stream. The core stream passes into the compressor and main turbojet core of the engine, driving the fan system. The bypass stream allows the engine to accelerate a larger mass of air to a lower exhaust velocity, improving propulsive efficiency in regimes where afterburning is not required, while also providing additional cooling and mixing benefits downstream when the two streams are recombined.

In mixed flow afterburning turbofans used in fighter aircraft, the bypass ratio is relatively low compared to that of modern commercial turbofans. High bypass engines used on transport aircraft bypass most of the inlet flow around the core, producing thrust primarily by accelerating a very large mass of air to a velocity only moderately above the flight speed; this configuration minimizes specific fuel consumption and jet noise but results in large fan diameters and lower exhaust velocities. In contrast, low bypass fighter engines retain a greater proportion of flow through the core which enables higher exhaust velocities, smaller frontal area, and better integration with supersonic inlets and variable area nozzles, at the expense of cruise fuel efficiency. The mixed flow arrangement considered in this work leverages a modest bypass stream to gain some of the efficiency and mixing benefits of a turbofan, while

preserving the high specific thrust and compact geometry required for fighter aircraft.

2.3 Compressor

Following the fan, the core flow enters the compressor stage which raises the stagnation pressure of the air before it reaches the combustor. Increasing the compressor exit pressure directly increases the thermal efficiency and specific thrust of the cycle, but it also drives a significant portion of the mechanical power requirements of the engine. Modern turbofan engines use multistage axial compressors that are designed to deliver high overall pressure ratios with high efficiency over the operating envelope. An axial compressor consists of alternating rows of rotating blades (rotors) and stationary vanes (stators). Each stage imparts a small pressure rise with typical pressure ratios of 1.2 to 1.4 per stage, stacking these stages to achieve the desired overall pressure ratio. The rotor blades add kinetic energy to the flow by increasing its tangential velocity component, while the downstream stator vanes diffuse this swirl and recover it as static pressure. For the mixed flow afterburning turbofan engines considered in this project, the compressor design strongly influences overall engine performance metrics such as overall pressure ratio, engine thrust, thrust specific fuel consumption. These parameters will feed directly into the parametric cycle analysis and into the training data for the surrogate model. A schematic of a typical multistage axial compressor is shown in Fig. 2.3.

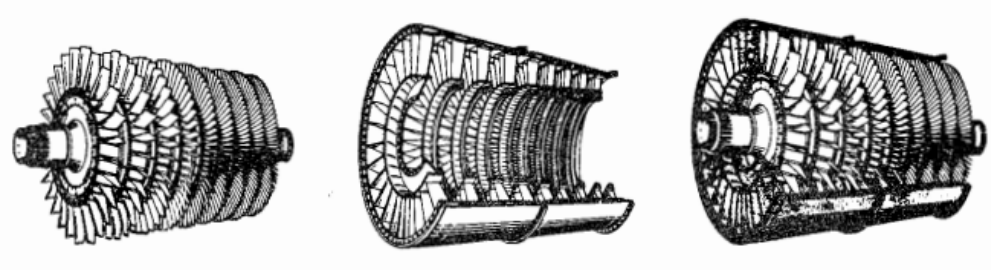


Figure 2.3 – Schematic of axial compressor [15]

2.4 Combustor

The combustor is the component in which chemical energy from the fuel is converted into an increase in the stagnation temperature of the core flow. Air leaving the compressor enters the combustor at high pressure and moderate temperature; fuel is injected and mixed with a portion of this air, and then is burned. The combustor provides reliable ignition and stable combustion over a wide range of inlet pressures and temperatures, and an acceptable exit temperature profile while minimizing total pressure loss. Several basic combustor geometries are used in turbofan engines, as shown in Fig. 2.4. Early engines often employed separate can combustors which simplifies development and maintenance but takes up more space. Annular combustors form a ring around the engine axis, which is typically a more compact and

aerodynamically efficient configuration at the cost of higher structural and manufacturing complexity. Can-annular designs combine features of both, with individual cans arranged within a common annular casing. Modern military turbofans such as those considered in this project typically use annular or can-annular combustors. In this project, the combustor is modeled through its overall pressure loss, exit stagnation temperature, and effective fuel to air ratio, which serve as inputs to the downstream turbine and to the cycle model used to generate the surrogate training data.

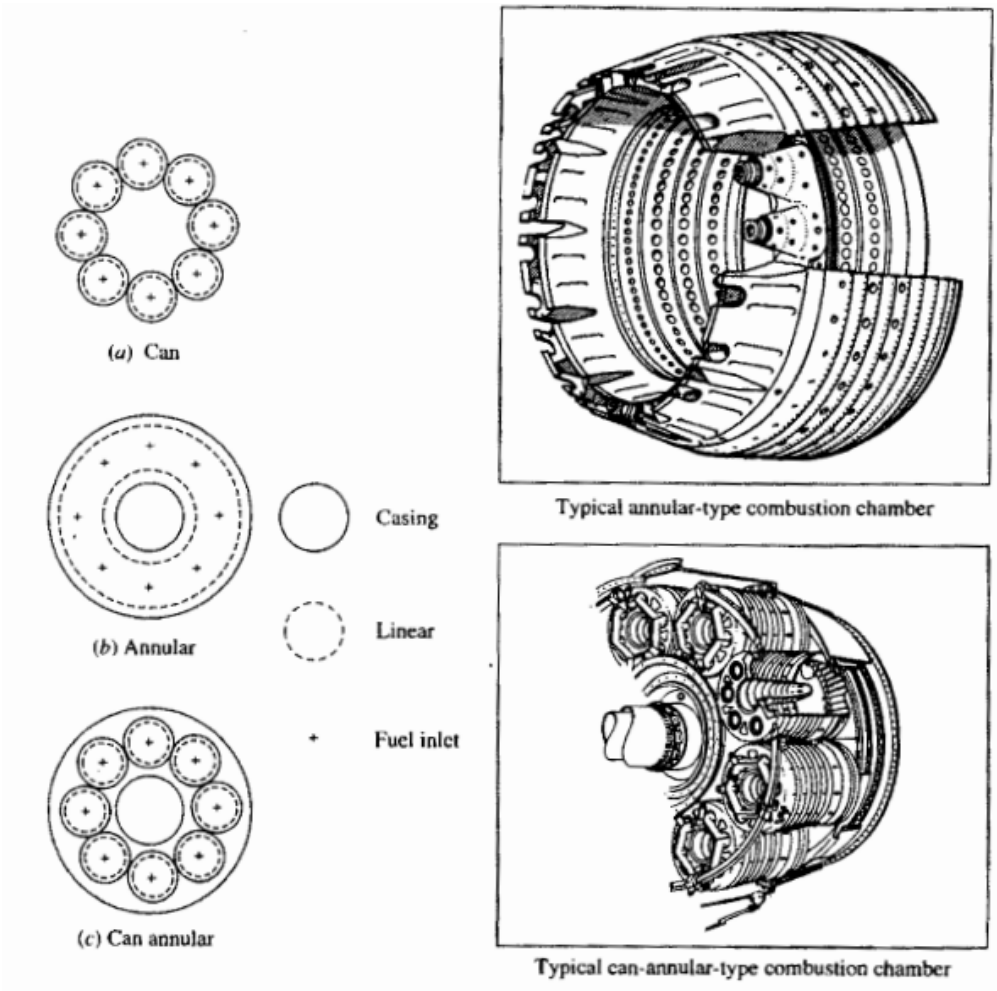


Figure 2.4 – Schematic of combustion chamber cross section [15]

2.5 Turbine

The turbine extracts mechanical power from the high temperature flow out of the combustor to drive the upstream compressor and fan. In a mixed flow afterburning turbofan this is accomplished by a high-pressure turbine which is coupled to the high-pressure compressor, and a low-pressure turbine which drives the fan through a concentric shaft system. As the gas expands through each turbine stage, its stagnation pressure and temperature decrease

while shaft work is produced. The turbine must extract enough power to meet the compressor and fan requirements while maintaining high efficiency and acceptable exit conditions for the mixer and afterburner. Like the compressor, the turbine in reality is a multiple stage axial machine consisting of alternating rows of stationary nozzles and rotating blades. The nozzles convert a portion of the gas enthalpy into kinetic energy by accelerating the flow, and the rotor blades then extract work from this high velocity flow. Because the turbine operates at very high temperatures, cooling is required to keep blade metal temperatures within allowable limits which is done with film cooling flows which will not be considered in flow rate contribution for the sake of model simplicity. The pressure and temperature ratios across the turbine as well as the turbine efficiencies all influence the overall cycle efficiency and the flow conditions at the turbine exit. For this project, the turbine is modeled at a system level through its stage or lumped efficiencies, pressure ratios, and work balance with the compressor and fan. These parameters determine the core exhaust stagnation state delivered to the mixer, which in turn sets the baseline for afterburner operation and nozzle expansion. A typical axial-flow turbine is shown in Fig. 2.5.

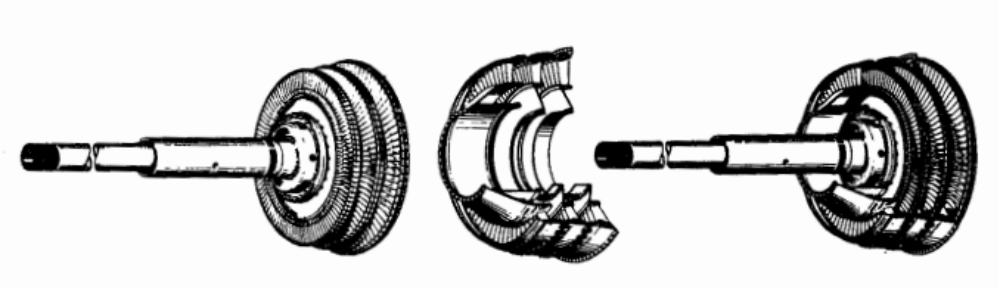


Figure 2.5 – Schematic of axial turbine [15]

2.6 Mixer

The mixer combines the hot core stream leaving the turbine with the less energetic bypass stream to form a single, mixed flow that feeds the afterburner and nozzle as illustrated in Fig. 2.6. In the configuration considered here, the bypass flow enters the mixer at station 16 and the core flow at station 6, separated by a splitter plate, and they are brought together within a ducted mixing region. By redistributing momentum and enthalpy between the two streams, the mixer sets the mass-averaged temperature, velocity, and composition of the flow at the mixed-flow station 6A. These mixed properties strongly influence afterburner performance, nozzle expansion, and the thrust and specific fuel consumption of the engine. In reality, mixing is governed by complex three dimensional turbulent flow dynamics and geometry dependent losses. These effects can introduce additional changes in total temperature and pressure for the mixed flow stream. To keep the cycle modeling feasible, this project adopts an idealized representation in which the core and bypass streams are perfectly mixed at station 6A, subject to conservation of mass, momentum, and energy, and with the net effect of irreversible processes captured through a prescribed total pressure loss across the mixer. This ideal mixer assumption provides a physically consistent and computationally

efficient link between turbine exit conditions, afterburner operation, and nozzle performance in the surrogate model.

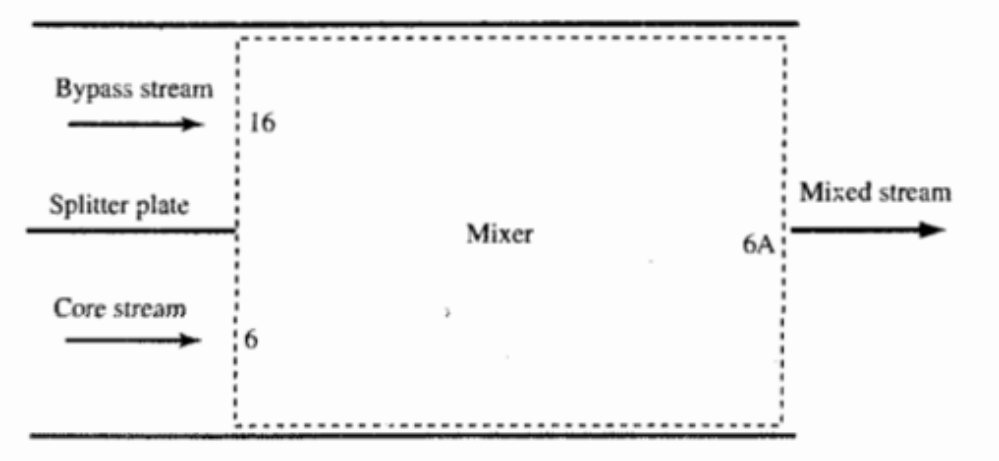


Figure 2.6 – Schematic of ideal flow mixer [15]

2.7 Afterburner

The afterburner is a secondary combustion system located downstream of the mixer, where additional fuel is burned in the mixed stream to produce large increases in thrust. As shown in Fig. 2.7, the flow first passes through a diffuser section that reduces Mach number and promotes stable combustion, then enters a region containing spray bars or fuel manifolds, flame stabilizers, and an annular liner. Because the flow at this station already contains substantial oxygen from the bypass stream, complete combustion can be achieved without additional compressor work. This results in a significant rise in stagnation temperature and mass flow that, when expanded through the nozzle, produces a large increase in net thrust.

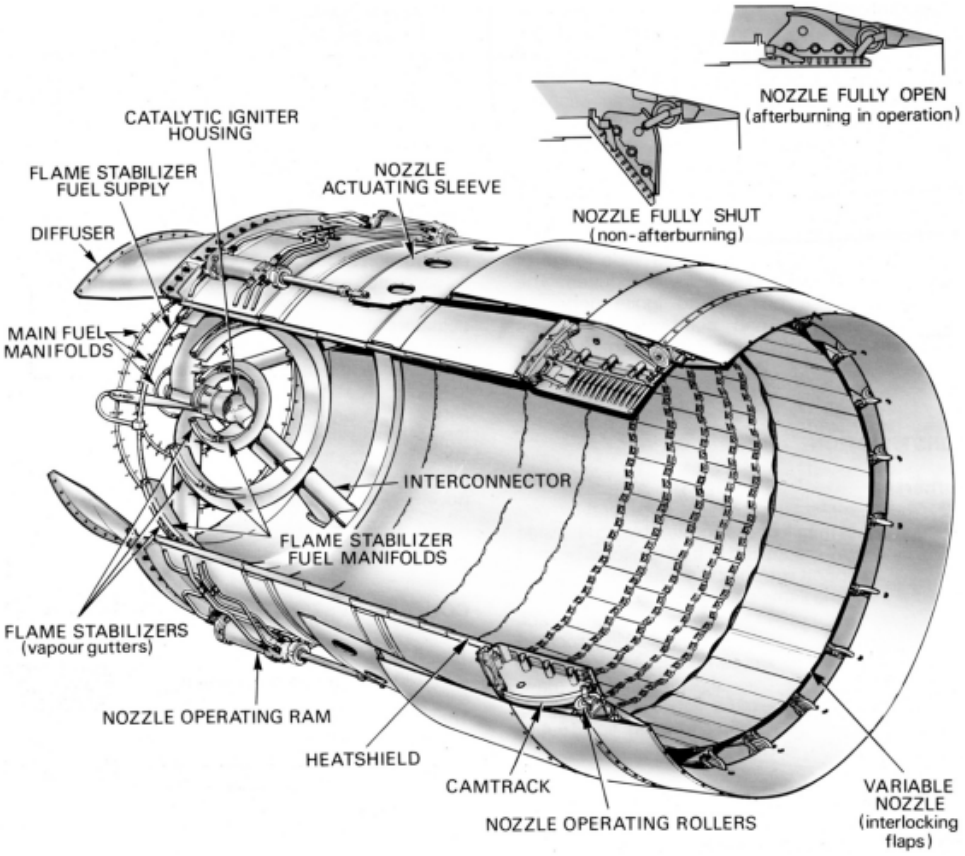


Figure 2.7 – Typical afterburner schematic [16]

Operation of the afterburner is highly flexible and is typically staged which allows partial or full augmentation depending on thrust demand. This comes at the cost of greatly increased specific fuel consumption and potential additional total pressure losses. When the afterburner is lit, the higher exhaust temperature and mass flow require the exhaust nozzle area to be increased to maintain acceptable turbine back pressure. In this project the afterburner is modeled at a one dimensional, system level as a region of essentially constant pressure heat addition in the mixed stream. The primary control variable is the afterburner fuel flow rate, which, together with the inlet conditions from the mixer and the nozzle geometry, determines the augmented exhaust state and the thrust and TSFC predicted by the surrogate model.

2.8 Variable Geometry Nozzle

The exhaust nozzle is the last engine component responsible for expanding the hot gases from the afterburner closer to ambient pressure to maximize thrust. In mixed flow afterburning turbofans this is accomplished with a variable geometry convergent divergent nozzle which can be seen in Fig. 2.8. The convergent section forms a throat where the flow can choke

reaching sonic conditions, downstream of which the divergent section allows further expansion to supersonic Mach numbers when required. By changing the position of the movable flaps, the engine can vary both the throat area and the exit area. This flexibility is essential because the mass flow rate, stagnation temperature, and pressure change significantly based on operating afterburning configuration.

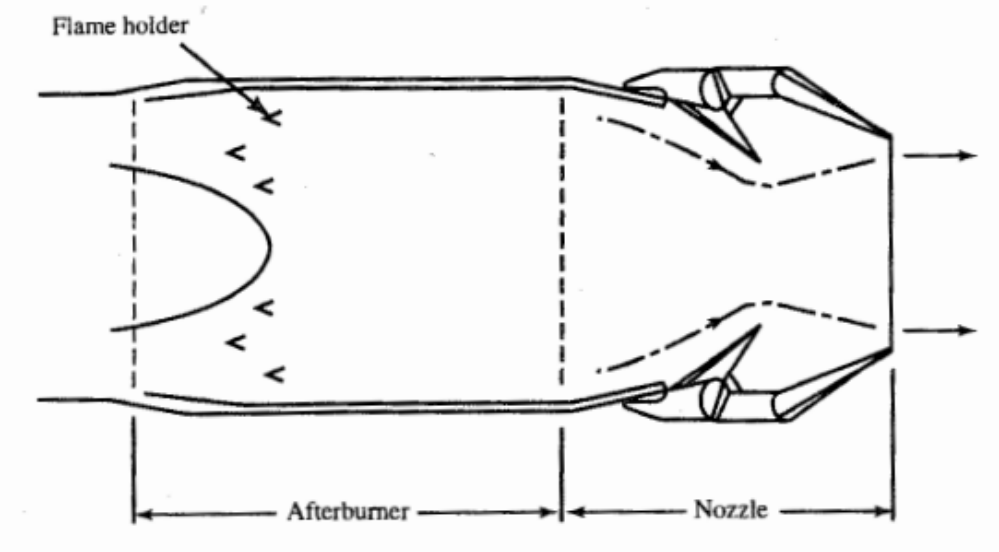


Figure 2.8 – Schematic of variable geometry nozzle [15]

When the afterburner is on, the increased temperature and mass flow can cause an excessive rise in turbine back pressure and exhaust static pressure if the nozzle area is fixed. Opening the nozzle increases the effective flow area which limits back pressure via flow expansion and allows the engine to utilize more of the additional thrust gained by using an afterburning configuration. Without afterburner use and partial throttle conditions, the nozzle is scheduled to a smaller area to maintain sufficient pressure ratio for efficient expansion and to meet thrust requirements without unnecessary fuel flow. In the parametric cycle model developed later in this project, the variable geometry nozzle is treated as a one dimensional converging diverging nozzle with controllable effective area; this nozzle area, together with afterburner fuel flow and upstream conditions, serves as a primary control variable for the surrogate model.

Chapter 3: Parametric Cycle Analysis

3.1 Fundamental Theory and Notation

The following sections establish the parametric cycle framework used to generate training data for the mixed-flow afterburning turbofan machine learning model. The analysis is posed as a forward mapping from an input vector of flight condition and engine design/setting parameters to an output vector of performance and internal operating states. For the ideal cycle, this mapping is written as $\vec{\boldsymbol{x}}_{\text{ideal}} \mapsto \vec{\boldsymbol{y}}_{\text{ideal}}$ and contains the selected freestream condition and idealized component ratios, and $\vec{\boldsymbol{y}}$ returns thrust, TSFC, and associated state variables under simplifying assumptions. The real-cycle mapping $\vec{\boldsymbol{x}}_{\text{real}} \mapsto \vec{\boldsymbol{y}}_{\text{real}}$ retains the same overall structure but incorporates practical losses and constraints through component efficiencies, total-pressure recoveries, and mixing/nozzle loss models, producing a higher-fidelity dataset that the final neural network is trained to reproduce across the operating envelope. Sections 3.2 and 3.3 present the full sets of relations for the ideal and real formulations, respectively, and define the variables that will later be sampled to construct the supervised training pairs. The governing equations utilized in the parametric model are adapted from Mattingly's gas turbine textbook [15], with derivation and algebraic proof being omitted.

3.1.1 Brayton Cycle Overview

In this subsection the fundamental thermodynamic relations for the ideal Brayton cycle are summarized, which form the basis for the engine cycles used in this project. The ideal Brayton cycle consists of four reversible processes: isentropic compression, constant-pressure heat addition, isentropic expansion, and constant-pressure heat rejection, as illustrated in Fig. 3.1. The working fluid is treated as a calorically perfect gas with constant c_p and γ, and all processes are assumed adiabatic and internally reversible. Under these assumptions, the cycle can be represented by defining non-dimensional total property ratios and then expressing work and heat transfer directly through changes in stagnation enthalpy.

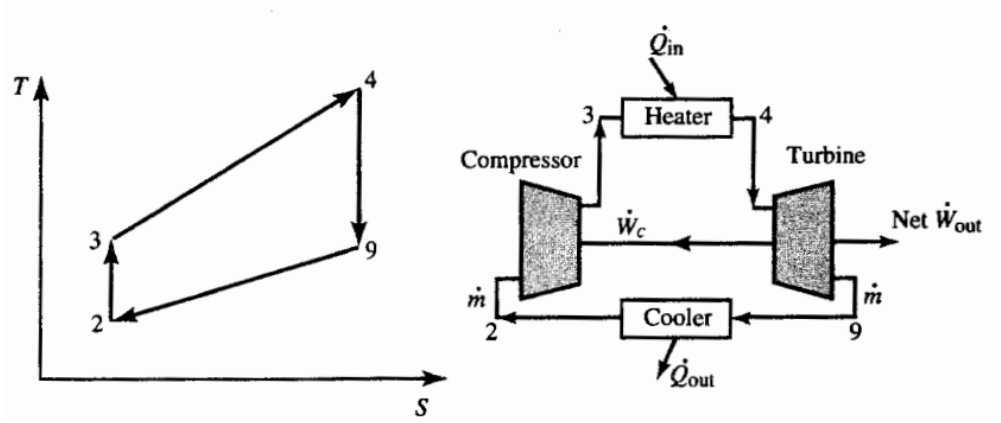


Figure 3.1 – T-s diagram of Brayton cycle [15]

For an ideal, adiabatic, and reversible compressor with total-pressure ratio π_c , the corresponding total-temperature ratio (τ_c) can be defined through isentropic relations as a function of the specific heat ratio. For an ideal turbine with total-pressure ratio π_t, the total-temperature ratio is defined similarly to the compressor through isentropic relations.

$$\tau_c = \frac{T_{t,3}}{T_{t,2}} = \pi_c^{\frac{\gamma-1}{\gamma}} \tag{3.1}$$

$$\tau_t = \frac{T_{t,5}}{T_{t,4}} = \pi_t^{\frac{\gamma-1}{\gamma}} \tag{3.2}$$

The compressor work per unit mass of fluid is defined by the change in specific enthalpy, which for a calorically perfect gas with constant c_p can be written as a function of the temperature difference across the compressor. The specific turbine work is defined similarly to the compressor work, but through the temperature drop across the turbine. These compression/expansion relations are intentionally written in a form that carries over to the mixed-flow turbofan cycle model, where total-temperature and total-pressure ratios remain the primary variables used to represent component interactions and to enforce shaft power balance in a one-dimensional steady system.

$$w_c = c_p \left(T_{t,3} - T_{t,2}\right) \tag{3.3}$$

$$w_t = c_p \left(T_{t,4} - T_{t,5}\right) \tag{3.4}$$

With stagnation states defined, the heat addition and rejection legs are represented as enthalpy changes across the constant-pressure processes. Under the same perfect-gas assumption, the specific heat added in the combustor and the specific heat rejected in the exhaust heat exchanger are written as seen in Eq. (3.1, 3.4). Although the full engine modeled later is an open cycle, these relations provide the same thermodynamic framework

required to define net work and efficiency and to interpret how changes in pressure ratio and peak temperature condition the performance of the engine.

$$q_{\text{in}} = c_p \left(T_{\text{t},4} - T_{\text{t},3}\right) \tag{3.5}$$
$$q_{\text{out}} = c_p \left(T_{\text{t},5} - T_{\text{t},2}\right) \tag{3.6}$$

Combining the compressor and turbine work expressions defines the net specific work output of the cycle based on the compressor and turbine states as shown below.

$$w_{\text{net}} = w_t - w_c = c_p \left[\left(T_{\text{t},4} - T_{\text{t},5}\right) - \left(T_{\text{t},3} - T_{\text{t},2}\right)\right] \tag{3.7}$$

The thermal efficiency of the ideal Brayton cycle is defined through the net specific work per heat input to the cycle. For the simplified Brayton cycle case with equal pressure ratios across the compressor and turbine and fixed minimum and maximum temperatures, the thermal efficiency of the cycle can also be rewritten in terms of compressor pressure ratio.

$$\eta_{\text{th}} = \frac{w_{\text{net}}}{q_{\text{in}}} = 1 - \frac{q_{\text{out}}}{q_{\text{in}}} = 1 - \frac{1}{\pi_{\text{c}}^{\frac{\gamma-1}{\gamma}}} = 1 - \frac{1}{\tau_c} \tag{3.8}$$

3.1.2 Station Numbering and Flow Assumptions

The mixed-flow afterburning turbofan considered in this project follows the standard station numbering convention used for two-stream turbofan engines with a mixed exhaust. A schematic of the station layout can be seen in Fig. 2.2. Both the ideal and real cycle models in Sections 3.2 and 3.3 treat the engine as a one-dimensional, steady-state system with uniform flow properties at each station. Component interactions are represented through total temperature and total pressure ratios and through mass-flow continuity; shaft power balance is enforced between the turbine, compressor, and fan. Flows from the bleed or cooling flows are not explicitly modeled, and all quantities are normalized with respect to static conditions in the free stream (T_0, p_0). The ideal mixed-flow model of Section 3.2 makes the following additional assumptions:

- All components (inlet, fan, compressor, turbine, mixer, afterburner, and nozzle) are adiabatic and internally reversible, with unit polytropic efficiencies and no total-pressure losses (e.g. $\pi_d = \pi_b = \pi_{AB} = \pi_M = \pi_n = 1$).
- The working fluid is treated as a calorically perfect gas with constant c_p and γ throughout the engine.

- The mixer is ideal: complete mixing of core and bypass streams produces a single uniform mixed stream at station 6A with no loss of total pressure and a mass-weighted total temperature.
- The exhaust nozzle expands the mixed stream exactly to ambient pressure so that $p_9 = p_0$, leading to a simplified thrust equation.
- Mechanical efficiency is unity and all shaft power extracted by the turbine is exactly balanced against compressor and fan power requirements.

Under these assumptions, each component can be represented by simple algebraic relations between non-dimensional total temperature and total pressure ratios, and the cycle is fully described by the ideal equation set in section 3.2. The real mixed-flow model of Section 3.3 retains the same station numbering and one-dimensional steady-flow assumptions while accounting for component efficiencies, pressure losses, and changing specific heats.

- Inlet pressure recovery, burner and afterburner pressure losses, mixer losses, and non-ideal nozzle expansion are represented by total-pressure ratios π_d, π_b, π_{AB}, π_M, and π_n less than 1.
- Real compressor, fan, and turbine behavior is modeled using polytropic efficiencies e_c, e_f, and e_t.
- Separate specific heats and ratios of specific heats are used for the core, turbine, and afterburner gases ($c_{p,c}$, γ_c , $c_{p,t}$, γ_t , $c_{p,AB}$, γ_{AB}), and mixed properties at station 6A are obtained from mass-weighted combinations.
- Mechanical efficiency $\eta_m < 1$ is included in the shaft power balance between the turbine and the compressor/fan system.
- The exhaust nozzle may not be ideally expanded with $p_9 \neq p_0$, so the pressure-thrust contribution is retained in the net thrust calculation.

These corrections preserve the same station framework as the ideal model but allow for a more realistic mapping from the input design parameters to thrust, TSFC, and engine characteristics. The detailed real-cycle relations used in the dataset generation are discussed in Section 3.3.

3.1.3 Dimensionless Quantities and Performance Metrics

In the parametric cycle analysis, each component is expressed in terms of non-dimensional total-temperature ratios τ and total-pressure ratios π, defined with respect to the station numbering introduced earlier. These non-dimensional ratios provide a compact way to parameterize the engine cycle and to relate design choices to thrust, specific fuel consumption,

and efficiency. The definitions of all necessary temperature and pressure ratios are shown in Tables 3.1 and 3.2 below.

Table 3.1 – Total-temperature ratios used in the mixed-flow afterburning turbofan cycle.

Symbol	Definition	Description
τ_r	$\frac{T_{\mathrm{t},0}}{T_0}$	Ram total-temperature ratio
τ_d	$\frac{T_{\mathrm{t},2}}{T_{\mathrm{t},0}}$	Diffuser total-temperature ratio
τ_f	$\frac{T_{\mathrm{t},13}}{T_{\mathrm{t},2}}$	Fan total-temperature ratio
τ_c	$\frac{T_{\mathrm{t},3}}{T_{\mathrm{t},2}}$	Compressor total-temperature ratio
τ_λ	$\frac{T_{\mathrm{t},4}}{T_0}$	Main combustor exit total-temperature ratio to ambient
τ_b	$\frac{T_{\mathrm{t},4}}{T_{\mathrm{t},3}}$	Main combustor total-temperature ratio
τ_t	$\frac{T_{\mathrm{t},5}}{T_{\mathrm{t},4}}$	Turbine total-temperature ratio
τ_{16}	$\frac{T_{\mathrm{t},16}}{T_{\mathrm{t},2}}$	Bypass stream total-temperature ratio (fan to mixer inlet)
τ_M	$\frac{T_{\mathrm{t},6A}}{T_{\mathrm{t},6}}$	Mixer total-temperature ratio (core reference)
τ_{AB}	$\frac{T_{\mathrm{t},7}}{T_{\mathrm{t},6A}}$	Afterburner total-temperature ratio
τ_n	$\frac{T_{\mathrm{t},9}}{T_{\mathrm{t},7}}$	Nozzle total-temperature ratio (unity for ideal nozzle)

Table 3.2 – Total-pressure ratios used in the mixed-flow afterburning turbofan cycle.

Symbol	Definition	Description
π_r	$\frac{p_{\mathrm{t},0}}{p_0}$	Ram total-pressure ratio
π_d	$\frac{p_{\mathrm{t},2}}{p_{\mathrm{t},0}}$	Diffuser total-pressure ratio
π_f	$\frac{p_{\mathrm{t},13}}{p_{\mathrm{t},2}}$	Fan total-pressure ratio
π_c	$\frac{p_{\mathrm{t},3}}{p_{\mathrm{t},2}}$	Compressor total-pressure ratio
π_b	$\frac{p_{\mathrm{t},4}}{p_{\mathrm{t},3}}$	Main combustor total-pressure ratio
π_t	$\frac{p_{\mathrm{t},5}}{p_{\mathrm{t},4}}$	Turbine total-pressure ratio
π_{16}	$\frac{p_{\mathrm{t},16}}{p_{\mathrm{t},2}}$	Bypass stream total-pressure ratio (fan to mixer inlet)
π_M	$\frac{p_{\mathrm{t},6A}}{p_{\mathrm{t},6}}$	Mixer total-pressure ratio (core reference)
π_{AB}	$\frac{p_{\mathrm{t},7}}{p_{\mathrm{t},6A}}$	Afterburner total-pressure ratio
π_n	$\frac{p_{\mathrm{t},9}}{p_{\mathrm{t},7}}$	Nozzle total-pressure ratio

For any ideal, internally reversible compression or expansion of a perfect gas, the total-temperature and total-pressure ratios are related by

$$\frac{T_{\mathrm{t,out}}}{T_{\mathrm{t,in}}} = \left(\frac{p_{\mathrm{t,out}}}{p_{\mathrm{t,in}}}\right)^{\frac{\gamma-1}{\gamma}} \tag{3.9}$$

In the ideal-cycle sections that follow, the individual ratios τ_f, τ_c, τ_t, etc. are obtained directly from the corresponding pressure ratios using Eq. (3.9). In the real-cycle model, these relations are modified by component efficiencies due to specific heats changing with temperature. The primary performance quantities used in this project are the specific thrust, thrust specific fuel consumption (TSFC), and the thermal, propulsive, and overall efficiencies. The specific thrust is defined as a function of the inlet mass flow rate $\dot{m}_0$, the nozzle exit mass flow rate $\dot{m}_9$, as well as freestream and nozzle exit velocities V_0 and V_9 , and the nozzle

exit area A_9.

$$\frac{F}{\dot{m}_0} = \frac{1}{\dot{m}_0}\left[\dot{m}_9 V_9 - \dot{m}_0 V_0 + (p_9 - p_0)A_9\right] \tag{3.10}$$

The thrust specific fuel consumption (TSFC) is defined through the total fuel mass flow rate $\dot{m}_f$ per specific thrust, where $\dot{m}_f$ is the sum of both the main combustor as well as the afterburner fuel mass flow rates.

$$S = \frac{\dot{m}_f}{F/\dot{m}_0} \tag{3.11}$$

3.2 Ideal Mixed-Flow Afterburning Turbofan Model

In this project, the ideal mixed-flow afterburning turbofan model is adapted directly from the relations found in Mattingly's text [15]. The inputs, outputs, and governing non-dimensional equations are also listed as they drive the architecture of the neural network model and training dataset generation.

3.2.1 Inputs and Outputs

The cycle is parameterized by the following input set:

$$\vec{\boldsymbol{x}}_{\text{ideal}} = \left[M_0,\ T_0,\ \gamma,\ c_p,\ h_{PR},\ T_{t4},\ T_{t7},\ \pi_c,\ \pi_f\right] \tag{3.12}$$

where M_0 and T_0 are the freestream Mach number and static temperature, γ and c_p are the ratio of specific heats and constant-pressure specific heat of the working fluid, h_{PR} is the fuel heating power, T_{t4} and T_{t7} are the main combustor and afterburner exit total temperatures, π_c is the overall compressor pressure ratio, and either the fan pressure ratio π_f or the bypass ratio α is specified. For consistency between the real and ideal cycle parametrization, π_f is taken as a design input parameter to determine the bypass ratio α.

The ideal-cycle outputs are taken as:

$$\vec{\boldsymbol{y}}_{\text{ideal}} = \left[\frac{F}{\dot{m}_0},\ f,\ f_{AB},\ f_0,\ S,\ \eta_T,\ \eta_P,\ \eta_O,\ \alpha\right] \tag{3.13}$$

where $F/\dot{m}_0$ is the specific thrust, f and f_{AB} are the main burner and afterburner fuel-air ratios (per unit core air), f_0 is the overall fuel-air ratio per unit freestream air, S is the thrust specific fuel consumption, and η_T, η_P, and η_O are the thermal, propulsive, and overall efficiencies.

3.2.2 Summary of Ideal Equations

The ram total-temperature ratio, τ_r ,converts the freestream static flow properties into the stagnation state associated with compressible deceleration at the inlet for the specified flight Mach number. In the dataset-generation code, τ_r is computed directly from the sampled (M_0, γ) pair in $\vec{\boldsymbol{x}}_{\text{ideal}}$, and it is immediately used to establish the normalized inlet total condition that appears throughout the remaining non-dimensional relations. This step anchors the cycle to the flight condition so that subsequent compressor, burner, and nozzle calculations are referenced consistently to the same freestream baseline.

$$\tau_r = 1 + \frac{\gamma - 1}{2} M_0^2 \tag{3.14}$$

Because the ideal cycle is parameterized using the fan pressure ratio π_f as a design input, the corresponding fan total-temperature ratio τ_f is obtained from the ideal isentropic relation rather than being solved implicitly. In code, this is evaluated algebraically from the sampled (π_f, γ) values in $\vec{\boldsymbol{x}}_{\text{ideal}}$, and the same approach is applied in parallel to compute τ_c from the sampled compressor pressure ratio π_c. These temperature ratios define how much stagnation temperature rise is associated with each compression process and therefore directly influence the bypass ratio solution, the turbine work requirement, and the final thrust and TSFC outputs.

$$\tau_f = \pi_f^{(\gamma-1)/\gamma} \tag{3.15}$$

$$\alpha = \frac{\tau_\lambda(\tau_c - \tau_r)}{\tau_r \tau_c(\tau_f - 1)} - \frac{\tau_c - 1}{\tau_f - 1} \tag{3.16}$$

In this ideal formulation, the bypass ratio α is not treated as an independent dataset input, but is instead implied by the selected cycle design and temperature levels through Eq. (3.16). Within the dataset generator, α is computed after τ_r, τ_f, τ_λ are known, which ensures the cycle remains fully determined by the chosen input set $\vec{\boldsymbol{x}}_{\text{ideal}}$. If Eq. (3.16) returns a negative value, the corresponding sample represents an unrealistic combination of inputs for the assumed ideal matching logic, so α is reset to zero to enforce a physically possible bypass ratio and to prevent invalid propagation into the mixer and thrust calculations.

Once α has been determined, the remaining core-state quantities are evaluated in a fixed order so that each subsequent relation uses only previously computed ratios and the sampled input temperatures. The turbine temperature ratio τ_t enforces the ideal shaft power balance implied by the selected compression work and bypass work split, which makes it the primary coupling between the upstream compression choices and the downstream mixed-stream state. The main combustor fuel-air ratio f is then obtained from an ideal energy balance that raises the core stream to the specified turbine inlet temperature level, and finally the mixer

temperature ratio τ_M combines the bypass and core contributions into the single mixed stream that feeds the afterburner and nozzle.

$$\tau_t = 1 - \frac{\tau_r}{\tau_\lambda}\left[\tau_c - 1 + \alpha(\tau_f - 1)\right] \tag{3.17}$$

$$f = \frac{c_p T_0}{h_{PR}}\left(\tau_\lambda - \tau_r \tau_c\right) \tag{3.18}$$

$$\tau_M = \frac{1}{1+\alpha}\left(1 + \alpha\frac{\tau_r \tau_f}{\tau_\lambda \tau_t}\right) \tag{3.19}$$

Eq. (3.19) models mixing as lossless in total pressure, with the mixed total temperature determined purely by mass-weighted addition of the core and bypass stagnation enthalpy. In the dataset-generation workflow, τ_M is treated as the definitive interface variable between the two-stream turbo machinery block and the single-stream afterburner and nozzle, so it is stored as an intermediate even when it is not included directly in $\vec{\boldsymbol{y}}_{\text{ideal}}$. This is important for later stages of the project because τ_M controls the required afterburner heat addition and strongly conditions the final exhaust state, which is why it must be computed for every sampled input vector.

Afterburning is represented by prescribing an afterburner exit total temperature T_{t7} as part of the ideal input set, which allows augmented and non-augmented operation to be generated using the same governing structure. In the dataset-generation code, $\tau_{\lambda AB}$ is formed directly from the sampled (T_{t7}, T_0), and the incremental fuel-air ratio f_{AB} is computed as the additional heat addition required to raise the mixed stream from its upstream level $\tau_\lambda \tau_t \tau_M$ to the specified afterburner temperature level. This produces the afterburner exhaust temperature ratio and therefore the thrust increase and TSFC penalty associated with afterburner operation, which is a key output behavior the neural network model must learn.

$$\tau_{\lambda AB} = \frac{T_{t7}}{T_0} \tag{3.20}$$

$$f_{AB} = \frac{c_p T_0}{h_{PR}}\left(\tau_{\lambda AB} - \tau_\lambda \tau_t \tau_M\right) \tag{3.21}$$

$$\frac{T_9}{T_0} = \frac{\tau_{\lambda AB}}{\tau_r \tau_f} \tag{3.22}$$

For a non-afterburning ideal cycle, the model enforces zero afterburner fuel addition by setting f_{AB}=0. In implementation, this branch is used whenever the dataset generation intends to represent a non-afterburning configuration, so the mixed-stream state is passed directly to the nozzle with no added heat. This ensures the training set contains consistent examples of both operating modes while keeping the governing equations algebraic and avoiding partial-afterburning configuration in the ideal model.

$$f_{AB} = 0 \tag{3.23}$$

$$\frac{T_9}{T_0} = \frac{\tau_\lambda \tau_t \tau_M}{\tau_r \tau_f} \tag{3.24}$$

With the exhaust temperature ratio T_9/T_0 determined, the nozzle relations convert the stagnation-based cycle solution into the kinematic quantities needed for thrust. In the ideal model the nozzle is assumed to expand the flow to ambient pressure, which removes the pressure-thrust term and allows the exit Mach number and velocity ratio to be expressed in closed form. In the dataset generator, $(M_9, V_9/a_0)$ are computed as the final flow-state variables before thrust and TSFC, and they are the primary drivers of both propulsive efficiency and specific thrust in the output vector.

$$M_9 = \sqrt{\frac{2}{\gamma - 1}\left(\tau_r\, \tau_f - 1\right)} \tag{3.25}$$

$$\frac{V_9}{a_0} = \sqrt{\frac{T_9}{T_0}}\, M_9 \tag{3.26}$$

The overall fuel-air ratio f_0 accounts for the fact that main combustor fuel is diluted by the total engine airflow when bypass flow is present, while afterburner fuel is added relative to the mixed stream. In code, f_0 is computed immediately after f, f_{AB}, and α because it is the fuel-normalization used in both TSFC and thermal efficiency calculations. The specific thrust is then evaluated from the difference between exhaust and freestream velocity ratios under the ideal expansion assumption, and TSFC follows directly as fuel flow per unit thrust, which completes the mapping from $\vec{\boldsymbol{x}}_{\text{ideal}}$ to the primary performance outputs in $\vec{\boldsymbol{y}}_{\text{ideal}}$.

$$f_0 = \frac{f}{1 + \alpha} + f_{AB} \tag{3.27}$$

$$\frac{F}{\dot{m}_0} = \frac{a_0}{g_c}\left(\frac{V_9}{a_0} - M_0\right) \tag{3.28}$$

$$S = \frac{f_0}{F/\dot{m}_0} \tag{3.29}$$

The efficiency metrics are computed last because they depend on the final exhaust velocity ratio and on the total fuel addition expressed through f_0. In the dataset generation workflow, η_T measures how effectively the added chemical energy is converted into jet kinetic energy, while η_P measures how effectively that jet kinetic energy is converted into useful propulsive power at the flight speed. Their product η_O provides a compact single number evaluation of cycle quality that is useful for both trend analysis and for constraining or regularizing later machine learning tasks, since it penalizes solutions that achieve thrust primarily through excessively high jet velocity or excessive fuel flow.

$$\eta_T = \frac{\gamma - 1}{2} \frac{c_p T_0}{f_0 h_{PR}} \left[\left(\frac{V_9}{a_0} \right)^2 - M_0^2 \right] \tag{3.30}$$

$$\eta_P = \frac{2M_0}{V_9/a_0 + M_0} \tag{3.31}$$

$$\eta_O = \eta_P \, \eta_T \tag{3.32}$$

3.3 Real Mixed-Flow Afterburning Turbofan Model

The real mixed-flow afterburning turbofan model used in this project follows the summary relations given by Mattingly [15] for the mixed-flow afterburning turbofan engine. In contrast to the ideal model of Section 3.2, the real model includes inlet pressure recovery, non-unity component efficiencies, polytropic compressor and turbine behavior, mixer losses, and a non-ideal nozzle. This section summarizes the input and output parameter sets and highlights the key non-dimensional relations.

3.3.1 Inputs and Outputs

The real-cycle model for the mixed-flow afterburning turbofan engine is parameterized by the following inputs:

$$\begin{aligned} \vec{\boldsymbol{x}}_{\text{real}} = \big[& M_0,\ T_0,\ \gamma_c,\ c_{p,c},\ \gamma_t,\ c_{p,t},\ \gamma_{AB},\ c_{p,AB},\ h_{PR},\ \pi_{d,\max},\ \pi_b,\ \pi_{AB},\ \pi_{M,\max},\ \pi_n, \\ & e_c,\ e_f,\ e_t,\ \eta_b,\ \eta_{AB},\ \eta_m,\ P_0/P_9,\ T_{t4},\ T_{t7},\ \pi_c,\ \pi_f \big] \end{aligned} \tag{3.33}$$

The subscripts c, t, and AB refer to the core (compressor), turbine, and afterburner gas properties, respectively; e_c, e_f, and e_t are polytropic efficiencies, η_b and η_{AB} are burner and afterburner efficiencies, η_m is the mechanical efficiency, and $\pi_{d,\max}$, $\pi_{M,\max}$, and π_n represent limiting or design values of inlet, mixer, and nozzle pressure ratios.

The primary outputs used in this project are:

$$\vec{\boldsymbol{y}}_{\text{real}} = \left[\frac{F}{\dot{m}_0},\ f,\ f_{AB},\ f_0,\ S,\ \alpha,\ \eta_T,\ \eta_P,\ \eta_O,\ \eta_c,\ \eta_t\right] \tag{3.34}$$

where α is the bypass ratio, f and f_{AB} are the main burner and afterburner fuel-air ratios, f_0 is the overall fuel-air ratio per unit freestream air, S is the thrust specific fuel consumption, and η_T, η_P, η_O are the thermal, propulsive, and overall cycle efficiencies. Additional internal quantities (stagnation temperatures, pressures, Mach numbers, and area ratios) are computed from the same equation set and are available for training and diagnostics.

3.3.2 Summary of Real-Cycle Relations

The real-cycle equations are grouped by component, with all relevant relations to dataset generation presented and discussed in this section. Because the real-cycle model allows different (c_p, γ) values in the compressor, turbine, and afterburner streams, the corresponding gas constants are defined first using Eq. (3.35). In the dataset generator, these constants ensure that all subsequent velocity, Mach number, and mass-flow-parameter relations remain consistent with the appropriate working-fluid model at each station. This separation is essential for the mixer and nozzle blocks, where property changes directly affect the predicted pressure recovery and exhaust velocity.

$$R_c = \frac{\gamma_c - 1}{\gamma_c}\, c_{p,c} \qquad R_t = \frac{\gamma_t - 1}{\gamma_t}\, c_{p,t} \qquad R_{AB} = \frac{\gamma_{AB} - 1}{\gamma_{AB}}\, c_{p,AB} \tag{3.35}$$

The freestream sound speed a_0 and velocity V_0 define the normalization used throughout the parametric cycle model, including the final use of V_9/a_0 in thrust and efficiency metrics. In dataset generation, a_0 is evaluated immediately after sampling $(M_0, T_0, \gamma_c, c_{pc})$ so that all downstream kinematic quantities are dimensionless and directly comparable across operating points. This step also anchors the cycle to the flight condition before component losses are applied.

$$a_0 = \sqrt{\gamma_c R_c T_0} \qquad V_0 = a_0 M_0 \tag{3.36}$$

The inlet block converts the freestream state into stagnation conditions and applies a practical total-pressure recovery model. In particular, τ_r and π_r establish the ideal stagnation rise associated with compressible deceleration, while η_r introduces a Mach-dependent penalty for supersonic inlet operation. Finally, Eq. (3.40) enforces a cap on diffuser performance so that the training dataset reflects realistic inlet losses rather than idealized recovery.

$$\tau_r = 1 + \frac{\gamma_c - 1}{2} M_0^2 \tag{3.37}$$

$$\pi_r = \tau_r^{\gamma_c/(\gamma_c - 1)} \tag{3.38}$$

$$\eta_r = \begin{cases} 1 & M_0 \leq 1 \\ 1 - 0.075(M_0 - 1)^{1.35} & M_0 > 1 \end{cases} \tag{3.39}$$

$$\pi_d = \pi_{d,\max}\, \eta_r \tag{3.40}$$

The temperature ratios τ_λ and τ_{AB} represent the prescribed heat-addition levels in the main combustor and afterburner, normalized to freestream static temperature. In the dataset generation workflow, T_{t4} and T_{t7} are treated as sampled or scheduled inputs in $\vec{\boldsymbol{x}}_{\text{real}}$, and these ratios provide the non-dimensional form needed by the rest of the parametric relations. This choice keeps the heat-addition modeling consistent across different flight conditions while still allowing strong coupling to nozzle expansion and TSFC.

$$\tau_\lambda = \frac{c_{p,t}\, T_{t4}}{c_{p,c}\, T_0} \tag{3.41}$$

$$\tau_{\lambda AB} = \frac{c_{p,AB}\, T_{t7}}{c_{p,c}\, T_0} \tag{3.42}$$

The compressor and fan blocks convert the selected pressure ratios (π_c, π_f) into temperature ratios (τ_c, τ_f) using polytropic efficiencies (e_c, e_f). In implementation, this is where the real-cycle model diverges most clearly from the ideal cycle, because the temperature rise is no longer the ideal isentropic value for a given π. The resulting τ_c and τ_f directly determine the spool work requirement and therefore strongly condition the turbine balance, bypass ratio solution, and ultimately the mixed-stream state.

$$\tau_c = \pi_c^{(\gamma_c - 1)/(\gamma_c e_c)} \tag{3.43}$$

$$\eta_c = \frac{\pi_c^{(\gamma_c - 1)/\gamma_c} - 1}{\tau_c - 1} \tag{3.44}$$

$$\tau_f = \pi_f^{(\gamma_c - 1)/(\gamma_c e_f)} \tag{3.45}$$

$$\eta_f = \frac{\pi_f^{(\gamma_c - 1)/\gamma_c} - 1}{\tau_f - 1} \tag{3.46}$$

The main burner fuel-air ratio f is obtained from an energy balance that accounts for the burner efficiency η_b and the fuel heating value h_{PR}. In the dataset generator, this equation converts the desired combustor exit temperature level (through τ_λ) into the required fuel addition relative to the core stream. This step is also a primary feasibility gate, since non-physical combinations of (T_{t4}, π_c, M_0) can produce invalid f values that must be rejected or clipped before the turbine block is evaluated. Furthermore, Eq. (3.48) provides the bypass ratio α implied by the selected fan and core design choices under real component behavior. In this project, π_f is used as a design input, and α is computed similarly to the ideal case so the cycle remains fully determined by $\vec{\boldsymbol{x}}_{\text{real}}$ rather than treating bypass ratio as an independent degree of freedom. Because α propagates into the turbine work split and the mixer area matching, it should be computed and validated before entering the turbine and mixer equations.

$$f = \frac{\tau_\lambda - \tau_r \tau_c}{\eta_b h_{PR}/(c_{p,c} T_0) - \tau_\lambda} \tag{3.47}$$

$$\alpha = \frac{\eta_m (1+f)\,(\tau_\lambda/\tau_r)\left[1 - (\pi_f/(\pi_c \pi_b))^{(\gamma_t - 1)e_t/\gamma_t}\right] - (\tau_c - 1)}{\tau_f - 1} \tag{3.48}$$

The turbine block enforces the shaft power balance between the compressor/fan system and the turbine, including mechanical efficiency η_m and the effect of fuel addition through $(1+f)$. In the dataset generator, (τ_t) is a key coupling variable that transfers upstream compression choices into a downstream total-pressure reduction via π_t. Reporting η_t alongside the performance outputs is valuable for ML training diagnostics, since it allows the model to learn not only thrust and TSFC trends but also whether the implied turbine operating point remains physically reasonable.

$$\tau_t = 1 - \frac{1}{\eta_m (1+f)} \frac{\tau_r}{\tau_\lambda} \left[\tau_c - 1 + \alpha\,(\tau_f - 1)\right] \tag{3.49}$$

$$\pi_t = \tau_t^{\gamma_t/[(\gamma_t - 1)e_t]} \tag{3.50}$$

$$\eta_t = \frac{1 - \tau_t}{1 - \tau_t^{1/e_t}} \tag{3.51}$$

The mixer relations combine the core and bypass streams into a single mixed exhaust stream while accounting for differing gas properties and mixing losses. The total-pressure ratio P_{t16}/P_{t6} establishes the relative stagnation levels of the two streams at the mixer inlet and is used to determine the bypass inlet Mach number M_{16}. This provides the required inlet conditions for the subsequent constant-area mixing relations and ensures that the model captures the sensitivity of mixing to upstream component pressure losses. The effective

bypass ratio α' and the mixed-stream properties $(c_{p,6A}, R_{6A}, \gamma_{6A})$ are computed as mass-weighted combinations consistent with the streams entering the mixer. In code, these values define the "equivalent gas" used downstream of the mixer for Mach number, nozzle expansion, and thrust calculations when the afterburner is off. The mixer temperature ratio τ_M is then computed to quantify the mixed stagnation temperature level that becomes the baseline for any afterburner heat addition.

$$\frac{P_{t16}}{P_{t6}} = \frac{\pi_f}{\pi_c \, \pi_b \, \pi_t} \tag{3.52}$$

$$M_{16} = \sqrt{\frac{2}{\gamma_c - 1} \left\{ \left[\frac{P_{t16}}{P_{t6}} \left(1 + \frac{\gamma_t - 1}{2} M_6^2 \right)^{\gamma_t/(\gamma_t - 1)} \right]^{(\gamma_c - 1)/\gamma_c} - 1 \right\}} \tag{3.53}$$

$$\alpha' = \frac{\alpha}{1 + f} \tag{3.54}$$

$$c_{p6A} = \frac{c_{pt} + \alpha' c_{pc}}{1 + \alpha'} \tag{3.55}$$

$$R_{6A} = \frac{R_t + \alpha' R_c}{1 + \alpha'} \tag{3.56}$$

$$\gamma_{6A} = \frac{c_{p6A}}{c_{p6A} - R_{6A}} \tag{3.57}$$

$$\frac{T_{t16}}{T_{t6}} = \frac{T_0 \, \tau_r \, \tau_f}{T_{t4} \, \tau_t} \tag{3.58}$$

$$\tau_M = \frac{c_{pt}}{c_{p6A}} \, \frac{1 + \alpha' \, (c_{pc}/c_{pt}) \, (T_{t16}/T_{t6})}{1 + \alpha'} \tag{3.59}$$

The next step solves the constant-area mixing constraint in a form that remains valid when gas properties differ between the bypass and core flow-streams. The functions $\phi(M, \gamma)$ and the composite parameter Φ compactly encode the momentum and continuity coupling between the core inlet Mach number M_6, the bypass inlet Mach number M_{16}, and the mixed-stream properties at station $6A$. Using Eq. (3.63) returns the mixer exit Mach number, and Eq. (3.64) yields the required area split A_{16}/A_6; both quantities are subsequently used to

compute the mixer pressure ratio and to diagnose whether the sampled point is physically realizable.

$$\phi(M_6, \gamma_6) = \frac{M_6^2 \left\{1 + \left[(\gamma_t - 1)/2\right] M_6^2\right\}}{\left(1 + \gamma_t M_6^2\right)^2} \tag{3.60}$$

$$\phi(M_{16}, \gamma_{16}) = \frac{M_{16}^2 \left\{1 + \left[(\gamma_c - 1)/2\right] M_{16}^2\right\}}{\left(1 + \gamma_c M_{16}^2\right)^2} \tag{3.61}$$

$$\Phi = \left[\frac{1 + \alpha'}{\dfrac{1}{\sqrt{\phi(M_6, \gamma_6)}} + \alpha' \sqrt{\dfrac{R_c\, \gamma_t}{R_t\, \gamma_c} \dfrac{T_{t16}/T_{t6}}{\phi(M_{16}, \gamma_{16})}}} \right]^2 \frac{R_{6A}\, \gamma_t}{R_t\, \gamma_{6A}}\, \tau_M \tag{3.62}$$

The mass-flow parameter MFP is introduced to express choked-flow and area-Mach behavior in a compact non-dimensional form that is consistent across different (γ, R) values. With MFP, the ideal mixer pressure ratio $\pi_{M,ideal}$ is computed based on the inlet conditions and the solved exit Mach number M_{6A}. The final mixer pressure ratio applies a design-limit multiplier to prevent unrealistically high pressure recovery in the training dataset, which is important for keeping the learned surrogate within feasible mixer-loss bounds.

$$M_{6A} = \sqrt{\frac{2\Phi}{1 - 2\gamma_{6A}\Phi + \sqrt{1 - 2(\gamma_{6A} + 1)\Phi}}} \tag{3.63}$$

$$\frac{A_{16}}{A_6} = \alpha' \sqrt{\frac{T_{t16}}{T_{t6}}} \left(\frac{M_6}{M_{16}}\right) \left[\frac{\gamma_t R_c}{\gamma_c R_t} \frac{1 + \left[(\gamma_t - 1)/2\right] M_6^2}{1 + \left[(\gamma_c - 1)/2\right] M_{16}^2}\right]^{1/2} \tag{3.64}$$

$$\mathrm{MFP}(M, \gamma, R) = \frac{M \sqrt{\gamma\, g_c / R}}{\left\{1 + \left[(\gamma - 1)/2\right] M^2\right\}^{(\gamma+1)/[2(\gamma-1)]}} \tag{3.65}$$

$$\pi_{M,\mathrm{ideal}} = \frac{(1 + \alpha')\sqrt{\tau_M}}{1 + A_{16}/A_6} \frac{\mathrm{MFP}(M_6, \gamma_t, R_t)}{\mathrm{MFP}(M_{6A}, \gamma_{6A}, R_{6A})} \tag{3.66}$$

$$\pi_M = \pi_{M,\max}\,\pi_{M,\text{ideal}} \tag{3.67}$$

Once all component pressure ratios are defined, the nozzle-inlet total-to-static pressure ratio is assembled as a product of the individual losses and compression/expansion ratios. This step is what couples inlet recovery, mixing losses, afterburner pressure loss, and nozzle loss into a single quantity that determines the achievable exit Mach number. This is also the point where under-expanded and over-expanded behavior is permitted through the specified P_9/P_0 term in $\vec{\boldsymbol{x}}_{\text{real}}$, enabling the model to retain pressure thrust in the final thrust relation.

$$\frac{P_{t9}}{P_9} = \frac{P_0}{P_9}\,\pi_r\,\pi_d\,\pi_c\,\pi_b\,\pi_t\,\pi_M\,\pi_{AB}\,\pi_n \tag{3.68}$$

The nozzle exit temperature depends on whether additional heat is added downstream of the mixer. For non-afterburning operation, the mixed-stream gas properties are retained at station 9 and the afterburner fuel flow is set to zero, which provides a clean baseline for comparing augmented and non-augmented performance. For afterburning operation, the gas properties are switched to the afterburner set $(c_{p,AB}, \gamma_{AB}, R_{AB})$ and the additional fuel-air ratio f_{AB} is computed from the required temperature rise and η_{AB}, ensuring consistent heat-addition accounting in the exhaust stream.

$$c_{p9} = c_{p6A}, \qquad R_9 = R_{6A}, \qquad \gamma_9 = \gamma_{6A}, \qquad f_{AB} = 0 \tag{3.69}$$

$$\frac{T_9}{T_0} = \frac{T_{t4}\,\tau_t\,\tau_M/T_0}{(P_{t9}/P_9)^{(\gamma_9-1)/\gamma_9}} \tag{3.70}$$

The nozzle exit temperature depends on whether additional heat is added downstream of the mixer. For non-afterburning operation, the mixed-stream gas properties are retained at station 9 and the afterburner fuel flow is set to zero, which provides a clean baseline for comparing augmented and non-augmented performance. For afterburning operation, the gas properties are switched to the afterburner set $(c_{p,AB}, \gamma_{AB}, R_{AB})$ and the additional fuel-air ratio f_{AB} is computed from the required temperature rise and η_{AB}, ensuring consistent heat-addition accounting in the exhaust stream.

$$c_{p9} = c_{pAB}, \qquad R_9 = R_{AB}, \qquad \gamma_9 = \gamma_{AB} \tag{3.71}$$

$$f_{AB} = \left(1 + \frac{f}{1+\alpha}\right) \frac{\tau_{\lambda AB} - \left(\frac{c_{p6A}}{c_{pt}}\right) \tau_\lambda \, \tau_t \, \tau_M}{\eta_{AB} \, \dfrac{h_{PR}}{c_{pc} T_0} - \tau_{\lambda AB}} \tag{3.72}$$

$$\frac{T_9}{T_0} = \frac{T_{t7}/T_0}{\left(P_{t9}/P_9\right)^{(\gamma_9 - 1)/\gamma_9}} \tag{3.73}$$

With Pt_9/P_9 and T_9/T_0 established, Eq. (3.74) returns the nozzle exit Mach number, and Eq. (3.75) converts it into the freestream normalized exhaust velocity V_9/a_0. The overall fuel-air ratio f_0 is then formed so that TSFC and efficiency metrics are expressed per unit freestream air, which is the normalization used for network training outputs. These variables are the immediate precursors to thrust and TSFC and are therefore the most sensitive indicators of whether a sampled cycle point is physically consistent.

$$M_9 = \sqrt{\frac{2}{\gamma_9 - 1} \left[\left(\frac{P_{t9}}{P_9} \right)^{(\gamma_9 - 1)/\gamma_9} - 1 \right]} \tag{3.74}$$

$$\frac{V_9}{a_0} = M_9 \sqrt{\frac{\gamma_9 R_9 T_9}{\gamma_c R_c T_0}} \tag{3.75}$$

$$f_0 = \frac{f}{1+\alpha} + f_{AB} \tag{3.76}$$

The remaining relations translate the solved exhaust state into the performance quantities used as supervised learning targets. Because the real cycle permits $P_9 \neq P_0$, the thrust relation retains the pressure-thrust contribution, which is necessary for representing off-design nozzle operation. The resulting $F/\dot{m}_0, S$ and (η_T, η_P, η_O) complete the forward mapping $\boldsymbol{x}_{\text{real}} \mapsto \boldsymbol{y}_{\text{real}}$ that defines the physics-guided dataset used to train the surrogate model.

$$\frac{F}{\dot{m}_0} = \frac{a_0}{g_c} \left[(1+f_0) \frac{V_9}{a_0} - M_0 + (1+f_0) \, \frac{R_9 \, (T_9/T_0)}{R_c \, (V_9/a_0)} \, \frac{(1 - P_0/P_9)}{\gamma_c} \right] \tag{3.77}$$

$$S = \frac{f_0}{F/\dot{m}_0} \tag{3.78}$$

$$\eta_P = \frac{2g_c V_0 (F/\dot{m}_0)}{a_0^2 \left[(1+f_0)(V_9/a_0)^2 - M_0^2\right]} \tag{3.79}$$

$$\eta_T = \frac{a_0^2 \left[(1+f_0)(V_9/a_0)^2 - M_0^2\right]}{2g_c f_0 h_{PR}} \tag{3.80}$$

$$\eta_O = \eta_P \, \eta_T \tag{3.81}$$

These relations fully define the real cycle mapping $\vec{\boldsymbol{x}}_{\text{real}} \mapsto \vec{\boldsymbol{y}}_{\text{real}}$ used for dataset generation in the following machine learning architecture chapter. In the following chapter, all inputs for the ideal and real sets are determined and total solution-spaces are generated via the aforementioned relations. This chapter has established a complete parametric cycle framework for both ideal and real mixed-flow afterburning turbofan operation, expressed as an explicit forward mapping from a sampled operating/configuration vector to thrust, TSFC, and internal station states. The real cycle relations in particular preserve the coupled mixer-afterburner-nozzle behavior that enables adaptive cycles, while maintaining physically interpretable variables suitable for feasibility checks. In the next chapter, these equations will be used as a dataset generator over the selected operating envelope, producing a training data-set for a neural network model that reproduces engine performance with higher computational efficiency. This learned model then serves as the computational backbone for exploring afterburner fuel and nozzle-area scheduling strategies under real-time thrust and efficiency optimization objectives.

Chapter 4: Machine Learning Architecture/Integration

4.1 Machine Learning Fundamentals

The parametric cycle analysis developed in Chapter 3 provides a deterministic forward model for the mixed-flow afterburning turbofan. For a specified set of flight conditions, thermodynamic properties, component parameters, and engine settings, the parametric cycle model returns thrust, fuel consumption, efficiency metrics, and internal station properties. The propulsion-side formulation follows the parametric cycle analysis basis established in Chapter 3 and anchored to standard gas-turbine cycle analysis methods [15]. The machine-learning formulation introduced in this chapter combines that deterministic physics model with standard engineering surrogate-modeling and supervised neural-network regression methods [17,18].

In this project, the neural network is used as an engineering surrogate model: a computationally inexpensive approximation to a deterministic reference model that can be evaluated rapidly after the reference behavior has been sampled over the relevant design space [17]. The role of the surrogate is therefore not to replace the governing thermodynamic formulation, but to approximate the input-output response of the parametric cycle analysis within a defined feasible operating envelope.

The reference cycle relationship can be represented as a nonlinear mapping from an input vector to an output vector,

$$\vec{\boldsymbol{y}} = \mathcal{F}_{\mathrm{PCA}}\left(\vec{\boldsymbol{x}}\right) \tag{4.1}$$

where $\vec{\boldsymbol{x}}$ contains the selected operating and design parameters, $\vec{\boldsymbol{y}}$ contains the corresponding performance and state quantities, and $\mathcal{F}_{\mathrm{PCA}}$ denotes the parametric cycle analysis equations.

Although the parametric cycle model is computationally inexpensive relative to higher-fidelity simulations such as computational fluid dynamics, repeated evaluation over large multidimensional design spaces can still become costly. This becomes especially important when the model is used for large parametric sweeps, optimization studies, sensitivity analysis, or potential control-oriented applications. The purpose of the machine learning model is therefore to approximate the deterministic cycle model with a fast surrogate after the

governing relationships have been sampled over a feasible operating envelope.

The present approach is best described as physics-guided machine learning because physical knowledge is used to define the input variables, generate the reference labels, filter infeasible operating points, select diagnostic outputs, and interpret model validity [19]. This differs from strict physics-informed neural-network formulations, where governing differential-equation residuals are commonly embedded directly into the training loss [20]. In this work, the physics enters primarily through the parametric cycle model, the feasibility filters, and the structure of the training data.

The neural-network surrogate is written as

$$\hat{\mathbf{y}} = \mathcal{F}_{\mathrm{NN}}\left(\mathbf{x}; \boldsymbol{\theta}\right) \tag{4.2}$$

where $\hat{\vec{y}}$ is the predicted output vector, $\mathcal{F}_{\mathrm{NN}}$ is the neural-network approximation of the parametric cycle model, and $\boldsymbol{\theta}$ represents the trainable weights and biases. The training process adjusts $\boldsymbol{\theta}$ so that the neural-network output agrees with the reference output generated by the parametric cycle equations.

4.1.1 Surrogate Modeling Objective

The objective of the surrogate model is to learn a fast approximation to the deterministic input-output relationship defined by the parametric cycle analysis. In the terminology of engineering surrogate modeling, the neural network acts as a lower-cost response approximation to a reference model that is more expensive or less convenient to evaluate repeatedly over a large design space [17]. In this project, the reference model is the mixed-flow afterburning turbofan cycle formulation developed in Chapter 3.

The training dataset is composed of supervised input-output pairs,

$$\mathcal{D} = \{(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{y}}_i)\}_{i=1}^{N} \tag{4.3}$$

where N is the number of accepted samples, $\vec{\boldsymbol{x}}_i$ is the i-th sampled input vector, and $\vec{\boldsymbol{y}}_i$ is the corresponding output vector computed by the parametric cycle model. The input vector contains the selected flight-condition, thermodynamic, component, and engine-setting parameters, while the output vector contains the resulting performance metrics and selected internal operating states.

The learned surrogate is intended to approximate the same forward relationship as the parametric cycle model,

$$\mathcal{F}_{\mathrm{NN}}\left(\vec{\boldsymbol{x}}; \boldsymbol{\theta}\right) \approx \mathcal{F}_{\mathrm{PCA}}\left(\vec{\boldsymbol{x}}\right) \tag{4.4}$$

within the sampled feasible operating envelope. Here, $\mathcal{F}_{\mathrm{NN}}$ denotes the neural-network approximation, $\mathcal{F}_{\mathrm{PCA}}$ denotes the parametric cycle analysis mapping, and $\boldsymbol{\theta}$ represents the trainable network parameters.

This distinction is important because the neural network is not treated as an independent propulsion model. Its validity is tied to the coverage of the generated dataset, the physical assumptions of the parametric cycle model, and the feasibility filters used during data construction. Predictions made far outside the sampled operating envelope are therefore not guaranteed to remain physically meaningful, even if the network produces finite numerical outputs. The model is trained as a multi-output regression surrogate rather than as a single-output predictor. The output vector may include specific thrust, thrust specific fuel consumption, efficiency metrics, fuel-air ratios, nozzle quantities, and selected station properties. Retaining internal operating states as outputs provides additional information for validation and helps determine whether the surrogate has learned a physically consistent representation of the cycle behavior rather than only matching the final performance quantities.

4.1.2 Multi-Layer Perceptron Formulation

The neural-network models used in this project are based on multi-layer perceptrons (MLPs), a standard feedforward architecture for supervised function approximation and regression problems [18]. An MLP maps an input vector through a sequence of affine transformations and nonlinear activation functions before producing an output vector. For the present surrogate model, the input vector is the engine operating/design vector and the output vector is the corresponding set of cycle-performance and internal-state quantities.

Let the network contain L layers. The input to the network is written as

$$\vec{\boldsymbol{a}}^{(0)} = \vec{\boldsymbol{x}} \tag{4.5}$$

where $\vec{\boldsymbol{a}}^{(0)}$ is the activation vector of the input layer. For a hidden layer l, the pre-activation vector is computed as

$$\vec{\boldsymbol{z}}^{(l)} = \boldsymbol{W}^{(l)}\vec{\boldsymbol{a}}^{(l-1)} + \vec{\boldsymbol{b}}^{(l)} \tag{4.6}$$

where $\boldsymbol{W}^{(l)}$ is the weight matrix, $\boldsymbol{b}^{(l)}$ is the bias vector, and $\boldsymbol{a}^{(l-1)}$ is the activation vector from the previous layer. The hidden-layer activation is then obtained from

$$\boldsymbol{a}^{(l)} = g_{\text{act}}\left(\boldsymbol{z}^{(l)}\right) \tag{4.7}$$

where $g_{\text{act}}(\cdot)$ is a nonlinear activation function applied elementwise.

The nonlinear activation functions are essential because the mixed-flow afterburning turbofan cycle is not a linear mapping from inputs to outputs. The parametric cycle equations contain nonlinear compressible-flow relations, pressure-ratio and temperature-ratio coupling, fuel-air ratio calculations, turbine work-balance constraints, mixer relations, nozzle choking logic, and pressure-thrust effects. The MLP does not encode each of these relations explicitly. Instead, the nonlinear hidden layers provide sufficient functional flexibility to approximate their combined input-output behavior over the sampled feasible domain.

The final layer of the network is written as

$$\hat{\vec{y}} = \boldsymbol{W}^{(L)}\vec{\boldsymbol{a}}^{(L-1)} + \vec{\boldsymbol{b}}^{(L)} \tag{4.8}$$

where $\hat{\vec{y}}$ is the predicted output vector. Because the target quantities are continuous dimensional or nondimensional cycle quantities, the output layer is left linear rather than being passed through a classification or probability-normalization function.

The complete set of trainable neural-network parameters is

$$\boldsymbol{\theta} = \left\{\boldsymbol{W}^{(1)}, \vec{\boldsymbol{b}}^{(1)}, \boldsymbol{W}^{(2)}, \vec{\boldsymbol{b}}^{(2)}, \ldots, \boldsymbol{W}^{(L)}, \vec{\boldsymbol{b}}^{(L)}\right\} \tag{4.9}$$

and the training process adjusts these parameters so that the neural-network prediction $\hat{\vec{y}}$ matches the reference output $\vec{\boldsymbol{y}}$ generated by the parametric cycle model.

This architecture is appropriate for the initial surrogate formulation because the problem is a continuous, fixed-vector regression problem rather than an image, sequence, graph, or spatial-field prediction problem. More specialized architectures are therefore not required for the baseline model. The predictive quality of the MLP is instead governed primarily by the physical coverage of the training dataset, the scaling of the inputs and outputs, the selected network capacity, and the feasibility of the sampled cycle-analysis points.

4.1.3 Supervised Regression Formulation

The surrogate-training problem is formulated as supervised regression because each sampled input vector is paired with a known reference output vector [18]. In this project, the reference outputs are not obtained from experimental engine measurements. Instead, they are generated deterministically by evaluating the parametric cycle analysis model developed in Chapter 3.

For the i-th sampled operating point, the supervised training pair is written as

$$(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{y}}_i) = (\vec{\boldsymbol{x}}_i, \mathcal{F}_{\mathrm{PCA}}(\vec{\boldsymbol{x}}_i)) \tag{4.10}$$

where $\vec{\boldsymbol{x}}_i$ is the sampled input vector and $\vec{\boldsymbol{y}}_i$ is the corresponding label produced by the parametric cycle model. The neural network is therefore trained to reproduce the sampled input-output response of the cycle-analysis equations rather than to infer propulsion behavior from unstructured data.

Because multiple engine-performance and internal-state quantities are predicted simultaneously, the problem is treated as multi-output regression. The target vector for each sample is written as

$$\vec{\boldsymbol{y}}_i \in \mathbb{R}^{n_y} \tag{4.11}$$

where n_y is the number of predicted output quantities. Depending on the model case, these

outputs may include specific thrust, thrust specific fuel consumption, efficiency metrics, fuel-air ratios, nozzle states, and selected station properties.

For a given input vector, the neural-network prediction is

$$\hat{\vec{y}}_i = \mathcal{F}_{\mathrm{NN}}\left(\vec{\boldsymbol{x}}_i; \boldsymbol{\theta}\right) \tag{4.12}$$

where $\boldsymbol{\theta}$ contains the trainable weights and biases. The corresponding prediction residual is

$$\vec{\boldsymbol{r}}_i = \hat{\vec{y}}_i - \vec{\boldsymbol{y}}_i \tag{4.13}$$

where $\vec{\boldsymbol{r}}_i$ contains the output-wise difference between the neural-network prediction and the parametric-cycle reference output.

Training adjusts the network parameters to minimize a selected loss function,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}\right) \tag{4.14}$$

where $\boldsymbol{\theta}^*$ denotes the trained parameter set and $\mathcal{L}$ quantifies the disagreement between the predicted outputs and the reference outputs over the training dataset.

This formulation makes the role of the neural network precise. The network does not replace the cycle equations as the source of physical authority. Instead, it approximates the deterministic mapping generated by those equations over the sampled feasible domain. The quality of the supervised regression model is therefore limited by the physical assumptions of the parametric cycle model, the coverage of the sampled input space, and the feasibility criteria used to accept or reject candidate operating points.

4.1.4 Loss Functions and Output Weighting

For supervised regression models, the loss function defines the scalar training objective used to measure disagreement between predicted outputs and reference labels [18]. In this project, the loss function measures the difference between the neural-network prediction and the corresponding output generated by the parametric cycle model. Using the prediction-residual concept introduced in Eq. (4.13), a generic mean-squared-error loss can be written as

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \widehat{\boldsymbol{y}}_i - \vec{\boldsymbol{y}}_i \right\|_2^2 \tag{4.15}$$

where N is the number of training samples, $\widehat{\boldsymbol{y}}_i$ is the neural-network prediction, and $\vec{\boldsymbol{y}}_i$ is the parametric-cycle reference output.

For a multi-output regression problem, the same loss can be written in component form

as

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n_y} \left(\hat{y}_{i,j} - y_{i,j} \right)^2 \tag{4.16}$$

where n_y is the number of output quantities and j indexes the individual output components.

The unweighted form in Eq. (4.16) is useful for defining the basic regression objective, but it is not always appropriate when the output vector contains quantities with different units and magnitudes. In the mixed-flow afterburning turbofan surrogate, the output vector may contain specific thrust, thrust specific fuel consumption, fuel-air ratios, efficiencies, temperature ratios, pressure ratios, and station properties. If raw dimensional errors are used directly, outputs with large numerical magnitudes can dominate the gradient updates even when smaller-magnitude outputs are physically important.

For this reason, the training loss is evaluated in normalized output space. The normalization procedure is described in Section 4.1.5. Using normalized outputs, a weighted mean-squared-error loss is written as

$$\mathcal{L}_W = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n_y} w_j \left(\widehat{\tilde{y}}_{i,j} - \tilde{y}_{i,j} \right)^2 \tag{4.17}$$

where $\tilde{y}_{i,j}$ is the normalized reference value, $\widehat{\tilde{y}}_{i,j}$ is the normalized neural-network prediction, and w_j is the loss weight assigned to the j-th output quantity.

The output weights allow selected quantities to be emphasized during training. For example, performance quantities such as specific thrust and thrust specific fuel consumption may be assigned higher importance when the primary objective is accurate cycle-performance prediction. Internal diagnostic quantities may also be retained and weighted because they help evaluate whether the network is approximating a physically consistent cycle state rather than only matching a small number of final performance outputs. This weighting strategy is part of the physics-guided training workflow because physically meaningful outputs and diagnostic states are deliberately represented in the training objective [19].

If all output weights are set equal after normalization, the loss treats each standardized output component with equal numerical importance. If nonuniform weights are used, the training objective becomes biased toward the selected quantities. The choice of weights must therefore reflect the intended use of the surrogate model. A model intended for thrust prediction may place greater emphasis on thrust and fuel-consumption metrics, while a model intended for cycle-state reconstruction may require stronger weighting of internal station properties and feasibility-related outputs.

A low training loss indicates that the network has fit the sampled training data, and a low validation loss suggests that the model generalizes reasonably within the sampled distribution. However, loss values alone do not prove that the surrogate has learned physically correct trends across the operating envelope. For this reason, the loss function is treated as only one part of the validation process. The trained models are also evaluated using parity

plots, residual plots, physical consistency checks, and parametric sweep comparisons against the deterministic parametric cycle model.

4.1.5 Input and Output Normalization

Input and output normalization is required because the variables used in the surrogate model have different units, magnitudes, and numerical ranges. The input vector may contain quantities such as Mach number, pressure ratios, temperature ratios, efficiencies, and component parameters, while the output vector may contain thrust-related quantities, fuel-air ratios, efficiencies, station temperatures, station pressures, and nozzle states. If these quantities are passed to the optimizer in raw form, gradient-based training can become poorly conditioned and large-magnitude variables can dominate the loss function [18].

For this reason, each input variable is standardized before it is passed into the neural network. For the j-th input variable of the i-th sample, the normalized input is written as

$$\tilde{x}_{i,j} = \frac{x_{i,j} - \mu_{x,j}}{\sigma_{x,j} + \epsilon} \tag{4.18}$$

where $x_{i,j}$ is the raw input value, $\mu_{x,j}$ is the mean of the j-th input variable, $\sigma_{x,j}$ is the corresponding standard deviation, and ϵ is a small positive constant included to prevent division by zero. The normalized value $\tilde{x}_{i,j}$ is the quantity used by the neural network during training.

The normalization statistics are computed using only the training partition of the dataset. The same training-derived means and standard deviations are then applied to the validation, test, sweep, and inference inputs. This prevents information from the validation or test sets from entering the training process through the preprocessing step and keeps the reported validation performance meaningful.

The output variables are standardized in the same manner. For the j-th output quantity of the i-th sample,

$$\tilde{y}_{i,j} = \frac{y_{i,j} - \mu_{y,j}}{\sigma_{y,j} + \epsilon} \tag{4.19}$$

where $y_{i,j}$ is the raw parametric-cycle reference output, $\mu_{y,j}$ is the training-set mean of the j-th output variable, and $\sigma_{y,j}$ is the corresponding training-set standard deviation. The weighted loss introduced in Section 4.1.4 is evaluated using these normalized output quantities. This ensures that output weights w_j represent deliberate modeling priorities rather than merely compensating for differences in physical units or numerical scale.

During training, the neural network predicts normalized outputs,

$$\widehat{\tilde{y}}_{i,j} = \mathcal{F}_{\mathrm{NN},j}\left(\tilde{\vec{x}}_i; \boldsymbol{\theta}\right) \tag{4.20}$$

where $\mathcal{F}_{\mathrm{NN},j}$ denotes the j-th component of the neural-network output. After prediction, the

normalized output is transformed back to its physical dimensional or non-dimensional form using

$$\hat{y}_{i,j} = \widehat{\tilde{y}}_{i,j} \left(\sigma_{y,j} + \epsilon\right) + \mu_{y,j} \tag{4.21}$$

where $\hat{y}_{i,j}$ is the recovered prediction in the original output coordinates.

Normalization changes only the numerical coordinates used by the optimizer. It does not change the physical reference solution generated by the parametric cycle model, the feasibility filters used to construct the dataset, or the dimensional and nondimensional quantities reported in the results. All physical validation, parity plots, residual plots, parametric sweep comparisons, and cycle-consistency checks are performed after the neural-network outputs have been transformed back into their original physical coordinates.

The trained surrogate must therefore be stored with the input and output normalization parameters. During inference, a new input vector is first normalized using the stored input statistics, passed through the neural network, and then converted back to physical output coordinates using the stored output statistics. This makes the trained model reproducible and ensures that future predictions are evaluated in the same coordinate system used during training.

4.2 Physics-Guided Learning Approach

The surrogate framework developed in this project is described as physics-guided because the learning problem is structured around the parametric cycle analysis formulation rather than around unstructured numerical data alone. The neural network is trained using supervised regression, but the inputs, outputs, labels, feasibility limits, and validation procedures are all defined from propulsion-cycle physics. The reference labels are generated from the mixed-flow afterburning turbofan cycle model developed in Chapter 3, which is based on standard gas-turbine parametric cycle analysis methods [15].

This approach follows the broader idea of integrating scientific or engineering knowledge into machine-learning workflows [19]. In the present work, physical knowledge enters through the selection of thermodynamic input variables, the generation of reference outputs from the cycle solver, the rejection of infeasible operating points, the inclusion of diagnostic internal-state outputs, and the use of parametric sweep validation against the deterministic reference model. The neural network is therefore not treated as an independent source of propulsion physics. Instead, it is trained as a fast surrogate approximation to a physically structured reference model [17].

The method is also distinct from a strict physics-informed neural network formulation. In many physics-informed neural-network approaches, governing differential-equation residuals are embedded directly into the loss function and minimized during training [20]. The present project does not solve the cycle equations by embedding their residuals directly into the loss. Instead, the parametric cycle analysis is evaluated first to generate physically consistent

supervised labels, and the neural network is trained to approximate that resulting input-output map within the sampled feasible envelope.

4.2.1 Parametric Cycle Analysis as the Reference Model

The parametric cycle analysis developed in Chapter 3 serves as the reference model for the neural-network surrogate. For each sampled input vector, the cycle solver evaluates the mixed-flow afterburning turbofan equations and returns the corresponding performance quantities and internal station properties. The propulsion-side reference model is therefore anchored to standard gas-turbine parametric cycle analysis methods [15].

Using the parametric cycle model as the label generator provides a controlled way to construct large supervised datasets over specified flight conditions, component parameters, thermodynamic properties, and afterburning states. This is consistent with the use of surrogate models as fast approximations to more detailed reference models for repeated evaluation over a design space [17].

The neural network does not exceed the physical fidelity of the reference model used to generate its labels. Its predictions inherit the assumptions of the cycle formulation, including the station-based one-dimensional flow model, component efficiency and pressure-loss assumptions, mixer and nozzle modeling assumptions, and the feasibility limits imposed during data generation. Consequently, the surrogate is expected to be reliable only within the sampled feasible operating envelope represented by the training data. This reference-model structure provides a continuous chain from the Chapter 3 cycle equations, to the generated datasets, to the trained neural-network models, and finally to the validation studies presented in later chapters.

4.2.2 Physics-Guided Versus Physics-Informed Learning

The present project distinguishes between physics-guided learning and strict physics-informed neural-network formulations. In physics-informed neural networks, the governing-equation residuals are commonly embedded directly into the training objective and minimized along with any available data mismatch terms [20]. In the broader physics-guided approach, scientific or engineering knowledge is used to structure the data, variables, constraints, model outputs, and validation process, while the primary learning problem may still remain a supervised regression problem [19].

Conceptually, a residual-based physics-informed loss may be written as

$$\mathcal{L}_{\text{informed}} = \mathcal{L}_{\text{data}} + \omega_{\text{phys}}\mathcal{L}_{\text{residual}} \tag{4.22}$$

where $\mathcal{L}_{\text{data}}$ measures agreement with data, $\mathcal{L}_{\text{residual}}$ penalizes violations of the governing relations, and ω_{phys} controls the relative importance of the residual penalty. In this type of formulation, the network prediction is evaluated inside the governing equations during training.

The approach used in this project is different. The parametric cycle equations are solved first, and the accepted cycle-analysis outputs become supervised reference labels. The training objective is therefore based on matching the neural-network output to the parametric-cycle reference output,

$$\mathcal{L}_{\text{guided}} = \mathcal{L}_{\text{data}}\left(\widehat{\boldsymbol{y}}, \vec{\boldsymbol{y}}_{\text{PCA}}\right) \tag{4.23}$$

where $\widehat{\boldsymbol{y}}$ denotes the neural-network prediction and $\vec{\boldsymbol{y}}_{\text{PCA}}$ denotes the output generated by the parametric cycle model.

A fully residual-based implementation would require the neural-network outputs to be inserted back into the coupled cycle relations during training. Although the cycle model is primarily algebraic rather than field-based, this would still require residual definitions for thermodynamic compatibility, component work balance, fuel-air ratio relations, mixer behavior, nozzle choking logic, pressure-thrust behavior, and wet/dry operating modes. These relations include conditional branches and feasibility checks, making direct residual enforcement more complicated than the supervised surrogate objective required for this project.

For this reason, physical guidance is introduced through the construction of the training data and the validation workflow rather than through a complete residual-based loss. The input variables are physically meaningful, the labels are generated by the parametric cycle solver, infeasible operating points are filtered before training, diagnostic internal states are retained as outputs, and the trained model is evaluated against the deterministic cycle model.

This distinction also defines the limitations of the surrogate. Because the cycle equations are not enforced directly at every prediction point, a low training or validation loss does not guarantee physical validity outside the sampled feasible envelope. Statistical regression accuracy must therefore be supported by physically interpretable validation, including consistency checks and parametric sweep comparisons against the reference cycle model.

4.2.3 Physical Constraints in the Training Workflow

Physical constraints are incorporated into the learning workflow before training by rejecting candidate cycle evaluations that do not correspond to meaningful engine operating states. This is a central part of the physics-guided approach: the neural network is not trained on arbitrary numerical outputs, but on parametric-cycle solutions that satisfy the feasibility requirements implied by the propulsion model [15,19].

Conceptually, the accepted training dataset can be written as

$$\mathcal{D}_{\text{train}} = \left\{\left(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{y}}_i\right) : \vec{\boldsymbol{x}}_i \in \Omega,\ \mathcal{F}_{\text{PCA}}\left(\vec{\boldsymbol{x}}_i\right) \in \Omega_{\text{phys}}\right\} \tag{4.24}$$

where Ω is the sampled input envelope and Ω_{phys} is the set of physically acceptable cycle solutions. The detailed feasibility requirements are discussed separately in Section 4.4.

The filtered solution set excludes failed or nonphysical cycle evaluations before they are used as supervised labels. These checks include thermodynamic validity, fuel-flow validity,

component compatibility, mixer and nozzle feasibility, and reasonable performance bounds. Applying these filters before training prevents the model from learning numerical artifacts produced by failed cycle solutions.

This filtering process improves the physical quality of the training data, but it does not guarantee that every neural-network prediction will be physically valid. Because the constraints are imposed primarily through the accepted data distribution, prediction errors may still occur near feasibility boundaries, sparsely sampled regions, or outside the training envelope. For this reason, physical validation is still required after training through consistency checks and parametric comparisons with the deterministic cycle model.

4.2.4 Internal State Prediction and Diagnostic Outputs

The surrogate model is trained to predict both final performance quantities and selected internal cycle states. This is important because thrust and thrust specific fuel consumption are downstream results of several coupled station calculations. A model may produce a reasonable final performance prediction while still producing internal quantities that are thermodynamically or component-wise inconsistent. Including internal states as outputs makes those inconsistencies observable and supports the physics-guided structure of the surrogate model [15,19].

The output vector can be written as

$$\vec{\boldsymbol{y}} = \begin{bmatrix} \vec{\boldsymbol{y}}_{\text{perf}} \\ \vec{\boldsymbol{y}}_{\text{state}} \end{bmatrix} \tag{4.25}$$

where $\vec{\boldsymbol{y}}_{\text{perf}}$ contains final performance quantities and $\vec{\boldsymbol{y}}_{\text{state}}$ contains selected internal state or diagnostic quantities.

Representative performance outputs include

$$\vec{\boldsymbol{y}}_{\text{perf}} = \begin{bmatrix} F/\dot{m}_0 & S & \eta_T & \eta_P & \eta_O \end{bmatrix}^T \tag{4.26}$$

where $F/\dot{m}_0$ is the specific thrust, S is the thrust specific fuel consumption, and η_T, η_P, and η_O are the thermal, propulsive, and overall efficiency metrics.

Representative diagnostic outputs for the real-cycle surrogate may include

$$\vec{\boldsymbol{y}}_{\text{state}} = \begin{bmatrix} f & f_{AB} & T_{t6A} & P_{t6A} & M_9 & A_9/A_t \end{bmatrix}^T \tag{4.27}$$

where f is the main-burner fuel-air ratio, f_{AB} is the afterburner fuel-air ratio, T_{t6A} and P_{t6A} are the mixed-stream total temperature and total pressure, M_9 is the nozzle exit Mach number, and A_9/A_t is the nozzle area ratio. The exact diagnostic outputs may vary by model case, but the purpose is the same: to expose the internal cycle behavior that leads to the final performance prediction.

Internal-state prediction provides three advantages. First, it supports error localization

by showing whether prediction errors originate primarily in fuel addition, mixing, nozzle expansion, or efficiency-related quantities. Second, it enables physical consistency checks. For example, an accurate thrust prediction is not sufficient if the associated fuel-air ratio, mixed total pressure, nozzle Mach number, or efficiency values are unreasonable. Third, it improves the value of parametric sweep validation because trends in internal quantities can be compared against the deterministic cycle model, not only the final thrust and fuel-consumption outputs.

The inclusion of additional outputs also increases the difficulty of the regression problem. Some internal quantities are more sensitive to small changes in the input vector than the final performance metrics, and the network capacity must be distributed across a larger output space. This motivates the normalized and weighted multi-output loss formulation introduced in Section 4.1.4 and the normalization procedure described in Section 4.1.5.

Predicting internal states does not guarantee physical correctness, but it makes physical inconsistency easier to detect. This reduces the degree to which the surrogate behaves as a purely black-box thrust predictor and provides a stronger basis for validation, interpretation, and future inverse or control-oriented extensions of the model.

4.3 Sampling and Dataset Construction

The neural-network surrogate requires a dataset that represents the operating envelope of the mixed-flow afterburning turbofan with sufficient coverage for supervised regression. In this project, the dataset is generated synthetically: candidate input vectors are sampled over prescribed bounds, mapped into physical engine and thermodynamic variables, evaluated using the parametric cycle model from Chapter 3, and retained only if the resulting cycle solution satisfies the feasibility requirements of the model [15]. The detailed feasibility criteria are treated separately in Section 4.4; the present section focuses on how candidate points are generated, organized, and divided for training and validation.

Because the input space contains coupled flight-condition, component, thermodynamic, mixer, nozzle, and afterburner variables, direct Cartesian gridding is not practical. Instead, the data-generation workflow uses space-filling sampling methods to distribute candidate points through the multidimensional input domain. Latin hypercube sampling is used for the ideal-cycle development stages because it provides stratified coverage of each input dimension without requiring a full factorial grid [21]. For the higher-dimensional real-cycle model, Sobol sampling is used to generate low-discrepancy candidate points with improved multidimensional coverage relative to purely random sampling [22].

The dataset construction process is also separated by operating mode. Dry and afterburning cases are generated as separate branches so that the resulting datasets remain consistent with the wet/dry expert structure used later in the neural-network models. For the larger real-cycle datasets, candidate generation and evaluation are performed in shards. Each shard is sampled, mapped to physical variables, evaluated by the cycle solver, filtered for feasible solutions, and written to storage before the accepted shards are merged into the final

dry, wet, and combined datasets. This organization keeps the sampling method, feasibility filtering, and dataset storage internally consistent.

4.3.1 High-Dimensional Sampling Requirements

The mixed-flow afterburning turbofan surrogate is trained over a multidimensional input space containing flight-condition variables, component pressure ratios, efficiencies, thermodynamic properties, mixer parameters, nozzle quantities, and afterburning-state variables. As the model progresses from the ideal-cycle cases to the real-cycle case, the number of sampled inputs increases substantially. This makes direct Cartesian gridding impractical.

If k sample levels are assigned to each of d input dimensions, the total number of grid points is

$$N_{\text{grid}} = k^d \tag{4.28}$$

where N_{grid} is the number of required cycle evaluations. For example, using ten levels per variable would require 10^{10} evaluations for a ten-dimensional model, 10^{19} evaluations for a nineteen-dimensional model, and 10^{23} evaluations for a twenty-three-dimensional model. This growth makes full factorial sampling infeasible for the real-cycle dataset.

Pure random sampling avoids the exponential structure of a full grid, but it can produce clustering in some regions and gaps in others, especially as the input dimension increases. For surrogate training, this is undesirable because the network requires broad candidate coverage over the operating envelope before feasibility filtering is applied. Space-filling sampling methods provide a more controlled alternative by distributing candidate points through the normalized input hypercube before mapping them to physical engine variables [21,22].

It is important to distinguish between candidate-space coverage and accepted-data coverage. Latin hypercube and Sobol sampling improve the distribution of candidate points in the prescribed input domain, but the final accepted dataset is shaped by the feasibility filters applied after cycle evaluation. Therefore, sparse regions in the accepted dataset may reflect physical infeasibility or strict operating constraints rather than a failure of the sampling method.

This project uses a staged sampling strategy. The lower-dimensional ideal-cycle models use Latin hypercube sampling to support rapid development and validation, while the higher-dimensional real-cycle model uses Sobol sampling and dataset sharding to improve candidate coverage and manage large-scale dataset generation. This progression follows the development path from Model A and Model B to the final real-cycle Model C surrogate.

4.3.2 Latin Hypercube Sampling

Latin hypercube sampling (LHS) is used to generate candidate points for the lower-dimensional ideal-cycle datasets. LHS divides each input dimension into equal-probability

intervals and samples each interval once, improving one-dimensional marginal coverage compared with purely random sampling [21]. This makes it useful for early surrogate development because the input envelope can be explored without requiring a full factorial grid.

For a dataset containing N candidate points and d sampled input variables, each normalized coordinate is constructed from a randomly permuted interval index and a random offset within that interval. The normalized sample coordinate for the i-th sample and j-th input variable can be written as

$$\xi_{i,j} = \frac{\ell_{i,j} + \nu_{i,j}}{N} \tag{4.29}$$

where $\ell_{i,j}$ is the permuted interval index for the j-th variable and $\nu_{i,j} \in [0, 1)$ is a random number that places the point within that interval. This construction ensures that each input variable is sampled once in each of its N normalized intervals.

The normalized coordinate is then mapped to the physical input bounds using

$$x_{i,j} = x_{j,\min} + \xi_{i,j} \left(x_{j,\max} - x_{j,\min} \right) \tag{4.30}$$

where $x_{j,\min}$ and $x_{j,\max}$ are the lower and upper bounds of the j-th physical input variable. After mapping, the candidate points are evaluated by the parametric cycle solver and passed to the feasibility-filtering stage described in Section 4.4.

In this project, LHS is used for the ideal-cycle development stages because the corresponding input spaces are smaller and the method provides efficient stratified coverage of each variable. This supports the construction of Model A and Model B datasets before moving to the higher-dimensional real-cycle surrogate. The main limitation of LHS is that it guarantees stratification only in the one-dimensional marginal distributions. It does not, by itself, guarantee uniform coverage of all multidimensional interactions among variables. This limitation motivates the use of Sobol sampling for the larger real-cycle dataset described in the next subsection.

4.3.3 Sobol Sampling and Dataset Sharding

Sobol sampling is used for the higher-dimensional real-cycle dataset because Sobol sequences are low-discrepancy point sets designed to distribute candidate points through a unit hypercube more evenly than purely random sampling [22]. This makes the method useful for Model C, where the real-cycle dry and afterburning generators require substantially larger input spaces than the ideal-cycle models. In the implemented dataset generator, the dry real-cycle branch uses a nineteen-dimensional candidate space, while the afterburning branch uses a twenty-three-dimensional candidate space.

For the i-th Sobol candidate point, the normalized input vector is written as

$$\vec{\boldsymbol{\xi}}_i = \begin{bmatrix} \xi_{i,1} & \xi_{i,2} & \cdots & \xi_{i,d} \end{bmatrix}^T \tag{4.31}$$

where d is the number of sampled input dimensions and each component lies within the normalized interval $0 \leq \xi_{i,j} \leq 1$. The implemented generator uses a scrambled Sobol sequence to reduce regularity artifacts while retaining the low-discrepancy character of the candidate set.

The normalized coordinates are then mapped to physical cycle-analysis inputs. For variables mapped linearly over their prescribed bounds, the transformation is

$$x_{i,j} = x_{j,\min} + \xi_{i,j} \left(x_{j,\max} - x_{j,\min} \right) \tag{4.32}$$

where $x_{j,\min}$ and $x_{j,\max}$ are the lower and upper limits of the j-th input variable. Most variables are mapped using this form. Selected pressure-ratio variables may instead be sampled in logarithmic coordinates and then transformed back to physical values so that the sampled range is distributed more appropriately over multiplicative pressure-ratio scales.

For the real-cycle dataset, Sobol sampling is combined with dataset sharding. Sharding is an implementation strategy rather than a separate sampling distribution. Instead of generating and storing all candidate points at once, the dataset is built from smaller subsets. The s-th shard of normalized candidate points can be written as

$$\mathcal{S}_{\xi}^{(s)} = \left\{ \vec{\boldsymbol{\xi}}_1^{(s)}, \vec{\boldsymbol{\xi}}_2^{(s)}, \ldots, \vec{\boldsymbol{\xi}}_{N_s}^{(s)} \right\} \tag{4.33}$$

where N_s is the number of candidates generated in that shard.

Each shard is mapped to physical input variables, evaluated using the parametric cycle model, filtered for feasible cycle solutions, and written to storage. The accepted dataset is then assembled from the accepted samples in each shard,

$$\mathcal{D}_{\text{accepted}} = \bigcup_{s=1}^{N_{\text{shard}}} \mathcal{D}_{\text{accepted}}^{(s)} \tag{4.34}$$

where $\mathcal{D}_{\text{accepted}}^{(s)}$ is the accepted subset from the s-th shard and N_{shard} is the total number of shards.

This procedure allows the dry and afterburning real-cycle datasets to be generated separately while controlling memory usage, preserving intermediate progress, and monitoring acceptance behavior during dataset construction. Sobol sampling provides broad candidate coverage for the high-dimensional input domain, while sharding makes the large-scale cycle-evaluation workflow manageable for the final Model C dataset.

4.3.4 Input Bounds and Operating Envelope Definition

The input bounds define the operating envelope over which the neural-network surrogate is intended to approximate the parametric cycle model. These bounds are part of the model definition: the trained network is not a universal turbofan model, but a surrogate for the sampled mixed-flow afterburning turbofan cycle space considered in this project. The se-

lected variables and their physical interpretation follow the parametric cycle formulation developed in Chapter 3 [15].

For each sampled input variable, the admissible range is written as

$$x_{j,\min} \leq x_j \leq x_{j,\max} \tag{4.35}$$

where x_j is the j-th input variable and $x_{j,\min}$ and $x_{j,\max}$ are its lower and upper bounds. The complete sampled input domain is therefore

$$\Omega = \{\vec{\boldsymbol{x}} \in \mathbb{R}^{n_x} : x_{j,\min} \leq x_j \leq x_{j,\max}, \quad j = 1, 2, \ldots, n_x\} \tag{4.36}$$

where n_x is the number of sampled input variables.

The bounds are selected to balance broad operating-envelope coverage against the need to obtain a usable feasible dataset after cycle evaluation. Very narrow bounds would limit the usefulness of the surrogate, while overly broad bounds would increase the number of failed or nonphysical cycle solutions. The individual feasibility requirements applied after evaluation are discussed in Section 4.4.

The effective input envelope differs between dry and afterburning operation. In dry operation, the afterburner is inactive and afterburner-specific variables are either omitted or fixed at their no-augmentation values. In afterburning operation, additional variables associated with afterburner heat addition, pressure loss, and nozzle matching become active. This separation is consistent with the wet/dry dataset structure used later in the mixture-of-experts surrogate.

Table 4.1 summarizes the main categories of variables bounded during dataset construction. The exact numerical bounds depend on the model case and are selected to match the ideal-cycle or real-cycle dataset being generated.

Table 4.1 – Input-bound categories used for dataset construction.

Category	Representative variables	Purpose
Flight condition	$M_0,\ T_0,\ P_0$	Defines freestream operating state
Gas properties	$\gamma,\ c_p,\ h_{\mathrm{PR}}$	Defines thermochemical assumptions
Compression system	$\pi_c,\ \pi_f,\ \eta_c,\ \eta_f$	Defines compressor and fan behavior
Combustion system	$\tau_\lambda,\ \eta_b,\ \pi_b$	Defines main-burner heat addition and loss
Turbine system	$\eta_t,\ \pi_t,\ \tau_t$	Defines turbine work extraction behavior
Mixer system	$\alpha,\ \pi_M,\ A_{16}/A_6$	Defines mixed-flow compatibility and losses
Afterburner system	$\tau_{\mathrm{AB}},\ \eta_{\mathrm{AB}},\ \pi_{\mathrm{AB}}$	Defines augmented operation
Nozzle system	$A_9/A_t,\ \eta_n,\ \pi_n$	Defines expansion and pressure-thrust behavior

The input bounds also determine how validation results should be interpreted. Predictions evaluated well inside the sampled feasible envelope are interpolation cases, while predictions near the envelope boundaries are more sensitive to local data density and feasibility limits. Predictions outside the defined bounds are extrapolation cases and should not be treated as physically reliable without additional validation against the parametric cycle model.

4.3.5 Training, Validation, and Test Splitting

After feasibility filtering, the accepted dataset is shuffled and divided into training, validation, and test partitions. This separation follows standard supervised-learning practice, where the training set is used to update model parameters, the validation set is used to monitor generalization during model development, and the test set is reserved for final held-out evaluation [18].

For an accepted dataset containing N samples, the split used in this project is

$$N_{\text{train}} = 0.70N, \qquad N_{\text{val}} = 0.15N, \qquad N_{\text{test}} = 0.15N \tag{4.37}$$

where N_{train}, N_{val}, and N_{test} are the numbers of training, validation, and test samples, respectively. In implementation, these values are rounded as needed so that every accepted sample belongs to exactly one partition.

The training partition is used during backpropagation to compute gradients and update the neural-network weights and biases. The validation partition is not used for parameter updates; instead, it provides an independent measure of model behavior during training and helps identify overfitting. The test partition is held out from both parameter updates and training-time model monitoring so that it can be used for final baseline statistical evaluation.

For the wet/dry expert strategy, dry and afterburning datasets are split within their respective operating modes before training the corresponding expert networks. This preserves the separation between non-augmented and augmented cycle behavior and prevents the partitioning procedure from mixing operating modes before expert assignment.

Consistent with the normalization procedure described in Section 4.1.5, input and output normalization statistics are computed from the training partition only. The same stored means and standard deviations are then applied unchanged to the validation and test partitions. This prevents validation or test data from entering the training process through preprocessing.

The resulting held-out test partitions provide the data used for baseline statistical validation in Chapter 5, including parity and residual analysis. The independent parametric sweep studies in Chapter 6 then provide additional physical validation by comparing surrogate predictions against the deterministic parametric cycle model over controlled variations of selected input variables.

4.4 Feasibility Filtering

The sampling methods described in Section 4.3 generate candidate input vectors over the prescribed operating envelope, but not every candidate produces a physically meaningful cycle solution. Some sampled combinations may violate thermodynamic requirements, produce invalid fuel-flow states, fail mixer or nozzle compatibility checks, or result in unreasonable performance quantities. Feasibility filtering is therefore applied after parametric cycle evaluation and before the data are used for neural-network training. This filtering step is part of the physics-guided dataset construction process. The parametric cycle model provides the reference solution, while the feasibility checks determine whether that solution is acceptable for supervised learning. Only candidate points that satisfy the physical and numerical requirements of the cycle model are retained in the accepted dataset [15,19].

The accepted dataset can be written as

$$\mathcal{D}_{\text{accepted}} = \{(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{y}}_i) : \vec{\boldsymbol{x}}_i \in \Omega,\ \vec{\boldsymbol{y}}_i = \mathcal{F}_{\text{PCA}}\left(\vec{\boldsymbol{x}}_i\right),\ \vec{\boldsymbol{y}}_i \in \Omega_{\text{phys}}\} \tag{4.38}$$

where Ω is the sampled input domain and Ω_{phys} is the set of physically acceptable cycle outputs. The following subsections define the feasibility checks used to reject failed, nonphysical, or unsuitable cycle evaluations before training.

4.4.1 Thermodynamic Feasibility Requirements

The first feasibility filters applied during dataset generation are thermodynamic validity checks. These checks remove candidate samples that produce undefined, non-finite, or physically impossible thermodynamic states before those samples can enter the training dataset. This step is necessary because many downstream cycle quantities depend on temperatures, pressures, specific heats, gas constants, and temperature ratios. If any of these quantities are invalid, the remaining cycle calculation may produce numerical artifacts rather than meaningful engine behavior.

For the ideal-cycle models, the initial checks require the sampled flight and gas-property inputs to be finite and physically positive where required. The static temperature, specific heat, and fuel heating value must be positive, and the ratio of specific heats must exceed unity:

$$T_0 > 0 \tag{4.39}$$

$$\gamma > 1 \tag{4.40}$$

$$c_p > 0 \tag{4.41}$$

$$h_{PR} > 0 \tag{4.42}$$

In Model A, γ, c_p, and h_{PR} are fixed constants that are stored with each row for consistency. In Model B, γ and h_{PR} are sampled directly, while c_p is derived from the sampled γ and the fixed gas constant. This preserves thermodynamic consistency between the specific heat, gas

constant, and ratio of specific heats.

The ideal-cycle datasets also require the nondimensional temperature ratios computed from the sampled inputs to remain real, finite, and positive. The ram temperature ratio, compressor temperature ratio, fan temperature ratio, turbine temperature ratio, and mixer temperature ratio are all checked during cycle evaluation. In particular, the turbine and mixer temperature ratios must satisfy

$$\tau_t > 0 \tag{4.43}$$

$$\tau_M > 0 \tag{4.44}$$

because negative or zero values would correspond to invalid stagnation temperature states. Candidate samples that fail these checks are rejected.

A similar check is applied to the nozzle exit static-temperature ratio. For the ideal-cycle models, the nozzle calculation requires

$$\frac{T_9}{T_0} > 0 \tag{4.45}$$

and the associated nozzle Mach-number argument must remain positive and finite. These requirements ensure that the computed exit Mach number and velocity are real. Although the nozzle-specific implications are discussed later, the temperature-ratio check is fundamentally thermodynamic because the exit kinetic state is derived from the available stagnation-to-static temperature drop.

For the real-cycle model, the thermodynamic feasibility requirements are more detailed because separate gas properties are used for the compressor-side, turbine-side, and afterburner-side flow states. The sampled ratios of specific heats must satisfy

$$\gamma_c > 1 \tag{4.46}$$

$$\gamma_t > 1 \tag{4.47}$$

$$\gamma_{AB} > 1 \tag{4.48}$$

and the corresponding specific heats and derived gas constants must remain positive:

$$c_{p,c} > 0, \qquad c_{p,t} > 0, \qquad c_{p,AB} > 0 \tag{4.49}$$

$$R_c > 0, \qquad R_t > 0, \qquad R_{AB} > 0 \tag{4.50}$$

where the gas constants are derived from the sampled thermodynamic properties. This prevents internally inconsistent gas-property combinations from entering the real-cycle dataset.

The real-cycle model also checks the basic temperature and fuel-energy scales,

$$T_0 > 0, \qquad T_{t4} > 0, \qquad T_{t7} > 0, \qquad h_{PR} > 0 \tag{4.51}$$

with the additional requirement that the main burner exit total temperature exceeds the

freestream static temperature,

$$T_{t4} > T_0 \tag{4.52}$$

For afterburning operation, the afterburner exit total temperature is required to be at least as large as the main burner exit total temperature,

$$T_{t7} \geq T_{t4} \tag{4.53}$$

while dry operation is handled separately through the dry-mode branch, where afterburner-specific quantities are fixed to their inactive values. These checks enforce the intended distinction between non-afterburning and afterburning samples.

The real-cycle model further requires the ram, diffuser, compressor, and fan temperature and pressure ratios to remain finite and positive. The ram temperature ratio and pressure ratio must satisfy

$$\tau_r > 0 \tag{4.54}$$

$$\pi_r > 0 \tag{4.55}$$

and the diffuser pressure recovery must remain positive. The compressor and fan temperature ratios are also checked so that

$$\tau_c > 1 \tag{4.56}$$

$$\tau_f > 1 \tag{4.57}$$

which is consistent with compression across the core compressor and fan. The heat-addition temperature ratios τ_λ and $\tau_{\lambda AB}$ are likewise required to be finite and positive.

These thermodynamic feasibility requirements form the first screening layer of the dataset-generation process. They do not determine whether the entire engine cycle is feasible by themselves, but they prevent invalid thermodynamic states from propagating into the fuel-flow, turbine, mixer, nozzle, and performance calculations. The remaining feasibility filters build on these checks by enforcing physically meaningful fuel addition, bypass ratio, mixer compatibility, nozzle expansion, thrust, and efficiency behavior.

4.4.2 Fuel-Flow and Bypass-Ratio Constraints

After the thermodynamic validity checks are satisfied, the dataset generator applies additional constraints to the fuel-flow quantities and bypass ratio. These variables are especially important because they determine whether the cycle solution represents a meaningful turbofan operating state. A sampled candidate may produce finite temperatures and pressure ratios while still requiring an unrealistic burner fuel-air ratio, afterburner fuel-air ratio, or bypass ratio. Such cases are rejected before the sample is stored for training.

The main burner fuel-air ratio is required to be finite, nonnegative, and bounded above.

In general, the accepted main fuel-air ratio must satisfy

$$f_{\min} \leq f \leq f_{\max} \tag{4.58}$$

with $f_{\min} = 0$ for all generated datasets. For Models A and B, the upper bound used for the main burner fuel-air ratio was

$$f_{\max} = 0.06 \tag{4.59}$$

while the final real-cycle Model C dataset used

$$f_{\max} = 0.08 \tag{4.60}$$

The larger upper bound in Model C reflects the wider real-cycle operating envelope used for the final surrogate.

The afterburner fuel-air ratio is handled according to the operating mode. In dry operation, the afterburner is inactive and the generator explicitly sets

$$f_{AB} = 0 \tag{4.61}$$

This convention is used so that dry and wet datasets can share a consistent output schema while still preserving the physical distinction between non-afterburning and afterburning operation.

In afterburning operation, the afterburner fuel-air ratio must be finite, nonnegative, and within the accepted augmented operating range:

$$f_{AB,\min} \leq f_{AB} \leq f_{AB,\max} \tag{4.62}$$

with $f_{AB,\min} = 0$. In Model A, the first ideal-cycle generator enforced nonnegative afterburner fuel addition but did not impose a strict upper bound. In Model B, the afterburner fuel-air ratio was bounded by

$$f_{AB,\max} = 0.08 \tag{4.63}$$

and in Model C the bound was increased to

$$f_{AB,\max} = 0.10 \tag{4.64}$$

to accommodate the broader real-cycle afterburning dataset. Samples that produce negative or excessive afterburner fuel addition are rejected because they do not represent acceptable augmented operation within the selected model envelope.

The total fuel-air ratio used for performance calculations is computed from the main-burner contribution and the afterburner contribution. Since the main-burner fuel is added only to the core stream while the afterburner fuel is applied downstream of the mixer, the resulting f_0 must remain finite and nonnegative. In Model C, this total fuel-air ratio is explic-

itly checked during the final performance calculation before thrust specific fuel consumption and efficiency quantities are accepted.

The bypass ratio is also treated as a feasibility-critical quantity. In the ideal-cycle models, α is solved from the ideal matching relation rather than freely accepted from the sampled inputs. In the real-cycle model, α is likewise computed from the real-cycle work balance and mixing relations. Therefore, even if all directly sampled inputs lie within their bounds, the resulting bypass ratio may still be outside the acceptable engine range.

For Models A and B, the accepted bypass-ratio range was

$$0.05 \leq \alpha \leq 3.0 \tag{4.65}$$

For Model C, the lower sampled bound was set to zero, but the feasibility check rejects values at or below this limit. The accepted real-cycle bypass ratio therefore satisfies

$$0 < \alpha \leq 3.0 \tag{4.66}$$

This prevents the real-cycle surrogate from being trained on negative or zero-flow bypass states while still allowing very low-bypass operation.

The combination of fuel-flow and bypass-ratio constraints is important because these quantities strongly influence all final performance outputs. Excessive fuel-air ratios can produce unrealistic thrust and temperature behavior, while invalid bypass ratios can corrupt the turbine work balance, mixer relations, and total fuel-air ratio. Applying these checks before training ensures that the accepted dataset represents operating states consistent with the intended mixed-flow turbofan configuration.

A summary of the fuel-flow and bypass-ratio feasibility bounds used across the model-development stages is provided in Table 4.2.

Table 4.2 – Fuel-flow and bypass-ratio feasibility bounds used during dataset generation

Model	Main fuel-air ratio	Afterburner fuel-air ratio	Bypass ratio
Model A	$0 \leq f \leq 0.06$	$f_{AB} \geq 0$	$0.05 \leq \alpha \leq 3.0$
Model B	$0 \leq f \leq 0.06$	$0 \leq f_{AB} \leq 0.08$	$0.05 \leq \alpha \leq 3.0$
Model C	$0 \leq f \leq 0.08$	$0 \leq f_{AB} \leq 0.10$	$0 < \alpha \leq 3.0$

These bounds do not imply that every point inside the fuel-flow and bypass-ratio limits is automatically feasible. They act as one layer of the full feasibility filter. A candidate sample must also satisfy the thermodynamic, mixer, nozzle, thrust, TSFC, and efficiency constraints before it is retained for training.

4.4.3 Mixer and Nozzle Feasibility Requirements

Mixer and nozzle feasibility requirements are applied because the mixed-flow afterburning turbofan cycle depends strongly on the compatibility between the core stream, bypass stream, mixed stream, and exhaust nozzle. A candidate sample may satisfy the basic thermodynamic, fuel-flow, and bypass-ratio constraints while still producing an invalid mixer state or nozzle expansion. These cases are rejected before the sample is accepted for training.

For the ideal-cycle Models A and B, the mixer is represented through the ideal mixed-stream temperature ratio. The computed mixer temperature ratio must be finite and positive,

$$\tau_M > 0 \tag{4.67}$$

and the mixed-stream total temperature entering the downstream afterburner and nozzle system is computed from this accepted state. In the ideal formulation, the mixed-stream total pressure P_{t6A} is not explicitly resolved by the generator and is retained as a placeholder diagnostic value. Therefore, the ideal mixer feasibility check is primarily a temperature-ratio and algebraic-validity check rather than a pressure-loss closure.

The ideal nozzle feasibility checks ensure that the exit static-temperature ratio and exit Mach-number calculation are physically meaningful. The nozzle exit temperature ratio must satisfy

$$\frac{T_9}{T_0} > 0 \tag{4.68}$$

and the nozzle Mach-number argument must also be positive,

$$\tau_r \tau_f - 1 > 0 \tag{4.69}$$

so that the computed nozzle exit Mach number is real. The resulting exit Mach number and nondimensional exit velocity must also be finite and positive:

$$M_9 > 0 \tag{4.70}$$

$$\frac{V_9}{a_0} > 0 \tag{4.71}$$

These checks remove ideal-cycle samples that would produce invalid nozzle kinematics before final thrust and efficiency quantities are evaluated.

For the real-cycle Model C, the mixer feasibility checks are more detailed. The real-cycle generator first evaluates the compatibility of the core and bypass streams entering the mixer. The core-stream mixer inlet Mach number M_6 is treated as a fixed representative value in the current Model C implementation because it appears in the mixer relations but is not exposed as an independent sampled input. The bypass-stream Mach number M_{16} is then computed from the pressure compatibility relation. The intermediate quantities used to define this state must remain finite, and the pressure ratio between the bypass and core mixer inlets

must satisfy

$$\frac{P_{t16}}{P_{t6}} > 0 \tag{4.72}$$

The bypass Mach-number argument must be nonnegative so that M_{16} is real, and the resulting bypass Mach number must be positive:

$$M_{16} > 0 \tag{4.73}$$

The real-cycle mixer also computes an effective bypass ratio entering the mixer, mixed-stream gas properties, the bypass-to-core total-temperature ratio, and the mixer temperature ratio. These quantities must satisfy

$$\alpha' \geq 0 \tag{4.74}$$
$$c_{p6A} > 0 \tag{4.75}$$
$$R_{6A} > 0 \tag{4.76}$$
$$\gamma_{6A} > 1 \tag{4.77}$$
$$\frac{T_{t16}}{T_{t6}} > 0 \tag{4.78}$$
$$\tau_M > 0 \tag{4.79}$$

with the additional thermodynamic consistency requirement

$$c_{p6A} > R_{6A} \tag{4.80}$$

so that the mixed-stream ratio of specific heats remains physically valid.

The constant-area mixer closure introduces additional feasibility requirements. The mixing functions ϕ_6, ϕ_{16}, and Φ must be finite and positive. The discriminant appearing in the mixed-stream Mach-number solution must be nonnegative, and the inner term of the square root expression must remain positive:

$$\phi_6 > 0 \tag{4.81}$$
$$\phi_{16} > 0 \tag{4.82}$$
$$\Phi > 0 \tag{4.83}$$
$$1 - 2(\gamma_{6A} + 1)\Phi \geq 0 \tag{4.84}$$

These requirements ensure that the mixed-stream Mach number M_{6A} is real and positive.

The resulting mixer area ratio and mass-flow parameters must also be finite and positive:

$$M_{6A} > 0 \tag{4.85}$$

$$\frac{A_{16}}{A_6} > 0 \tag{4.86}$$

$$\mathrm{MFP}_6 > 0 \tag{4.87}$$

$$\mathrm{MFP}_{6A} > 0 \tag{4.88}$$

The real-cycle generator also checks the mixer total-pressure ratio. The ideal mixer pressure ratio is first computed from the constant-area mixer relations, then multiplied by the sampled maximum mixer pressure recovery. Both the ideal and corrected mixer pressure ratios must remain finite and positive:

$$\pi_{M,\mathrm{ideal}} > 0 \tag{4.89}$$

$$\pi_M > 0 \tag{4.90}$$

These checks prevent invalid mixer pressure recovery from propagating into the downstream nozzle pressure-ratio calculation.

The real-cycle nozzle feasibility requirements begin with the assembled nozzle-inlet total-to-static pressure ratio,

$$\frac{P_{t9}}{P_9} = \frac{P_0}{P_9}\pi_r\pi_d\pi_c\pi_b\pi_t\pi_M\pi_{AB}\pi_n \tag{4.91}$$

The generator requires this quantity to be finite and greater than unity:

$$\frac{P_{t9}}{P_9} > 1 \tag{4.92}$$

This condition ensures that a valid expansion from the nozzle inlet total state to the exit static state can be computed.

The station 9 gas properties are then selected according to operating mode. In afterburning operation, the station 9 gas properties correspond to the afterburner gas model. In dry operation, they correspond to the mixed-stream gas properties. In both cases, the exit gas properties and exit static-temperature ratio must satisfy

$$c_{p9} > 0 \tag{4.93}$$

$$R_9 > 0 \tag{4.94}$$

$$\gamma_9 > 1 \tag{4.95}$$

$$\frac{T_9}{T_0} > 0 \tag{4.96}$$

$$c_{p9} > R_9 \tag{4.97}$$

These constraints prevent invalid exit gas-property combinations from being used in the

nozzle Mach-number and velocity calculations.

The nozzle Mach number is computed from the pressure-ratio expansion argument. The argument must be finite and nonnegative:

$$\left(\frac{P_{t9}}{P_9}\right)^{(\gamma_9-1)/\gamma_9} - 1 \geq 0 \tag{4.98}$$

The resulting exit Mach number and nondimensional exit velocity must be finite and positive:

$$M_9 > 0 \tag{4.99}$$

$$\frac{V_9}{a_0} > 0 \tag{4.100}$$

The area ratio A_9/A_t is then computed using the mass-flow-parameter form. The throat and exit mass-flow parameters must be finite and positive, and the resulting nozzle area ratio must also be positive:

$$\mathrm{MFP}_t > 0 \tag{4.101}$$

$$\mathrm{MFP}_9 > 0 \tag{4.102}$$

$$\frac{A_9}{A_t} > 0 \tag{4.103}$$

Finally, the real-cycle generator checks the mixed-stream total temperature and total pressure used as internal diagnostic outputs. These quantities are computed after the mixer and turbine states are accepted and must satisfy

$$T_{t6A} > 0 \tag{4.104}$$

$$P_{t6A} > 0 \tag{4.105}$$

with both quantities required to be finite. These diagnostic states are retained in the dataset because they provide useful information for evaluating whether the trained surrogate has preserved the internal behavior of the mixed-flow cycle.

These mixer and nozzle feasibility requirements remove samples that would otherwise contaminate the training dataset with invalid algebraic closures, non-real Mach numbers, nonphysical area ratios, or impossible pressure expansions. This is particularly important for Model C, where the mixer and nozzle states are coupled to real-cycle pressure losses, afterburner behavior, and the pressure-thrust contribution in the final thrust equation.

4.4.4 Efficiency and Performance Bounds

The final feasibility layer checks whether the completed cycle solution produces physically meaningful performance outputs. These checks are applied after the thermodynamic, fuel-flow, bypass-ratio, mixer, and nozzle requirements have been satisfied. This ordering is

important because thrust, thrust specific fuel consumption, and efficiency metrics depend on the complete downstream cycle solution.

The first performance requirement is that the specific thrust must be finite and positive. In general, the accepted samples must satisfy

$$\frac{F}{\dot{m}_0} > F_{\min} \tag{4.106}$$

where $F_{\min}$ is the minimum allowable specific-thrust threshold used by the corresponding dataset generator. In the first ideal-cycle generator, $F_{\min}$ was effectively zero, so samples were rejected if the specific thrust was non-finite or non-positive. In Model B, the minimum allowable specific thrust was set to

$$F_{\min} = 10^{-6} \tag{4.107}$$

and in Model C it was set to

$$F_{\min} = 10^{-8} \tag{4.108}$$

using the SI-consistent specific-thrust units used by the generator. These small positive thresholds prevent near-zero or negative thrust states from being accepted as valid engine operating points.

The thrust specific fuel consumption is then computed from the total fuel-air ratio and specific thrust,

$$S = \frac{f_0}{F/\dot{m}_0} \tag{4.109}$$

The accepted samples must produce a finite and positive value of S. For Models B and C, the minimum accepted value was

$$S_{\min} = 10^{-12} \tag{4.110}$$

which acts primarily as a numerical guard against zero, negative, or undefined TSFC values. The first ideal-cycle generator similarly rejects samples when the computed TSFC is non-finite or non-positive.

The total fuel-air ratio is also checked at the final performance stage. For the mixed-flow afterburning turbofan datasets, the total fuel-air ratio is

$$f_0 = \frac{f}{1+\alpha} + f_{AB} \tag{4.111}$$

and the accepted samples must satisfy

$$f_0 \geq 0 \tag{4.112}$$

with f_0 required to be finite. This check prevents invalid fuel-flow combinations from being used in the final thrust, TSFC, and efficiency calculations.

The efficiency quantities are retained as both performance metrics and diagnostic indicators. For Models A and B, the thermal, propulsive, and overall efficiencies are computed after the thrust and TSFC calculations. The implemented feasibility check requires these values to be finite:

$$\eta_T,\ \eta_P,\ \eta_O \in \mathbb{R} \tag{4.113}$$

where non-finite efficiency values cause the candidate sample to be rejected. In these ideal-cycle generators, the efficiency values are not used as hard upper-bound gates; instead, they act as diagnostic performance outputs after the main thrust and fuel-consumption checks have been passed.

For the real-cycle Model C generator, additional care is taken in the efficiency calculation because the real thrust expression includes both momentum thrust and pressure-thrust contributions. The full-thrust-based propulsive and overall efficiencies are computed for diagnostic purposes, but the primary output vector uses momentum-based propulsive and overall efficiencies to prevent the pressure-thrust term from distorting the propulsive-efficiency interpretation. The total specific thrust is decomposed as

$$\left(\frac{F}{\dot{m}_0}\right) = \left(\frac{F}{\dot{m}_0}\right)_{\text{mom}} + \left(\frac{F}{\dot{m}_0}\right)_{\text{press}} \tag{4.114}$$

where the first term represents the momentum contribution and the second term represents the pressure-thrust contribution. The momentum-based propulsive efficiency is then used as the primary η_P output for Model C:

$$\eta_{P,\text{phys}} = \frac{2g_c V_0 \left(F/\dot{m}_0\right)_{\text{mom}}}{a_0^2\left[(1+f_0)\left(V_9/a_0\right)^2 - M_0^2\right]} \tag{4.115}$$

and the corresponding overall efficiency is

$$\eta_{O,\text{phys}} = \eta_{P,\text{phys}}\eta_T \tag{4.116}$$

These quantities are checked for finite values before the sample is accepted.

The thermal efficiency calculation also requires the kinetic-energy term and fuel-power denominator to remain finite and positive. In Model C, the relevant kinetic term is

$$K = (1+f_0)\left(\frac{V_9}{a_0}\right)^2 - M_0^2 \tag{4.117}$$

and the denominator terms used in the efficiency definitions must satisfy

$$a_0^2 K > 0 \tag{4.118}$$

$$2 g_c f_0 h_{PR} > 0 \tag{4.119}$$

These checks prevent samples with invalid kinetic-energy differences or invalid fuel-energy scaling from entering the final dataset.

A summary of the primary performance feasibility thresholds is provided in Table 4.3.

Table 4.3 – Performance feasibility bounds used during dataset generation

Model	Specific thrust requirement	TSFC requirement	Efficiency requirement
Model A	$F/\dot{m}_0 > 0$	$S > 0$	finite η_T, η_P, η_O
Model B	$F/\dot{m}_0 > 10^{-6}$	$S > 10^{-12}$	finite η_T, η_P, η_O
Model C	$F/\dot{m}_0 > 10^{-8}$	$S > 10^{-12}$	finite $\eta_T, \eta_{P,\mathrm{phys}}, \eta_{O,\mathrm{phys}}$

The efficiency and performance checks complete the feasibility-filtering process. A sample is only retained if it produces finite, positive thrust, finite positive fuel-consumption behavior, acceptable total fuel addition, and valid efficiency diagnostics. This final screening step is essential because the neural network is trained only on accepted samples. If invalid performance states were retained, the surrogate could learn trends associated with failed cycle solutions rather than physically meaningful mixed-flow turbofan operation.

4.5 Wet and Dry Mixture-of-Experts Strategy

The mixed-flow afterburning turbofan does not behave as a single smooth input-output mapping across dry and augmented operation. When the afterburner is inactive, the downstream cycle is governed by the mixed exhaust stream and nozzle expansion without additional heat addition. When the afterburner is active, additional fuel addition, afterburner pressure loss, higher nozzle entry temperature, and nozzle-area matching become dominant parts of the cycle response. Treating both operating modes with a single undifferentiated regression model can therefore increase the burden placed on the network.

To reduce this burden, the surrogate is organized using separate dry and afterburning expert models. This follows the general mixture-of-experts concept, where different expert models are assigned to different regimes of an input-output mapping [23]. In contrast to adaptive mixture-of-experts formulations that learn a gating function from the data, the present work uses a deterministic binary gate because the afterburner mode is explicitly known from the engine operating state.

The resulting architecture separates the dry and afterburning mappings while preserving a common prediction framework. Each expert is trained on samples from its corresponding operating mode, with its own dataset partitioning, normalization parameters, and trainable

network weights. During prediction, the afterburner mode flag selects the appropriate expert, producing a dry-cycle prediction when augmentation is inactive and an afterburning-cycle prediction when augmentation is active.

4.5.1 Afterburning Mode Separation

Dry and afterburning operation represent different propulsion regimes within the same mixed-flow turbofan architecture. As established in the parametric cycle formulation, dry operation excludes afterburner heat addition, while afterburning operation introduces additional fuel addition, pressure loss, higher nozzle-entry total temperature, and stronger coupling with nozzle-area matching [15]. These changes alter both the magnitude and sensitivity of the cycle outputs.

For this reason, the dataset is separated by afterburning mode before training. The dry dataset is written as

$$\mathcal{D}_{\text{dry}} = \{(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{y}}_i) : m_{AB,i} = 0\} \tag{4.120}$$

and the afterburning dataset is written as

$$\mathcal{D}_{\text{wet}} = \{(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{y}}_i) : m_{AB,i} = 1\} \tag{4.121}$$

where m_{AB} is the binary afterburner mode flag.

This separation allows each expert network to specialize in one operating regime. The dry expert learns the non-augmented cycle response, while the afterburning expert learns the augmented response that includes afterburner and nozzle-matching effects. Separating the modes reduces the need for a single network to represent a sharp change in cycle behavior across the afterburner activation boundary.

Each mode-specific dataset is split, normalized, and used to train its corresponding expert model independently. The resulting dry and afterburning experts therefore have separate trainable parameters and separate scaling statistics, while retaining the same overall supervised-regression framework developed in the preceding sections.

4.5.2 Binary Expert-Gating Formulation

The wet and dry mixture-of-experts model uses a binary gate to select the appropriate surrogate for a requested operating condition. This follows the general mixture-of-experts concept, where different expert models represent different regimes of an input-output mapping [23]. Unlike adaptive mixture-of-experts formulations that learn the gating function from data, the present gate is deterministic because the afterburner mode is known during both dataset generation and model evaluation. The gate is based on the afterburner mode indicator m_{AB}, where $m_{AB} = 0$ selects dry operation and $m_{AB} = 1$ selects afterburning operation.

The gated surrogate can be written as

$$\hat{\mathbf{y}} = (1 - m_{AB})\,\mathcal{F}_{dry}\left(\mathbf{x}_{dry}; \boldsymbol{\theta}_{dry}\right) + m_{AB}\mathcal{F}_{wet}\left(\mathbf{x}_{wet}; \boldsymbol{\theta}_{wet}\right) \tag{4.122}$$

where $\mathcal{F}_{dry}$ is the dry expert, $\mathcal{F}_{wet}$ is the afterburning expert, $\boldsymbol{\theta}_{dry}$ is the trained parameter set of the dry expert, and $\boldsymbol{\theta}_{wet}$ is the trained parameter set of the wet expert. When $m_{AB} = 0$, the afterburning expert contribution is removed and the prediction becomes

$$\hat{\mathbf{y}} = \mathcal{F}_{dry}\left(\mathbf{x}_{dry}; \boldsymbol{\theta}_{dry}\right) \tag{4.123}$$

When $m_{AB} = 1$, the dry expert contribution is removed and the prediction becomes

$$\hat{\mathbf{y}} = \mathcal{F}_{wet}\left(\mathbf{x}_{wet}; \boldsymbol{\theta}_{wet}\right) \tag{4.124}$$

The input vectors for the two experts may contain similar variables, but they are not required to be identical in interpretation. In dry operation, the afterburner-related quantities are inactive or fixed by the dry-mode definition. In afterburning operation, the afterburner variables become active and contribute directly to the downstream thermal state, fuel-air ratio, and nozzle expansion. Therefore, the dry and wet experts are trained using their own datasets, normalization constants, and learned parameter sets.

The binary gate also simplifies the training process. Instead of training one larger model on a combined dataset containing both operating modes, each expert is optimized only over the subset of the operating envelope assigned to that mode. The training losses for the two experts can therefore be written separately as

$$\mathcal{L}_{dry} = \mathcal{L}\left(\mathcal{F}_{dry}\left(\mathbf{x}_{dry}; \boldsymbol{\theta}_{dry}\right), \mathbf{y}_{dry}\right) \tag{4.125}$$

$$\mathcal{L}_{wet} = \mathcal{L}\left(\mathcal{F}_{wet}\left(\mathbf{x}_{wet}; \boldsymbol{\theta}_{wet}\right), \mathbf{y}_{wet}\right) \tag{4.126}$$

where $\mathbf{y}_{dry}$ and $\mathbf{y}_{wet}$ are the corresponding reference outputs from the dry and wet datasets. Since the two experts are trained independently, errors in one operating mode do not directly modify the weights of the other expert.

This deterministic gating formulation is appropriate for the present project because the objective is to predict cycle behavior for explicitly specified dry or afterburning operation. The model is not required to infer whether the afterburner should be active; it is only required to evaluate the correct surrogate once the operating mode is selected. This keeps the architecture simple and makes the prediction pathway easy to interpret.

The main limitation of the binary gate is that it does not model a continuous transition between dry and afterburning operation. In a physical engine, the afterburner may be scheduled gradually through fuel staging, ignition logic, and nozzle-area changes. The present model instead treats dry and wet operation as two discrete regimes. This is sufficient for the surrogate models developed in this report, but future work could replace the deterministic gate with a learned or continuous gating function that blends expert outputs across intermediate augmentation states.

Despite this limitation, the binary expert gate provides a practical and effective architecture for the current dataset structure. It preserves the physical distinction between dry and afterburning operation, reduces the complexity of each regression problem, and allows the final surrogate to evaluate the appropriate expert using a simple mode-selection variable.

4.5.3 Connection to Models A, B, and C

The wet and dry mixture-of-experts strategy is used consistently throughout the staged model-development process. Models A, B, and C all preserve the same basic idea: dry and afterburning operation are treated as separate supervised regression problems, and a binary gate selects the corresponding expert during prediction. The primary difference between the models is not the gating logic, but the dimensionality and physical completeness of the input space.

Model A is the fixed-property ideal-cycle surrogate. Its dataset uses the same stored input-column scaffold later retained by Model B,

$$\vec{\boldsymbol{x}}_A = \begin{bmatrix} M_0 & T_0 & \gamma & c_p & h_{PR} & T_{t4} & T_{t7,\mathrm{cmd}} & \pi_c & \pi_f & m_{AB} \end{bmatrix}^T \tag{4.127}$$

but γ, c_p, and h_{PR} are constants over the dataset. As a result, these quantities do not function as independent learned input directions in Model A. Their influence is absorbed into the fitted network weights as part of the fixed ideal-cycle mapping. The effective independent sampling dimension is therefore lower than the stored input dimension. For Model A, the dry branch samples M_0, T_0, π_c, π_f, and τ_λ, while the wet branch additionally samples τ_{AB}.

Model B keeps the same ideal-cycle dataset scaffold, but removes the fixed-property simplification. The ratio of specific heats and fuel heating value are varied across the dataset, and the specific heat is derived from the sampled γ using the fixed gas constant. Thus, the same stored input-vector structure is retained, but γ, c_p, and h_{PR} become active thermochemical variables rather than constants embedded in the mapping. This increases the dimensionality of the learned relationship while preserving the same ideal-cycle output structure used in Model A.

The Model B input vector can therefore be written using the same column structure as Model A,

$$\vec{\boldsymbol{x}}_B = \begin{bmatrix} M_0 & T_0 & \gamma & c_p(\gamma) & h_{PR} & T_{t4} & T_{t7,\mathrm{cmd}} & \pi_c & \pi_f & m_{AB} \end{bmatrix}^T \tag{4.128}$$

but the interpretation is different. In Model A, the gas-property columns are constant schema fields. In Model B, they are part of the active input space. The dry branch therefore samples seven independent directions, while the wet branch samples eight independent directions due to the additional afterburner-temperature-ratio input.

Model C is the real-cycle production surrogate. It no longer uses the fixed-property ideal-cycle simplification and instead expands the input vector to include the real-cycle parameters required by the higher-fidelity equation set. The Model C input vector includes separate gas-

property and component-behavior variables, pressure-loss parameters, efficiency parameters, burner and afterburner thermal states, and nozzle pressure information. The Model C input vector can be written compactly as

$$\vec{\boldsymbol{x}}_C = \left[\vec{\boldsymbol{x}}_{\text{flight}}^T \quad \vec{\boldsymbol{x}}_{\text{gas}}^T \quad \vec{\boldsymbol{x}}_{\text{loss}}^T \quad \vec{\boldsymbol{x}}_{\text{comp}}^T \quad \vec{\boldsymbol{x}}_{\text{burn}}^T \quad \vec{\boldsymbol{x}}_{\text{mix/noz}}^T \quad m_{AB}\right]^T \tag{4.129}$$

where the grouped inputs represent the flight condition, gas-property parameters, component pressure-loss parameters, compression-system parameters, combustion and afterburner parameters, mixer/nozzle parameters, and the binary afterburner mode flag.

This produces a substantially higher-dimensional regression problem than the ideal-cycle models. In the implemented real-cycle generator, the stored Model C input vector contains 26 input columns. Candidate sampling uses separate dry and wet branches, with the wet branch requiring additional active afterburner variables relative to the dry branch.

The dimensional progression is summarized in Table 4.4. The stored input dimension refers to the number of input columns retained in the training matrix. The active sampled dimension refers to the number of independent variables varied during candidate generation for each operating branch.

Table 4.4 – Dimensional progression from Model A to Model C.

Model	Stored input dimension	Active dry dimension	Active wet dimension
Model A	10	5	6
Model B	10	7	8
Model C	26	19	23

This progression was intentional. Model A first established that the mixture-of-experts surrogate could learn the ideal mixed-flow afterburning turbofan mapping under fixed thermochemical assumptions. Model B then tested whether the same architecture could handle variation in γ, c_p, and h_{PR} without changing the overall dataset scaffold. Model C extended the same framework to the full real-cycle input space, where none of the primary thermodynamic or component-loss terms are treated as fixed ideal constants.

The mixture-of-experts strategy therefore provides continuity across the three models while allowing the physical fidelity and dimensionality to increase. The same dry/wet gating concept is preserved, but each successive model requires the expert networks to learn a richer mapping. Chapter 5 describes the specific architecture, dataset size, training behavior, and validation performance of each model in this staged development sequence.

4.6 Chapter Summary

This chapter established the machine learning framework used to construct the mixed-flow afterburning turbofan surrogate models. The parametric cycle analysis developed in Chapter 3 was treated as the reference forward model, while the neural network was formulated as

a supervised regression surrogate trained to approximate the same input-output relationship over a feasible sampled operating envelope.

The chapter first introduced the multi-layer perceptron architecture used by the surrogate models. The MLP was selected because the cycle-analysis problem can be represented as a fixed-size vector regression task, where engine input parameters are mapped to continuous performance and state outputs. The supervised learning formulation, loss-function structure, output weighting, and normalization process were then described to explain how the neural networks are trained and evaluated.

The physics-guided nature of the workflow was also defined. The approach used in this project is not a fully physics-informed neural network in which the cycle equations are embedded directly into the training loss. Instead, the physics enters through the parametric cycle data generator, feasibility filters, operating-envelope definition, dry/wet mode separation, and physical trend validation. This allows the training process to remain computationally practical while keeping the learned mapping anchored to the propulsion model.

The sampling and dataset-construction strategy was then discussed. Because the engine model contains many interacting input variables, structured Cartesian grids are not practical for dataset generation. Latin hypercube sampling was used during the lower-dimensional ideal-cycle development stages, while Sobol-based sampling and dataset sharding were used for the higher-dimensional real-cycle dataset. The final accepted datasets were globally shuffled and split using a 70–15–15 training, validation, and test partition.

Feasibility filtering was presented as a critical part of the data-generation process. Candidate samples were rejected if they produced invalid thermodynamic states, unacceptable fuel-air ratios, invalid bypass ratios, failed mixer or nozzle states, nonphysical thrust or TSFC values, or invalid efficiency diagnostics. These filters define the feasible region of the sampled operating envelope and prevent the surrogate from learning numerical artifacts from failed cycle solutions.

Finally, the wet and dry mixture-of-experts strategy was introduced. Dry and afterburning operation were treated as separate regression problems because afterburner activation changes the structure of the cycle response. A binary expert gate selects the dry or wet surrogate according to the specified afterburner mode. This same strategy is retained across Models A, B, and C, while the input dimensionality and physical completeness increase from the fixed-property ideal-cycle surrogate to the full real-cycle production surrogate. The framework developed in this chapter provides the foundation for the model-specific training and validation work presented in Chapter 5.

Chapter 5: Model Training and Validation

5.1 Model Development Overview

The machine-learning framework developed in Chapter 4 is implemented through three progressively more complete surrogate models. Each model uses the same general supervised-regression structure, but the physical fidelity of the reference cycle model and the dimensionality of the input-output mapping are increased in stages. This staged development makes it possible to validate the neural-network approach first on simpler ideal-cycle mappings before applying the same methodology to the higher-dimensional real-cycle formulation.

Model A serves as the fixed-property ideal-cycle baseline. In this model, thermodynamic properties and fuel heating value are treated as constants, so the network primarily learns the ideal mixed-flow afterburning turbofan cycle response. Model B extends this baseline by introducing variable thermochemical inputs while retaining the ideal-cycle structure. Model C then applies the surrogate methodology to the real-cycle formulation, including component efficiencies, pressure losses, mixer behavior, afterburner effects, nozzle behavior, and additional diagnostic state outputs.

The dry and afterburning expert separation introduced in Section 4.5 is retained throughout the model sequence. Each model is therefore trained using separate dry and afterburning experts, allowing the non-augmented and augmented cycle responses to be learned independently. The purpose of Chapter 5 is to document the implementation of these trained models, present their baseline training and validation behavior, and establish that each surrogate reproduces the corresponding parametric cycle reference data with sufficient accuracy for the later sweep and extrapolation studies.

5.1.1 Staged Development Strategy

The surrogate models were developed using a staged strategy rather than training the full real-cycle model as the first implementation. This approach reduced development risk by separating the neural network learning problem from the increasing complexity of the propulsion model. Each model stage was designed to answer a specific question about the

feasibility of the surrogate framework.

Model A was used as the baseline proof-of-capability case. It tested whether the selected feedforward neural network structure could learn the ideal mixed-flow afterburning turbofan mapping when the gas properties and heating value were held constant. This provided the simplest controlled case for verifying the dataset-generation process, dry/wet expert separation, normalization workflow, and training scripts.

Model B then retained the same ideal-cycle framework while introducing variation in thermochemical quantities. The purpose of this stage was to determine whether the same surrogate structure could learn sensitivity to γ, c_p, and h_{PR} without the additional complication of real-cycle component losses. Model B therefore served as an intermediate bridge between the fixed-property ideal model and the fully expanded real-cycle model.

Model C was developed as the final high-dimensional surrogate. This stage used the real mixed-flow afterburning turbofan equations and included the component pressure losses, polytropic efficiencies, burner and afterburner efficiencies, mixer pressure recovery, nozzle loss, and mode-dependent gas properties. Unlike the earlier ideal-cycle stages, Model C treated the full input vector as active and did not rely on fixed primary input constants. This sequence allowed each level of complexity to be validated before moving to the next. As a result, errors in the final real-cycle surrogate could be interpreted relative to the simpler ideal-cycle models rather than treated as unexplained failures of the entire framework.

5.1.2 Relationship Between Models A, B, and C

Models A, B, and C are not independent surrogate concepts. They form a progressive model hierarchy in which each stage removes one major simplification from the previous stage. The purpose of this hierarchy is to build confidence in the surrogate framework before applying it to the full real-cycle mixed-flow afterburning turbofan problem.

Model A represents the lowest-complexity case. It uses the ideal-cycle equations with fixed thermochemical properties. Although the stored input matrix includes columns for γ, c_p, and h_{PR}, these values remain constant over the generated dataset. As a result, Model A does not learn independent sensitivity to gas-property variation. Instead, these quantities are embedded into the fixed ideal-cycle mapping learned by the network.

Model B retains the same ideal-cycle structure and dataset scaffold as Model A, but activates the thermochemical dimensions. In this stage, γ and h_{PR} are varied during dataset generation, and c_p is derived consistently from the gas constant and the sampled value of γ. Model B therefore tests whether the same surrogate architecture can learn a broader ideal-cycle mapping when the gas properties are no longer constant.

Model C extends the same methodology to the real-cycle formulation. In this case, the input vector is expanded to include the additional variables required by the real parametric cycle analysis, including component efficiencies, pressure recovery terms, pressure losses, mode-dependent gas properties, afterburner behavior, mixer pressure recovery, and nozzle loss. Unlike Model A, none of the primary input-vector components are treated as fixed

constants in the full Model C dataset. The model therefore represents the highest-fidelity and highest-dimensional surrogate developed in this project.

This progression is important because it separates three different sources of difficulty. Model A primarily tests the neural network implementation and the basic ideal-cycle mapping. Model B tests whether the same framework can handle thermochemical variability. Model C tests whether the framework can scale to the full real-cycle input space with losses, efficiencies, and mode-dependent component behavior.

The dry/wet expert separation remains consistent across all three models. Each stage uses separate dry and afterburning experts, with the binary afterburner gate selecting the appropriate network during inference. Therefore, the main architectural idea is preserved while the physical fidelity and input dimensionality increase from Model A to Model C.

5.2 Common Neural Network Implementation

Models A, B, and C are implemented using a common neural-network training framework. The same general software structure is retained across the model sequence so that differences in performance can be attributed primarily to the underlying cycle formulation, input dimensionality, output definition, and training dataset rather than to changes in the implementation procedure. Each model is trained as a supervised feedforward surrogate using the dry and afterburning expert separation defined in Section 4.5.

The implementation is organized around three groups of scripts. The data preparation scripts load the accepted cycle-analysis datasets, select the required input and output columns, separate dry and afterburning cases when needed, and compute the training-set normalization parameters. The training scripts construct the corresponding neural-network architecture, initialize the trainable parameters, evaluate the normalized loss function, update the network weights, monitor validation behavior, and save the best-performing model checkpoint. The prediction scripts load the trained model package, apply the stored input normalization, evaluate the appropriate expert network, and transform the predicted outputs back to their physical dimensional or nondimensional coordinates.

This organization is used to keep the training process reproducible across the three model stages. Model A, Model B, and Model C differ in the number of input variables, the number and type of output quantities, the dataset size, and the level of physical fidelity in the reference cycle model. However, the overall implementation pattern remains the same: prepare the accepted data, train the mode-specific expert networks, monitor validation behavior, store the trained parameters and scaling constants, and evaluate the resulting surrogate on held-out test data.

The following subsections describe the common implementation details used across the trained models. The emphasis is on the practical construction of the training arrays, neural-network parameter initialization, forward evaluation, gradient-based parameter updates, loss monitoring, and saved prediction workflow. General methodology such as supervised re-

gression, normalization, dataset splitting, and wet/dry expert selection was established in Chapter 4 and is referenced here only where needed to clarify the implementation.

5.2.1 Input and Output Data Preparation

The data preparation scripts convert the accepted cycle-analysis datasets into the input and target matrices used by the training scripts. This step selects the required input columns, selects the supervised output columns, applies the mode-specific dry or afterburning conventions, and stores the metadata required to reproduce the prediction workflow. The prepared arrays are then passed to the training, validation, and test split procedure described in Section 5.2.2.

For Model A, the ideal-cycle dry expert uses the input vector

$$\vec{\boldsymbol{x}}_{A,dry} = \begin{bmatrix} M_0 & T_0 & \log(\pi_c) & \log(\pi_f) & \tau_\lambda \end{bmatrix}^T \tag{5.1}$$

while the ideal-cycle afterburning expert uses

$$\vec{\boldsymbol{x}}_{A,wet} = \begin{bmatrix} M_0 & T_0 & \log(\pi_c) & \log(\pi_f) & \tau_\lambda & \tau_{AB} \end{bmatrix}^T. \tag{5.2}$$

The logarithmic pressure-ratio inputs are used in the preparation scripts to improve numerical conditioning before normalization. In Model A, the gas properties and fuel heating value are fixed by the ideal-cycle formulation rather than treated as independently sampled network inputs.

For Model B, the same ideal-cycle structure is retained, but variable thermochemical inputs are added. The dry and afterburning input vectors are

$$\vec{\boldsymbol{x}}_{B,dry} = \begin{bmatrix} M_0 & T_0 & \log(\pi_c) & \log(\pi_f) & \tau_\lambda & \gamma & h_{PR} \end{bmatrix}^T, \tag{5.3}$$

$$\vec{\boldsymbol{x}}_{B,wet} = \begin{bmatrix} M_0 & T_0 & \log(\pi_c) & \log(\pi_f) & \tau_\lambda & \tau_{AB} & \gamma & h_{PR} \end{bmatrix}^T. \tag{5.4}$$

In the current Model B implementation, c_p is not used as an independent neural-network input because it is determined consistently from γ and the fixed gas constant used in the ideal-cycle formulation.

The primary ideal-cycle output vector is similar for Models A and B. For dry operation, the supervised primary outputs are

$$\vec{\boldsymbol{y}}_{\text{ideal},dry} = \begin{bmatrix} F/\dot{m}_0 & S & \alpha & f & \eta_T & \eta_P \end{bmatrix}^T, \tag{5.5}$$

while for afterburning operation the afterburner fuel-air ratio is added,

$$\vec{\boldsymbol{y}}_{\text{ideal},wet} = \begin{bmatrix} F/\dot{m}_0 & S & \alpha & f & f_{AB} & \eta_T & \eta_P \end{bmatrix}^T. \tag{5.6}$$

The mixed-stream total temperature T_{t6A} is retained as an auxiliary supervised output for the ideal-cycle experts, while f_0 and η_O are retained as consistency-check quantities rather

than primary training targets.

For Model C, the real-cycle input vector is larger and is written in grouped form for readability:

$$\vec{\boldsymbol{x}}_C = \begin{bmatrix} \vec{\boldsymbol{x}}_{\text{gas}}^T & \vec{\boldsymbol{x}}_{\text{loss}}^T & \vec{\boldsymbol{x}}_{\text{eff}}^T & \vec{\boldsymbol{x}}_{\text{op}}^T \end{bmatrix}^T . \tag{5.7}$$

The grouped inputs are defined as

$$\vec{\boldsymbol{x}}_{\text{gas}} = \begin{bmatrix} \gamma_c & c_{p,c} & \gamma_t & c_{p,t} & \gamma_{AB} & c_{p,AB} & h_{PR} \end{bmatrix}^T , \tag{5.8}$$

$$\vec{\boldsymbol{x}}_{\text{loss}} = \begin{bmatrix} \pi_{d,\max} & \pi_b & \pi_{AB} & \pi_{M,\max} & \pi_n \end{bmatrix}^T , \tag{5.9}$$

$$\vec{\boldsymbol{x}}_{\text{eff}} = \begin{bmatrix} e_c & e_f & e_t & \eta_b & \eta_{AB} & \eta_m \end{bmatrix}^T , \tag{5.10}$$

$$\vec{\boldsymbol{x}}_{\text{op}} = \begin{bmatrix} M_0 & T_0 & P_0/P_9 & T_{t4} & T_{t7} & \pi_c & \pi_f & m_{AB} \end{bmatrix}^T . \tag{5.11}$$

This grouping preserves the full real-cycle input definition while avoiding a single unreadable input-vector expression. In the implementation, selected positive variables such as π_c, π_f, and h_{PR} are transformed logarithmically before normalization.

The Model C output targets are separated into primary outputs and auxiliary diagnostic outputs. The primary output vector is

$$\vec{\boldsymbol{y}}_{C,p} = \left[F/\dot{m}_0,\ f,\ f_{AB},\ f_0,\ S,\ \alpha,\ \eta_T,\ \eta_{P,phys},\ \eta_{O,phys},\ \eta_c,\ \eta_t \right]^T \tag{5.12}$$

where $\eta_{P,phys}$ and $\eta_{O,phys}$ are the physically bounded propulsive and overall efficiency targets used for learning. The report-form efficiency quantities are retained in the raw data for checking and interpretation, but they are not used as primary supervised targets.

The auxiliary diagnostic output vector is

$$\vec{\boldsymbol{y}}_{C,a} = \begin{bmatrix} T_{t6A} & P_{t6A} & M_9 & A_9/A_t & \tau_c & \tau_f & \tau_t \end{bmatrix}^T . \tag{5.13}$$

These auxiliary outputs provide internal-state supervision for the real-cycle surrogate and support the physical consistency checks discussed later in the validation results.

For each prepared dataset, the selected input and target vectors are assembled into matrix form,

$$X = \begin{bmatrix} \vec{\boldsymbol{x}}_1^T \\ \vec{\boldsymbol{x}}_2^T \\ \vdots \\ \vec{\boldsymbol{x}}_N^T \end{bmatrix} , \qquad Y = \begin{bmatrix} \vec{\boldsymbol{y}}_1^T \\ \vec{\boldsymbol{y}}_2^T \\ \vdots \\ \vec{\boldsymbol{y}}_N^T \end{bmatrix} . \tag{5.14}$$

For Model C, the complete supervised target matrix is assembled from the primary and auxiliary targets as

$$Y_C = \begin{bmatrix} Y_{C,p} & Y_{C,a} \end{bmatrix} . \tag{5.15}$$

The preparation scripts store the selected input names, output names, transformation metadata, normalization constants, and train/validation/test indices with each model package. This ensures that the same column ordering and preprocessing operations are used during training, testing, parametric sweeps, and later prediction calls.

5.2.2 Training, Validation, and Test Data Splitting

After the input and output matrices are assembled, the accepted samples are shuffled and partitioned using the 70–15–15 training, validation, and test split defined in Section 4.3.5. The split is performed after feasibility filtering so that each partition contains only accepted cycle solutions.

For the wet/dry expert models, dry and afterburning samples are partitioned within their respective operating modes before training the corresponding expert networks. This preserves the mode separation used throughout the surrogate architecture and prevents dry and augmented samples from being mixed before expert assignment.

The training partition is used to compute the normalization statistics and to update the neural-network parameters. The validation partition is used during training to monitor model behavior and select the saved checkpoint. The test partition is held out until final baseline evaluation and is used for the parity and residual plots presented in Section 5.3.

5.2.3 Network Initialization and Forward Propagation

The trained surrogate models are implemented as feedforward neural networks using normalized input and output variables. The equations in this subsection define the implementation notation used by the training scripts, including the activation functions, hidden-layer structure, parameter initialization, and forward-pass organization. The general multi-layer perceptron formulation was introduced in Section 4.1.2; the present subsection focuses on the specific architecture choices used for Models A, B, and C.

For a generic fully connected layer, the forward operation is

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)} \tag{5.16}$$

$$\boldsymbol{a}^{(l)} = g_{\text{act}}\left(\boldsymbol{z}^{(l)}\right) \tag{5.17}$$

where $\boldsymbol{W}^{(l)}$ is the weight matrix of layer l, $\boldsymbol{b}^{(l)}$ is the bias vector, $\boldsymbol{z}^{(l)}$ is the pre-activation vector, $\boldsymbol{a}^{(l)}$ is the post-activation vector, and $g_{\text{act}}(\cdot)$ is the activation function. The first activation input is $\boldsymbol{a}^{(0)} = \boldsymbol{x}^{*}$.

For the ideal-cycle Model A and Model B experts, the hidden layers use the hyperbolic tangent activation function,

$$g_{\text{act}}(z) = \tanh(z) \tag{5.18}$$

applied elementwise to the pre-activation vector. The ideal-cycle experts use relatively compact networks because their reference mappings are lower dimensional than the real-cycle Model C mapping.

The dry and wet experts use the same structural pattern, but with different hidden-layer widths depending on the model and operating mode. A residual connection is included after the first two hidden layers. If $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ are the outputs of the first and second activation layers, the residual addition is

$$\mathbf{a}^{(r)} = \mathbf{a}^{(1)} + \mathbf{a}^{(2)} \tag{5.19}$$

This residual vector is then passed through the remaining fully connected layers before reaching the final output layer. The normalized output prediction for Models A and B can therefore be written compactly as

$$\hat{\mathbf{y}}^* = \mathcal{F}_{ideal}\left(\mathbf{x}^*; \boldsymbol{\theta}\right) \tag{5.20}$$

where $\boldsymbol{\theta}$ represents all trainable weights and biases in the selected dry or wet expert.

The initialized hidden-layer widths for the ideal-cycle experts are summarized in Table 5.1. These widths were selected to provide sufficient regression capacity while keeping the ideal-cycle models lightweight relative to the full real-cycle surrogate.

Table 5.1 – Hidden-layer widths used for the Model A and Model B ideal-cycle experts

Model	Expert	Hidden-layer widths
Model A	Dry	[128, 128, 96, 64]
Model A	Wet	[160, 160, 128, 96]
Model B	Dry	[160, 160, 128, 96]
Model B	Wet	[192, 192, 160, 128]

The real-cycle Model C surrogate uses a deeper shared-trunk architecture with separate output heads for primary performance quantities and auxiliary diagnostic quantities. This structure allows the lower layers to learn a common representation of the real-cycle input state while the output heads specialize in different groups of targets. The shared trunk first maps the normalized real-cycle input vector into a learned feature vector,

$$\boldsymbol{\psi}_C = \mathcal{T}\left(\boldsymbol{x}_C^*; \boldsymbol{\theta}_T\right) \tag{5.21}$$

where $\mathcal{T}$ is the shared trunk and $\boldsymbol{\theta}_T$ contains the trunk parameters. The primary and auxiliary heads then produce separate normalized predictions:

$$\widehat{\boldsymbol{y}}_{C,p}^* = \mathcal{H}_p\left(\boldsymbol{\psi}_C; \boldsymbol{\theta}_p\right) \tag{5.22}$$

$$\widehat{\boldsymbol{y}}_{C,a}^* = \mathcal{H}_a\left(\boldsymbol{\psi}_C; \boldsymbol{\theta}_a\right) \tag{5.23}$$

where $\mathcal{H}_p$ is the primary-output head and $\mathcal{H}_a$ is the auxiliary-output head. This two-

head structure allows the real-cycle model to learn the main performance quantities and the internal diagnostic states using a common feature representation.

The Model C trunk uses five fully connected layers with widths

$$[256,\ 256,\ 192,\ 192,\ 128] \tag{5.24}$$

and each output head uses three fully connected hidden layers with widths

$$[128,\ 96,\ 64] \tag{5.25}$$

Model C uses a leaky-rectified linear activation function,

$$g_{\text{act}}(z) = \begin{cases} z, & z \geq 0 \\ 0.05z, & z < 0 \end{cases} \tag{5.26}$$

where the negative-slope value of 0.05 is an implementation choice. The Model C weights are initialized using He initialization, which was developed for networks with rectifier-type nonlinearities [24]. Biases are initialized to zero.

For all models, forward propagation is performed on mini-batches arranged in channel-batch form. If a mini-batch contains N_b samples and d_x input features, the normalized input matrix is transposed before being passed to the network:

$$\mathbf{X}_b^* \in \mathbb{R}^{d_x \times N_b} \tag{5.27}$$

This format is used consistently during training, validation, and test-set prediction.

The forward pass returns normalized predictions. Dimensional or nondimensional cycle quantities are recovered only after inference using the stored output normalization parameters. This keeps training and gradient evaluation in the normalized coordinate system while ensuring that validation plots, consistency checks, and parametric sweep comparisons are performed in physical output coordinates.

5.2.4 Backpropagation and Parameter Updates

The training scripts update the network parameters using mini-batch gradient-based optimization. For each mini-batch, the network performs a forward pass, evaluates the normalized training loss, computes gradients with respect to the trainable parameters, and updates those parameters using the Adam optimizer [25]. The loss function used in each case follows the normalized multi-output loss framework defined in Section 4.1.4, with model-specific output weighting where applicable.

For a mini-batch containing N_b samples, the normalized input and target matrices are passed through the network to produce normalized predictions. The mini-batch loss is eval-

uated as

$$\mathcal{L}_b = \mathcal{L}\left(\hat{\mathbf{Y}}_b^*, \mathbf{Y}_b^*; \boldsymbol{\theta}\right) \tag{5.28}$$

where $\hat{\mathbf{Y}}_b^*$ is the normalized network prediction, $\mathbf{Y}_b^*$ is the normalized reference output, and $\boldsymbol{\theta}$ contains all trainable weights and biases.

The gradient of the mini-batch loss with respect to the network parameters is computed using backpropagation:

$$\mathbf{g}_b = \nabla_{\boldsymbol{\theta}} \mathcal{L}_b \tag{5.29}$$

In the MATLAB implementation, this gradient calculation is performed using automatic differentiation through the forward pass and loss-function evaluation. This allows the same training framework to handle the ideal-cycle experts used for Models A and B and the multi-output-head architecture used for Model C.

For training iteration k, Adam forms exponentially weighted first- and second-moment estimates of the gradient:

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1 - \beta_1)\, \mathbf{g}_k \tag{5.30}$$

$$\mathbf{v}_k = \beta_2 \mathbf{v}_{k-1} + (1 - \beta_2)\, \mathbf{g}_k^2 \tag{5.31}$$

where $\mathbf{m}_k$ is the first-moment estimate, $\mathbf{v}_k$ is the second-moment estimate, β_1 and β_2 are the Adam decay coefficients, and $\mathbf{g}_k^2$ denotes the elementwise squared gradient. Bias-corrected moment estimates are then formed as

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_k}{1 - \beta_1^k} \tag{5.32}$$

$$\hat{\mathbf{v}}_k = \frac{\mathbf{v}_k}{1 - \beta_2^k} \tag{5.33}$$

The parameter update is then

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \zeta_k \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{v}}_k} + \epsilon} \tag{5.34}$$

where ζ_k is the optimizer step size at iteration k, and ϵ is a small numerical stabilization constant.

A scheduled step size is used to reduce the update magnitude as training progresses. The general form of the schedule is

$$\zeta_n = \zeta_0 \delta_{\mathrm{lr}}^{\lfloor n/E_d \rfloor} \tag{5.35}$$

where ζ_0 is the initial optimizer step size, δ_{lr} is the decay factor, E_d is the decay interval in epochs, and n is the current epoch index. For Models A and B, a minimum step size was

enforced:

$$\zeta_n = \max\left[\zeta_{\min}, \zeta_0 \delta_{\mathrm{lr}}^{\lfloor n/E_d \rfloor}\right] \tag{5.36}$$

The training hyperparameters used for the surrogate experts are summarized in Table 5.2, where N_b is the mini-batch size, ζ_0 is the initial optimizer step size, δ_{lr} is the decay factor, E_d is the decay interval in epochs, and $\zeta_{\min}$ is the minimum step size when enforced.

Table 5.2 – Optimization and parameter-update settings used during training

Model	Expert	Epochs	N_b	ζ_0	δ_{lr}	E_d	$\zeta_{\min}$
A	Dry	250	512	1.0×10^{-3}	0.5	25	1.0×10^{-5}
A	Wet	100	512	1.0×10^{-3}	0.5	25	1.0×10^{-5}
B	Dry	250	512	1.0×10^{-3}	0.5	25	1.0×10^{-5}
B	Wet	250	512	1.0×10^{-3}	0.5	25	1.0×10^{-5}
C	Dry	250	4096	3.0×10^{-4}	0.3	60	–
C	Wet	250	4096	3.0×10^{-4}	0.3	60	–

The smaller mini-batch size used for Models A and B is appropriate for the lower-dimensional ideal-cycle datasets. Model C uses a larger mini-batch size because the real-cycle datasets contain substantially more accepted samples and a higher-dimensional input-output mapping. In all cases, the optimizer updates the dry and wet experts independently, so gradients from one operating mode do not directly affect the trainable parameters of the other expert. During training, validation loss is evaluated at the epoch level. The validation data are not used for parameter updates; instead, they provide an independent measure of generalization during optimization and are used to retain the best-performing checkpoint for later test-set evaluation.

5.2.5 Training Loss and Validation Loss Monitoring

Training and validation loss histories were monitored throughout optimization for all dry and wet experts. These histories provide an implementation-level record of whether each network learned the sampled cycle mapping and whether validation behavior remained consistent with the training response. For each epoch, the mean training loss was computed from the mini-batches processed during parameter updates, while a separate validation loss was evaluated using the corresponding validation subset.

If the mini-batch loss for batch b in epoch n is denoted by $\mathcal{L}_{b,n}$, then the reported epoch training loss is

$$\mathcal{L}_{train}^{(n)} = \frac{1}{B_n} \sum_{b=1}^{B_n} \mathcal{L}_{b,n} \tag{5.37}$$

where B_n is the number of mini-batches processed during epoch n. The validation loss is

evaluated by passing the validation data through the current network without parameter updates:

$$\mathcal{L}_{val}^{(n)} = \mathcal{L}\left(\hat{\mathbf{Y}}_{val}^{*}, \mathbf{Y}_{val}^{*}\right) \tag{5.38}$$

where $\hat{\mathbf{Y}}_{val}^{*}$ is the normalized prediction on the validation subset and $\mathbf{Y}_{val}^{*}$ is the corresponding normalized target matrix.

The validation history is used as an independent training diagnostic because it is evaluated on samples that are not used for parameter updates. A consistent decrease or plateau in both training and validation loss indicates stable optimization, while a divergence between the two histories can indicate overfitting or loss of generalization within the sampled dataset.

For Models A and B, the training scripts monitored the validation loss during optimization and used a patience-based criterion to terminate training when the validation loss failed to improve for a specified number of epochs. In these models, the patience value was set to 20 epochs. This allowed the ideal-cycle experts to stop once validation performance ceased improving, rather than continuing to reduce the training loss without a corresponding improvement in validation behavior.

For Model C, the validation loss was also evaluated at each epoch, but the final training runs did not use strict early stopping. Instead, training was allowed to proceed through the full epoch schedule while the best-performing validation checkpoint was retained. Therefore, Model C used validation monitoring for model selection rather than for hard termination of the training loop.

The training and validation histories are presented later in this chapter for each model and operating mode. Since the dry and wet experts are trained independently, separate loss-history plots are reported for each expert. These plots are treated as baseline training diagnostics rather than final proof of physical reliability. Final model assessment also includes test-set comparisons, parity plots, residual plots, and the parametric sweep studies presented in Chapter 6.

5.2.6 Model Storage and Prediction Workflow

After training, each surrogate expert is stored as a MATLAB model artifact so that prediction can be performed without rerunning the dataset generator or training script. The stored artifacts contain the trained network, selected input and output names, normalization constants, transformation metadata, training history where applicable, and validation-checkpoint information. This allows the trained surrogates to be reused consistently for test-set evaluation, parametric sweeps, extrapolation studies, and runtime benchmarking.

For Models A and B, the final deployable files store the trained network and a `modelInfo` structure. This structure contains the column definitions, scaling constants, transformation rules, architecture information, and best validation information needed for repeatable prediction. Separate learnable-parameter files are also saved for inspection, but the deployable

files are the primary artifacts used by the prediction scripts.

For Model C, the best validation checkpoint is stored separately for the dry and wet experts. Each checkpoint contains the trained network, model metadata, training history, and the model specification structure. The specification structure stores the data-package information required to reproduce the same input transformations and output scaling used during training.

The main deployment artifacts are summarized in Table 5.3.

Table 5.3 – Stored model artifacts used for prediction

Model	Expert	Primary prediction artifact
Model A	Dry	`ideal_dry_model_A_deploy.mat`
Model A	Wet	`ideal_wet_model_A_deploy.mat`
Model B	Dry	`ideal_dry_model_B_deploy.mat`
Model B	Wet	`ideal_wet_model_B_deploy.mat`
Model C	Dry	`real_dry_model_C_best.mat`
Model C	Wet	`real_wet_model_C_best.mat`

The prediction workflow begins with a new physical input point or batch of input points. The prediction script verifies that the required input variables are present and finite, applies the same transformations used during training, and then normalizes the transformed inputs using the stored training-set statistics. For the ideal-cycle Models A and B, this includes the logarithmic pressure-ratio transformations applied to π_c and π_f. For Model C, the prediction script applies the transformation rules stored in the model metadata.

The transformed and normalized inputs are then passed through the trained network corresponding to the selected expert. The network returns normalized predictions, which are converted back to dimensional or nondimensional cycle coordinates using the stored output normalization constants. If an output transformation was used during training, the inverse transformation is applied before the result is returned. The final prediction output is packaged as tables and matrices containing the predicted primary quantities, auxiliary quantities, and diagnostic or consistency-check quantities where applicable.

For each full model, a unified routed predictor is used above the individual dry and wet expert predictors. The routed predictor reads the afterburner mode indicator m_{AB}, separates the input rows by operating mode, evaluates the corresponding expert, and recombines the predicted rows into the original input order. For Model C, rows with $m_{AB} < 0.5$ are assigned to the dry expert, while rows with $m_{AB} \geq 0.5$ are assigned to the wet expert.

This workflow ensures that all predictions use the same scaling, transformation, and dry/wet routing assumptions used during training. As a result, the stored surrogate models can be evaluated consistently across the test-set comparison scripts, parametric sweep studies, extrapolation studies, and runtime benchmark scripts.

5.3 Model-Specific Training and Baseline Validation

The following sections defined the common surrogate architecture, data preparation workflow, loss formulation, normalization process, and prediction pipeline. The model-specific discussion is therefore limited to the training behavior and baseline validation results for each surrogate family. This avoids repeating the common implementation details already established and keeps the focus on whether each model stage successfully learned its corresponding cycle mapping.

For each model family, the dry and wet experts are assessed using training history plots, test-set parity plots, residual plots, and consistency-check plots where applicable. These figures provide a first verification that the trained networks reproduce the cycle-analysis outputs on withheld test data. The broader physical interpretation of the trained models, including parametric sweeps, extrapolation behavior, and runtime benchmarking, is reserved for Chapter 6.

5.3.1 Model A: Fixed-Property Ideal-Cycle Surrogate

Model A represents the fixed-property ideal-cycle surrogate. This model was used as the baseline proof-of-capability case for the neural network framework, since the gas properties and fuel heating value were held constant while the primary cycle variables were varied. Separate dry and wet experts were trained, allowing the non-afterburning and afterburning mappings to be learned independently.

The dry expert training history is shown in Fig. 5.1. The training and validation losses decrease rapidly during the early epochs and then approach a low-loss plateau. The close tracking between the training and validation curves indicates that the dry expert learned the ideal-cycle mapping without a large separation between the fitted training data and the held-out validation data.

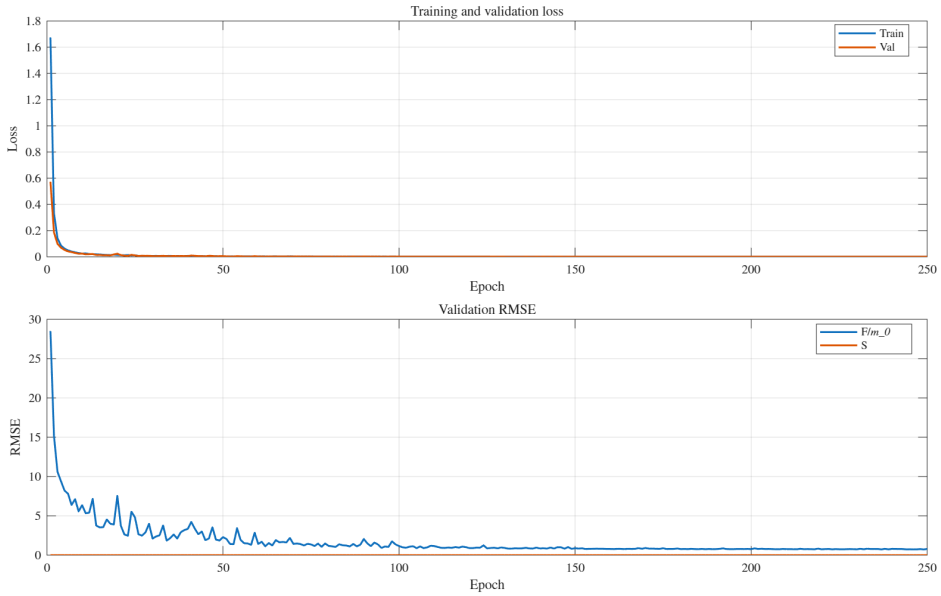


Figure 5.1 – Model A dry expert training and validation loss history

The wet expert training history is shown in Fig. 5.2. The afterburning case introduces an additional fuel-addition mode and a larger range of exit thermodynamic states, but the same general convergence behavior is observed. The validation loss decreases with the training loss and remains stable after the initial rapid reduction. This suggests that the wet expert also learned a generalizable mapping over the accepted afterburning operating envelope.

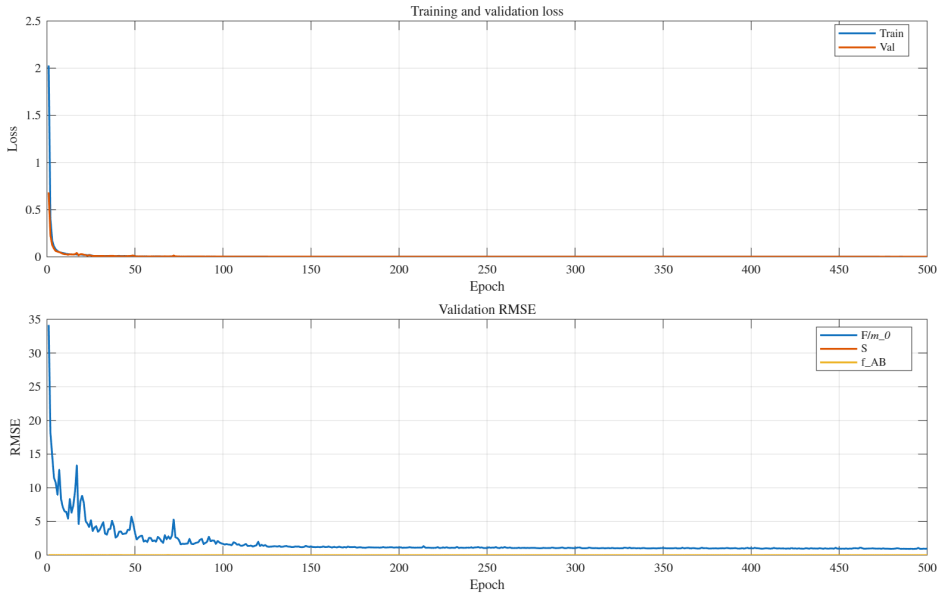


Figure 5.2 – Model A wet expert training and validation loss history

The dry expert parity plots are shown in Fig. 5.3. The predicted values for specific thrust,

TSFC, bypass ratio, and main-burner fuel-air ratio lie closely along the one-to-one reference line. This indicates that the fixed-property dry ideal-cycle mapping is learned accurately across the withheld test samples. The bypass-ratio prediction shows slightly more visible scatter than the other dry-output quantities, but the overall trend remains strongly aligned with the reference cycle-analysis solution.

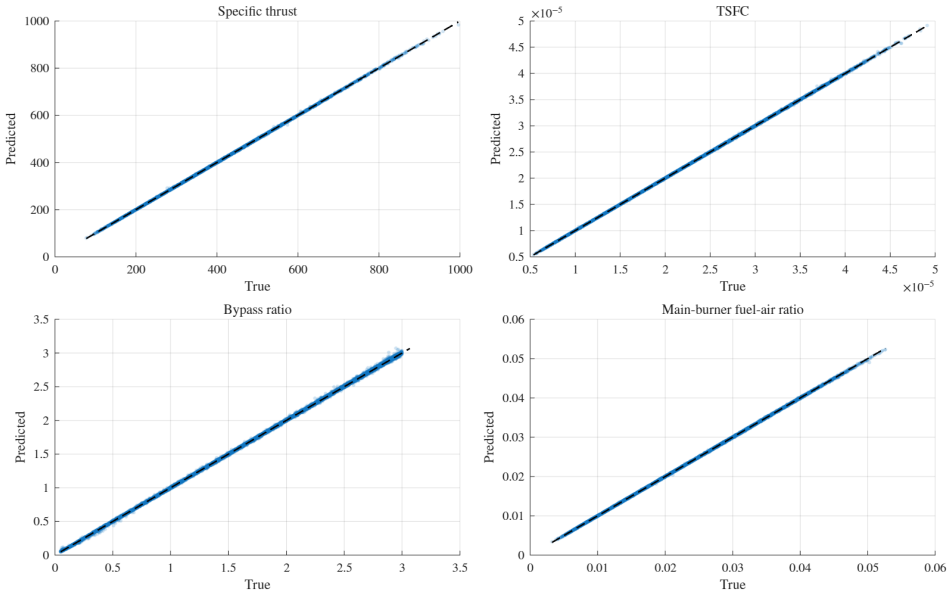


Figure 5.3 – Model A dry expert test-set parity plots

The wet expert parity plots are shown in Fig. 5.4. The afterburning expert also shows strong agreement between predicted and reference quantities. Specific thrust, TSFC, bypass ratio, main-burner fuel-air ratio, afterburner fuel-air ratio, and the mixed-stream total temperature diagnostic remain close to the one-to-one line. This confirms that the wet expert captures both the baseline ideal-cycle mapping and the additional response associated with afterburner heat addition.

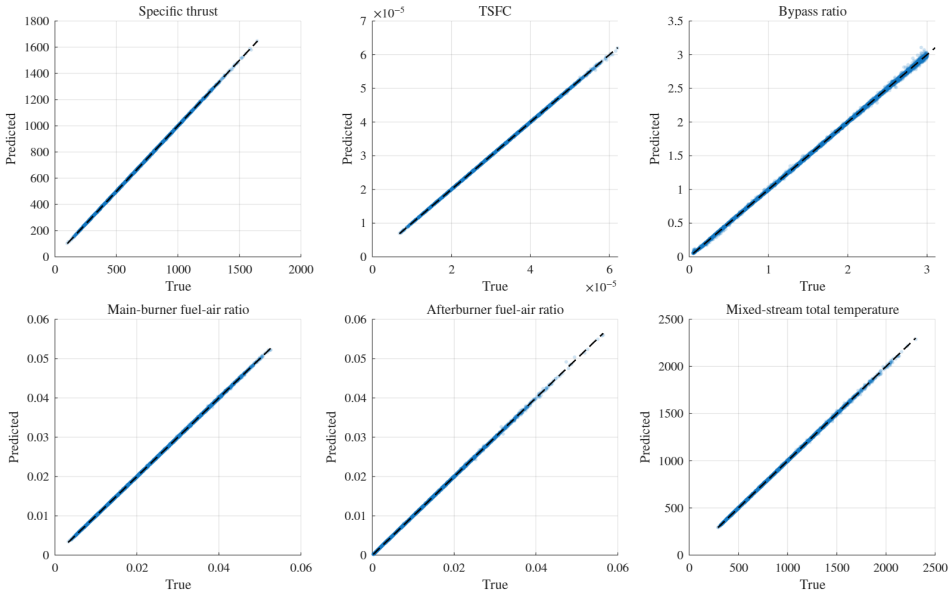


Figure 5.4 – Model A wet expert test-set parity plots

The dry expert residual plots are shown in Fig. 5.5. The residuals for specific thrust and TSFC are generally centered near zero over the tested ranges of freestream Mach number, compressor pressure ratio, and burner temperature ratio. Some increase in residual scatter appears near the higher-demand portions of the envelope, especially at larger freestream Mach number and high burner temperature ratio. This behavior is expected for the regions where the cycle outputs vary more strongly with the inputs.

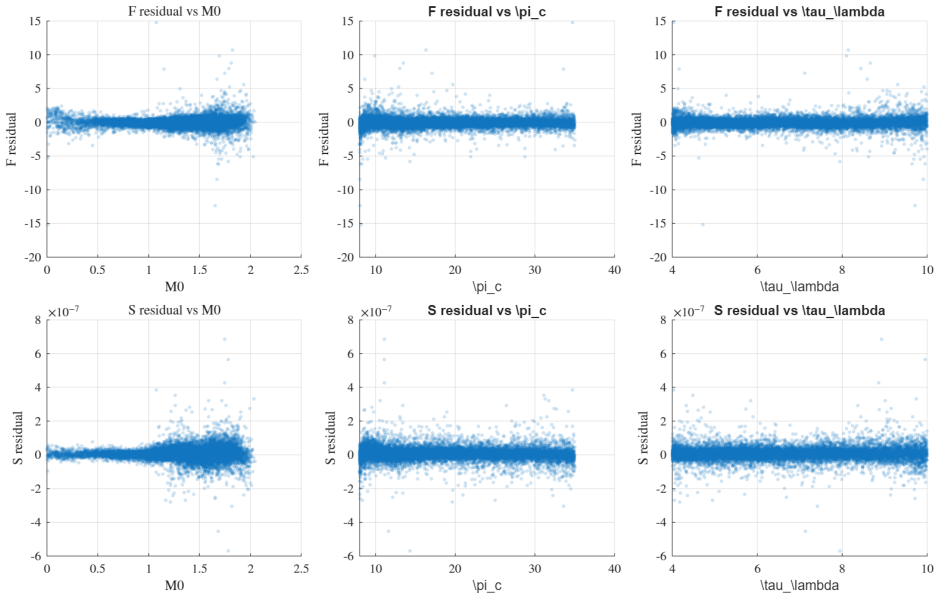


Figure 5.5 – Model A dry expert test-set residual plots

The wet expert residual scatter plots are shown in Fig. 5.6. The residuals remain centered near zero for the primary afterburning outputs over the tested Mach number, compressor pressure ratio, burner temperature ratio, and afterburner temperature-ratio ranges. As with the dry expert, the scatter increases slightly in portions of the envelope associated with higher thermal loading and larger thrust response. The residual structure does not show a dominant systematic bias across the plotted input variables.

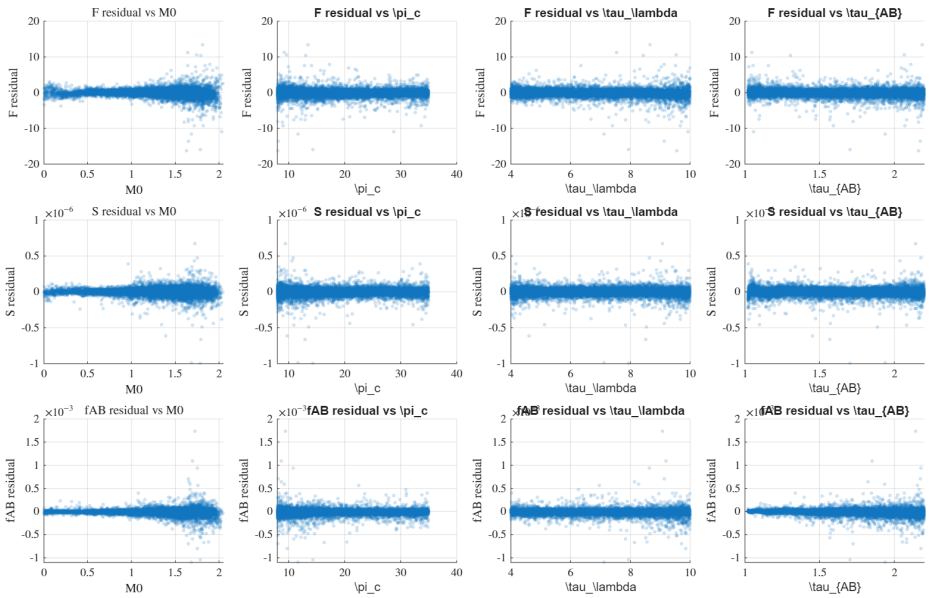


Figure 5.6 – Model A wet expert test-set residual plots

The dry expert consistency checks are shown in Fig. 5.7. These plots compare the predicted values against algebraic consistency checks for total fuel-air ratio and overall efficiency. The close alignment with the one-to-one reference line indicates that the dry expert predictions preserve the expected relationships between f_0, η_T, η_P, and η_O after denormalization.

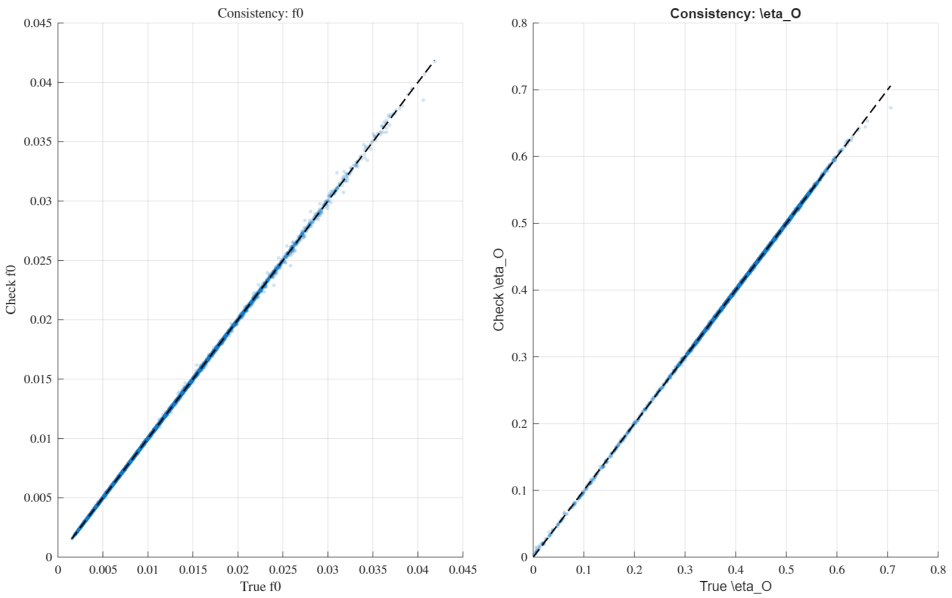


Figure 5.7 – Model A dry expert test-set consistency checks

The wet expert consistency checks are shown in Fig. 5.8. The afterburning expert also maintains close agreement between the predicted quantities and the reconstructed consistency quantities. This is important because the wet case includes both main-burner and afterburner fuel addition, so the total fuel-air-ratio relationship is more demanding than in the dry case.

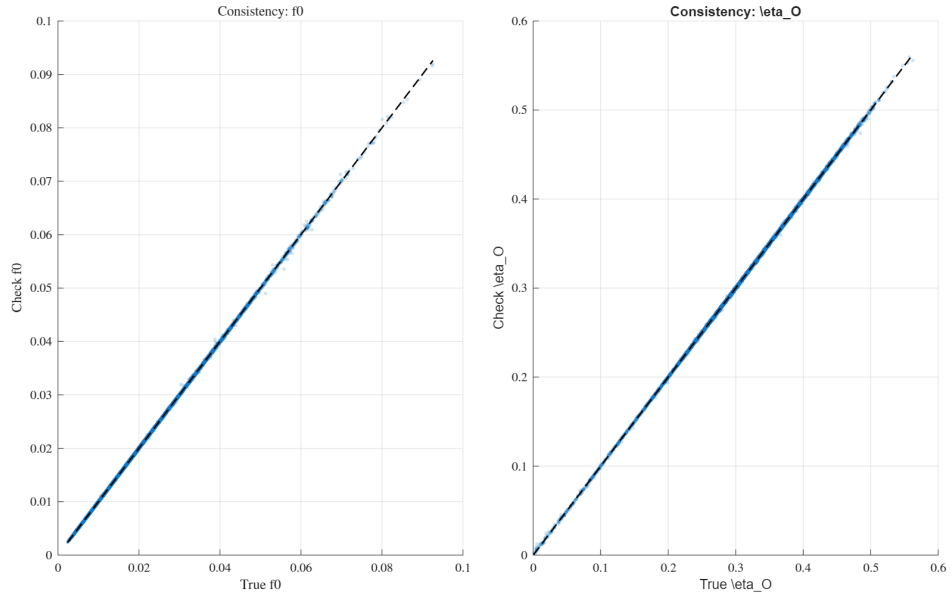


Figure 5.8 – Model A wet expert test-set consistency checks

Overall, the Model A results confirm that the selected neural network framework can

reproduce the fixed-property ideal mixed-flow afterburning turbofan mapping with high accuracy on withheld test data. The training histories show stable convergence, the parity plots show close agreement with the reference cycle solver, the residual plots remain centered near zero, and the consistency plots verify that the principal algebraic relationships are preserved. Model A therefore establishes the baseline capability of the surrogate approach before thermochemical variability and real-cycle losses are introduced in Models B and C.

5.3.2 Model B: Variable-Thermochemistry Ideal-Cycle Surrogate

Model B extends the ideal-cycle surrogate framework by activating the thermochemical input dimensions that were fixed in Model A. The ideal mixed-flow afterburning turbofan equations are retained, but γ and h_{PR} vary across the dataset, while c_p is derived consistently from the sampled gas properties. This makes Model B an intermediate stage between the fixed-property ideal-cycle surrogate and the full real-cycle surrogate.

The dry expert training history is shown in Fig. 5.9. The training and validation losses decrease sharply during the early portion of training and then settle into a low-loss region. The validation RMSE for specific thrust shows larger early-epoch variation than the TSFC validation RMSE, which is expected because specific thrust has a larger dimensional range and stronger dependence on the sampled operating condition. After the initial transient, the validation behavior remains stable.

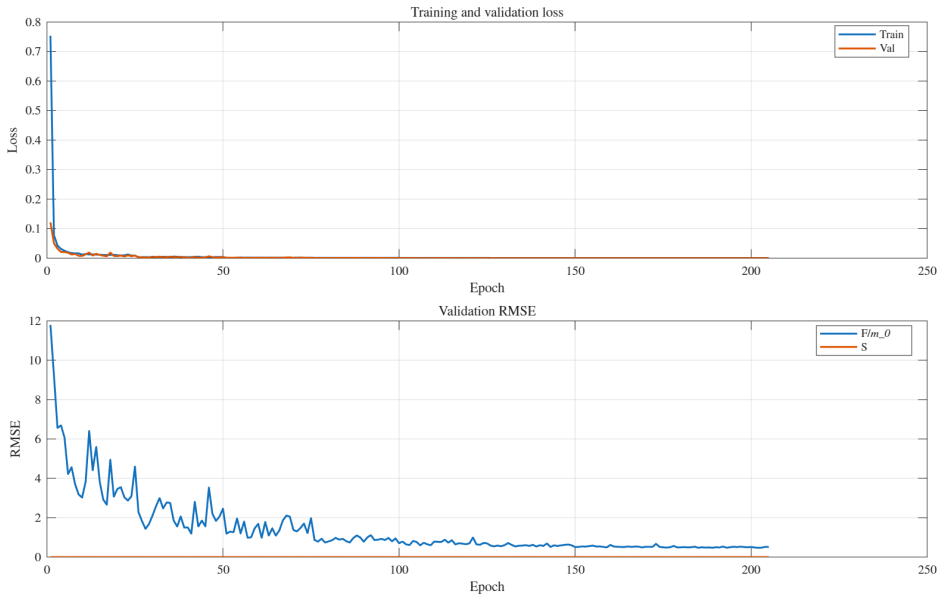


Figure 5.9 – Model B dry expert training and validation loss history

The wet expert training history is shown in Fig. 5.10. The same overall convergence trend is observed for the afterburning expert. The added afterburner fuel-air ratio output introduces another supervised quantity, but the loss history still shows rapid initial convergence followed

by a stable low-loss plateau. This indicates that the wet Model B expert can learn the variable-property afterburning ideal-cycle mapping without unstable training behavior.

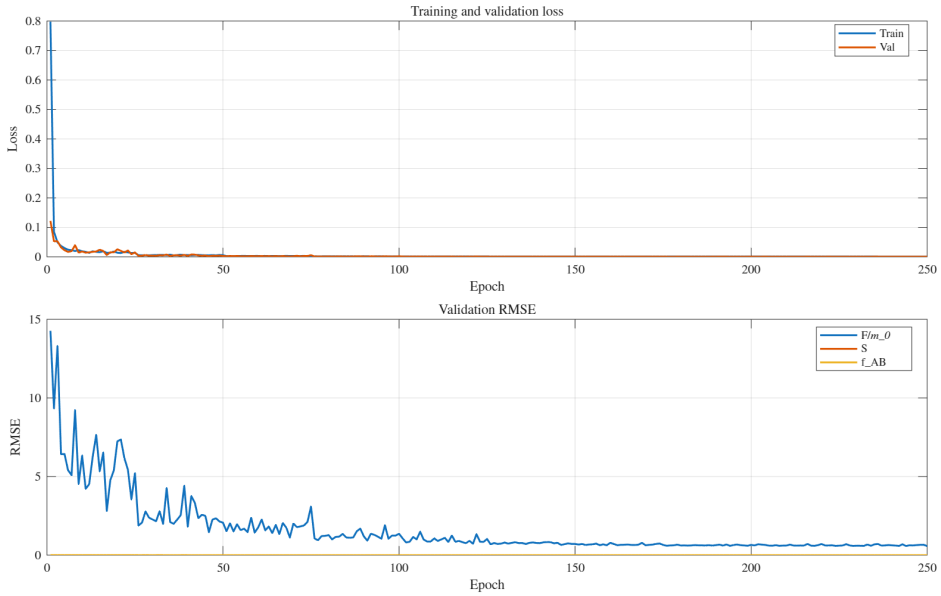


Figure 5.10 – Model B wet expert training and validation loss history

The dry expert parity plots are shown in Fig. 5.11. The predicted values for specific thrust, TSFC, bypass ratio, and main-burner fuel-air ratio remain closely aligned with the one-to-one reference line. This confirms that the dry expert learned the ideal-cycle mapping even after thermochemical variation was introduced. Compared with Model A, this is a more demanding regression problem because the network must account for the effect of changing γ, c_p, and h_{PR} rather than treating them as fixed constants.

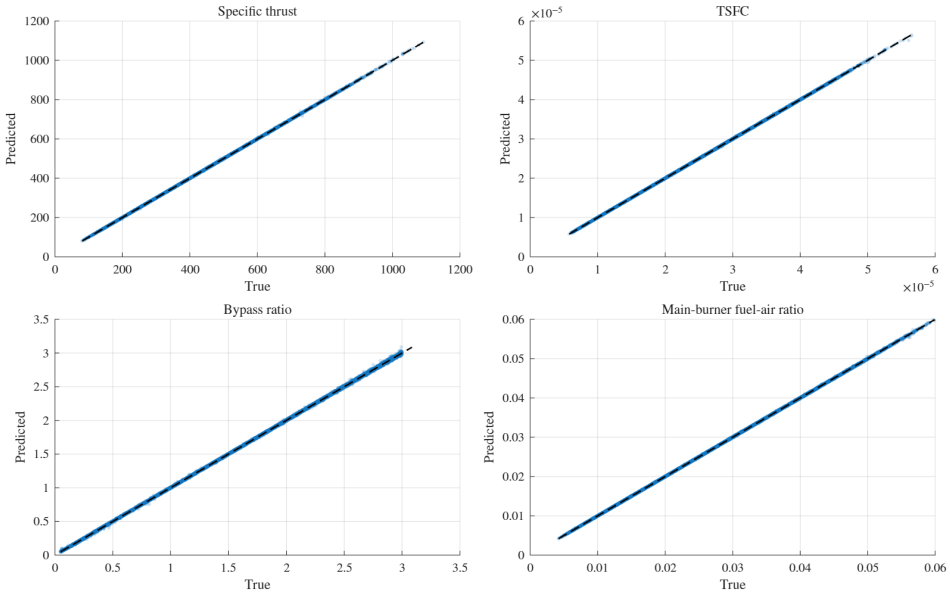


Figure 5.11 – Model B dry expert test-set parity plots

The wet expert parity plots are shown in Fig. 5.12. The wet Model B expert shows strong agreement for specific thrust, TSFC, bypass ratio, main-burner fuel-air ratio, afterburner fuel-air ratio, and mixed-stream total temperature. The afterburner fuel-air ratio and mixed-stream total temperature predictions are especially important because they confirm that the model captures the additional heating response while also accounting for gas-property variation.

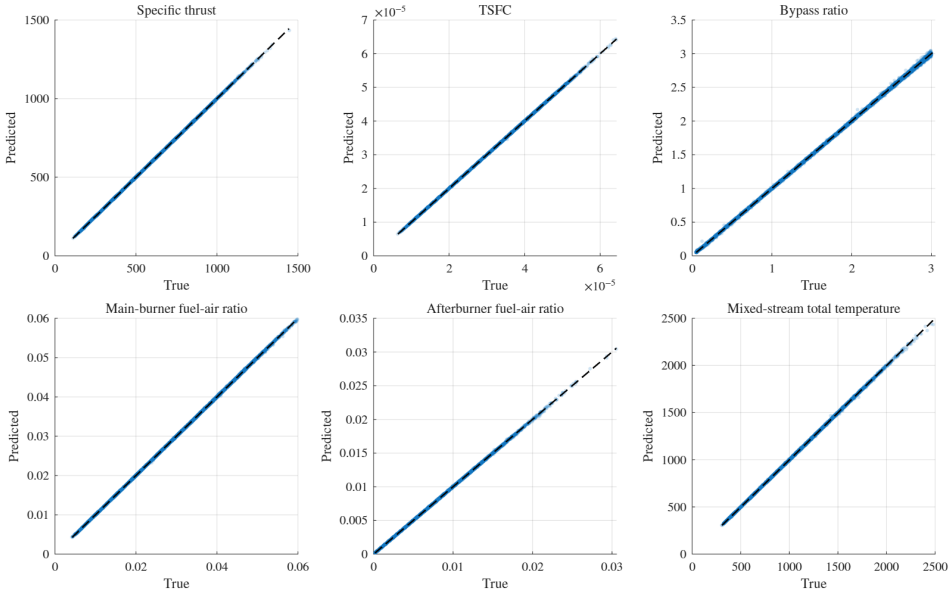


Figure 5.12 – Model B wet expert test-set parity plots

The dry expert residual plots are shown in Fig. 5.13. The residuals for specific thrust

and TSFC remain centered near zero over the plotted ranges of freestream Mach number, compressor pressure ratio, burner temperature ratio, γ, and h_{PR}. Some increase in scatter is visible at higher Mach number and higher burner temperature ratio, where the cycle response changes more strongly with the inputs. The absence of a strong residual drift with γ or h_{PR} indicates that the thermochemical dimensions were learned without introducing a dominant systematic bias.

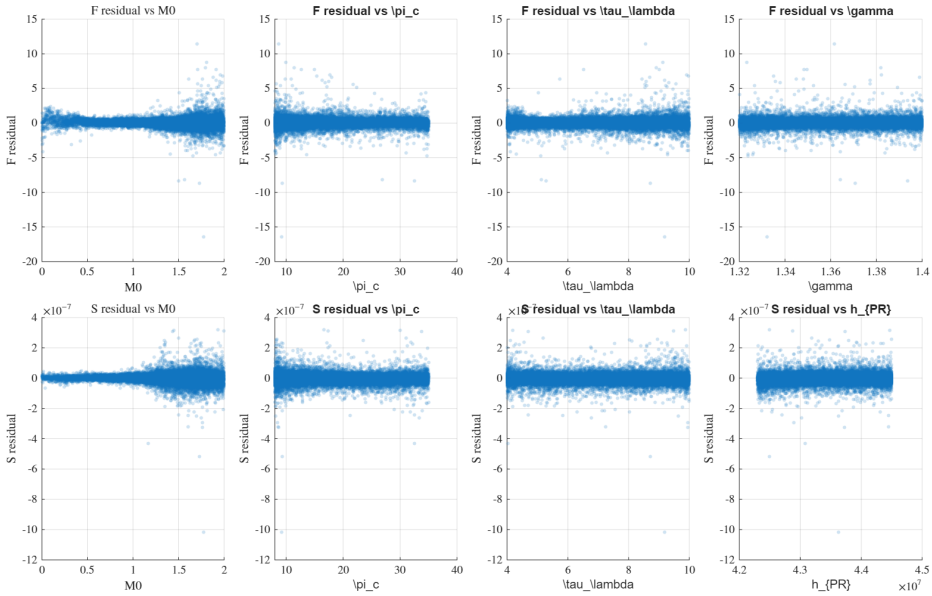


Figure 5.13 – Model B dry expert test-set residual plots

The wet expert residual plots are shown in Fig. 5.14. The residuals remain distributed around zero for the primary afterburning outputs. The afterburner fuel-air ratio residuals show increased scatter near the higher afterburner temperature-ratio region, which is expected because the wet mapping depends directly on the additional heat addition imposed downstream of the mixer. Even with this increased scatter, the residuals do not show a dominant monotonic bias across the plotted operating variables.

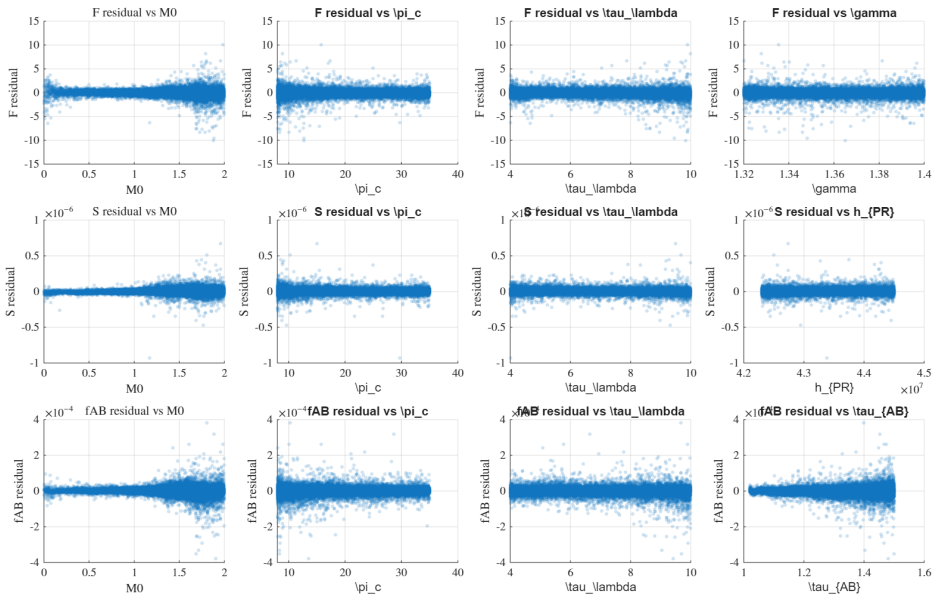


Figure 5.14 – Model B wet expert test-set residual plots

The dry expert consistency checks are shown in Fig. 5.15. The predicted total fuel-air ratio and overall efficiency are compared against values reconstructed from the corresponding predicted components. The close one-to-one agreement indicates that the dry Model B predictions preserve the principal algebraic consistency relationships after denormalization.

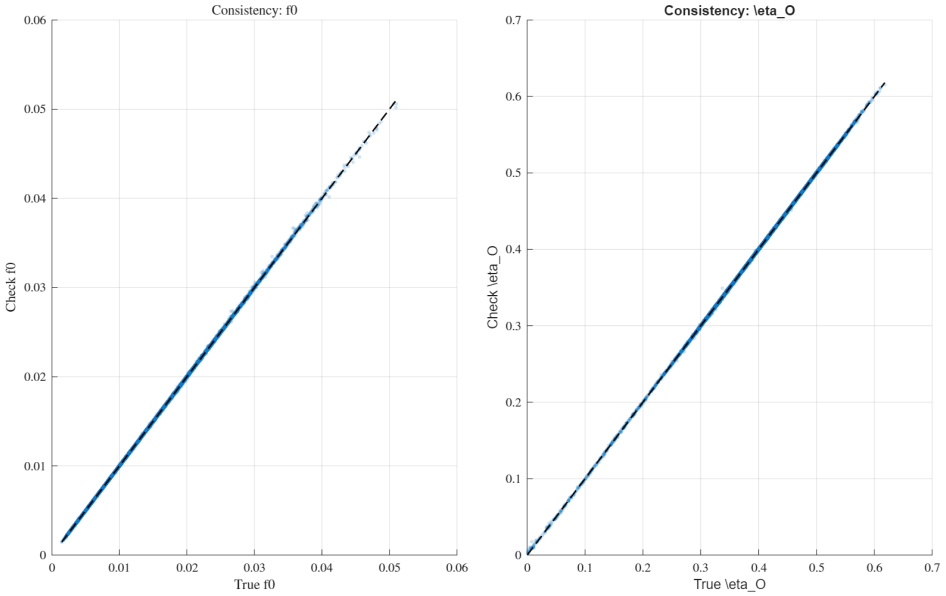


Figure 5.15 – Model B dry expert test-set consistency checks

The wet expert consistency checks are shown in Fig. 5.16. The total fuel-air-ratio consistency check is more demanding for the wet expert because both f and f_{AB} contribute to the

total fuel-air ratio. The close agreement shown in the figure indicates that the wet expert preserves this relationship while also maintaining consistency between η_T, η_P, and η_O.

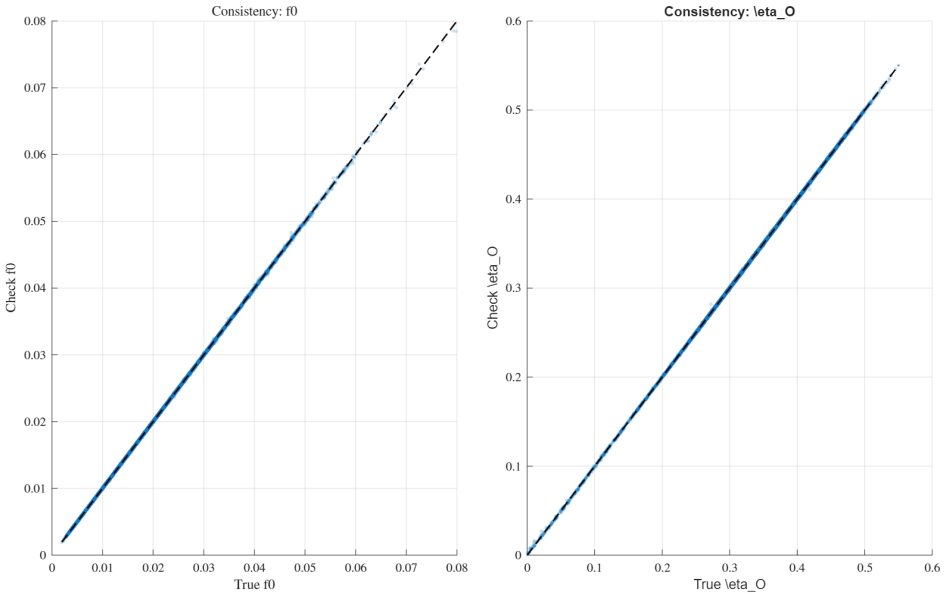


Figure 5.16 – Model B wet expert test-set consistency checks

Overall, the Model B results show that the same neural network framework can learn the ideal mixed-flow afterburning turbofan mapping when thermochemical variation is introduced. The training histories show stable convergence, the parity plots show close agreement with the cycle-analysis reference data, the residual plots remain centered near zero, and the consistency plots confirm that the main algebraic relationships are retained. Model B therefore provides the intermediate validation step needed before moving from ideal-cycle surrogates to the high-dimensional real-cycle formulation used in Model C.

5.3.3 Model C: Real-Cycle Surrogate

Model C is the full real-cycle surrogate developed in this project. Unlike Models A and B, Model C includes component losses, polytropic efficiencies, burner and afterburner efficiencies, mixer pressure recovery, nozzle loss, mode-dependent gas properties, and auxiliary internal-state outputs. This makes Model C the highest-dimensional and most physically complete surrogate stage.

The dry expert training history is shown in Fig. 5.17. The total training and validation losses decrease rapidly during the early epochs and then continue decreasing more gradually over the remainder of training. The validation RMSE curves for the selected primary outputs also decrease smoothly, indicating that the dry expert learns the real-cycle mapping without unstable divergence between the training and validation data. The bypass-ratio output has the largest early validation RMSE, which is expected because α is strongly coupled to the compressor, fan, turbine, and mixer matching relations in the real-cycle formulation.

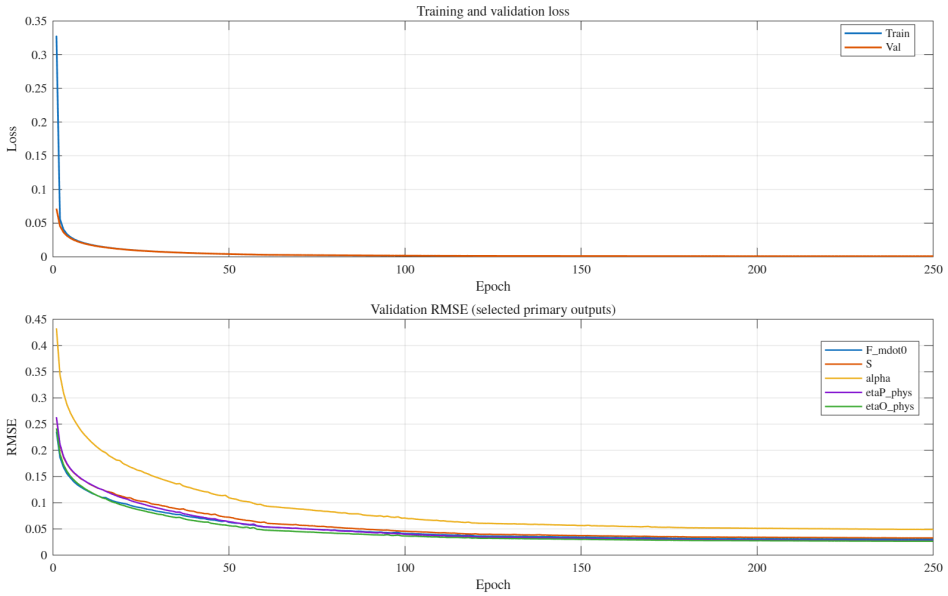


Figure 5.17 – Model C dry expert training and validation loss history

The wet expert training history is shown in Fig. 5.18. The wet expert follows the same general convergence pattern, with rapid early loss reduction followed by a stable low-loss region. The validation RMSE curves show that the network learns the major afterburning performance quantities while accounting for the additional dependence on afterburner heat addition, afterburner efficiency, afterburner pressure loss, and the wet-mode gas-property inputs. The TSFC validation RMSE remains one of the slower quantities to settle, which is reasonable because TSFC depends on the combined prediction of thrust and fuel-flow quantities.

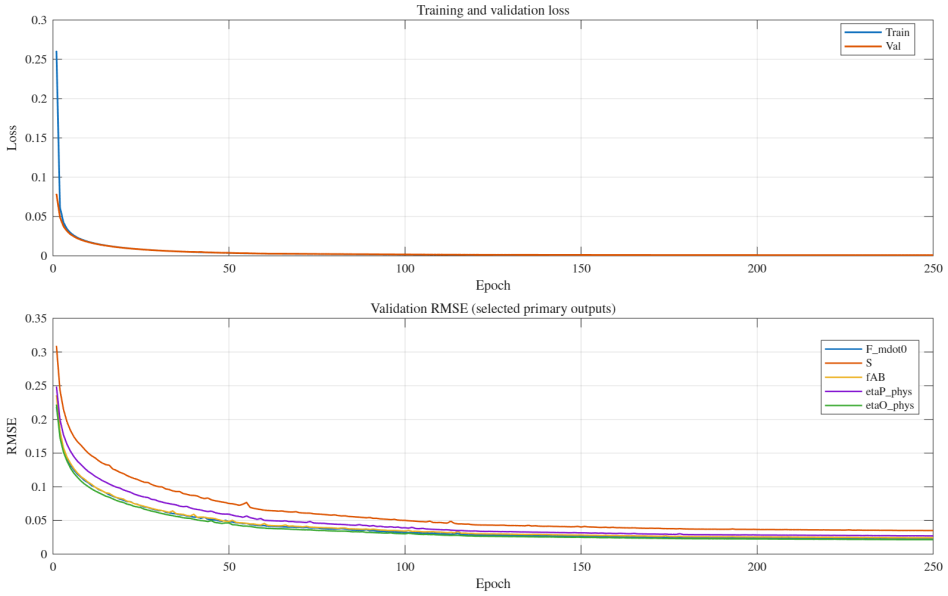


Figure 5.18 – Model C wet expert training and validation loss history

The combined Model C test-set parity and error-distribution plots are shown in Fig. 5.19. Unlike the Model A and Model B figures, the Model C baseline validation plot combines the dry and wet expert test-set predictions into a single routed comparison. This reflects the final prediction workflow, where the afterburner mode indicator assigns each input row to the appropriate expert and the outputs are recombined for evaluation.

The parity plots show close agreement between the predicted and reference cycle-analysis values for specific thrust, TSFC, bypass ratio, physical propulsive efficiency, physical overall efficiency, nozzle exit Mach number, and nozzle area ratio. The agreement for M_9 and A_9/A_t is especially important because these are auxiliary internal-state quantities rather than only final performance metrics. Their accurate prediction indicates that the Model C surrogate is not merely fitting thrust and TSFC, but is also learning cycle states associated with the nozzle expansion process.

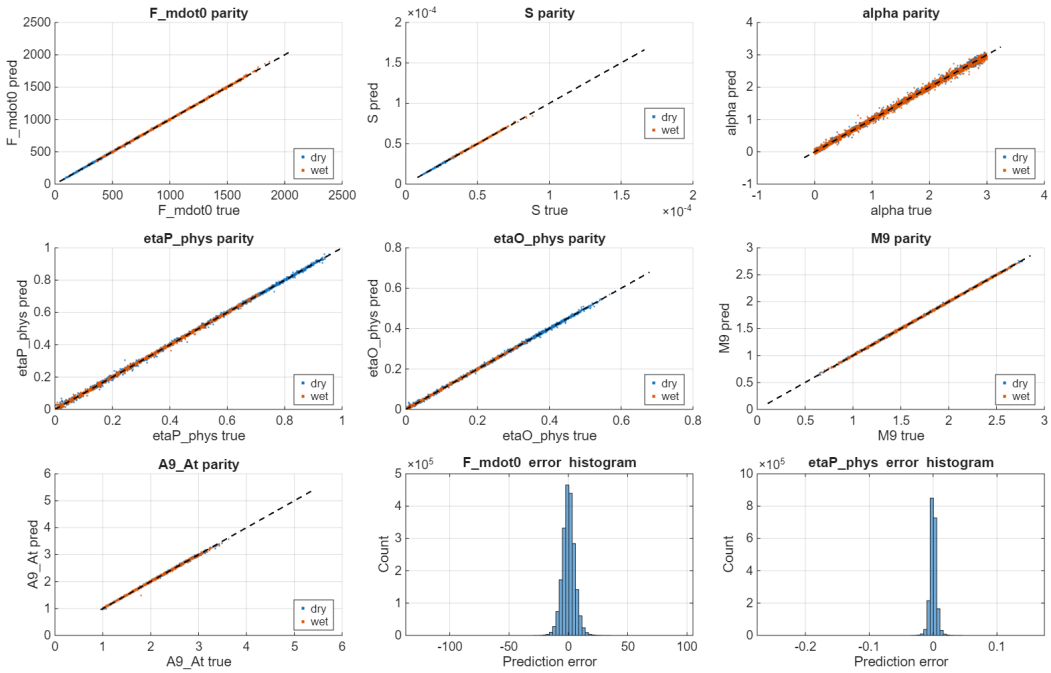


Figure 5.19 – Model C combined dry and wet test-set parity and error-distribution plots

The largest visible scatter occurs in the bypass-ratio and low-efficiency regions. This is expected because Model C solves a more difficult regression problem than the ideal-cycle models. The real-cycle bypass ratio depends on component efficiencies, pressure losses, turbine work extraction, and mixer compatibility, while the efficiency quantities depend on several coupled predicted outputs. Even with this increased complexity, the predicted values remain closely distributed around the one-to-one reference trends.

The error histograms included in Fig. 5.19 provide a compact residual summary for the combined routed test set. The specific thrust and physical propulsive-efficiency errors are sharply concentrated near zero, indicating that the dominant prediction errors are small and centered. This supports the conclusion that the Model C surrogate does not exhibit a large

systematic bias over the combined dry and wet validation set.

Overall, the Model C results show that the surrogate framework scales from the ideal-cycle cases to the full real-cycle formulation. The dry and wet training histories demonstrate stable convergence, while the combined test-set parity plots show that the routed Model C predictor reproduces both primary performance quantities and selected auxiliary cycle states. Model C therefore serves as the final forward surrogate used for the broader parametric sweep, extrapolation, and runtime studies discussed in Chapter 6.

5.4 Chapter Summary

This chapter presented the development and baseline validation of the neural network surrogate models used in this project. The model hierarchy was introduced as a staged progression from a fixed-property ideal-cycle surrogate, to a variable-thermochemistry ideal-cycle surrogate, and finally to a full real-cycle surrogate. This staged approach allowed the neural network framework to be tested first on controlled ideal-cycle mappings before being applied to the higher-dimensional real-cycle problem.

The common training workflow was then described, including dataset preparation, dry/wet expert separation, training-validation-test splitting, normalization, network initialization, forward propagation, backpropagation, parameter updates, validation monitoring, model storage, and prediction-time routing. These shared implementation details provide the foundation for all three model families and ensure that differences in model behavior can be interpreted primarily as consequences of increasing physical complexity.

The final section presented the model-specific training and baseline validation results for Models A, B, and C. Model A verified that the selected neural network framework could reproduce the fixed-property ideal mixed-flow afterburning turbofan mapping. Model B showed that the same framework could handle thermochemical variation within the ideal-cycle formulation. Model C demonstrated that the surrogate approach could be extended to the full real-cycle formulation while predicting both primary performance quantities and selected auxiliary internal states.

The training histories, parity plots, residual plots, and consistency checks showed that the trained models learned their corresponding cycle-analysis mappings with stable convergence and reasonable generalization to withheld test data. These baseline results establish the trained forward surrogates as suitable candidates for the broader validation studies presented in Chapter 6, where their behavior is evaluated through parametric sweeps, extrapolation studies, runtime comparisons, and physical trend analysis.

Chapter 6: Results and Analysis

6.1 Results and Validation Overview

This chapter presents the main validation and analysis results for the trained surrogate models. The purpose of the chapter is not only to show that the models produce low test-set error, but also to examine whether the trained surrogates reproduce physically reasonable cycle trends across the operating envelope. The analysis therefore focuses on parametric sweep behavior, extrapolation behavior, runtime performance, and the practical implications of using the trained networks as forward prediction tools.

The baseline test-set results presented in Chapter 5 showed that Models A, B, and C learned their corresponding dry and afterburning cycle mappings with reasonable accuracy on withheld samples. Chapter 6 extends that validation by testing the trained models in structured studies. Instead of evaluating random test points only, each study varies selected engine or flight parameters while holding the remaining inputs fixed at feasible reference conditions. This makes the predicted trends easier to interpret and allows direct comparison between the neural network surrogate and the deterministic parametric cycle analysis solver.

The first major validation study is the parametric sweep study. For each model and operating mode, a feasible anchor point is selected from the accepted dataset. One input variable is swept at a time while all other inputs are held fixed at the selected feasible anchor point. The deterministic cycle solver and the trained surrogate are then evaluated over the same sweep. This provides a direct visual comparison between the reference cycle behavior and the neural network prediction for quantities such as specific thrust, TSFC, fuel-air ratio, bypass ratio, and selected internal cycle states.

The second study examines extrapolation behavior. The extrapolation study uses the same anchor-based one-variable-at-a-time structure as the parametric sweep study, but extends the swept variable beyond the original training bounds. These plots are used to identify how the surrogate behaves near and outside the sampled domain. The original training bounds and dataset-supported regions are shown on the plots, which helps distinguish ordinary interpolation error from behavior caused by leaving the reliable training envelope.

The final quantitative study is the runtime benchmark. This comparison measures the computational time required for direct deterministic cycle evaluation and compares it against the runtime of the trained neural network surrogate. The runtime benchmark is important

because the practical value of the surrogate model depends not only on prediction accuracy, but also on whether it provides a meaningful speed advantage for large sweeps, control studies, or optimization workflows.

Only the final results-oriented plots are discussed in this chapter. Additional diagnostic plots used during script development and anchor verification are not included in the main report body. This keeps the chapter focused on the performance of the trained models rather than the debugging process used to construct the studies.

6.1.1 Purpose of Final Validation Studies

The final validation studies were performed to evaluate the trained surrogate models beyond random test-set accuracy. The test-set parity and residual plots in Chapter 5 confirmed that the networks could reproduce withheld samples from the generated datasets. However, a surrogate intended for engineering analysis must also reproduce physically meaningful trends when individual cycle parameters are varied in a controlled manner.

The validation studies in this chapter therefore focus on structured model behavior. Parametric sweeps are used to compare the neural network predictions against deterministic cycle-analysis results over selected input ranges. Extrapolation studies are used to examine the model response near and outside the sampled training envelope. Runtime benchmarks are used to measure whether the trained surrogates provide a practical computational advantage over direct cycle evaluation.

Together, these studies assess three important qualities of the surrogate framework: predictive accuracy, physical trend consistency, and computational efficiency. These qualities determine whether the trained models are useful not only as regression fits, but also as fast forward models for design-space exploration and future control-oriented studies.

6.1.2 Evaluation Metrics

The final validation studies use both numerical and visual evaluation metrics. The primary comparison is between the deterministic parametric cycle analysis result and the neural network prediction for the same input condition. For a reference output y_i and surrogate prediction $\hat{y}_i$, the pointwise residual is defined as

$$r_i = \hat{y}_i - y_i \tag{6.1}$$

A residual near zero indicates close agreement between the surrogate and the cycle-analysis reference solution.

For a set of N evaluated points, the root-mean-square error is used as a compact measure

of average prediction error:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{6.2}$$

The mean absolute error is also useful because it is less dominated by isolated large-error points:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{6.3}$$

For outputs with different magnitudes, the relative error is used to compare prediction quality across variables:

$$E_i^{\mathrm{rel}} = \frac{\hat{y}_i - y_i}{y_i} \tag{6.4}$$

When reported as a percentage, E_i^{rel} is multiplied by 100.

In addition to scalar error values, parity plots are used to assess prediction quality visually. A parity plot compares $\hat{y}$ against y, with perfect agreement corresponding to the one-to-one reference line. This format makes bias, scatter, and output-dependent error trends easier to identify than a single aggregate metric.

For runtime benchmarking, the primary metric is the speedup factor between the deterministic cycle-analysis solver and the trained surrogate:

$$\mathcal{R}_{\mathrm{speed}} = \frac{t_{\mathrm{PCA}}}{t_{\mathrm{NN}}} \tag{6.5}$$

where t_{PCA} is the runtime of the deterministic parametric cycle analysis evaluation and t_{NN} is the runtime of the neural network surrogate. A larger value of $\mathcal{R}_{\mathrm{speed}}$ indicates a greater computational advantage for the surrogate model.

These metrics are interpreted together rather than independently. Low error values indicate numerical agreement, parity plots show whether the agreement is unbiased, residual plots reveal where the model begins to degrade, and runtime speedup determines whether the surrogate provides practical value for large-scale sweeps or future control-oriented applications.

6.1.3 Distinction Between Statistical and Physical Validation

Statistical validation and physical validation serve different purposes in this project. Statistical validation measures how accurately the neural network predicts withheld samples drawn from the same accepted dataset distribution used during training. This includes test-set parity plots, residual plots, RMSE, MAE, and relative-error statistics. These metrics

are necessary because they show whether the network has learned the sampled input-output mapping rather than only memorizing the training subset.

However, statistical validation alone is not sufficient for an engineering surrogate. A model may produce low average test-set error while still failing to reproduce physically meaningful trends when an input variable is varied systematically. This is especially important for propulsion cycle analysis, where the expected behavior of thrust, TSFC, fuel-air ratio, bypass ratio, and efficiency must remain consistent with the governing thermodynamic relationships.

Physical validation therefore examines the shape and direction of predicted trends. In the parametric sweep studies, one input variable is varied while the remaining inputs are held fixed at a feasible anchor point. The neural network prediction is then compared directly against the deterministic cycle solver over the same sweep. Agreement in both magnitude and trend indicates that the surrogate is learning the underlying cycle behavior, not only matching isolated test points. Statistical validation evaluates pointwise predictive accuracy over a sampled dataset, while physical validation evaluates whether the trained model preserves the expected engineering behavior of the cycle. Both forms of validation are required. The test-set results establish baseline regression accuracy, while the sweep and extrapolation studies determine whether the trained surrogate remains useful as an engineering analysis tool across structured operating changes.

6.2 Parametric Sweep Validation Method

The parametric sweep study was designed to test whether the trained surrogate models reproduce deterministic cycle-analysis trends when individual input variables are varied in a controlled manner. Unlike the random test-set comparisons presented in Chapter 5, the sweep study evaluates structured one-variable-at-a-time changes. This makes the physical response of each surrogate easier to interpret.

For each model and operating mode, a feasible anchor point was selected from the accepted dataset. The anchor point represents a physically valid engine state from the same feasible manifold used for training. One input variable was then swept while all remaining input variables were held fixed at the selected anchor value. The deterministic parametric cycle analysis solver and the trained neural network surrogate were evaluated over the same sweep points.

The general sweep construction is

$$x_j = \text{sweep}, \qquad x_k = x_{k,\text{anchor}} \quad \text{for} \quad k \neq j$$

where x_j is the swept input variable and all other inputs remain fixed at their feasible anchor values. Each sweep used 200 points over the selected in-domain interval.

The anchor selection process was designed to avoid choosing a point that was technically feasible but poorly suited for sweeping. Candidate anchors were taken from accepted dataset rows near the center of the normalized input space. For each candidate, the deterministic

cycle solver was used to estimate the feasible sweep interval for every planned sweep variable. The selected anchor maximized the usable sweep span across the planned variables while also retaining a mild preference for centrality in the input space. This ensured that the resulting sweeps represented meaningful in-domain validation cases rather than edge cases that immediately left the feasible region.

For each swept variable, the raw training bounds were first used as the candidate interval. The deterministic cycle solver was then evaluated over a coarse search grid to identify the contiguous physically valid interval containing the selected anchor point. The final plotted sweep interval was restricted to this contiguous valid region. This prevented invalid deterministic solutions from being connected through the plotted curves.

The swept variables used for each model family are summarized in Table 6.1.

Table 6.1 – Input variables used in the parametric sweep validation study

Model	Operating mode	Swept variables
Model A	Dry	$M_0,\ \pi_c,\ \pi_f,\ \tau_\lambda$
Model A	Afterburning	$M_0,\ \pi_c,\ \pi_f,\ \tau_\lambda,\ \tau_{AB}$
Model B	Dry	$M_0,\ \pi_c,\ \pi_f,\ \tau_\lambda$
Model B	Afterburning	$M_0,\ \pi_c,\ \pi_f,\ \tau_\lambda,\ \tau_{AB}$
Model C	Dry	$M_0,\ \pi_c,\ \pi_f,\ T_{t4},\ \pi_{M,\max}$
Model C	Afterburning	$M_0,\ \pi_c,\ \pi_f,\ T_{t4},\ T_{t7},\ \pi_{AB}$

A second support check was applied after deterministic evaluation. Although a sweep can remain inside the raw input bounds, it can still drive a solved cycle quantity outside the region represented by the accepted training data. This was especially important for bypass ratio, since α is not an independent prescribed input in the cycle solution and may leave the dataset-supported range during a one-variable sweep. To avoid treating these points as ordinary in-domain validation cases, the plotted and summarized sweep points were restricted using the 0.5th to 99.5th percentile range of α from the accepted dataset.

The outputs plotted for the ideal-cycle models included specific thrust, TSFC, bypass ratio, fuel-air ratios, and efficiency quantities. The real-cycle model used the same primary performance quantities, with the additional effects of real component losses and variable gas properties included through the Model C input vector. In each plot, the deterministic cycle-analysis result is shown against the corresponding neural network prediction over the same supported sweep interval.

This validation method provides a direct comparison between the learned surrogate response and the governing parametric cycle analysis. Agreement between the deterministic and neural network curves indicates that the surrogate has learned the local physical trend associated with the swept variable. Disagreement indicates either local prediction error, insufficient support from the accepted training manifold, or sensitivity to a region where the trained model has limited data coverage.

6.2.1 Anchor-Point Sweep Procedure

The parametric sweep study used an anchor-point procedure to ensure that each sweep began from a physically feasible engine state. Rather than selecting an arbitrary midpoint of the input ranges, each anchor was taken from an accepted dataset row. This ensured that the fixed values assigned to the non-swept variables corresponded to a valid cycle solution already represented in the training data.

For a selected anchor point, the input vector can be written as

$$\mathbf{x}_{\text{anchor}} = \begin{bmatrix} x_{1,\text{anchor}} & x_{2,\text{anchor}} & \cdots & x_{n_x,\text{anchor}} \end{bmatrix}^T \tag{6.6}$$

where n_x is the number of input variables for the corresponding model and operating mode. During a sweep of input variable x_j, all other variables remain fixed at their anchor values:

$$\mathbf{x}(\chi) = \begin{bmatrix} x_{1,\text{anchor}} & \cdots & x_j(\chi) & \cdots & x_{n_x,\text{anchor}} \end{bmatrix}^T \tag{6.7}$$

where χ is the dimensionless sweep coordinate.

Each candidate sweep point was evaluated using the deterministic cycle solver. Points that produced invalid cycle states, infeasible fuel-air ratios, unphysical bypass ratios, invalid nozzle states, or failed thermodynamic checks were excluded from the supported sweep interval. The final sweep curve therefore represents the continuous valid region containing the anchor point, rather than the entire raw input bound if portions of that bound produce invalid cycle solutions.

This procedure is important because the input space is not uniformly feasible. A point can lie inside the prescribed bounds for the swept variable while still producing an invalid cycle solution once the remaining variables are held fixed. The anchor-point procedure avoids plotting through these invalid regions and allows the neural network prediction to be compared only against deterministic solutions that are physically admissible.

The resulting procedure for each sweep is as follows:

1. Select a feasible anchor point from the accepted dataset.
2. Choose one input variable to sweep.
3. Hold all remaining inputs fixed at the anchor values.
4. Evaluate the deterministic cycle solver over the candidate sweep range.
5. Retain the contiguous feasible interval containing the anchor point.
6. Evaluate the trained surrogate over the same retained interval.
7. Compare the deterministic and surrogate output trends.

Using feasible anchor points provides a more rigorous validation case than a simple midpoint sweep. It ensures that each plotted trend begins from a known valid operating state and that the comparison between the deterministic solver and neural network surrogate is performed over physically meaningful sweep segments.

6.2.2 Single-Variable Perturbation Method

The parametric sweep study used a single-variable perturbation method. In each sweep, only one input variable was varied while all other inputs were held fixed at the selected feasible anchor point. This approach isolates the effect of one parameter at a time and makes the resulting surrogate trends easier to compare against the deterministic cycle solver.

For a selected swept variable x_j, the perturbation is defined as

$$x_j(\chi) = x_{j,\min} + \chi\,(x_{j,\max} - x_{j,\min})\,, \qquad 0 \le \chi \le 1 \tag{6.8}$$

where $x_{j,\min}$ and $x_{j,\max}$ define the retained feasible sweep interval. The remaining variables are unchanged:

$$x_k(\chi) = x_{k,\text{anchor}}, \qquad k \neq j \tag{6.9}$$

This method does not attempt to represent a full engine control schedule or a mission trajectory. Instead, it provides a controlled sensitivity study. The deterministic solver establishes the reference response of the cycle to the selected input perturbation, and the neural network prediction is compared against that response over the same set of sweep points.

The main advantage of this method is interpretability. If the surrogate and cycle-analysis curves agree during a sweep of M_0, π_c, π_f, τ_λ, T_{t4}, or another selected variable, then the trained network is reproducing the local physical sensitivity associated with that input. If the curves diverge, the error can be associated with a specific direction in the input space rather than with an unstructured random sample.

The method also makes edge behavior easier to identify. Since only one variable is changed at a time, deviations near the end of a sweep can be interpreted relative to the swept parameter and its interaction with the fixed anchor state. This is useful for distinguishing ordinary interpolation error from behavior caused by approaching the boundary of the feasible cycle manifold.

The limitation of the method is that it does not test all coupled input variations simultaneously. Real engine operation may involve coordinated changes in pressure ratio, turbine inlet temperature, afterburner temperature, and nozzle behavior. Therefore, the single-variable perturbation results should be interpreted as local trend-validation studies rather than complete dynamic operating schedules.

6.2.3 Reference PCA and Neural Network Comparison

For each retained sweep interval, the deterministic parametric cycle analysis solver was treated as the reference model. The same input points were passed to both the PCA solver and the trained neural network surrogate, allowing the predicted output trends to be compared directly. This comparison was made using the dimensional output quantities after the neural network predictions were denormalized and inverse-transformed where required.

The PCA solver provides the physics-based reference response for each single-variable perturbation. Since the solver evaluates the governing cycle-analysis equations directly, its output defines the expected behavior of the engine model under the assumptions established in Chapter 3. The neural network output is then interpreted as an approximation of this reference mapping rather than as an independent source of truth.

The comparison focuses on both magnitude agreement and trend agreement. Magnitude agreement indicates that the surrogate predicts the correct output values at individual sweep points. Trend agreement indicates that the surrogate captures the correct response shape as the swept input changes. Both are necessary for a useful engineering surrogate. A model that matches a few isolated points but predicts the wrong trend would not be reliable for design-space exploration or control-oriented studies.

For the ideal-cycle models, the comparison emphasizes specific thrust, TSFC, fuel-air ratio, bypass ratio, and efficiency behavior. For the real-cycle model, the comparison also reflects the influence of component losses, mode-dependent gas properties, mixer pressure recovery, and nozzle behavior. This makes the Model C sweeps more demanding because the surrogate must approximate a higher-dimensional and more strongly coupled reference mapping.

The plotted sweep figures use the same convention throughout the chapter. The deterministic PCA solution is shown as the reference curve, while the neural network prediction is shown over the same supported sweep points. Close overlap between the two curves indicates that the surrogate has learned the corresponding local cycle response. Visible separation between the curves indicates local prediction error, poor training support in that region, or sensitivity to coupled variables held fixed at the selected anchor state. This direct comparison between PCA and neural network predictions forms the basis of the parametric sweep validation results presented in the following sections.

6.3 Dry-Mode Parametric Sweep Results

The dry-mode parametric sweep results evaluate the non-afterburning experts for Models A, B, and C. In these cases, afterburner fuel addition is inactive, so the predicted response is governed by the baseline mixed-flow turbofan cycle. The dry-mode studies therefore provide a controlled comparison of how each surrogate reproduces the deterministic PCA response before afterburner heat addition is introduced.

The results are organized by swept variable rather than by model family. This allows the

response of Models A, B, and C to be compared along the same physical direction in the input space. For each sweep, the deterministic PCA solution is treated as the reference curve, while the neural network prediction is plotted over the same supported interval. Agreement between the two curves indicates that the surrogate has learned the local dry-cycle trend associated with the selected input variable.

The dry-mode sweep discussion begins with freestream Mach number, followed by compressor pressure ratio, fan pressure ratio, burner temperature ratio, bypass-ratio behavior, and component-efficiency-related effects. The section then concludes with a brief summary of the dry-mode results and the main differences observed between the ideal-cycle and real-cycle surrogates.

6.3.1 Mach Number Sweep

The dry-mode Mach number sweeps evaluate the response of each surrogate as the freestream Mach number is varied while all remaining inputs are held fixed at the selected feasible anchor point. This sweep is important because freestream Mach number affects the ram temperature ratio, ram pressure ratio, flight velocity term, nozzle expansion behavior, and resulting propulsion efficiencies. Therefore, agreement between the PCA and neural network curves indicates that the surrogate has learned an important flight-condition sensitivity of the cycle.

The Model A dry-mode Mach number sweep is shown in Fig. 6.1. Since Model A uses fixed gas properties and the ideal-cycle formulation, this case represents the simplest Mach sensitivity test. The neural network prediction follows the deterministic PCA trend closely across the supported interval. The specific thrust, TSFC, fuel-air ratio, bypass ratio, and efficiency outputs remain well aligned with the reference solution, confirming that the fixed-property ideal-cycle expert captures the dry-cycle Mach-number response.

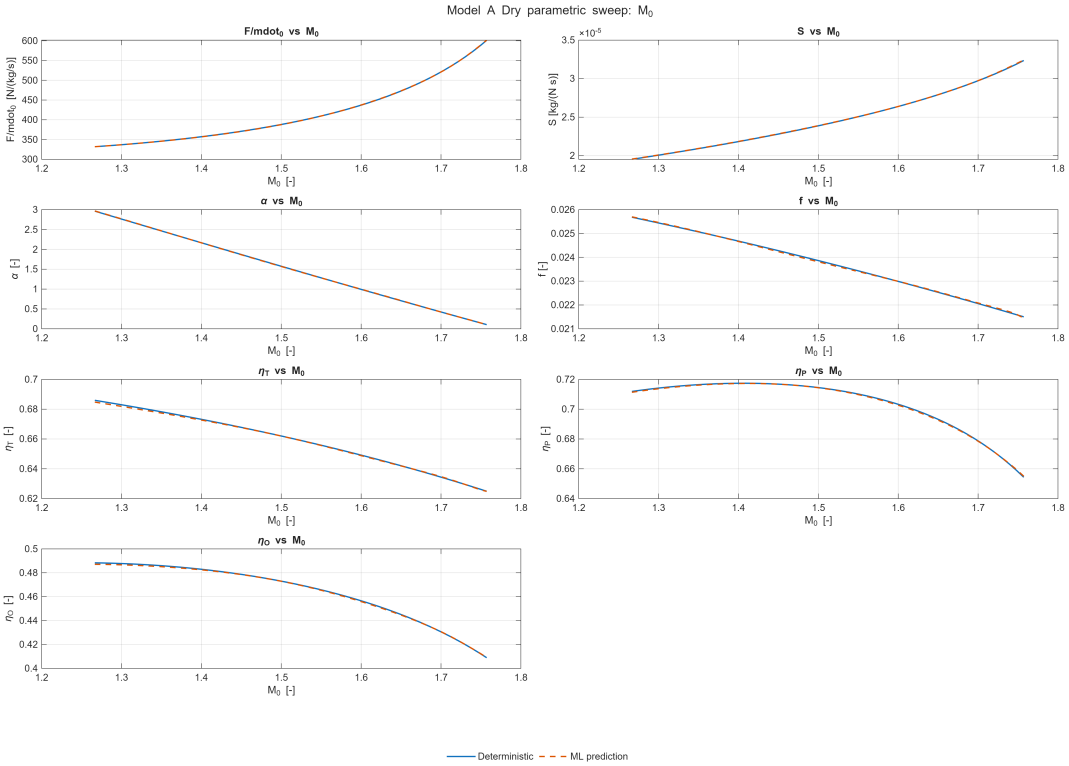


Figure 6.1 – Model A dry-mode parametric sweep over freestream Mach number

The Model B dry-mode Mach number sweep is shown in Fig. 6.2. Model B retains the ideal-cycle formulation but includes thermochemical variation through the active gas property inputs. The surrogate continues to track the deterministic PCA response over the Mach sweep, showing that the added thermochemical dimensions do not prevent the network from learning the main dry-mode flight condition sensitivity. The agreement is consistent with the Model B test-set validation results from Chapter 5.

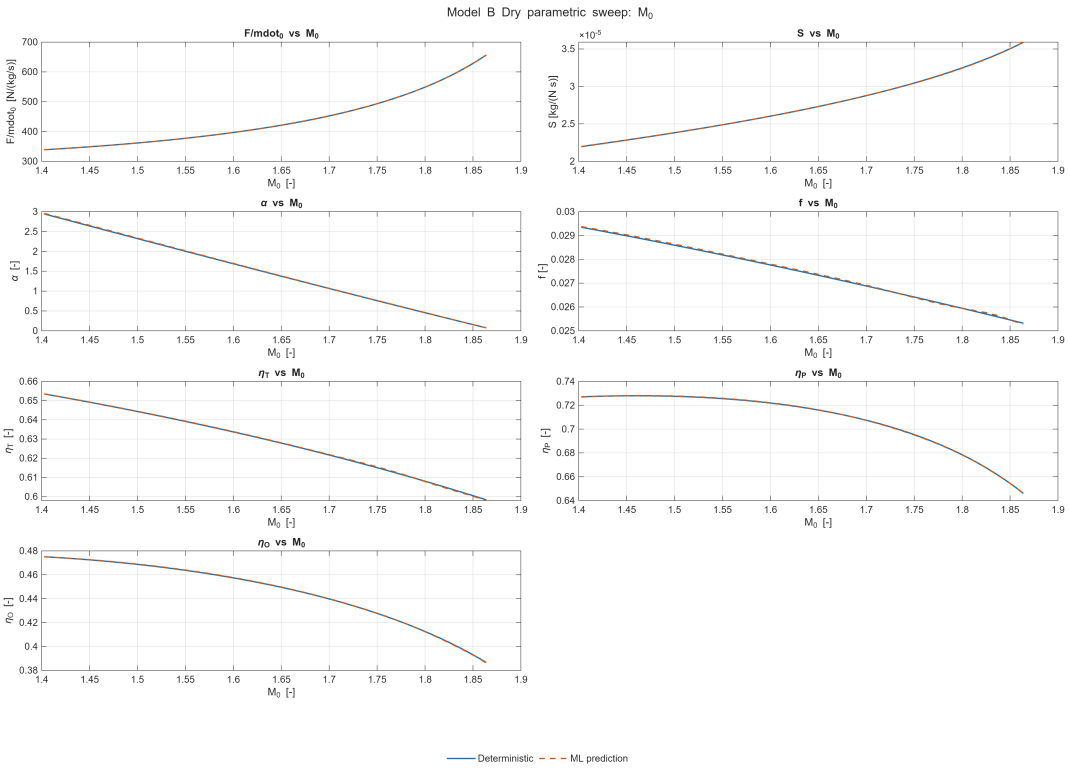


Figure 6.2 – Model B dry-mode parametric sweep over freestream Mach number

The Model C dry-mode Mach number sweep is shown in Fig. 6.3. This is the most demanding Mach sweep because the real-cycle formulation includes component losses, polytropic efficiencies, mixer pressure recovery, nozzle pressure loss, and variable gas properties. Despite the increased dimensionality, the surrogate captures the dominant deterministic trends across the supported Mach-number interval. Slight deviations between the deterministic solution and surrogate predictions can be seen in the specific thrust as well as predicted bypass ratio. The general trend alignment indicates that the real-cycle dry expert learned the primary Mach dependence of the cycle, including the coupled influence of ram effects and nozzle expansion on the final performance quantities.

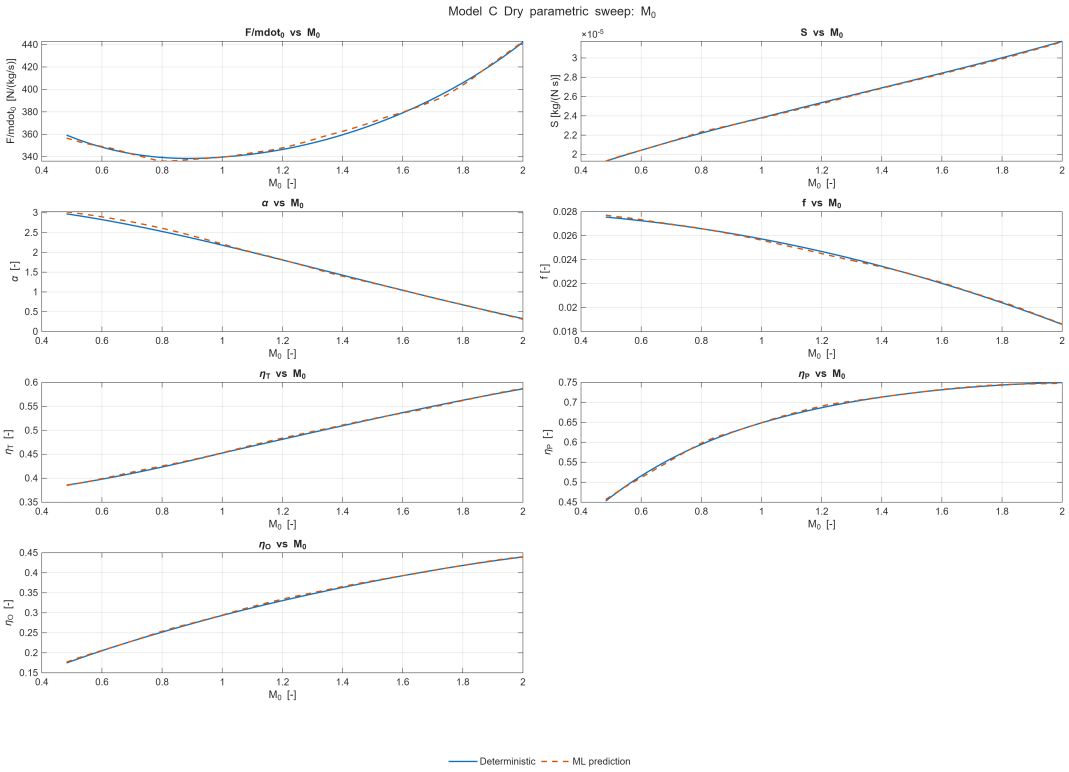


Figure 6.3 – Model C dry-mode parametric sweep over freestream Mach number

Overall, the Mach number sweeps show strong agreement between the deterministic PCA solver and the trained dry-mode surrogates. Models A and B closely reproduce the ideal-cycle response, while Model C captures the same general physical trend in the higher-dimensional real-cycle setting. This supports the conclusion that all three dry experts learned the flight-condition sensitivity associated with freestream Mach number.

6.3.2 Compressor Pressure Ratio Sweep

The dry-mode compressor pressure ratio sweeps evaluate how each surrogate responds to changes in π_c while the remaining inputs are held fixed at the feasible anchor point. Compressor pressure ratio is one of the primary cycle design variables because it affects the compressor exit state, main burner fuel requirement, turbine work extraction, and nozzle expansion conditions. Agreement in this sweep indicates that the surrogate has learned the dry-cycle sensitivity to core compression.

The Model A dry-mode compressor pressure ratio sweep is shown in Fig. 6.4. The neural network prediction closely tracks the deterministic PCA reference over the supported sweep interval. The specific thrust, TSFC, fuel-air ratio, bypass ratio, and efficiency trends remain well aligned with the ideal-cycle solution. This result confirms that the fixed-property ideal-cycle expert captures the basic effect of changing core pressure ratio in dry operation.

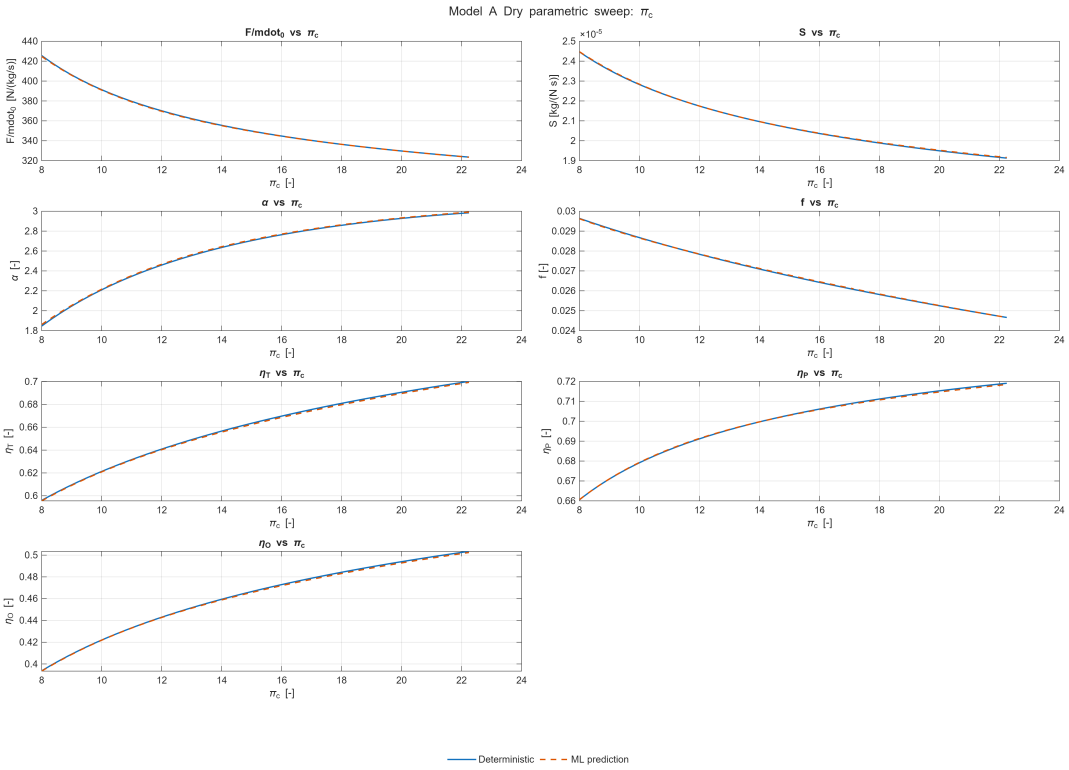


Figure 6.4 – Model A dry-mode parametric sweep over compressor pressure ratio

The Model B dry-mode compressor pressure ratio sweep is shown in Fig. 6.5. The surrogate again follows the deterministic PCA trend closely, even though the thermochemical quantities are active input dimensions in Model B. This indicates that the variable-property ideal-cycle expert learns the compressor pressure ratio response without losing the baseline trend behavior observed in Model A.

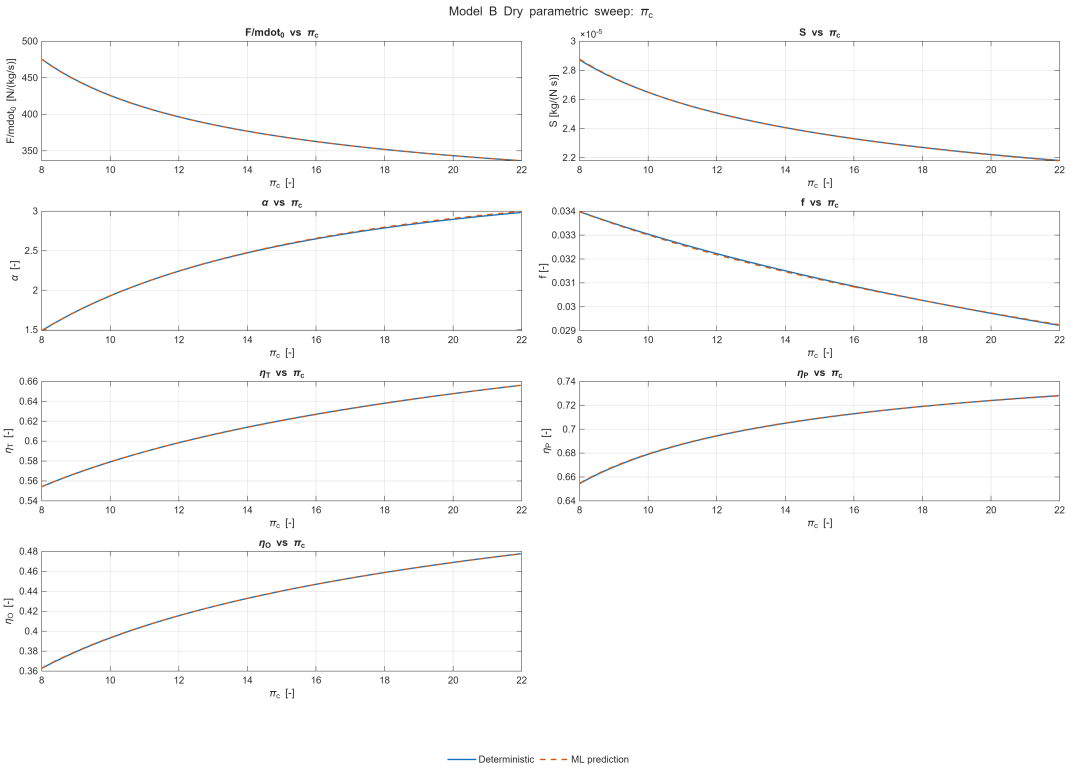


Figure 6.5 – Model B dry-mode parametric sweep over compressor pressure ratio

The Model C dry-mode compressor pressure ratio sweep is shown in Fig. 6.6. This case is more complex because the real-cycle response includes component efficiencies, combustor pressure loss, turbine pressure ratio, mixer pressure recovery, and nozzle loss. Despite this added coupling, the Model C dry expert reproduces the general trend over the supported interval. Deviation can be seen in all of the measured trends more than the mach number trends, with the statistical drift remaining small in magnitude. The general trend agreement shows that the real-cycle surrogate captures the primary influence of core compression on dry-mode performance.

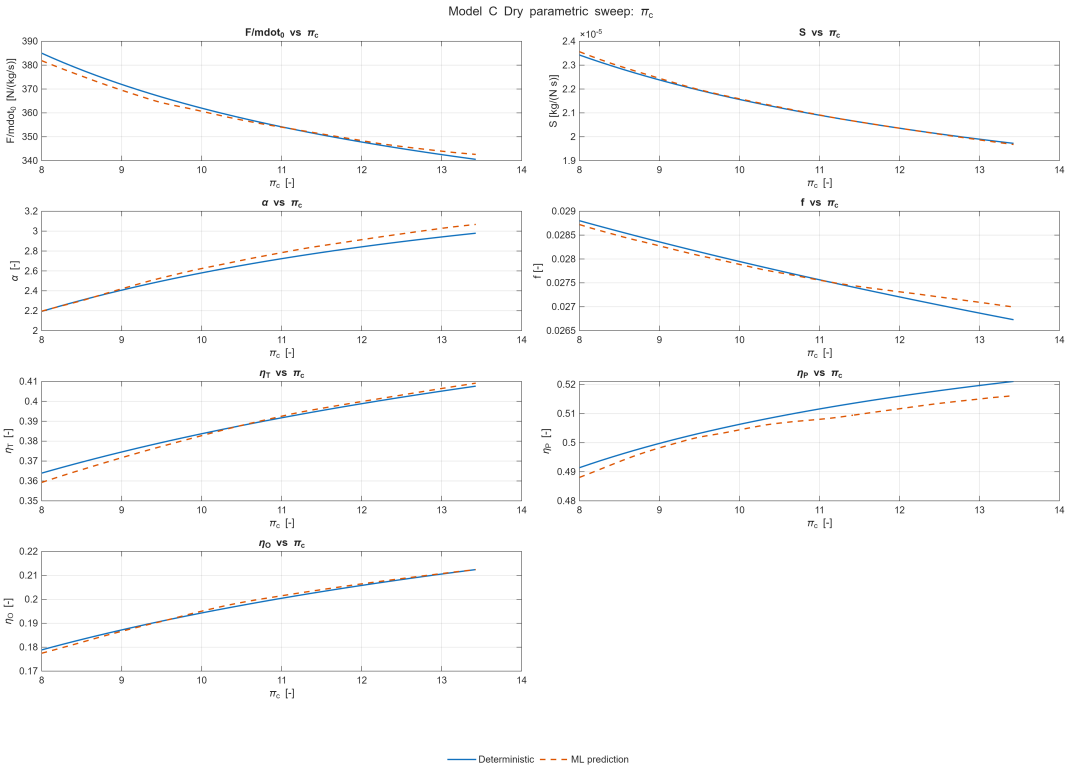


Figure 6.6 – Model C dry-mode parametric sweep over compressor pressure ratio

Overall, the compressor pressure ratio sweeps show consistent agreement between the deterministic PCA solver and the neural network surrogates. Models A and B reproduce the ideal-cycle π_c response with strong agreement, while Model C preserves the same general sensitivity after real-cycle losses and efficiency effects are included at a slight accuracy penalty. This indicates that core compression behavior is one of the better-learned dry-mode directions across the full surrogate hierarchy.

6.3.3 Fan Pressure Ratio Sweep

The dry-mode fan pressure ratio sweeps evaluate how each surrogate responds to changes in π_f while all other inputs remain fixed at the feasible anchor point. Fan pressure ratio directly affects the bypass stream state and therefore influences the mixed-flow balance between the core and bypass streams. This makes the π_f sweep an important test of whether the surrogate has learned the mixed-flow coupling rather than only the core-cycle response.

The Model A dry-mode fan pressure ratio sweep is shown in Fig. 6.7. The neural network prediction follows the deterministic PCA reference closely over the supported interval for quantities such as the specific thrust, bypass ratio, as well as thermal and propulsive efficiencies. The trained model struggles with predicting the thrust specific fuel consumption trend by a narrow offset and also under-predicts the overall efficiency of the cycle for the lower range in π_f. additionally the model approximates the fuel to air ratio within reason, but fails

to model the independence of this ratio relative to the fan pressure ratio. The fixed-property dry expert captures the fan-pressure-ratio sensitivity in the baseline ideal-cycle formulation with reasonable accuracy.

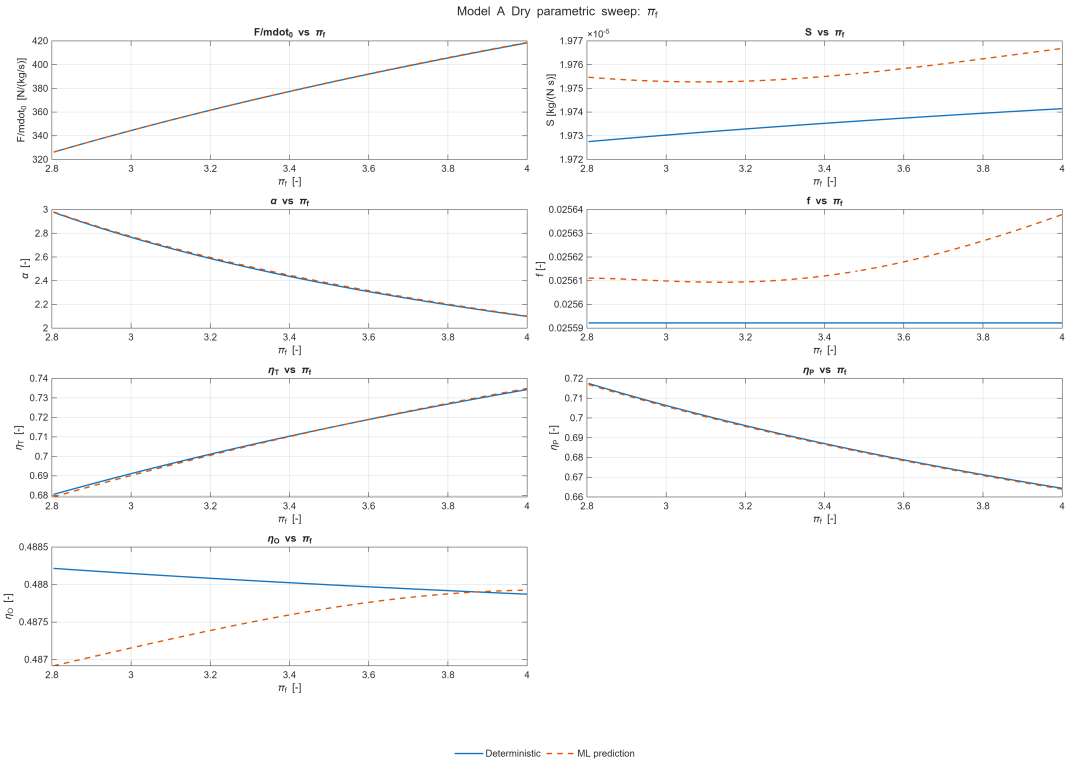


Figure 6.7 – Model A dry-mode parametric sweep over fan pressure ratio

The Model B dry-mode fan pressure ratio sweep is shown in Fig. 6.8. The surrogate follows a similar trend in accuracy to Model B, capturing the deterministic trends, although with deviations. This is expected because Model B must represent the same fan and mixer sensitivity while also accounting for active variation in thermochemical quantities. Additionally, since both models A and B are trained on data created from the same equation set, the same failure mode is expected even with the difference in dimensionality of the input set. with the added input dependence, the trends predicted by Model B remain consistent with the PCA reference solution.

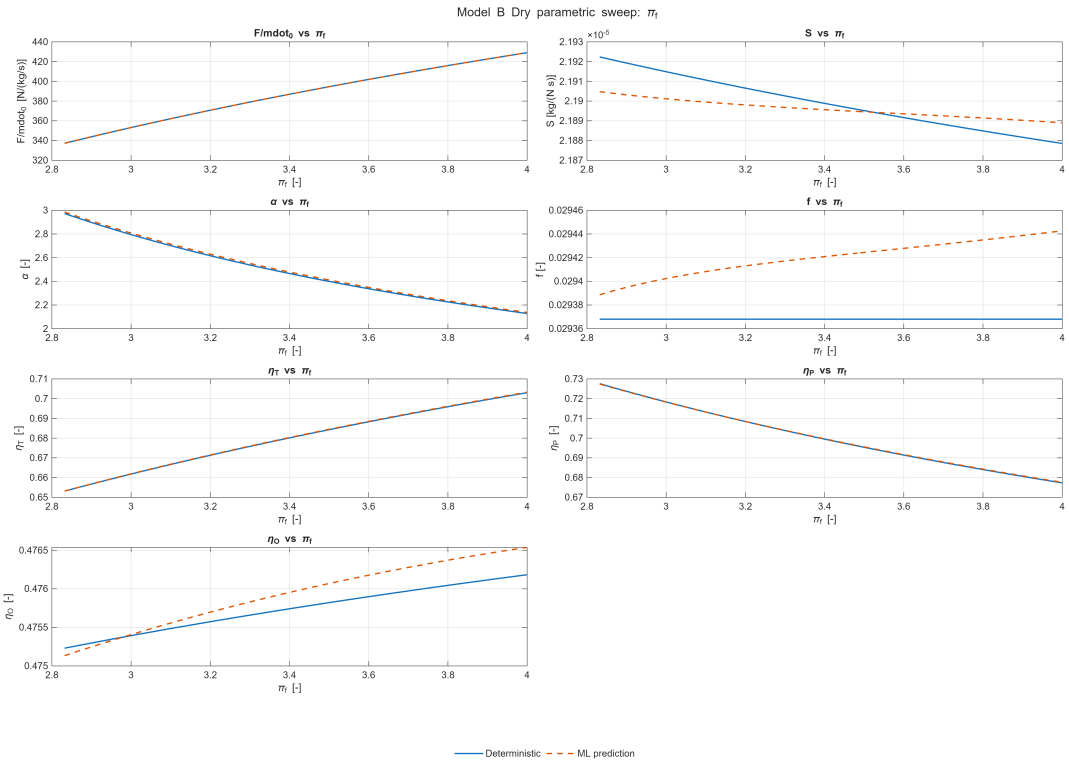


Figure 6.8 – Model B dry-mode parametric sweep over fan pressure ratio

The Model C dry-mode fan pressure ratio sweep is shown in Fig. 6.9. This sweep is more demanding because the real-cycle response to fan pressure ratio is coupled to component efficiencies, turbine work balance, mixer pressure recovery, and downstream nozzle expansion. The surrogate reproduces the overall shape of the deterministic response, but visible offsets appear in the thrust specific fuel consumption and efficiency metrics. The same problematic prediction of the fuel to air ratio as models A and B can also be seen, though with more statistical noise. Additionally, the trends predicted by the surrogate all exhibit slight oscillatory behavior where the model loses some accuracy at select locations. These deviations indicate that the fan-pressure-ratio direction is more difficult for the real-cycle surrogate than the Mach number and compressor pressure ratio directions.

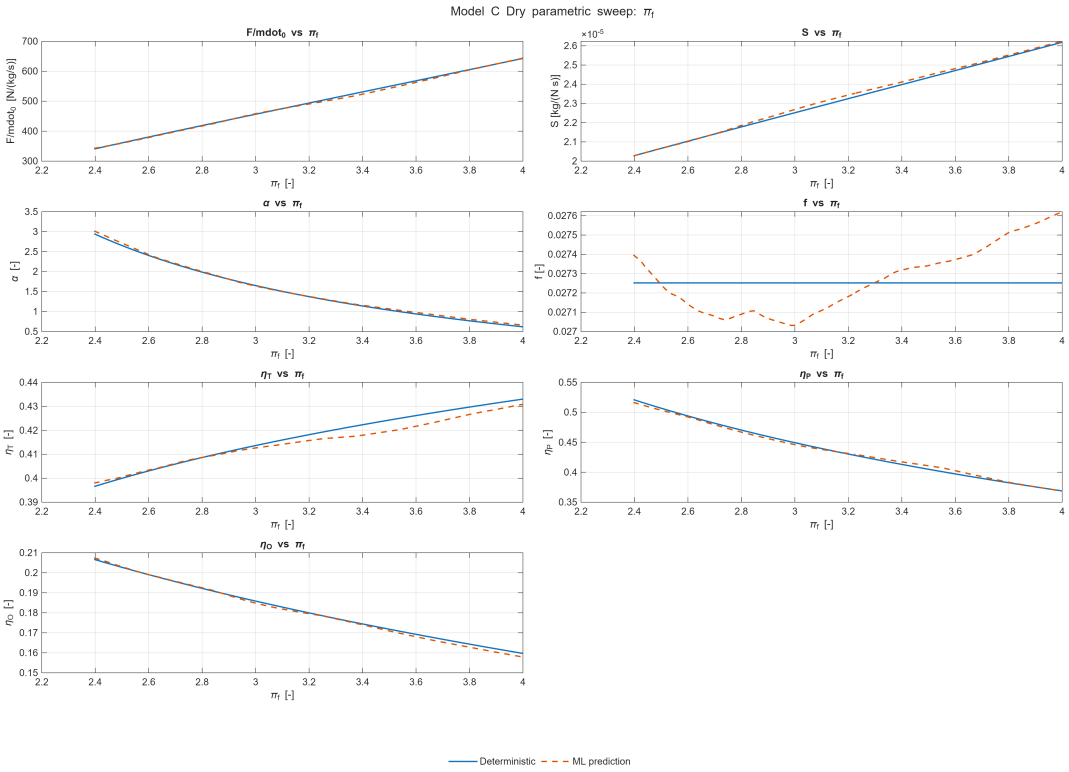


Figure 6.9 – Model C dry-mode parametric sweep over fan pressure ratio

Overall, the fan pressure ratio sweeps show that the ideal-cycle surrogates capture the bypass-stream sensitivity with reasonable agreement. Model C also captures the dominant trend, but with larger local deviations caused by the higher-dimensional real-cycle coupling. This makes fan pressure ratio one of the more informative dry-mode sweep directions for identifying where the real-cycle surrogate could benefit from additional training support or refined loss weighting.

6.3.4 Burner Temperature Ratio Sweep

The dry-mode burner temperature ratio sweeps evaluate the thermal-loading response of each surrogate. For Models A and B, the swept variable is the dimensionless burner temperature ratio τ_λ. For Model C, the corresponding real-cycle sweep is performed using the burner exit total temperature T_{t4}. These quantities control the thermal energy added in the main combustor and therefore strongly influence fuel-air ratio, turbine work balance, specific thrust, and TSFC.

The Model A dry-mode burner temperature ratio sweep is shown in Fig. 6.10. The neural network prediction follows the deterministic PCA reference less closely than previous parameters across the supported interval. The specific thrust and fuel-air ratio increase consistently with increasing thermal loading, while the efficiency and TSFC trends remain aligned with the ideal-cycle reference solution with some oscillatory behavior. The cyclical

over-prediction and under-prediction in most metrics is at low magnitude but indicates potential needed refinement. This confirms that the fixed-property dry expert captures the general thermal sensitivity of the ideal mixed-flow cycle, but could use some refined loss weighting and potential hyperparameter adjustment.

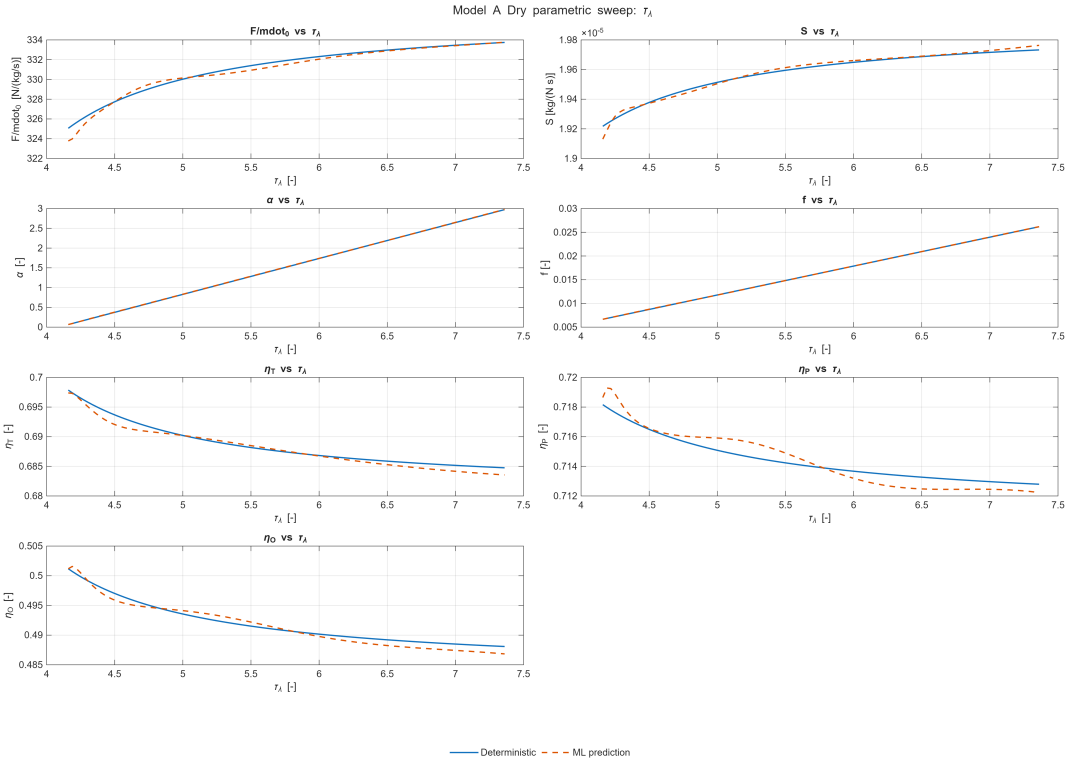


Figure 6.10 – Model A dry-mode parametric sweep over burner temperature ratio

The Model B dry-mode burner temperature ratio sweep is shown in Fig. 6.11. The surrogate again reproduces the main deterministic PCA trends but with less visible oscillatory behavior. This result is important because Model B must learn the thermal response while also accounting for active thermochemical variation in γ, c_p, and h_{PR}. The close agreement indicates that the variable-property ideal-cycle expert preserves the expected relationship between combustor thermal loading and dry-cycle performance. Additionally the increase in neuron count relative to model A has increased the predicted trend fitting relative to the deterministic solution.

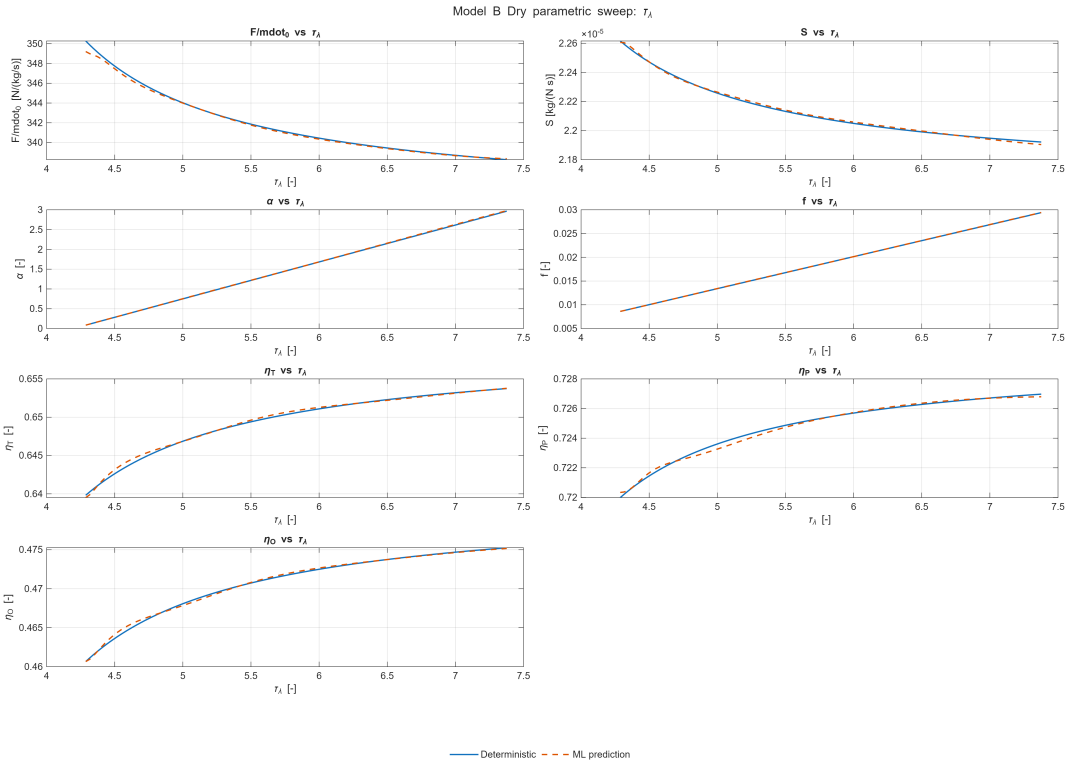


Figure 6.11 – Model B dry-mode parametric sweep over burner temperature ratio

The Model C dry-mode burner exit total-temperature sweep is shown in Fig. 6.12. This is the real-cycle equivalent of the ideal burner temperature ratio sweep. The surrogate shows significantly larger deviations relative to the ideal-cycle models. The most noticeable differences occur in specific thrust, TSFC, bypass ratio, and fuel-air ratio, particularly near the lower end of the supported T_{t4} interval. This indicates that the real-cycle thermal-loading direction is significantly more difficult for the network to learn because it is coupled to component efficiencies, turbine work extraction, mixer behavior, and nozzle expansion.

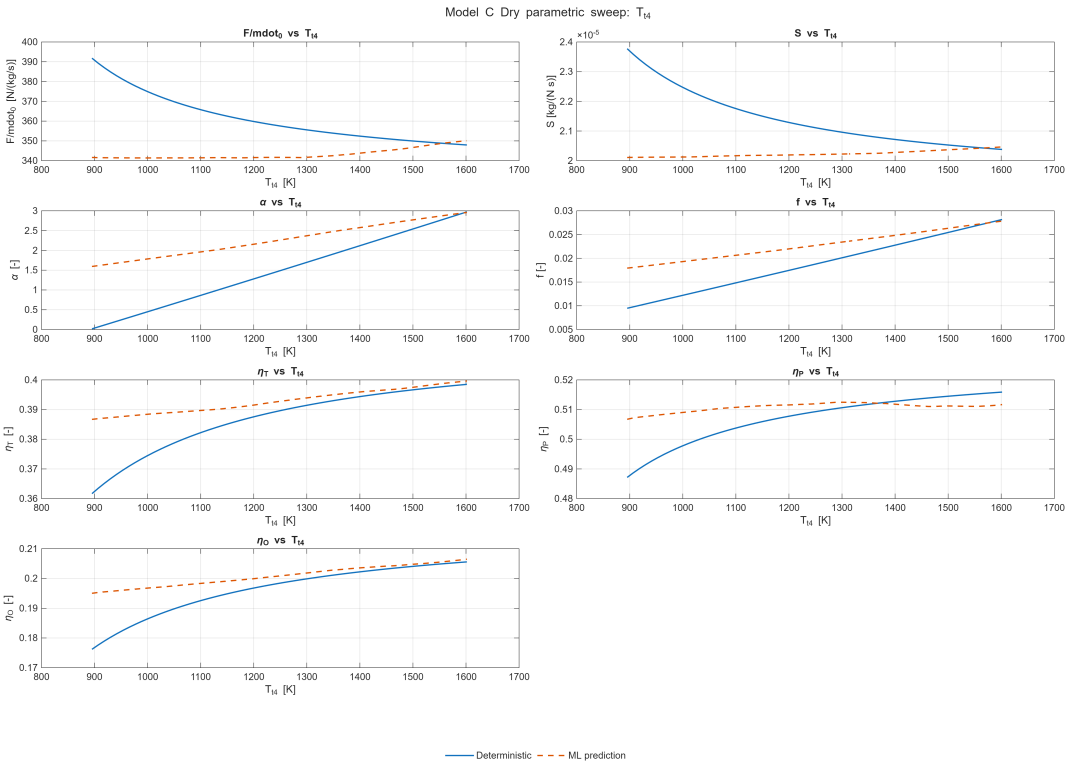


Figure 6.12 – Model C dry-mode parametric sweep over burner exit total temperature

Overall, the burner temperature ratio sweeps show reasonable thermal-response agreement for the ideal-cycle surrogates and more challenging behavior for the real-cycle surrogate. Models A and B reproduce the deterministic PCA trends closely, while Model C struggles with the general response trends and exhibits larger local deviations. This makes the burner thermal-loading direction one of the key dry-mode areas where future Model C refinement would be most beneficial.

6.3.5 Mixer Pressure Recovery Sweep

The mixer pressure recovery sweep was performed only for Model C because mixer pressure loss is included only in the real-cycle formulation. The swept variable is $\pi_{M,\max}$, which represents the maximum mixer pressure recovery used in the real-cycle mixer model. This parameter influences the mixed-stream total pressure downstream of the mixer and therefore affects the available nozzle pressure ratio, exit Mach number, thrust, and TSFC.

The Model C dry-mode mixer pressure recovery sweep is shown in Fig. 6.13. The surrogate captures the main response of the deterministic PCA solution over the supported interval, but visible deviations appear in selected output quantities. This is expected because $\pi_{M,\max}$ acts through the mixer and downstream nozzle rather than through a direct thermal or compression input. As a result, its effect is distributed through several coupled real-cycle relations.

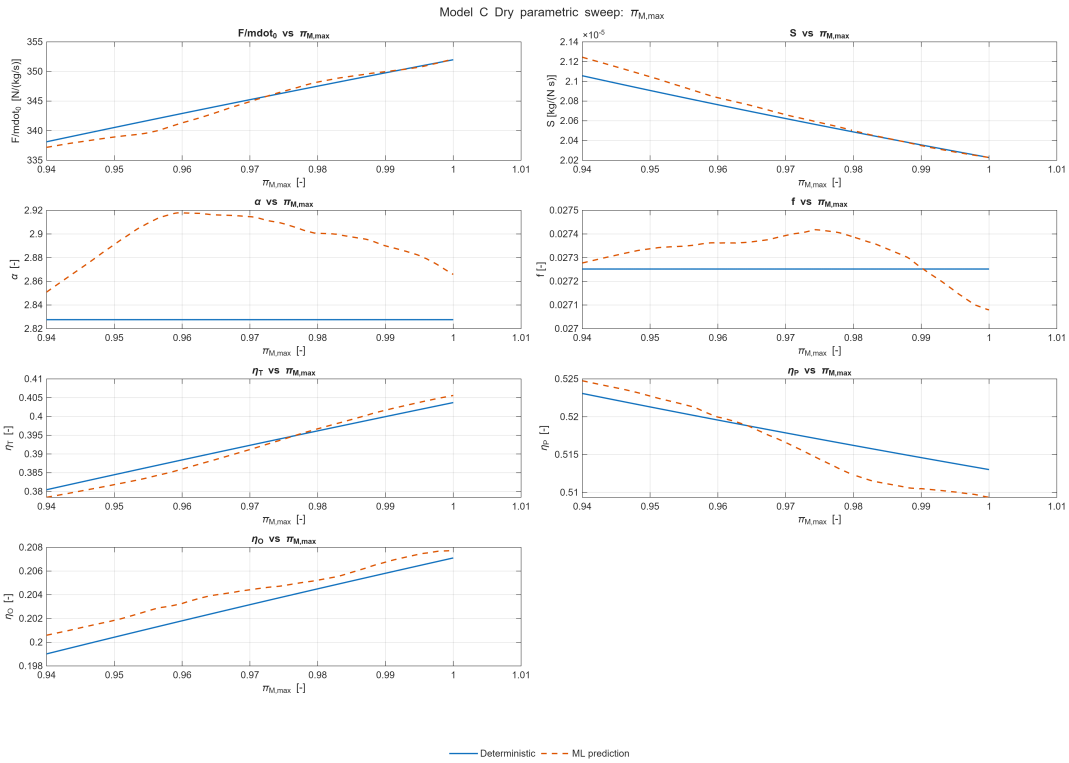


Figure 6.13 – Model C dry-mode parametric sweep over maximum mixer pressure recovery

The thrust and TSFC trends remain reasonably consistent with the reference solution, indicating that the surrogate has learned the dominant performance effect of changing mixer pressure recovery. Larger differences in predicted efficiency trends suggest that this loss-related direction is more difficult for the network than the primary flight-condition and compressor-pressure directions. Additionally the model fails in recognizing the independent nature of the bypass ratio and fuel to air ratio relative to changes in $\pi_{M,\max}$. The machine learning model predicts the expected values for the bypass ratio and fuel to air ratio within a reasonable tolerance, but seems to hallucinate some small dependency. This is a reasonable result because mixer pressure recovery is coupled to the mixed-stream total pressure, nozzle expansion process, and final performance quantities rather than appearing as a simple independent output driver.

Overall, the mixer pressure recovery sweep shows that Model C captures the general dry-mode effect of mixer pressure loss, but with less precision than the strongest sweep directions. This result identifies mixer pressure recovery as a useful target for future refinement of the real-cycle surrogate, especially through additional training density, output weighting, or explicit physics-consistency penalties involving the mixer and nozzle states.

6.3.6 Dry-Mode Sweep Summary

The dry-mode parametric sweeps show that the trained surrogates generally reproduce the deterministic PCA trends over the supported in-domain sweep intervals. Models A and B

show the strongest agreement, which is expected because both use the ideal-cycle formulation and have lower input dimensionality than the real-cycle model. Model A verifies the fixed-property baseline case, while Model B confirms that the same framework remains effective after thermochemical variability is introduced.

Model C also captures the dominant dry-cycle trends, especially for freestream Mach number and compressor pressure ratio. Larger deviations appear in the fan pressure ratio, burner exit total temperature, and mixer pressure recovery sweeps. These directions are more strongly coupled to the real-cycle loss, mixer, turbine work-balance, and nozzle-expansion relations, making them more difficult for the surrogate to approximate with the same level of precision as the ideal-cycle models.

Overall, the dry-mode sweeps support the use of the trained models as forward surrogates within the sampled operating envelope. The results also identify the most important refinement targets for Model C: improved representation of thermal-loading effects, bypass-stream coupling, and mixer/nozzle loss behavior.

6.4 Afterburning-Mode Parametric Sweep Results

The afterburning-mode parametric sweep results evaluate the wet experts for Models A, B, and C. In these cases, afterburner heat addition is active and the cycle response includes both the baseline turbofan behavior and the additional effects of downstream fuel addition. These sweeps are therefore more demanding than the dry-mode cases because the surrogate must reproduce the coupled response of thrust, TSFC, main-burner fuel flow, afterburner fuel flow, bypass ratio, and efficiency quantities.

The results are again organized by swept variable rather than by model family. This allows the afterburning response of Models A, B, and C to be compared along consistent physical directions in the input space. The ideal-cycle models are evaluated using sweeps over freestream Mach number, compressor pressure ratio, fan pressure ratio, burner temperature ratio, and afterburner temperature ratio. The real-cycle Model C sweeps use the corresponding real-cycle variables, including burner exit total temperature, afterburner exit total temperature, and afterburner pressure ratio.

For each sweep, the deterministic PCA solution is treated as the reference curve and the neural network prediction is plotted over the same supported interval. Agreement between the two curves indicates that the wet expert has learned the afterburning cycle response associated with the selected input variable. The section concludes with a brief summary of the afterburning-mode results and the main differences observed between the ideal-cycle and real-cycle surrogate behavior.

6.4.1 Mach Number Sweep

The afterburning-mode Mach number sweeps evaluate the response of each wet expert as the freestream Mach number is varied while all remaining inputs are held fixed at the selected feasible anchor point. This sweep tests whether the surrogate can reproduce the combined effect of flight condition, ram compression, afterburner heat addition, and nozzle expansion on the augmented cycle.

The Model A afterburning-mode Mach number sweep is shown in Fig. 6.14. The neural network prediction follows the deterministic PCA reference closely across the supported interval. The specific thrust, TSFC, bypass ratio, main-burner fuel-air ratio, afterburner fuel-air ratio, and efficiency trends remain well aligned with the ideal-cycle solution. This confirms that the fixed-property wet expert captures the dominant Mach-number sensitivity of the augmented ideal-cycle mapping.

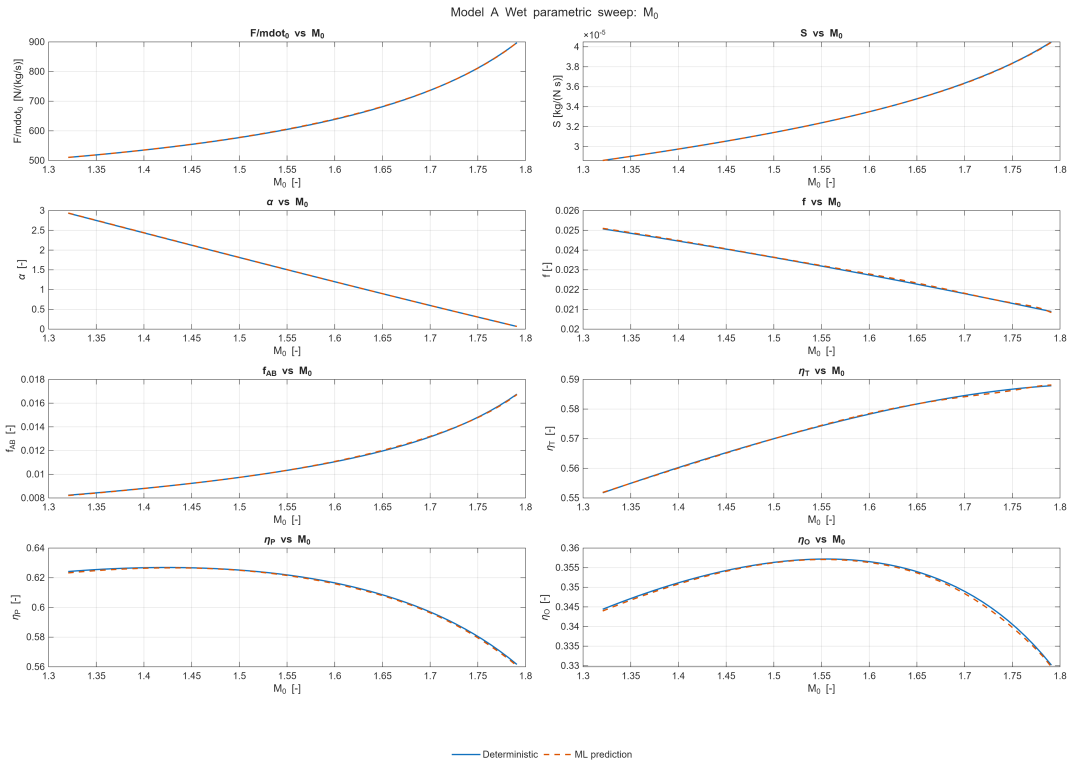


Figure 6.14 – Model A afterburning-mode parametric sweep over freestream Mach number

The Model B afterburning-mode Mach number sweep is shown in Fig. 6.15. Model B retains the same ideal-cycle structure as Model A, but includes active thermochemical variation. The surrogate continues to reproduce the deterministic PCA response over the Mach sweep, indicating that the added gas-property dependence does not prevent the wet expert from learning the primary flight-condition trend.

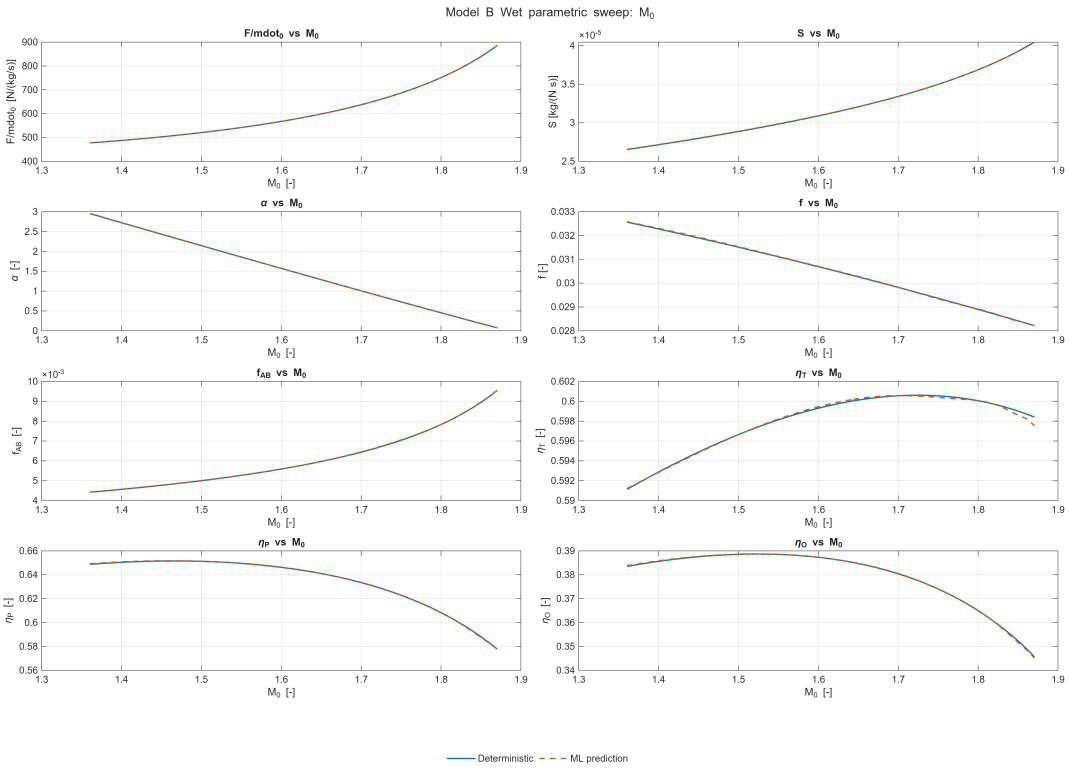


Figure 6.15 – Model B afterburning-mode parametric sweep over freestream Mach number

The Model C afterburning-mode Mach number sweep is shown in Fig. 6.16. This is the most demanding Mach sweep because the real-cycle wet expert must account for component losses, afterburner pressure loss, mixer pressure recovery, nozzle loss, and mode-dependent gas properties. Despite the additional coupling, the surrogate captures the main deterministic trends over the supported Mach-number interval. The agreement indicates that Model C learned the principal freestream Mach number response for the augmented real-cycle formulation.

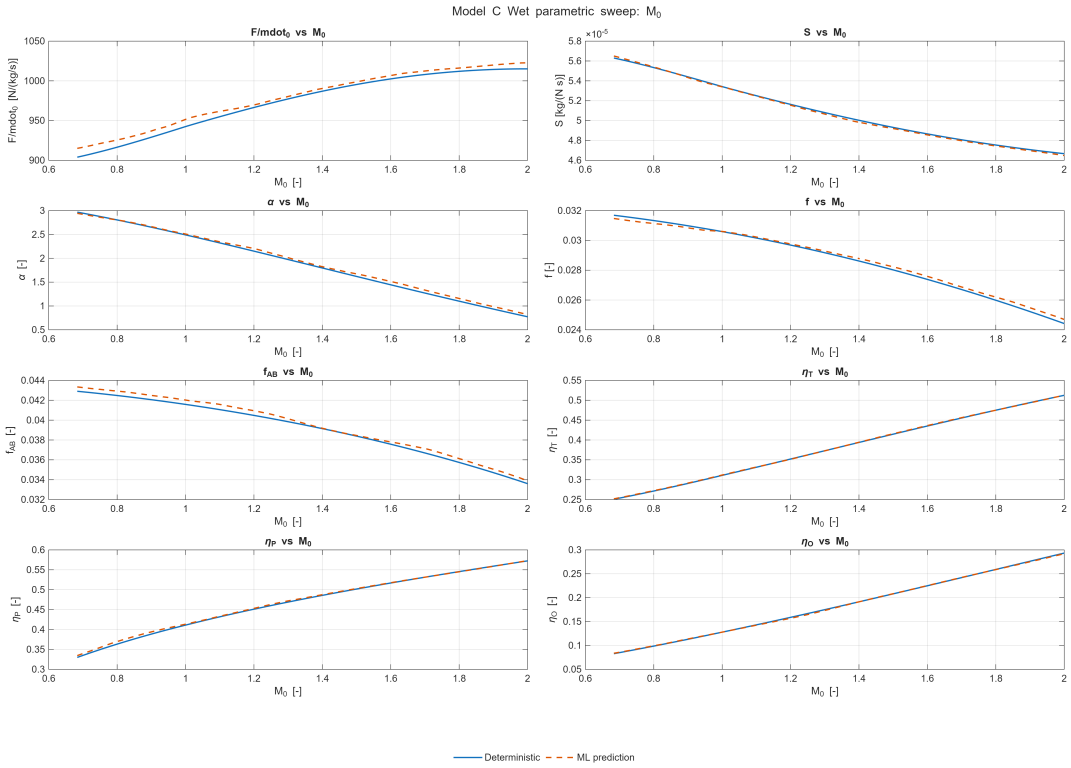


Figure 6.16 – Model C afterburning-mode parametric sweep over freestream Mach number

Overall, the afterburning-mode Mach number sweeps show strong trend agreement across all three model families. Models A and B closely reproduce the ideal-cycle augmented response, while Model C captures the same general flight-condition sensitivity in the full real-cycle setting. This indicates that freestream Mach number remains one of the better-learned sweep directions for both dry and afterburning operation.

6.4.2 Compressor Pressure Ratio Sweep

The afterburning-mode compressor pressure ratio sweeps evaluate the response of each wet expert as π_c is varied while the remaining inputs are held fixed at the feasible anchor point. Compressor pressure ratio affects the core compression process, main burner fuel requirement, turbine work balance, and the thermodynamic state entering the mixer. In afterburning operation, these effects also influence the state entering the afterburner and the resulting augmented nozzle expansion.

The Model A afterburning-mode compressor pressure ratio sweep is shown in Fig. 6.17. The neural network prediction closely tracks the deterministic PCA reference over the supported interval. The specific thrust, TSFC, bypass ratio, main-burner fuel-air ratio, afterburner fuel-air ratio, and efficiency trends remain well aligned with the ideal-cycle solution. This indicates that the fixed-property wet expert captures the core compression sensitivity of the augmented ideal cycle.

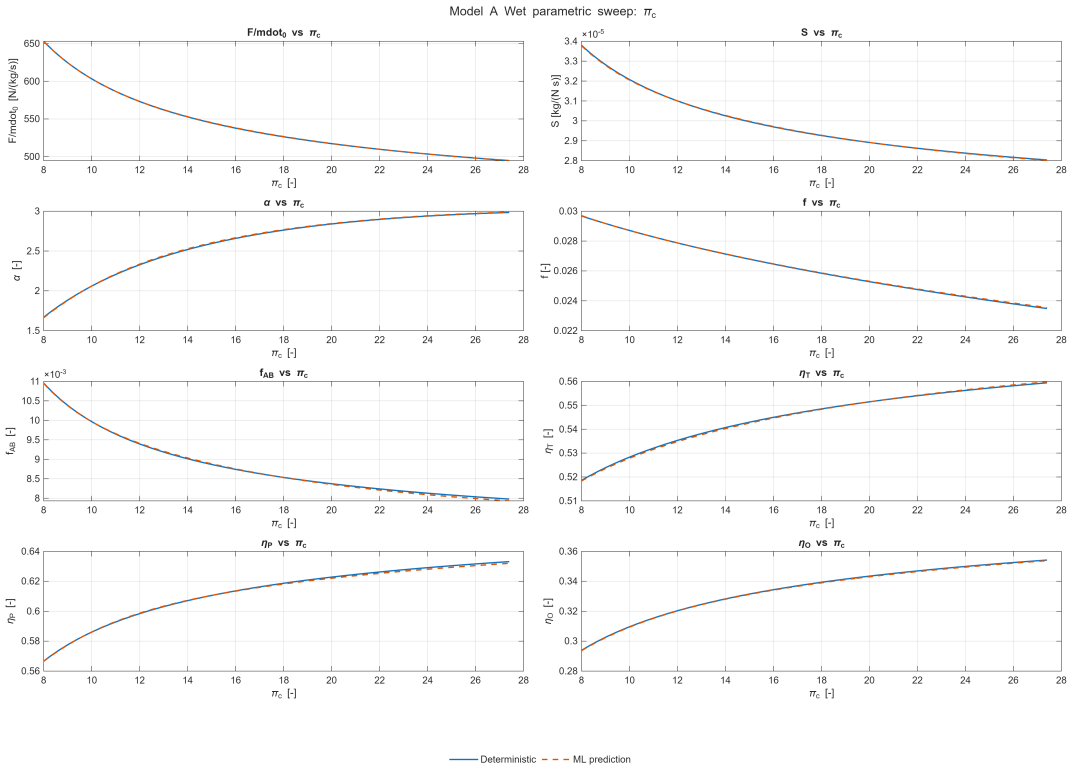


Figure 6.17 – Model A afterburning-mode parametric sweep over compressor pressure ratio

The Model B afterburning-mode compressor pressure ratio sweep is shown in Fig. 6.18. The surrogate again follows the deterministic PCA response closely. Since Model B includes active thermochemical variation, this result shows that the wet expert preserves the expected π_c trend while also accounting for variation in gas properties and fuel heating value.

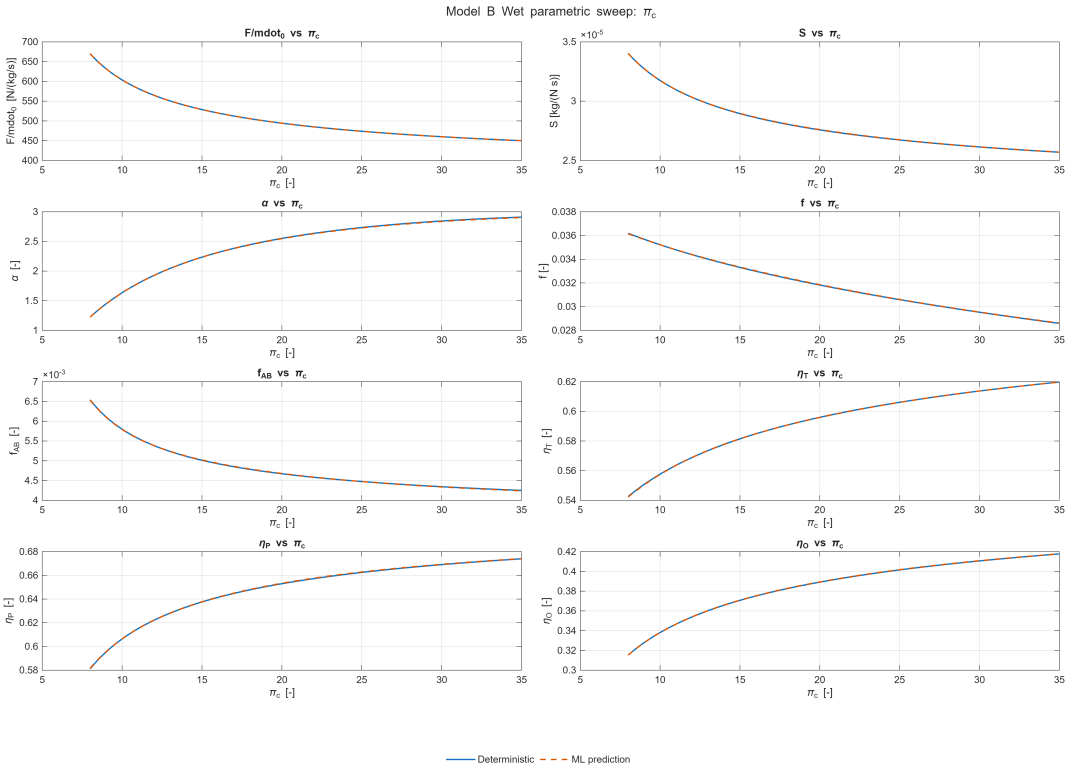


Figure 6.18 – Model B afterburning-mode parametric sweep over compressor pressure ratio

The Model C afterburning-mode compressor pressure ratio sweep is shown in Fig. 6.19. This case includes real component losses, afterburner pressure loss, mixer pressure recovery, and nozzle loss. Despite the increased coupling, the surrogate captures the deterministic trends over the supported interval, though with less accuracy than the ideal models. The general trend agreement indicates that the full real-cycle wet expert learned the primary effect of compressor pressure ratio on augmented-cycle performance.

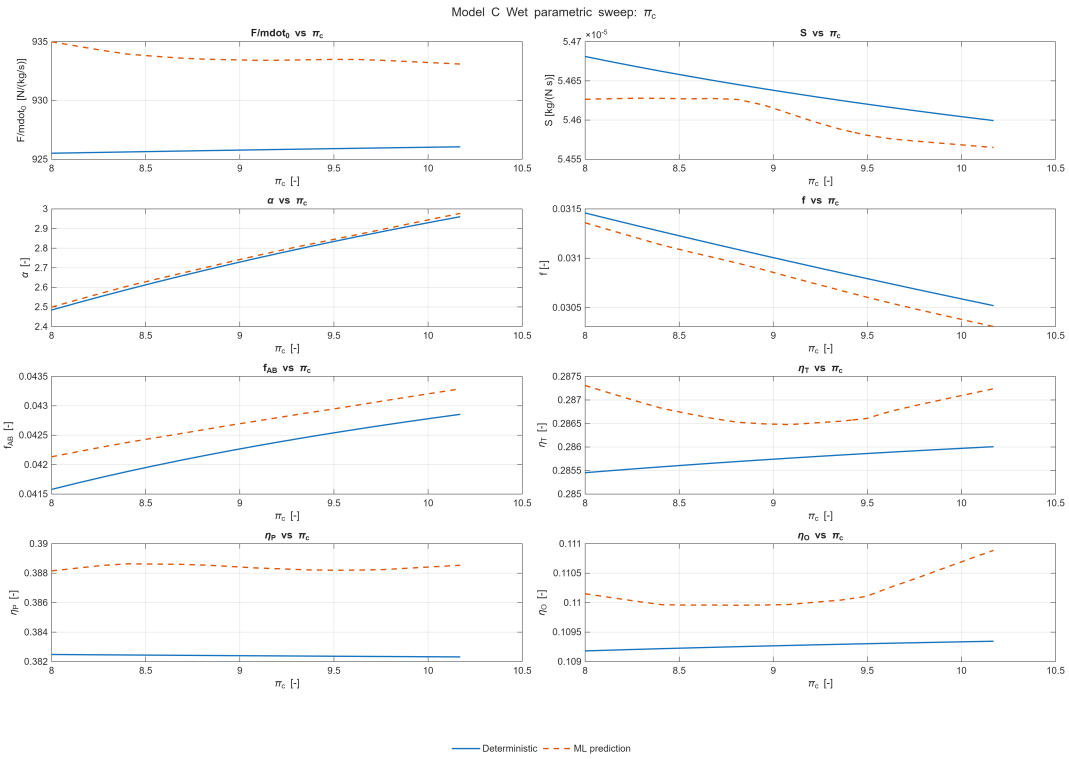


Figure 6.19 – Model C afterburning-mode parametric sweep over compressor pressure ratio

Overall, the compressor pressure ratio sweeps show strong agreement between the deterministic PCA solver and the wet surrogate predictions. Models A and B reproduce the ideal-cycle augmented response with close alignment, while Model C retains the same core compression trend in the higher-dimensional real-cycle formulation with slight accuracy loss. This indicates that π_c is one of the more reliably learned afterburning-mode directions across the model hierarchy.

6.4.3 Fan Pressure Ratio Sweep

The afterburning-mode fan pressure ratio sweeps evaluate the response of each wet expert as π_f is varied while all other inputs remain fixed at the selected feasible anchor point. Fan pressure ratio affects the bypass-stream state entering the mixer, which then influences the mixed-stream condition upstream of the afterburner. In afterburning operation, this makes π_f important because changes in bypass-stream compression can alter the temperature and pressure state that is subsequently augmented and expanded through the nozzle.

The Model A afterburning-mode fan pressure ratio sweep is shown in Fig. 6.20. The neural network prediction follows the deterministic PCA reference closely over the supported interval. The specific thrust, TSFC, bypass ratio, main-burner fuel-air ratio, afterburner fuel-air ratio, and efficiency quantities remain well aligned with the ideal-cycle solution. The same effect of the model hallucinating a trend dependency of the fuel to air ratio with the fan pressure ratio persists, though the deviation is small. This indicates that the fixed-property

wet expert captures the fan-pressure-ratio sensitivity of the augmented ideal-cycle mapping.

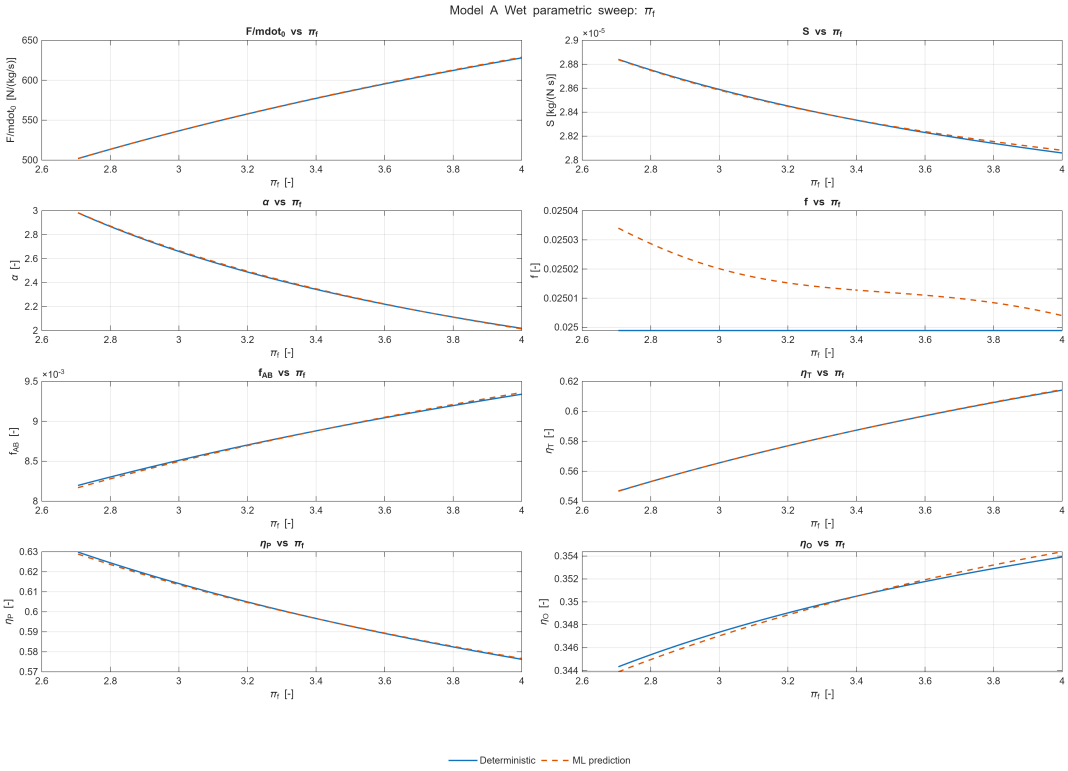


Figure 6.20 – Model A afterburning-mode parametric sweep over fan pressure ratio

The Model B afterburning-mode fan pressure ratio sweep is shown in Fig. 6.21. The surrogate captures the main deterministic trends while also accounting for the active thermochemical variation included in Model B. Small deviations are similarly visible to the fixed-property Model A case, but the overall response remains consistent with the PCA reference. The same fuel to air ratio low magnitude deviation is also observable in model B. This suggests that the variable-property wet expert still learns the general bypass-stream and mixer sensitivity associated with changing π_f.

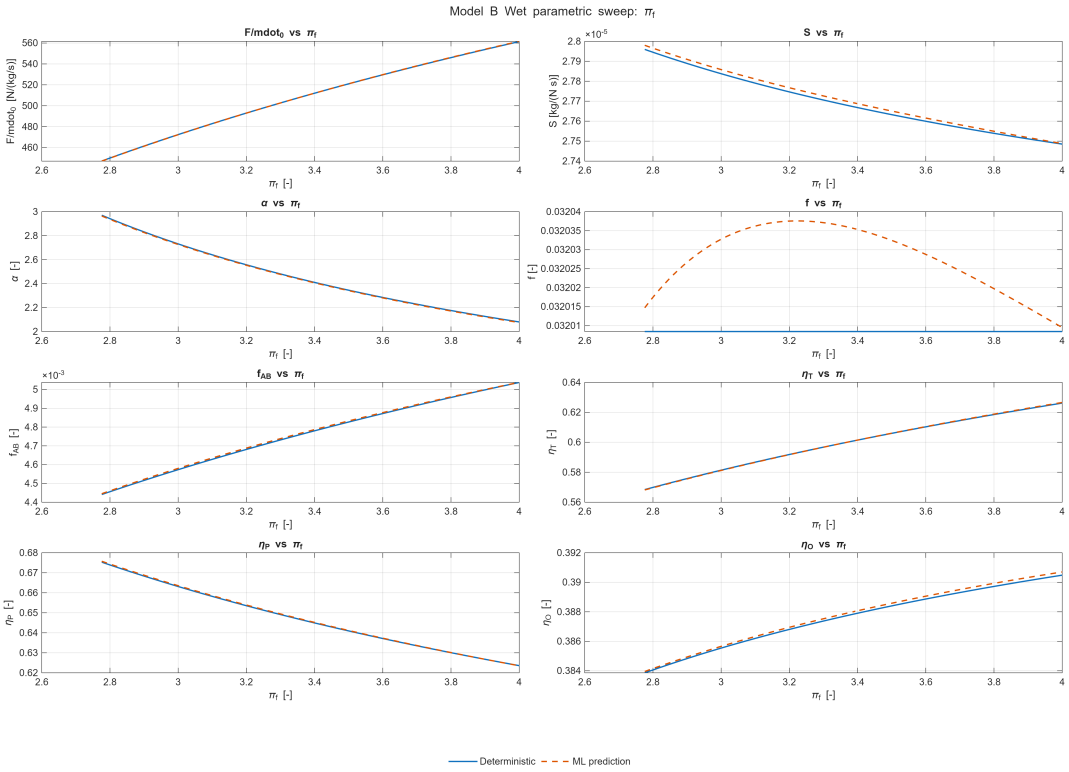


Figure 6.21 – Model B afterburning-mode parametric sweep over fan pressure ratio

The Model C afterburning-mode fan pressure ratio sweep is shown in Fig. 6.22. This sweep is more demanding because the real-cycle response couples the fan pressure ratio to component efficiencies, bypass-stream total pressure, mixer pressure recovery, afterburner pressure loss, and nozzle expansion. The surrogate reproduces the dominant PCA trend, although all metrics show larger local trend differences than those observed for the Mach number and compressor pressure ratio sweeps. Additionally the hallucinated fuel to air ratio dependency shows more statistical noise than the previous models but remains at a reasonably low error. This indicates that the fan-pressure-ratio direction remains one of the reasonably accurate directions for the real-cycle wet expert.

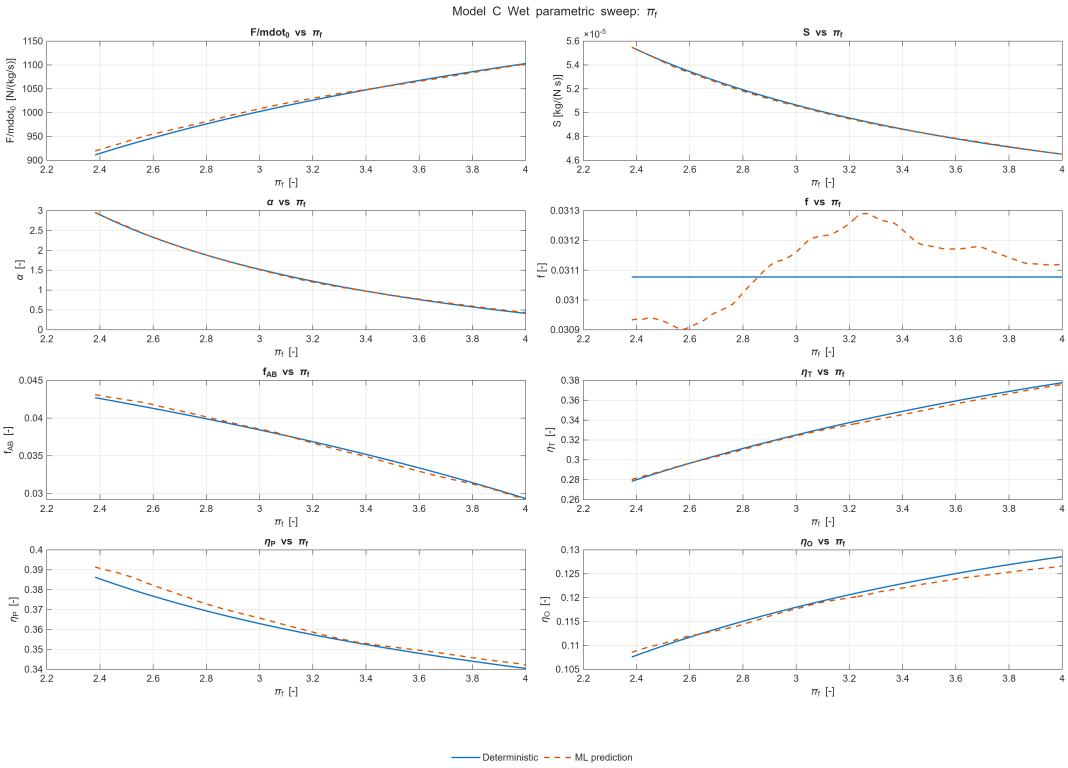


Figure 6.22 – Model C afterburning-mode parametric sweep over fan pressure ratio

Overall, the fan pressure ratio sweeps show that the wet ideal-cycle surrogates reproduce the deterministic PCA behavior with strong agreement. Model C also captures the principal real-cycle response, but with increased deviation in cycle and performance metrics due to the stronger coupling between the bypass stream, mixer, afterburner, and nozzle. This makes π_f an important direction for evaluating mixed-flow behavior in the afterburning surrogates.

6.4.4 Burner Temperature Ratio Sweep

The afterburning-mode burner temperature ratio sweeps evaluate the response of each wet expert to changes in main combustor thermal loading. For Models A and B, the swept variable is τ_λ. For Model C, the corresponding real-cycle sweep is performed using the burner exit total temperature T_{t4}. This sweep affects the core stream before mixing and therefore changes the thermodynamic state entering the mixer, afterburner, and nozzle. The burner temperature trends have been seen to be less accurately predicted by the surrogate in the non-afterburning configuration, and this is expected to continue to be the case for the afterburning configuration.

The Model A afterburning-mode burner temperature ratio sweep is shown in Fig. 6.23. The neural network prediction closely follows the deterministic PCA reference over the supported interval. The trends in specific thrust, TSFC, bypass ratio, main-burner fuel-air ratio, and fuel-air ratios remain well aligned with the ideal-cycle solution, although with visible

oscillatory behavior. This same inaccuracy can be seen in the efficiency predictions, as the surrogate prediction deviates from the deterministic solution. The general trend agreement confirms that the fixed-property wet expert captures the main combustor thermal sensitivity of the augmented ideal-cycle mapping.

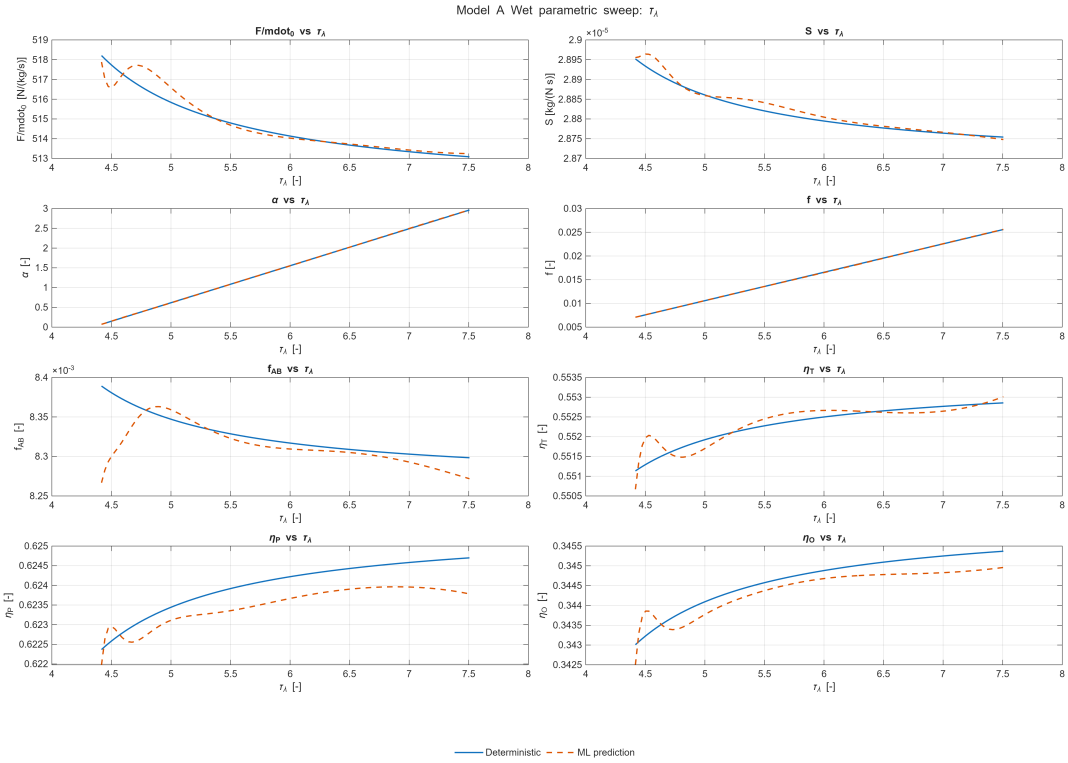


Figure 6.23 – Model A afterburning-mode parametric sweep over burner temperature ratio

The Model B afterburning-mode burner temperature ratio sweep is shown in Fig. 6.24. The surrogate again reproduces the general deterministic PCA trends with similar low magnitude oscillatory error. Since Model B includes active thermochemical variation, this result indicates that the wet expert preserves the expected combustor thermal-loading response while accounting for changes in gas properties and fuel heating value.

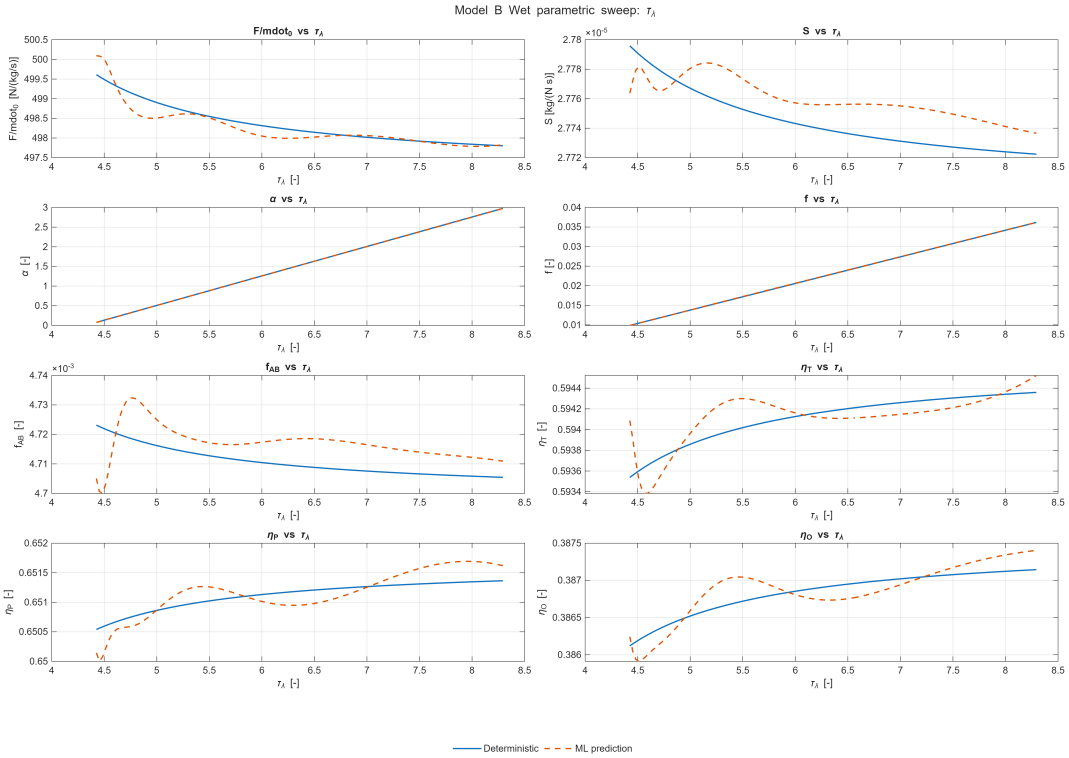


Figure 6.24 – Model B afterburning-mode parametric sweep over burner temperature ratio

The Model C afterburning-mode burner exit total-temperature sweep is shown in Fig. 6.25. This real-cycle sweep is more difficult because the effect of T_{t4} propagates through the turbine work balance, mixer state, afterburner fuel requirement, and nozzle expansion process. The surrogate captures the broad deterministic response for the required bypass ratio and fuel to air ratios, although larger local differences appear than in the ideal-cycle cases. Additionally, there is a significant discrepancy between the predicted and deterministic trend for the specific thrust trend, which show a need for further tuning. These deviations are consistent with the stronger coupling and higher dimensionality of the real-cycle afterburning formulation, making it challenging to model.

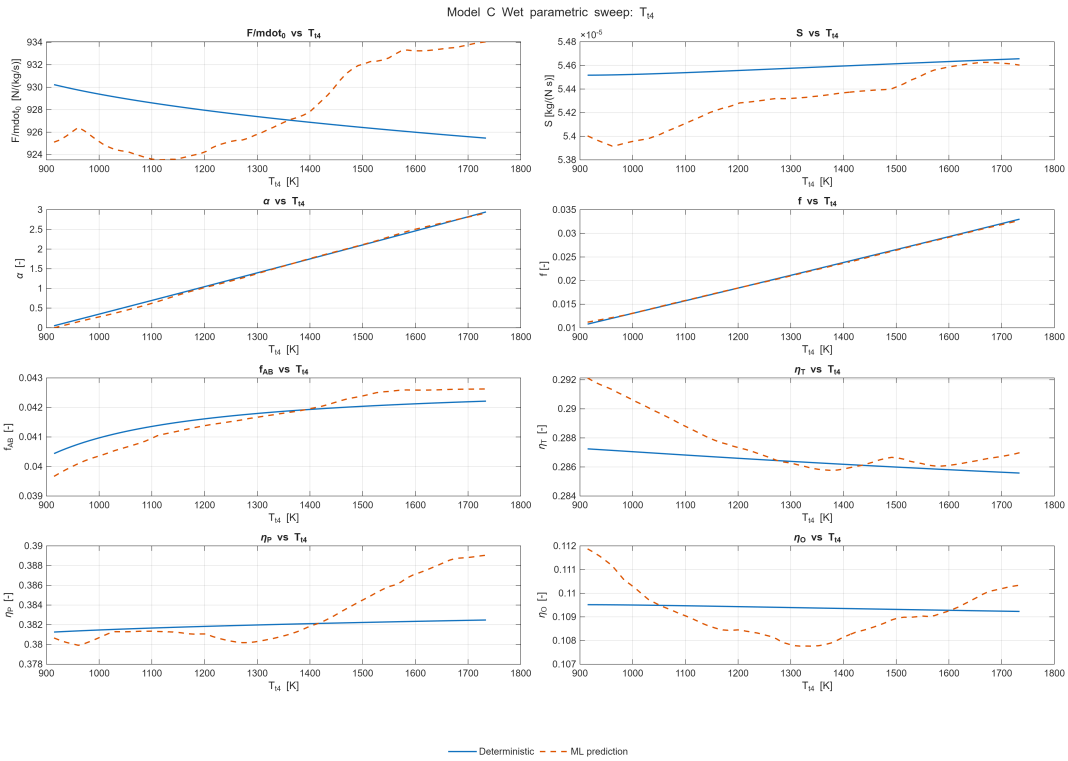


Figure 6.25 – Model C afterburning-mode parametric sweep over burner exit total temperature

Overall, the burner temperature ratio sweeps show reasonable agreement for the ideal-cycle wet experts and more challenging behavior for the real-cycle wet expert. Models A and B reproduce the deterministic combustor thermal-loading response closely. Model C captures the main trend, but with larger deviations in selected quantities because T_{t4} influences several coupled real-cycle relations downstream of the main burner.

6.4.5 Afterburner Temperature Ratio Sweep

The afterburner temperature ratio sweeps evaluate the response of each wet expert to changes in downstream heat addition. For Models A and B, the swept variable is τ_{AB}. For Model C, the corresponding real-cycle sweep is performed using the afterburner exit total temperature T_{t7}. This sweep is central to the augmented cycle because it directly affects afterburner fuel-air ratio, exhaust temperature, nozzle expansion, specific thrust, and TSFC.

The Model A afterburning-mode afterburner temperature ratio sweep is shown in Fig. 6.26. The neural network prediction closely follows the deterministic PCA reference across the supported interval. The increase in afterburner thermal loading is reflected in the thrust, afterburner fuel-air ratio, and TSFC trends, while the surrogate remains well aligned with the ideal-cycle solution. This confirms that the fixed-property wet expert captures the primary effect of afterburner heat addition. The model fails to capture the independence of the bypass

ratio and burner fuel to air ratio on the afterburner state, but maintains the prediction close to expected values.

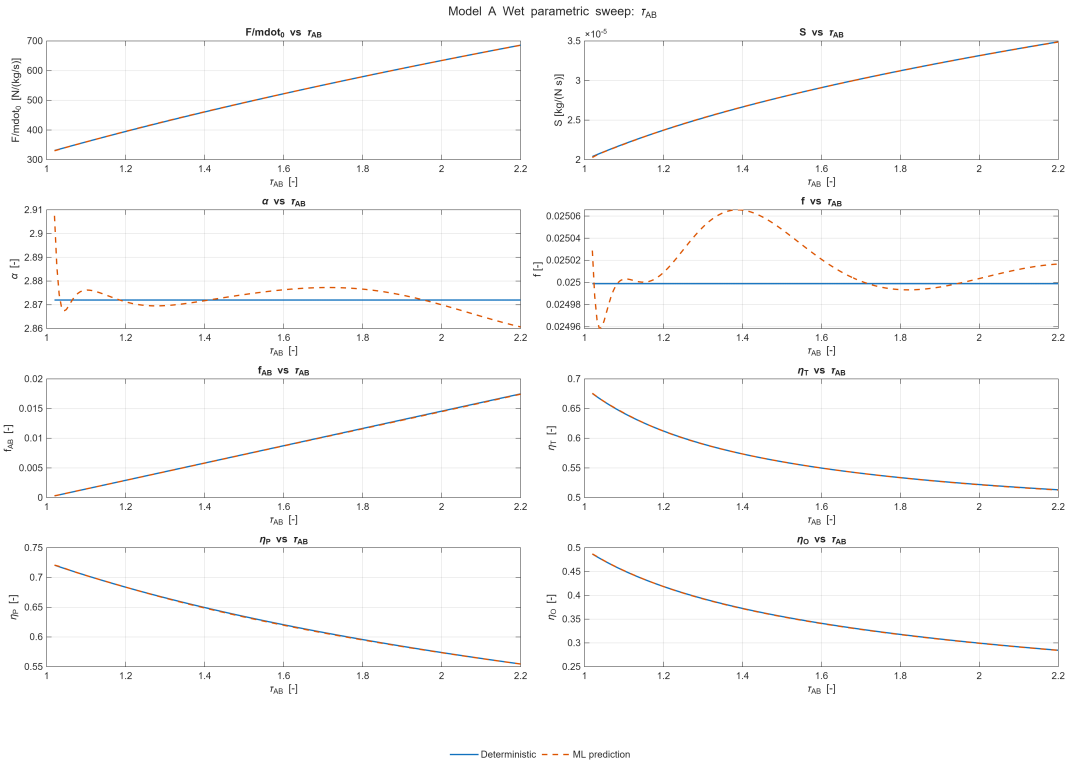


Figure 6.26 – Model A afterburning-mode parametric sweep over afterburner temperature ratio

The Model B afterburning-mode afterburner temperature ratio sweep is shown in Fig. 6.27. The surrogate again reproduces the main deterministic PCA trends while accounting for thermochemical variation. This indicates that the variable-property wet expert learns the dependence of afterburner fuel addition and augmented thrust on τ_{AB}, rather than only fitting the baseline burner and compression variables. The same failure to capture the independence of the bypass ratio and main burner fuel to air ratio is also observed, with the statistical error remaining low.

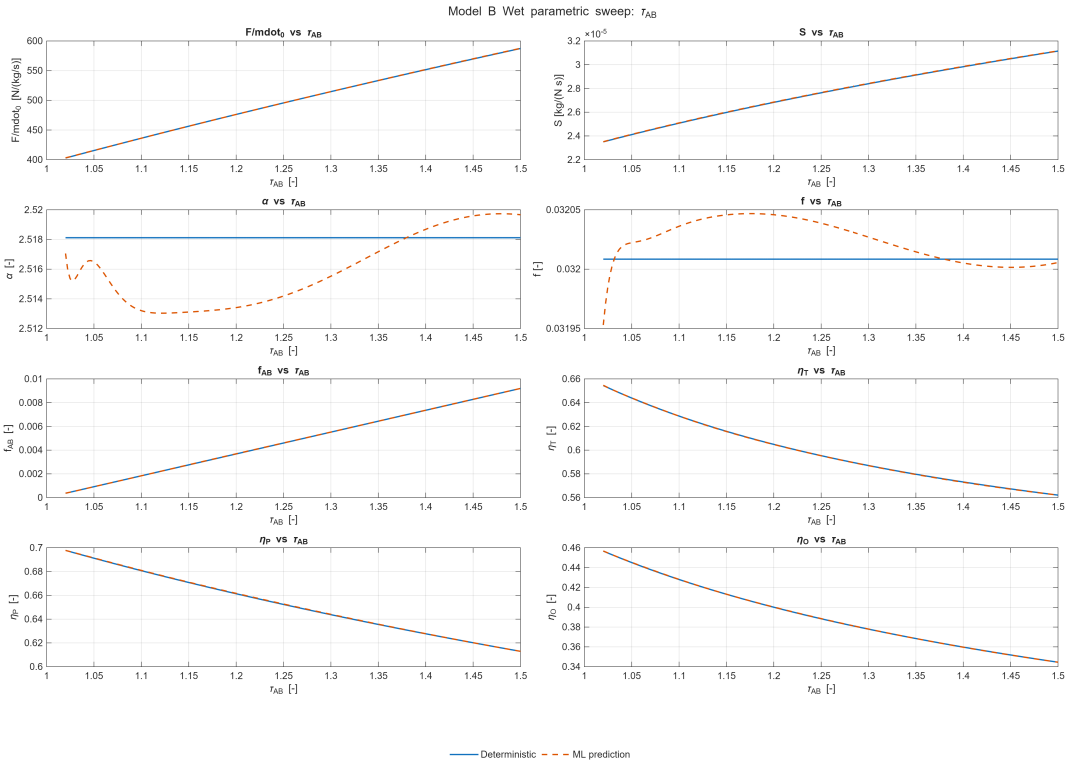


Figure 6.27 – Model B afterburning-mode parametric sweep over afterburner temperature ratio

The Model C afterburning-mode afterburner exit total-temperature sweep is shown in Fig. 6.28. This sweep is the real-cycle equivalent of the ideal afterburner temperature-ratio sweep. The surrogate captures the broad deterministic response as T_{t7} changes, including the associated changes in afterburner fuel-air ratio, specific thrust, and TSFC. Because T_{t7} is coupled to afterburner efficiency, afterburner pressure loss, gas-property variation, and nozzle expansion, this sweep is more intensive than the corresponding ideal-cycle cases and the resulting slight oscillatory error can be seen as a consequence. This error is most visible in the independent metrics over the sweep similarly to models A and B. Overall the trend agreement between the deterministic model and the surrogate remains strong over variations in T_{t7}.

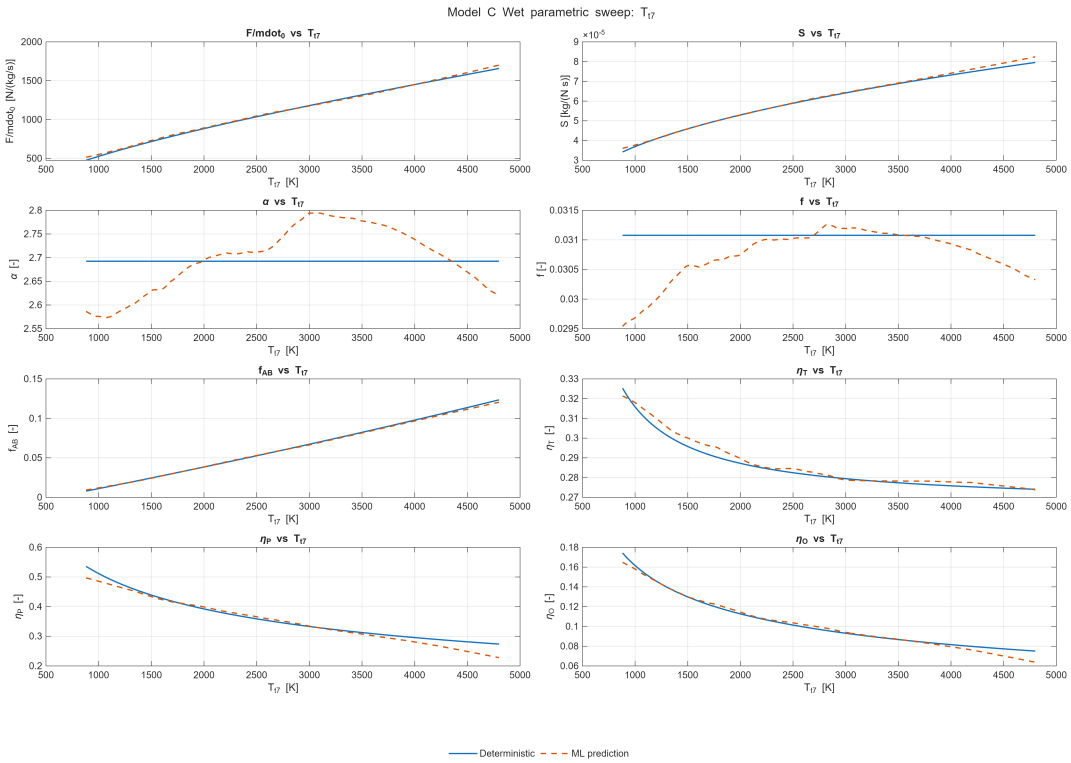


Figure 6.28 – Model C afterburning-mode parametric sweep over afterburner exit total temperature

Overall, the afterburner temperature sweeps show that the wet experts learned the dominant augmented-cycle heat-addition response. Models A and B closely follow the deterministic ideal-cycle solutions, while Model C captures the main real-cycle trend with the additional complexity introduced by component losses and mode-dependent gas properties. This result is important because afterburner heat addition is one of the primary variables that potential control studies would need to manipulate for thrust targeting.

6.4.6 Afterburning-Mode Sweep Summary

The afterburning-mode parametric sweeps show that the wet experts generally reproduce the deterministic PCA trends over the supported sweep intervals. Models A and B show close agreement across the Mach number, compressor pressure ratio, fan pressure ratio, burner temperature ratio, and afterburner temperature ratio sweeps. This confirms that the ideal-cycle wet experts learned both the baseline turbofan response and the additional effects of afterburner heat addition.

Model C also captures the dominant afterburning-mode trends, although larger local deviations appear in the more strongly coupled real-cycle directions. These differences are expected because the real-cycle wet mapping includes afterburner pressure loss, afterburner efficiency, mixer pressure recovery, nozzle loss, and mode-dependent gas properties. The

burner exit total temperature, fan pressure ratio, and afterburner exit total temperature sweeps are therefore more demanding than the corresponding ideal-cycle cases.

Overall, the afterburning-mode sweeps support the use of the trained wet experts as forward surrogates within the sampled operating envelope. The results also show that afterburner heat addition is captured well enough for the trained forward model to serve as a foundation for future inverse-control or thrust-targeting studies.

6.5 Extrapolation and Edge-Envelope Behavior

The extrapolation study evaluates how the trained surrogates behave near and outside the sampled training envelope. While the parametric sweep results focused primarily on supported in-domain behavior, extrapolation testing is needed to identify where the neural network predictions begin to lose physical reliability. This is especially important because a trained surrogate can still produce smooth numerical outputs outside the region where the underlying training data provides meaningful support.

The extrapolation plots use the same anchor-based sweep structure as the parametric sweep study, but the swept variable is extended beyond its original training range. The deterministic PCA solver and neural network surrogate are then compared over the extended interval where valid reference solutions are available. Training bounds and dataset-supported regions are marked on the plots so that interpolation, edge-envelope behavior, and true extrapolation can be distinguished visually.

The discussion is organized by model family. This structure is used because each model has a different level of physical complexity and therefore exhibits different extrapolation behavior. Models A and B provide the ideal-cycle baseline cases, while Model C shows how the full real-cycle surrogate behaves when component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, and mode-dependent gas properties are included.

The purpose of this section is not to claim that the surrogates are reliable outside the sampled envelope. Instead, the extrapolation study identifies the types of failure modes that appear when the models are pushed beyond their training support. These results help define the practical operating limits of the trained surrogates and motivate the model-improvement recommendations presented later in the chapter.

6.5.1 Model A Extrapolation Behavior

Model A provides the lowest-complexity extrapolation case because it uses the fixed-property ideal-cycle formulation. As a result, the Model A extrapolation plots are useful for distinguishing basic neural network edge-of-domain behavior from the additional complications introduced later by thermochemical variability and real-cycle losses. In each plot, the deterministic PCA solution is shown only where the cycle evaluation remains valid, while the neural network prediction continues over the extended sweep range.

The Model A dry Mach number extrapolation sweep is shown in Fig. 6.29. Inside the supported region, the neural network prediction follows the deterministic PCA solution closely. Once the sweep moves beyond the supported manifold boundary, the prediction continues smoothly but no longer follows a verified deterministic reference. The most visible deviation occurs at high Mach number, where the predicted thrust, TSFC, fuel-air ratio, and efficiency quantities continue as extrapolated neural network behavior rather than validated cycle behavior. At Mach numbers below the supported training boundary, the model maintains a reasonable prediction relative to the expected deterministic solution.

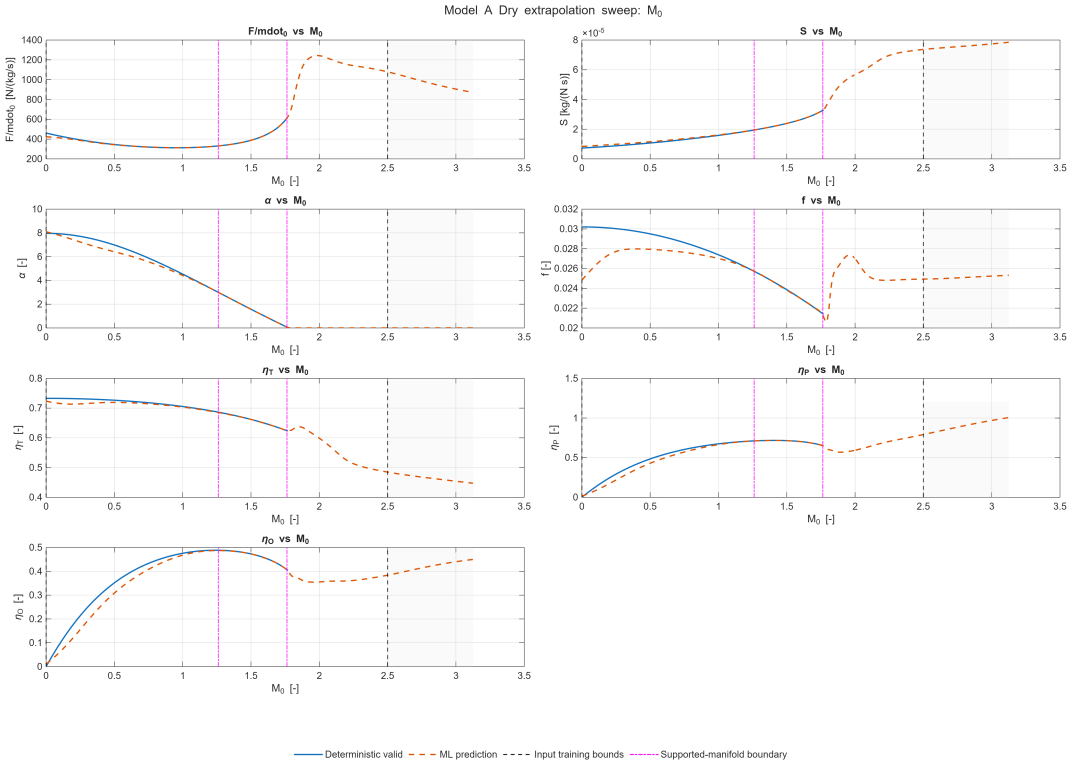


Figure 6.29 – Model A dry-mode extrapolation sweep over freestream Mach number

The Model A dry compressor pressure ratio extrapolation sweep is shown in Fig. 6.30. The surrogate remains well aligned with the deterministic solution across most of the valid and supported region. The largest departures occur near the lower edge of the compressor pressure ratio range, where the PCA solution changes rapidly and the network produces a smooth continuation outside the well-supported region. At higher compressor pressure ratios, the extrapolated prediction remains comparatively stable.

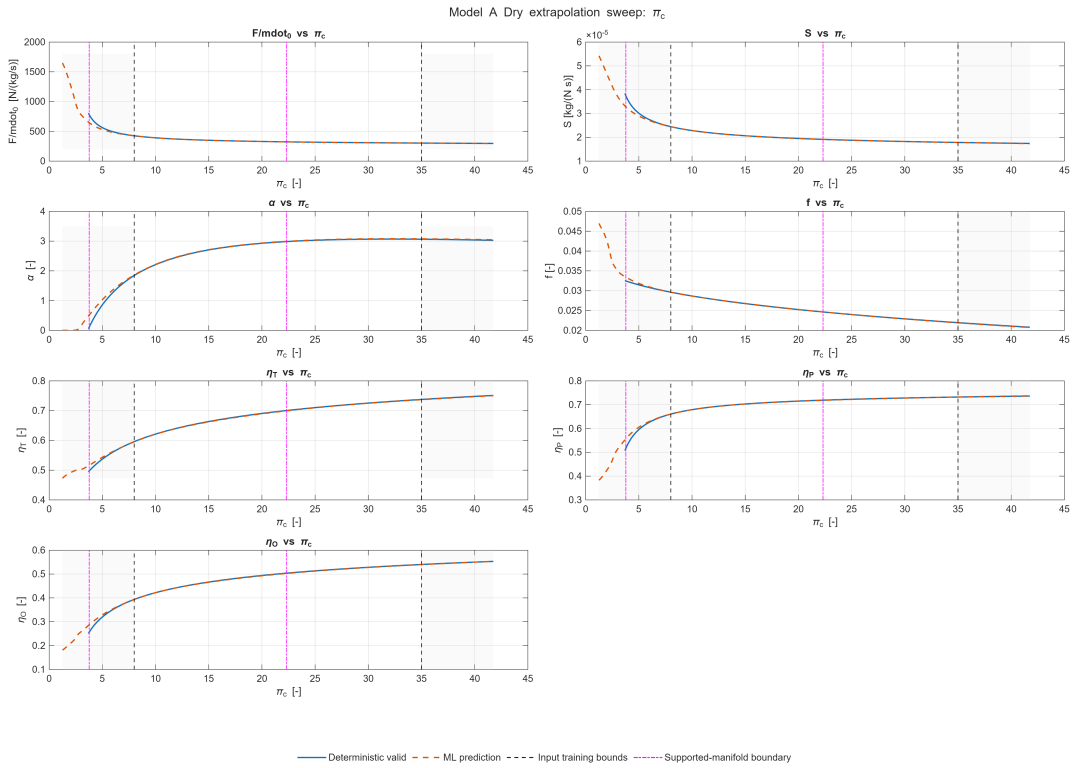


Figure 6.30 – Model A dry-mode extrapolation sweep over compressor pressure ratio

The Model A dry fan pressure ratio extrapolation sweep is shown in Fig. 6.31. This sweep shows a more pronounced edge-envelope sensitivity near the lower fan pressure ratio boundary. The deterministic solution changes sharply near the low-π_f region, while the neural network produces a smoother continuation that does not reproduce the sharp local behavior. The agreement improves once the sweep moves into the supported range, and the prediction remains close to the PCA trend across the upper portion of the sweep.

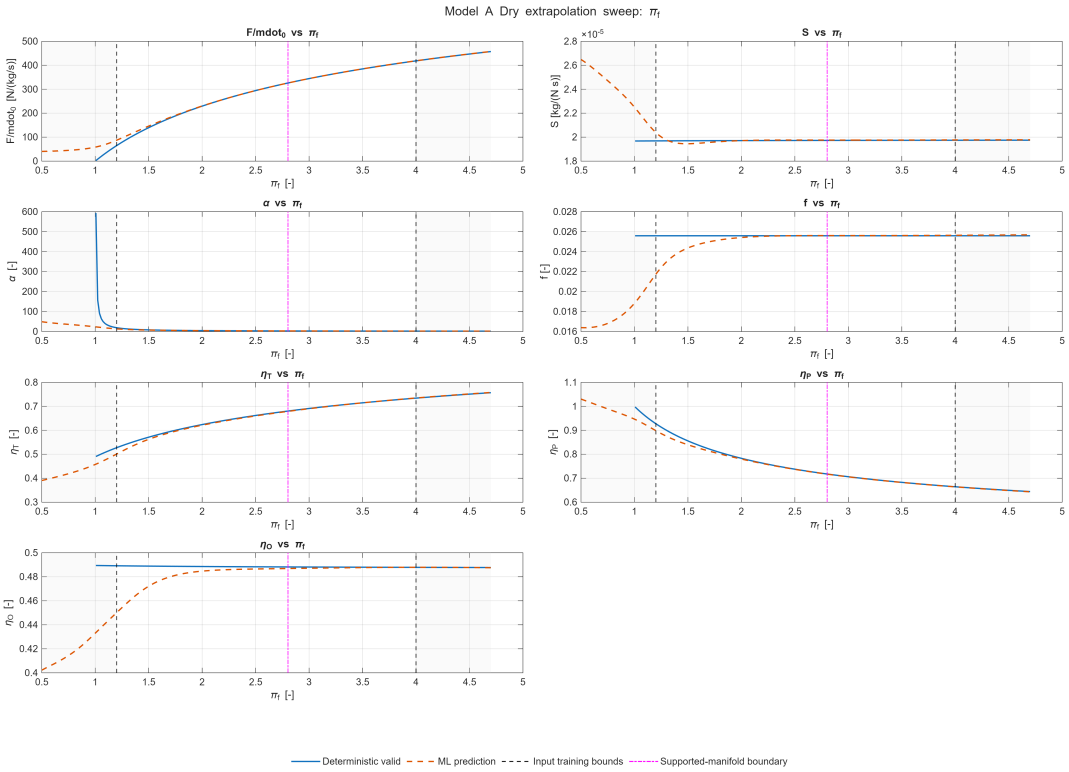


Figure 6.31 – Model A dry-mode extrapolation sweep over fan pressure ratio

The Model A dry burner temperature ratio extrapolation sweep is shown in Fig. 6.32. Inside the supported interval, the surrogate follows the deterministic trends for thrust, TSFC, bypass ratio, and main-burner fuel-air ratio. Outside the lower bound, the network produces a smooth but physically unreliable continuation. At the upper end of the sweep, the thrust and fuel-air-ratio predictions remain reasonable, while the efficiency predictions begin to drift more visibly from the deterministic trend. This indicates that efficiency quantities are more sensitive to extrapolation than the primary thrust and fuel-flow outputs.

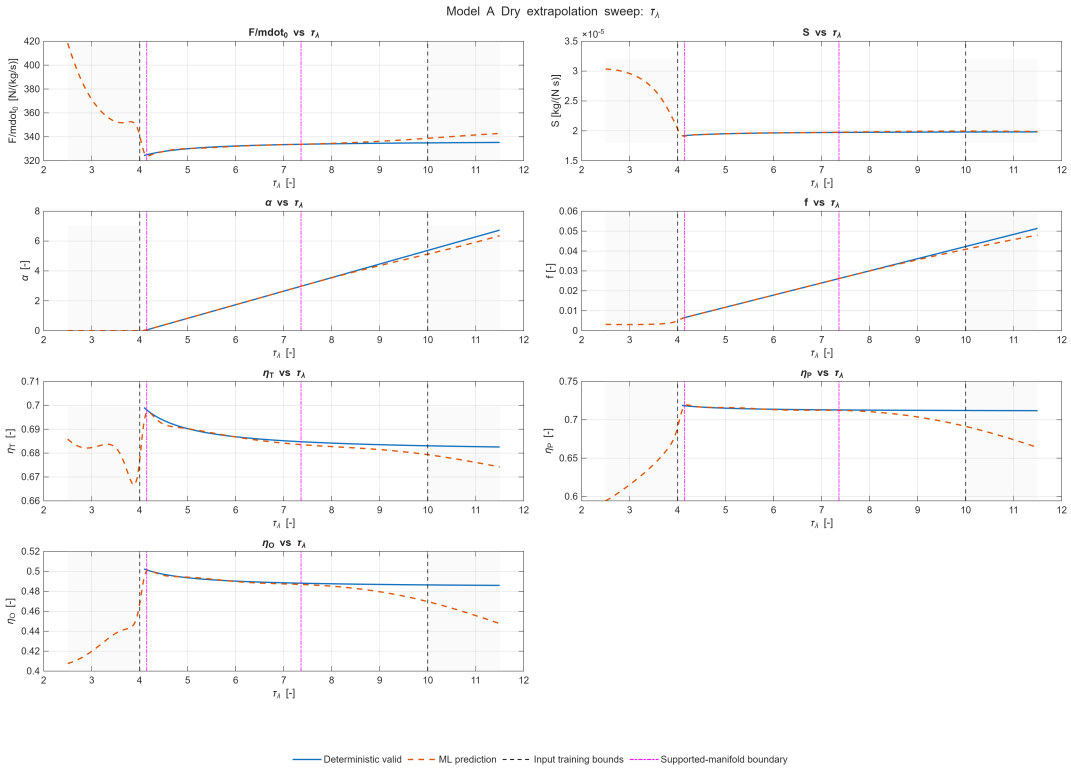


Figure 6.32 – Model A dry-mode extrapolation sweep over burner temperature ratio

The Model A wet Mach number extrapolation sweep is shown in Fig. 6.33. The in-domain portion again shows close agreement between the PCA and neural network curves. Beyond the supported Mach-number region, the wet expert produces large changes in predicted specific thrust, TSFC, afterburner fuel-air ratio, and efficiency quantities. In the low Mach number regions left of the supported training boundary the surrogate maintains a reasonable accuracy closely following the expected deterministic solutions. This is a clear example of the model generating mathematically smooth outputs outside the region where the augmented-cycle mapping is supported by the training data.

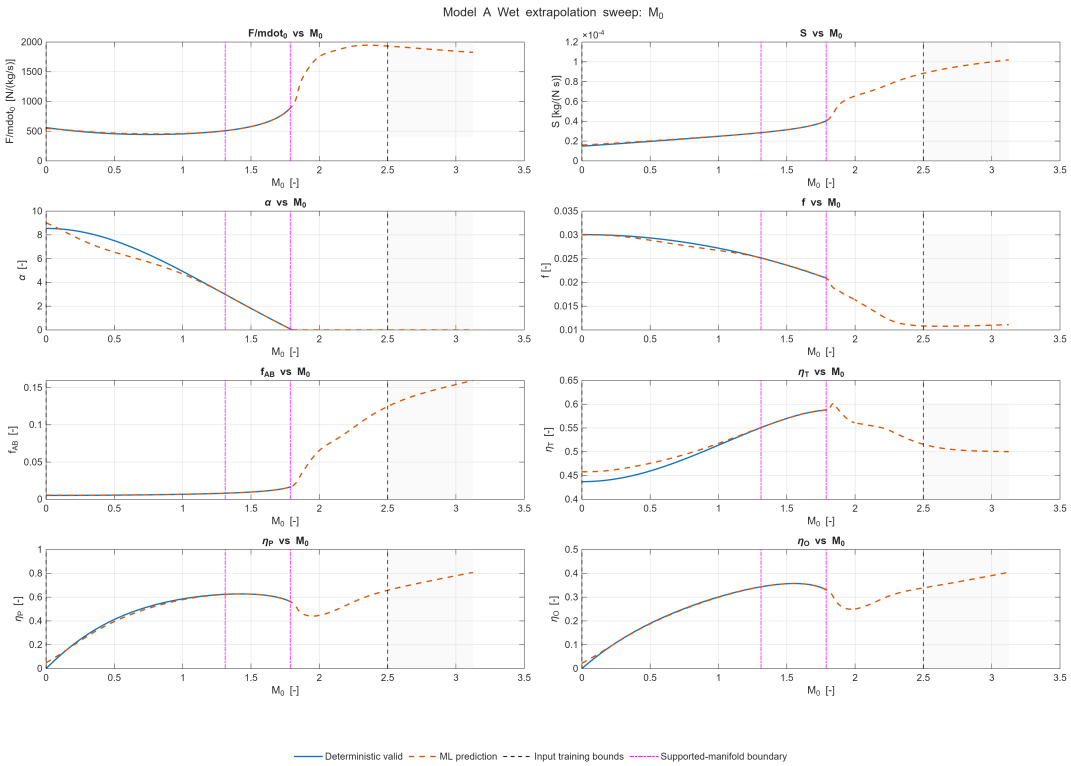


Figure 6.33 – Model A afterburning-mode extrapolation sweep over freestream Mach number

The Model A wet compressor pressure ratio extrapolation sweep is shown in Fig. 6.34. The surrogate tracks the deterministic PCA solution well through the supported region and across much of the upper pressure-ratio range. Larger deviations appear near the lower edge of the sweep, where thrust, TSFC, afterburner fuel-air ratio, and efficiency predictions become more sensitive. In large compressor pressure ratio areas outside of the supported training boundary the surrogate maintains its accuracy, matching the deterministic solution closely. This behavior is consistent with the dry case, where low compressor pressure ratio also produced noticeable edge behavior.

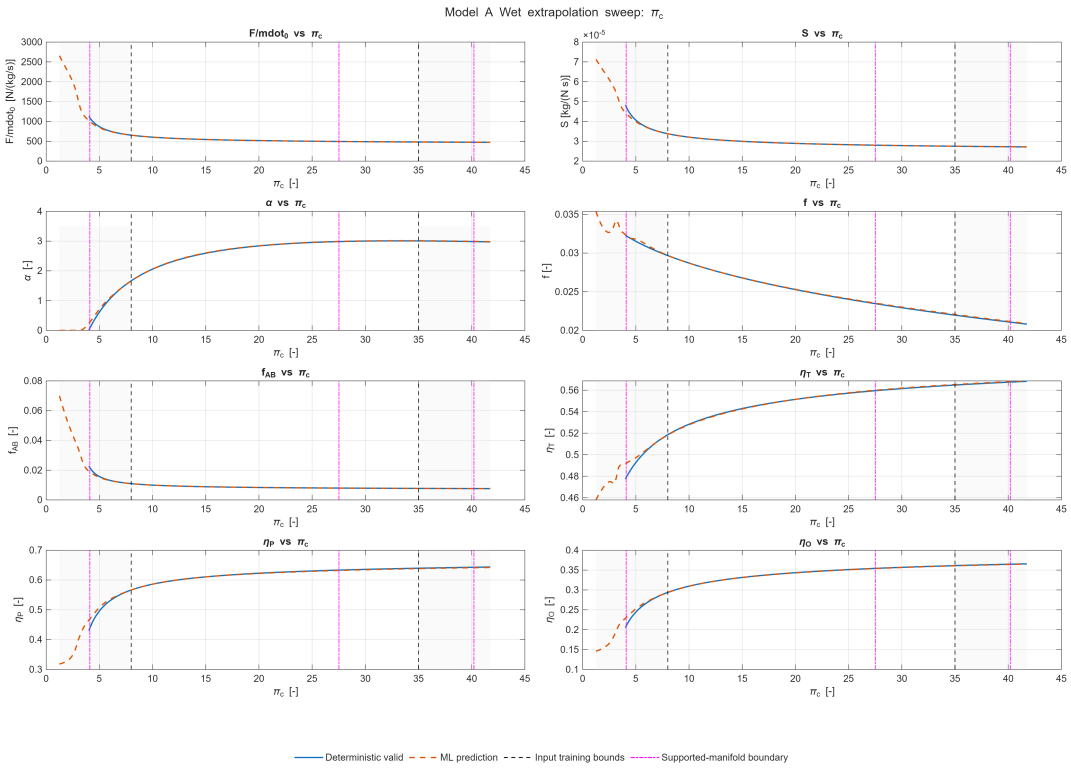


Figure 6.34 – Model A afterburning-mode extrapolation sweep over compressor pressure ratio

The Model A wet fan pressure ratio extrapolation sweep is shown in Fig. 6.35. As in the dry case, the lower fan pressure ratio region produces a large mismatch. The network prediction smooths through the edge behavior instead of reproducing the sharp deterministic response near the low-π_f boundary. Within the supported interval and toward the upper part of the sweep, the surrogate follows the PCA trend more closely.

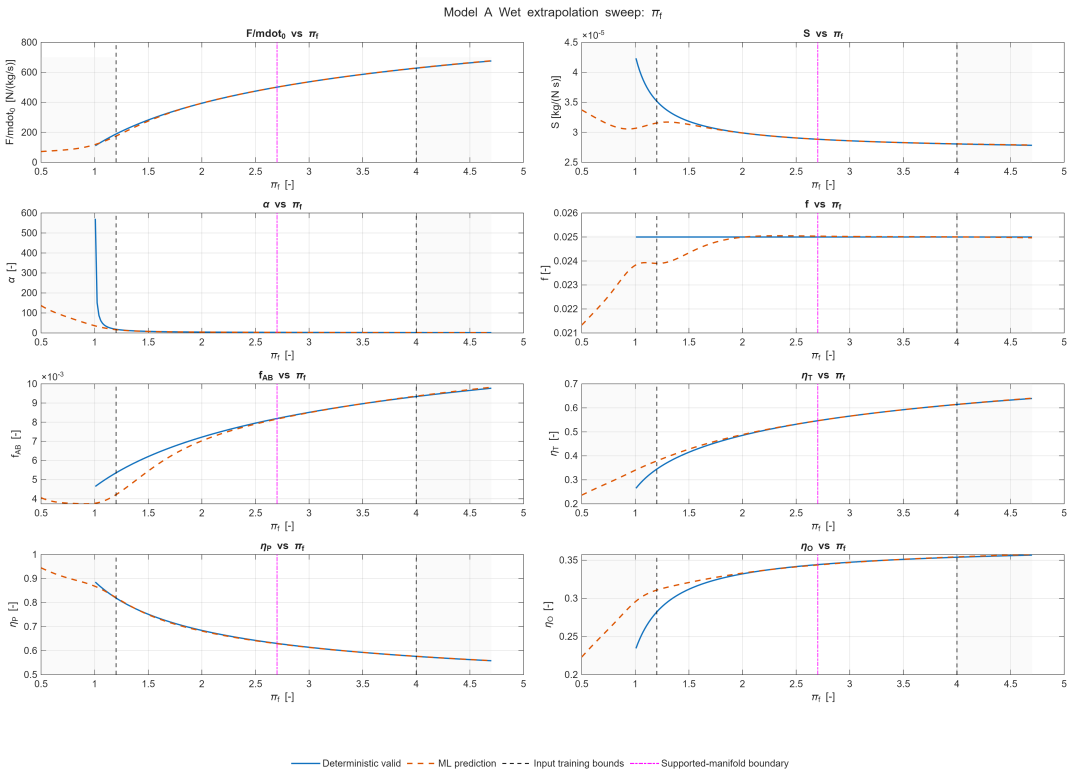


Figure 6.35 – Model A afterburning-mode extrapolation sweep over fan pressure ratio

The Model A wet burner temperature ratio extrapolation sweep is shown in Fig. 6.36. The surrogate remains closely aligned with the deterministic solution inside the supported region. Outside the lower training boundary, the predictions depart strongly from the reference trend, particularly for thrust, TSFC, afterburner fuel-air ratio, and efficiency quantities. At the upper end of the sweep, the primary thrust and fuel-flow trends remain comparatively stable, while the efficiency quantities show more visible extrapolation drift.

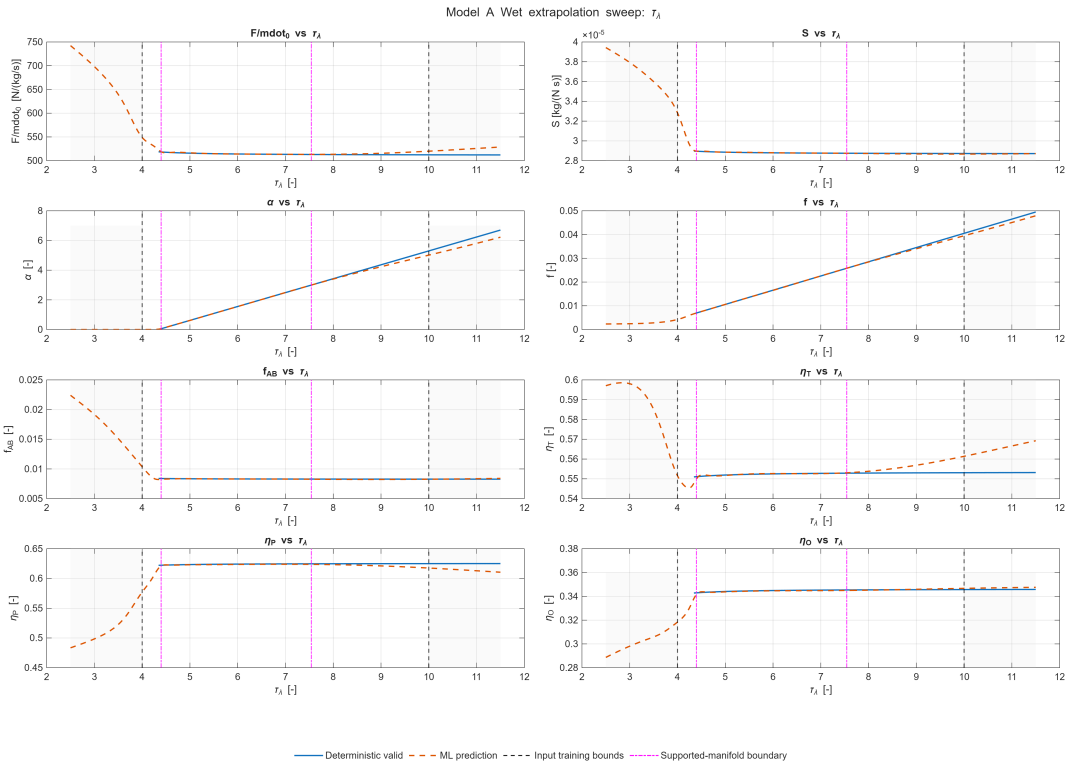


Figure 6.36 – Model A afterburning-mode extrapolation sweep over burner temperature ratio

The Model A wet afterburner temperature ratio extrapolation sweep is shown in Fig. 6.37. This is the most stable wet extrapolation direction for Model A. The surrogate closely follows the deterministic PCA trend over the plotted range for thrust, TSFC, afterburner fuel-air ratio, and efficiency outputs. When testing on the lower end of the afterburner temperature ratio, the predicted trend continues with some deviation even though an afterburner temperature ratio of less that unity does not make physical sense. This indicates that the wet expert learned a comparatively smooth and well-conditioned relationship between afterburner heat addition and the augmented-cycle outputs.

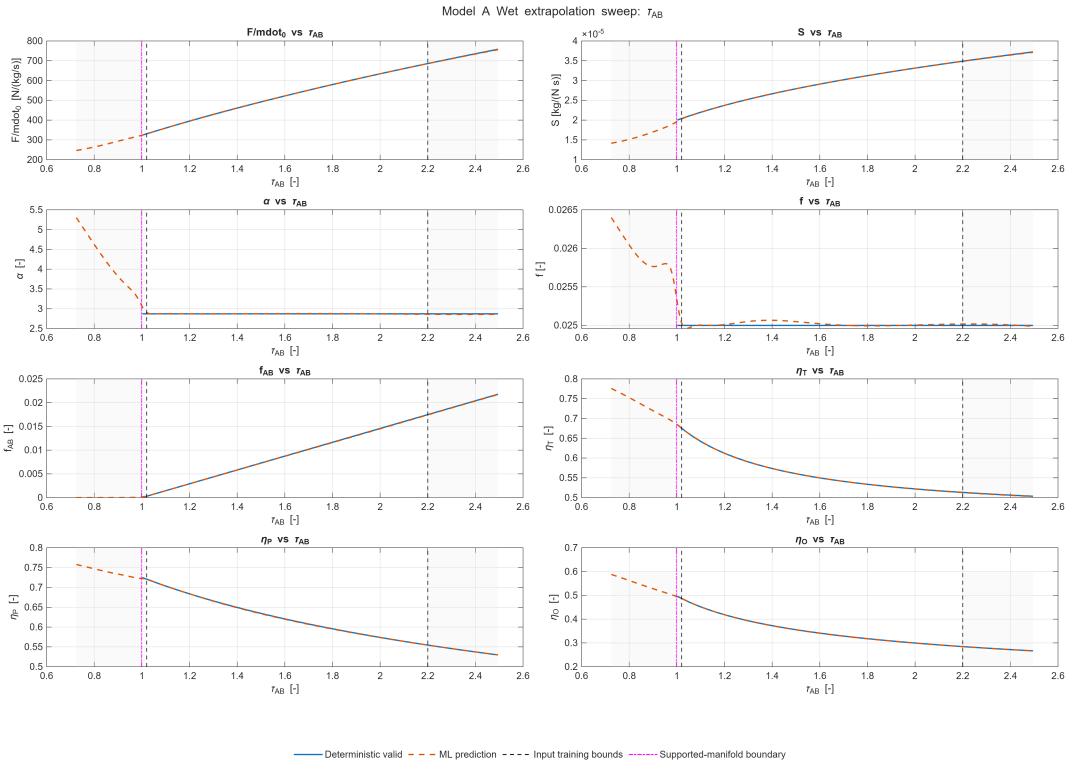


Figure 6.37 – Model A afterburning-mode extrapolation sweep over afterburner temperature ratio

Overall, Model A behaves well inside the supported training envelope and retains reasonable trend behavior for several upper-bound extrapolation directions. The dominant failure modes appear near lower fan pressure ratio, lower burner temperature ratio, lower compressor pressure ratio, and high Mach number beyond the supported manifold. These failures are not unexpected: the network is not constrained to satisfy the cycle equations outside the sampled domain, so it produces smooth extrapolated continuations rather than physically guaranteed solutions. The Model A results therefore establish the main interpretation used for the later models: in-domain agreement is meaningful, but out-of-envelope predictions must be treated as unreliable unless verified by the deterministic PCA solver.

6.5.2 Model B Extrapolation Behavior

Model B provides the variable-thermochemistry ideal-cycle extrapolation case. Compared with Model A, the governing cycle formulation remains ideal, but the surrogate must also account for variation in γ, c_p, and h_{PR}. The Model B extrapolation plots therefore show how the ideal-cycle surrogate behaves outside the sampled envelope once thermochemical variation has been introduced.

The Model B dry Mach number extrapolation sweep is shown in Fig. 6.38. Inside the supported region, the neural network prediction follows the deterministic PCA reference closely.

Beyond the upper supported Mach boundary, the network continues with a smooth extrapolated response while the deterministic valid region is limited. The largest deviations occur at high Mach number, where thrust, TSFC, and efficiency quantities become increasingly dependent on unsupported extrapolated behavior. Similarly to model A, the surrogate prediction extrapolates the trends well for low mach number regions on the left of the supported training boundary.

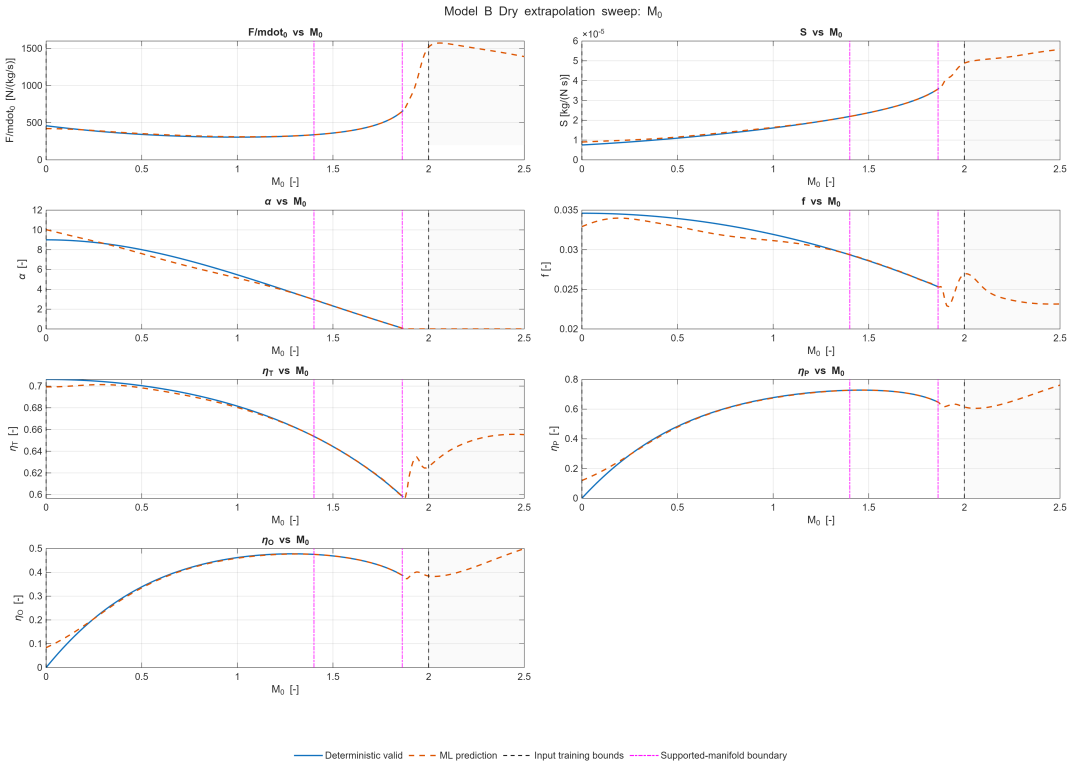


Figure 6.38 – Model B dry-mode extrapolation sweep over freestream Mach number

The Model B dry compressor pressure ratio extrapolation sweep is shown in Fig. 6.39. The surrogate is well aligned with the PCA solution through the supported interval and remains comparatively stable at higher compressor pressure ratios. The strongest edge behavior occurs near the lower pressure-ratio boundary, where the deterministic response changes rapidly and the network produces a smoother continuation. This is consistent with the Model A result, but the added thermochemical input variation makes the edge region slightly more demanding.

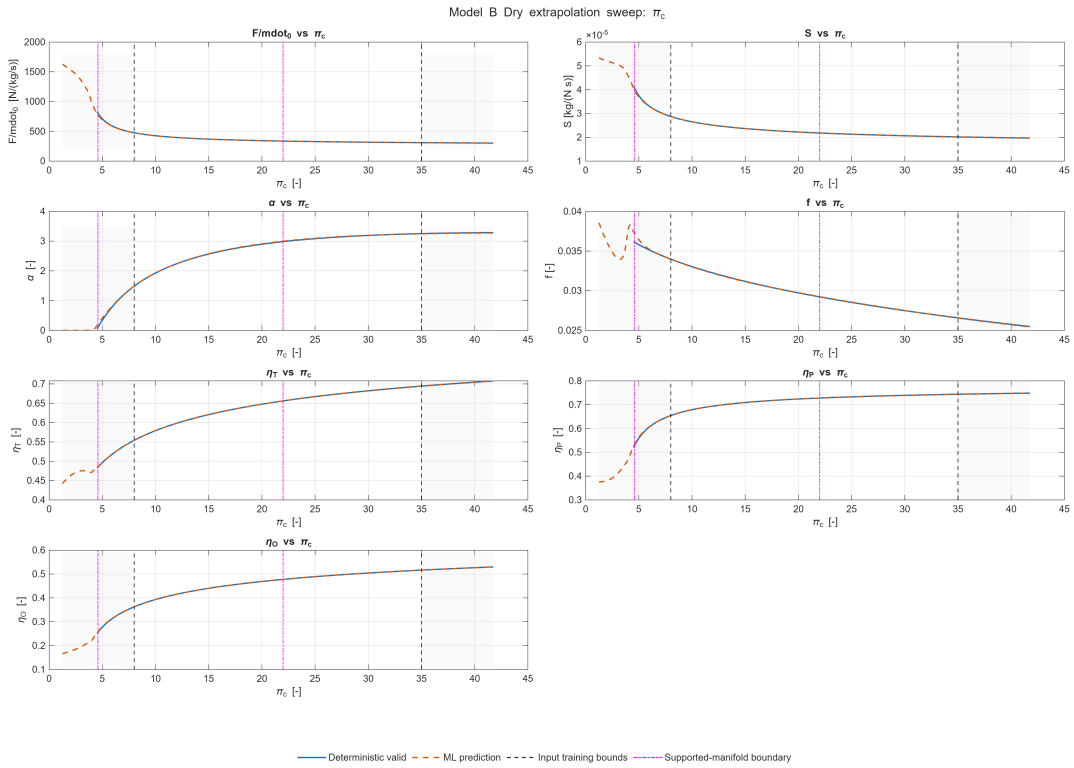


Figure 6.39 – Model B dry-mode extrapolation sweep over compressor pressure ratio

The Model B dry fan pressure ratio extrapolation sweep is shown in Fig. 6.40. This sweep shows a strong sensitivity near the lower fan pressure ratio boundary. The deterministic bypass-ratio behavior changes sharply in this region, while the neural network response remains smooth and cannot reproduce the near-boundary change with the same local structure. As the fan pressure ratio approaches unity, the deterministic model equations determine the bypass ratio to be infinite, and as such the solutions in the extrapolation near this boundary are not expected to make physical sense or retain accuracy as these solutions were excluded from training via the feasibility filtration. Once the sweep moves into the supported region, the surrogate tracks the PCA response more closely.

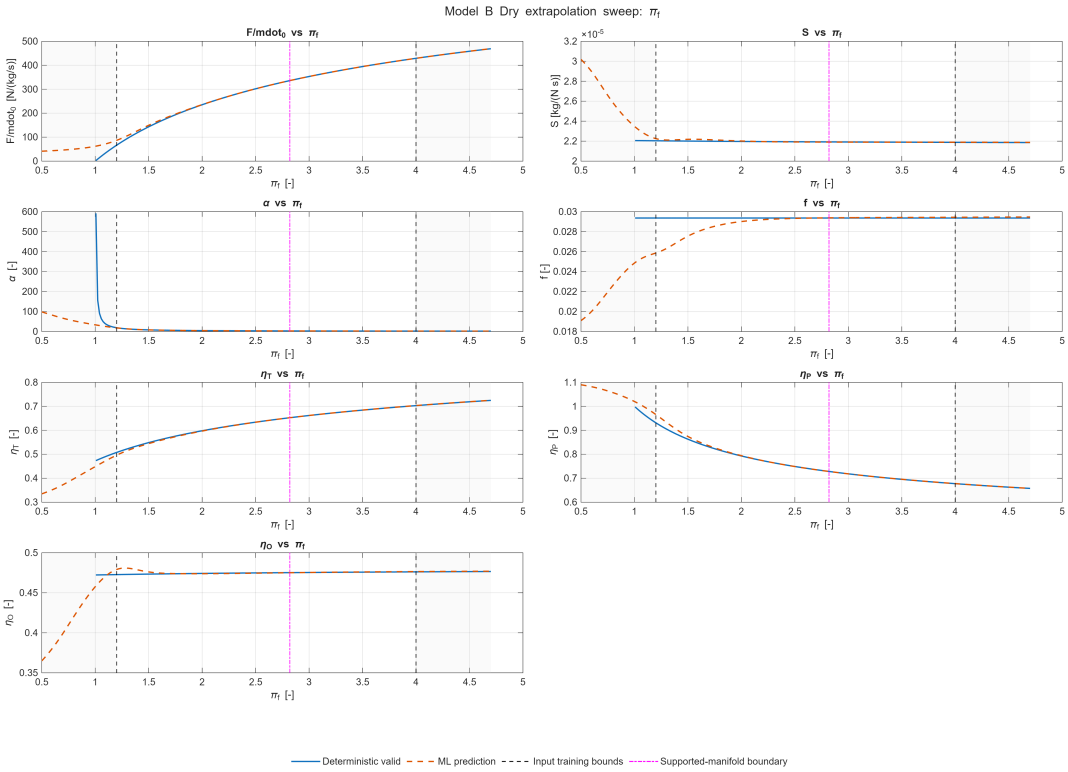


Figure 6.40 – Model B dry-mode extrapolation sweep over fan pressure ratio

The Model B dry burner temperature ratio extrapolation sweep is shown in Fig. 6.41. The surrogate reproduces the deterministic behavior through the main supported interval, but the lower thermal-loading region produces large extrapolation error. Below the supported burner temperature-ratio range, the model predicts a smooth continuation that is not physically reliable. The upper portion of the sweep is more stable, with the network retaining the main thrust, fuel-flow, and efficiency trends but with noticeable extrapolation drift present.

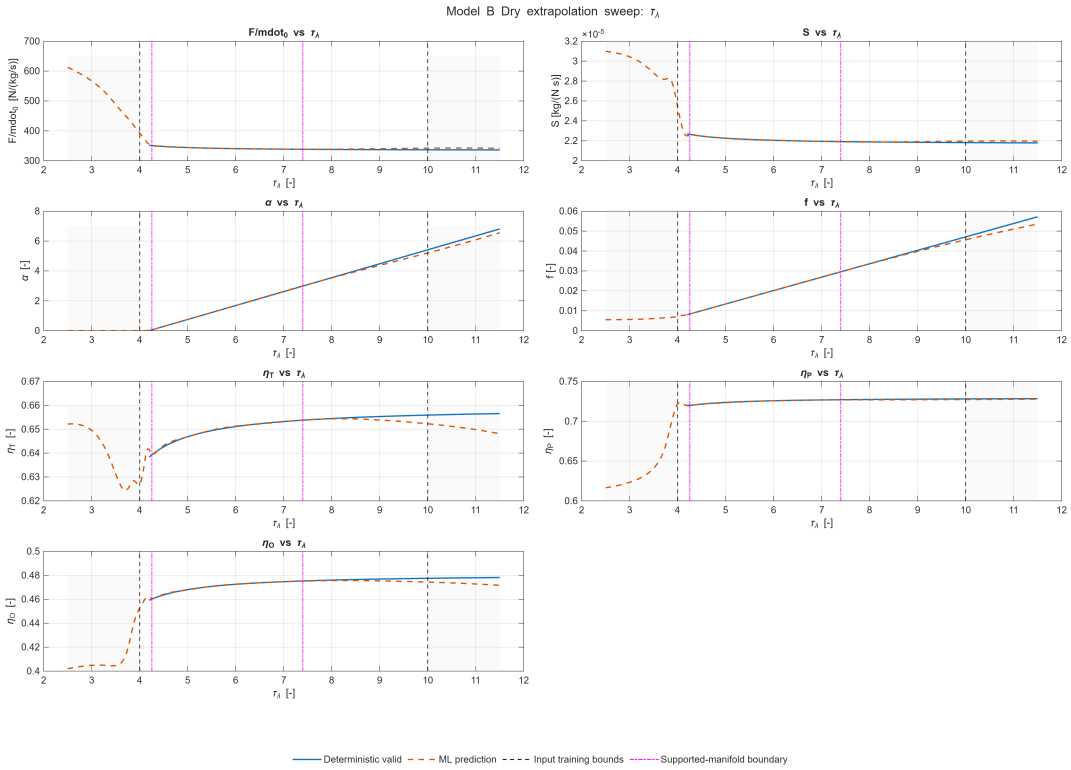


Figure 6.41 – Model B dry-mode extrapolation sweep over burner temperature ratio

The Model B wet Mach number extrapolation sweep is shown in Fig. 6.42. The surrogate follows the deterministic PCA solution well over the supported Mach-number interval. Outside the upper supported region, the predicted thrust, TSFC, afterburner fuel-air ratio, and efficiency quantities begin to depart from the verified reference behavior. Below the Lower mach number supported boundary the model retains its accuracy to the deterministic trends. This demonstrates the same high-Mach extrapolation limitation observed in Model A, now with the additional complexity of variable thermochemistry.

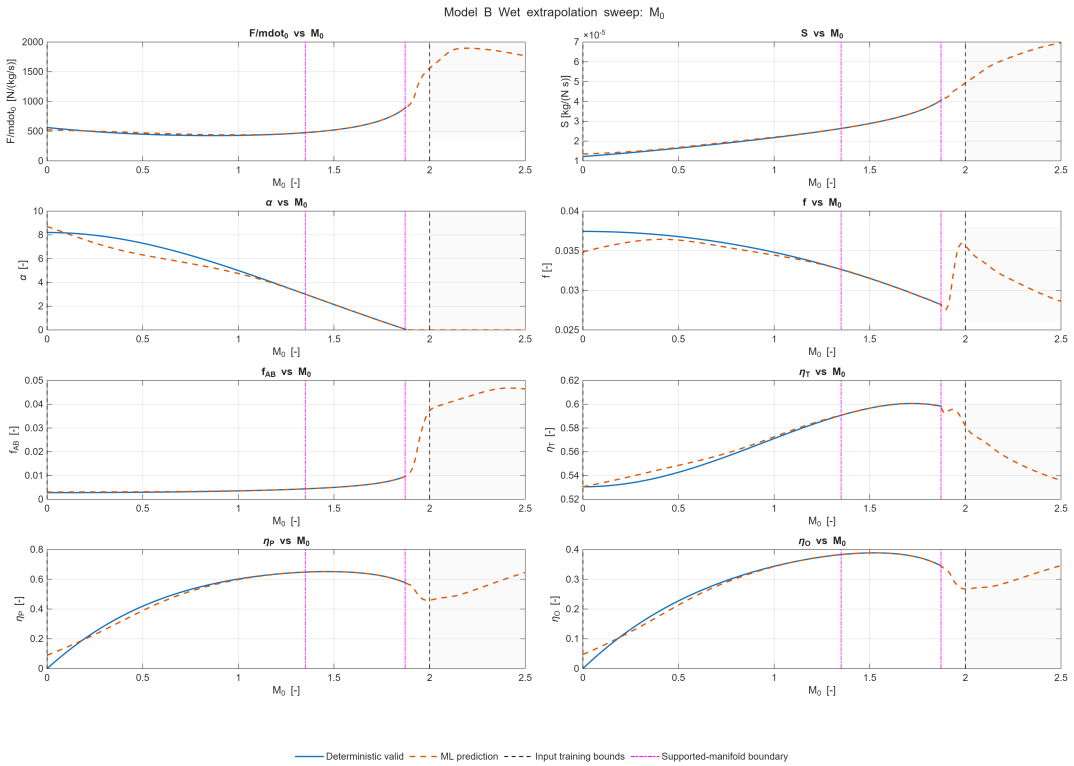


Figure 6.42 – Model B afterburning-mode extrapolation sweep over freestream Mach number

The Model B wet compressor pressure ratio extrapolation sweep is shown in Fig. 6.43. The surrogate remains close to the deterministic solution through most of the supported and upper pressure-ratio region. The slight mismatch appears near the lower compressor pressure ratio edge, where the afterburning fuel-flow and efficiency responses become more sensitive. This indicates that low-π_c operation is an edge-envelope failure direction for both dry and wet Model B experts.

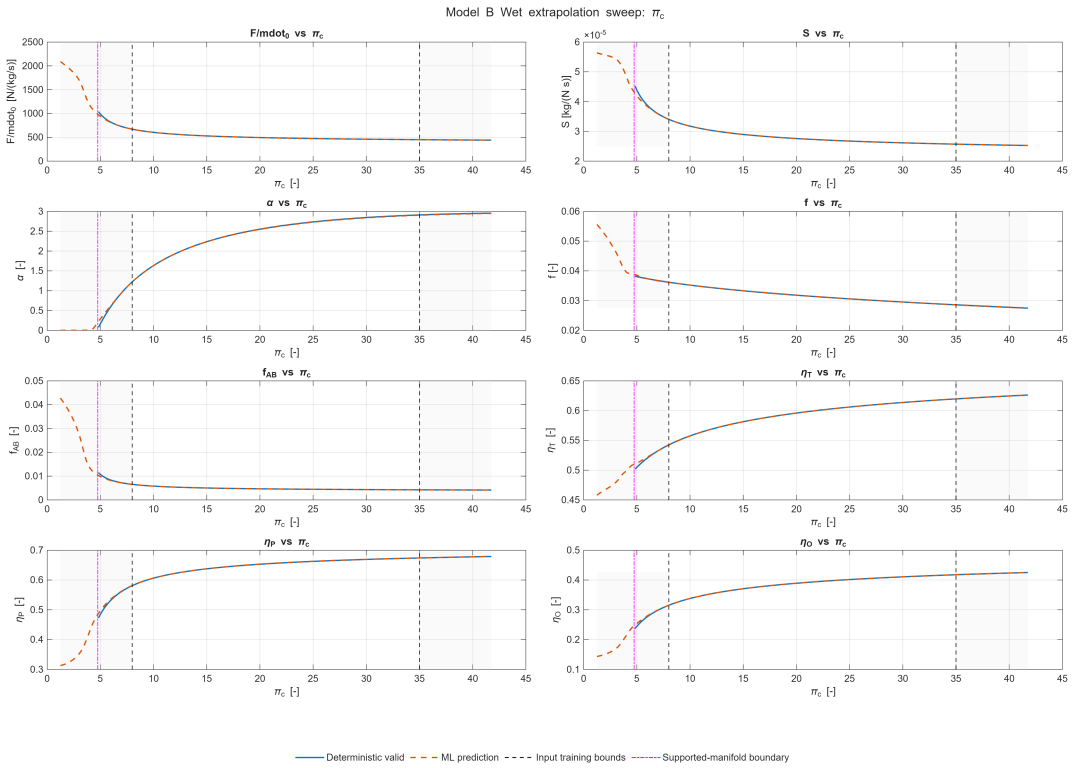


Figure 6.43 – Model B afterburning-mode extrapolation sweep over compressor pressure ratio

The Model B wet fan pressure ratio extrapolation sweep is shown in Fig. 6.44. As in the dry case, the lower fan pressure ratio region produces a severe local mismatch. The deterministic solution exhibits a sharp near-boundary response in the bypass-ratio-related behavior, while the neural network produces a smoother extrapolated curve. The agreement improves once the sweep enters the supported manifold and remains more stable toward the upper fan pressure ratio range.

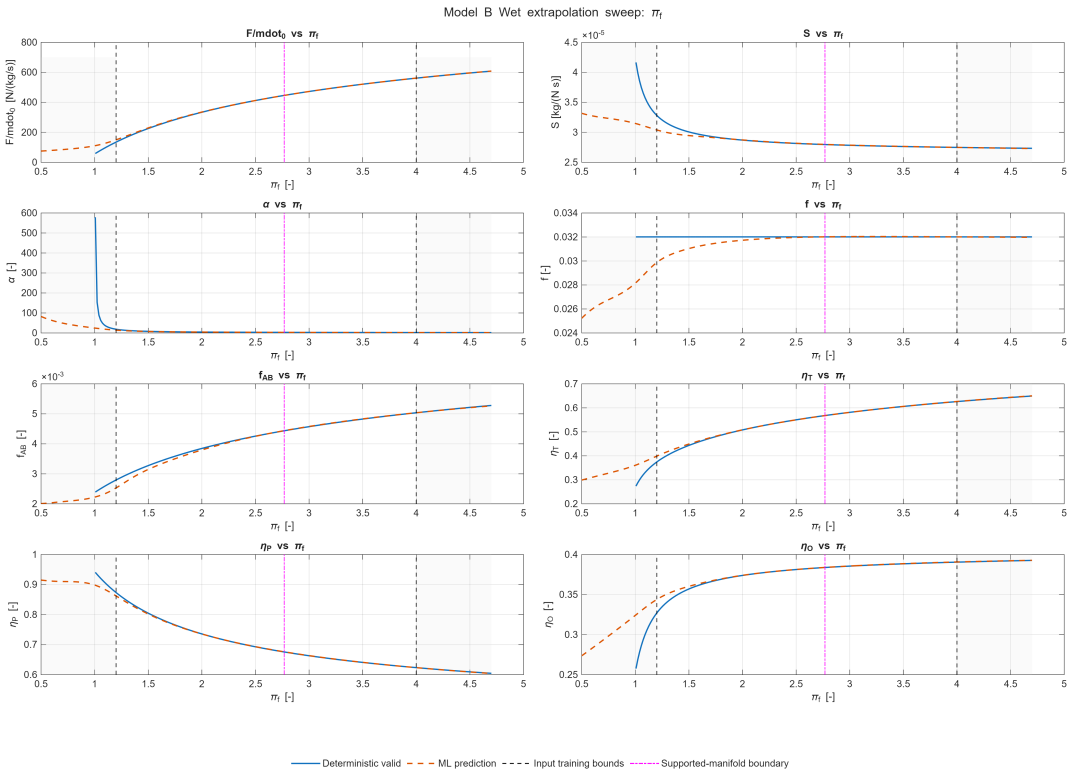


Figure 6.44 – Model B afterburning-mode extrapolation sweep over fan pressure ratio

The Model B wet burner temperature ratio extrapolation sweep is shown in Fig. 6.45. This is one of the clearest Model B failure cases outside the sampled envelope. Below the lower supported thermal-loading region, the predicted thrust, TSFC, afterburner fuel-air ratio, and efficiency quantities depart strongly from the deterministic trend. Inside the supported interval, the agreement improves substantially and the surrogate again follows the PCA response. This confirms that the large errors are primarily an extrapolation issue rather than a failure of the in-domain mapping.

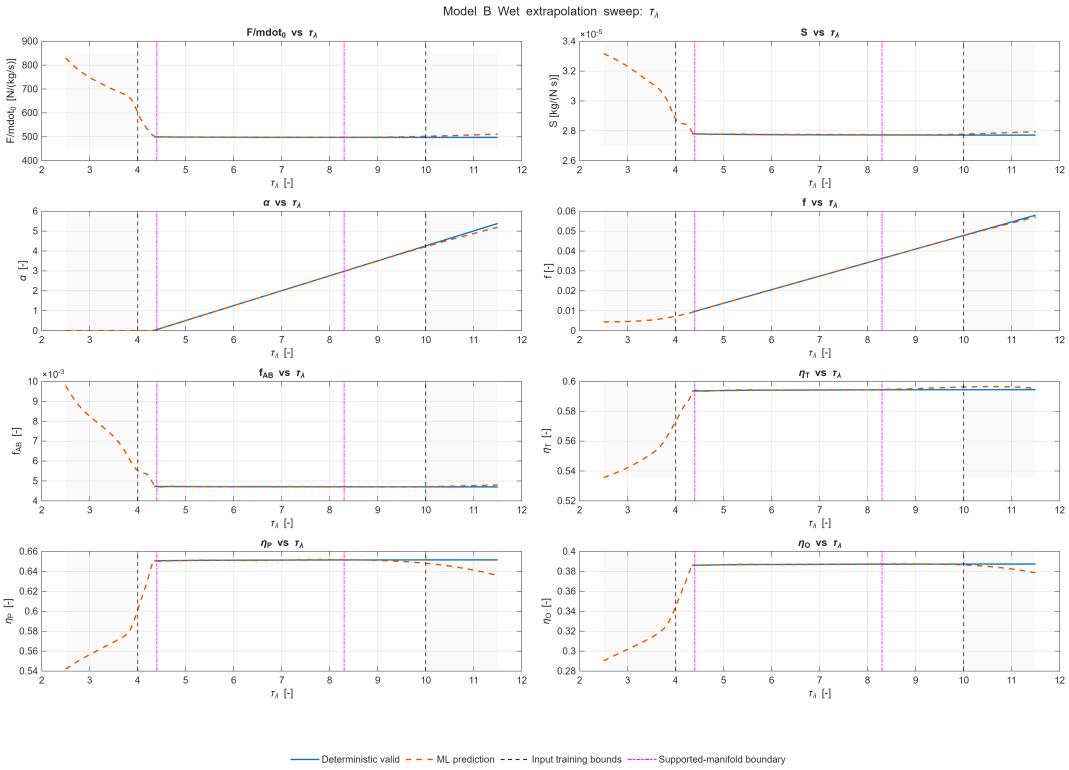


Figure 6.45 – Model B afterburning-mode extrapolation sweep over burner temperature ratio

The Model B wet afterburner temperature ratio extrapolation sweep is shown in Fig. 6.46. This sweep is comparatively well behaved. The surrogate follows the deterministic PCA trend closely over the plotted range for specific thrust, TSFC, bypass ratio, afterburner fuel-air ratio, and efficiency quantities. Similarly to Model A, on the low range of the afterburner temperature ratio extrapolation region the model produces a smooth output continuation. This indicates that the relationship between afterburner heat addition and the augmented ideal-cycle well learned for this model.

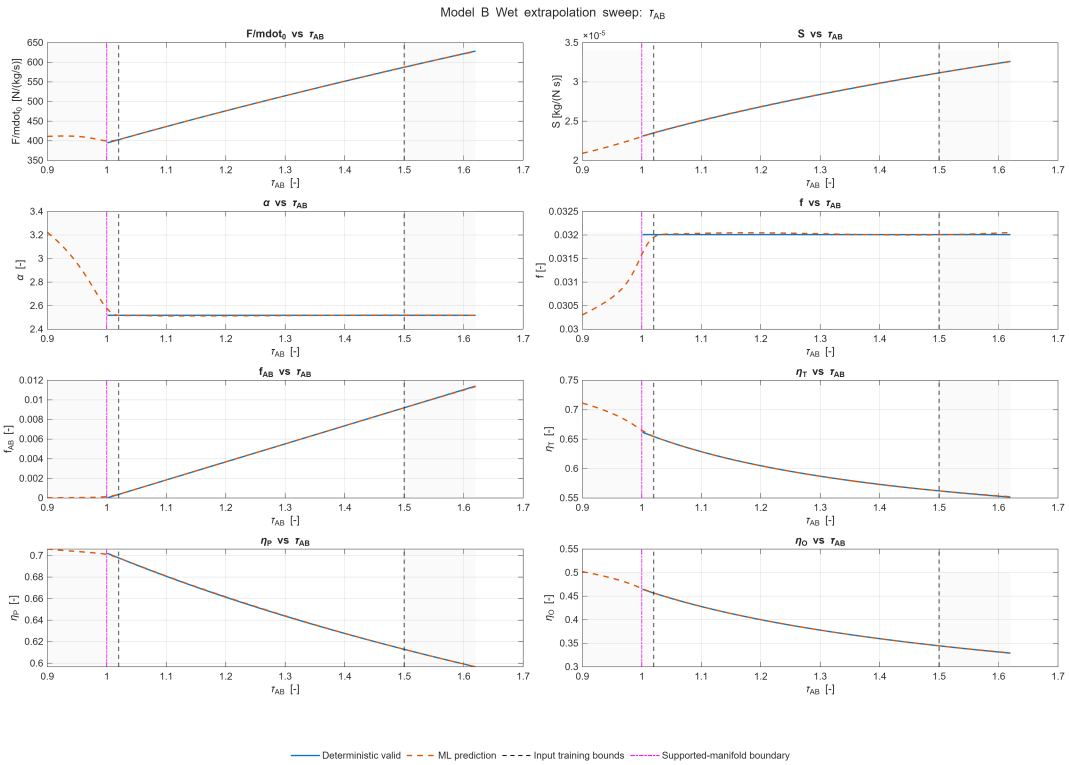


Figure 6.46 – Model B afterburning-mode extrapolation sweep over afterburner temperature ratio

Overall, Model B retains strong interpolation performance within the supported training envelope, but it exhibits clear failure modes when pushed into unsupported edge regions. The most significant deviations occur near lower fan pressure ratio, lower burner temperature ratio, lower compressor pressure ratio, and high Mach number beyond the supported manifold. The afterburner temperature ratio sweep is comparatively stable, suggesting that the direct afterburner heat-addition direction is better conditioned than several upstream cycle-parameter directions. These results reinforce the main conclusion from Model A: the surrogate is reliable as an in-domain forward model, but extrapolated predictions outside the sampled envelope require verification by the deterministic PCA solver.

6.5.3 Model C Extrapolation Behavior

Model C provides the full real-cycle extrapolation case. This is the most demanding extrapolation study because the surrogate must account for component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. As a result, the Model C plots show stronger edge-envelope sensitivity than the ideal-cycle Models A and B.

The Model C dry Mach number extrapolation sweep is shown in Fig. 6.47. The surrogate follows the deterministic PCA solution well through the supported region. Beyond the sampled Mach-number envelope, the prediction continues smoothly but begins to slightly

drift from the verified reference behavior, particularly in the high-Mach extension. This has been seen in the earlier less complex models, where the model proceeds to extrapolate a smooth trend beyond what is realistic. In the case of this particular Mach number sweep it can be seen that the surrogate predicts a continuous drop in the bypass ratio past physical limits. A negative bypass ratio is an obvious impossibility and shows the extrapolation limitations of neural network models as they attempt to approximate a statistical trend without regards to physical boundaries. On the lower end of the mach number sweep outside of the supported boundary it can be seen that the model maintains a high level of accuracy in the extrapolation, with predictions following closely to the deterministic solution.

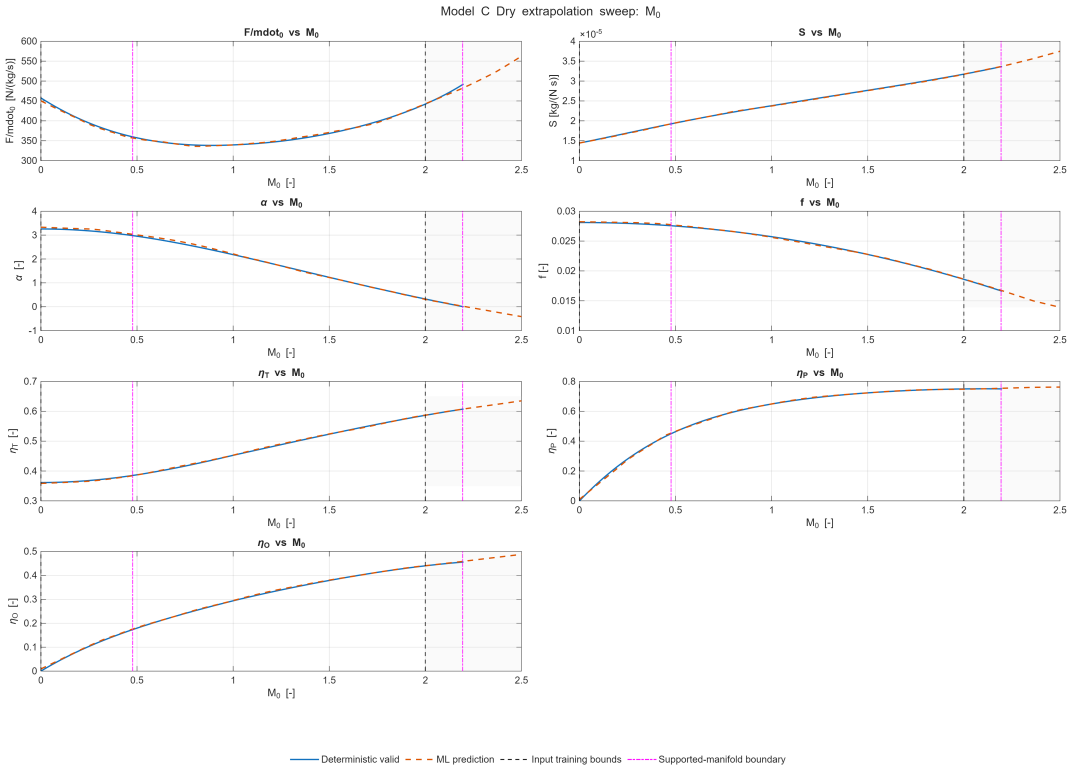


Figure 6.47 – Model C dry-mode extrapolation sweep over freestream Mach number

The Model C dry compressor pressure ratio extrapolation sweep is shown in Fig. 6.48. The model performs reasonably well through the supported region and across much of the upper pressure-ratio extension. The largest deviations from deterministic trends occur near the lower compressor pressure ratio edge, where the expected metrics change rapidly and the neural network prediction smooths through the edge behavior. This indicates that low π_c operation remains an edge-envelope sensitivity for the real-cycle dry expert. The surrogate maintains closely aligned to deterministic prediction in the high π_c range outside of the supported boundary, showing the extrapolation capability of the model with inputs far beyond the training set.

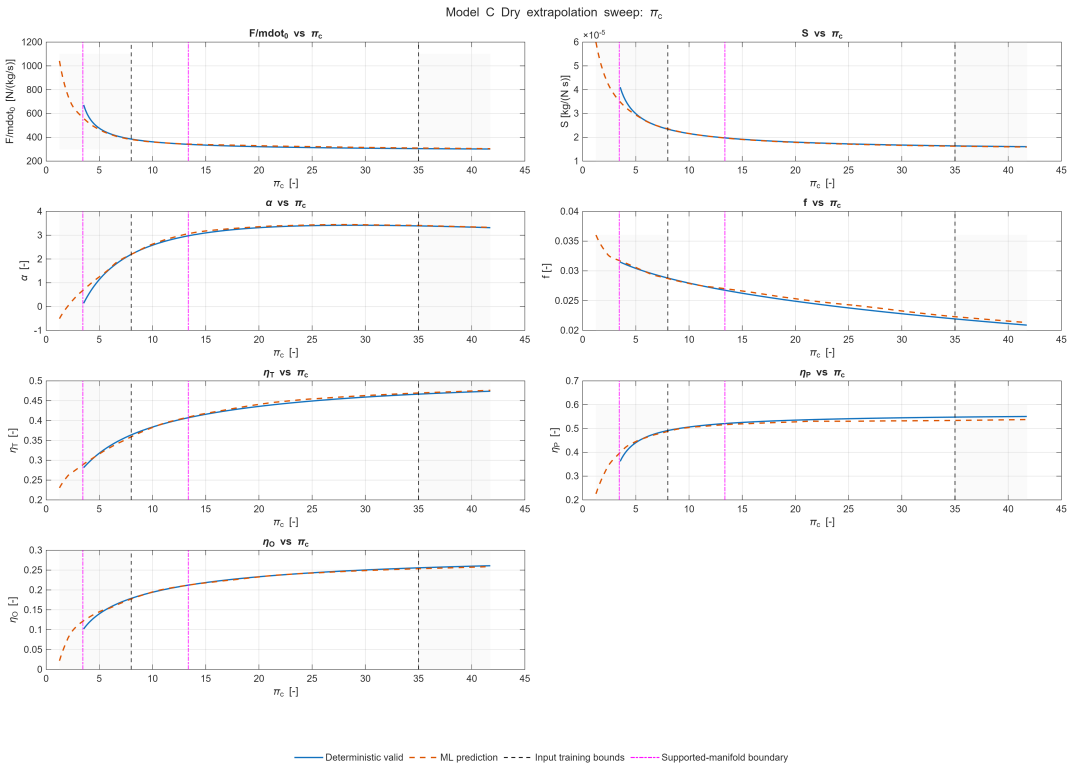


Figure 6.48 – Model C dry-mode extrapolation sweep over compressor pressure ratio

The Model C dry fan pressure ratio extrapolation sweep is shown in Fig. 6.49. This is one of the clear dry-mode failure directions for Model C where the required bypass ratio rises without bound. Near the lower fan pressure ratio boundary, the deterministic solution changes sharply, especially in bypass-ratio-related behavior which is coupled to the thrust and efficiency metrics. The neural network instead produces a smooth continuation and does not capture the sharp local structure. Trend alignment improves inside the supported region, but the sweep still shows that fan pressure ratio is a sensitive extrapolation direction for the real-cycle surrogate for low fan pressure operating modes.

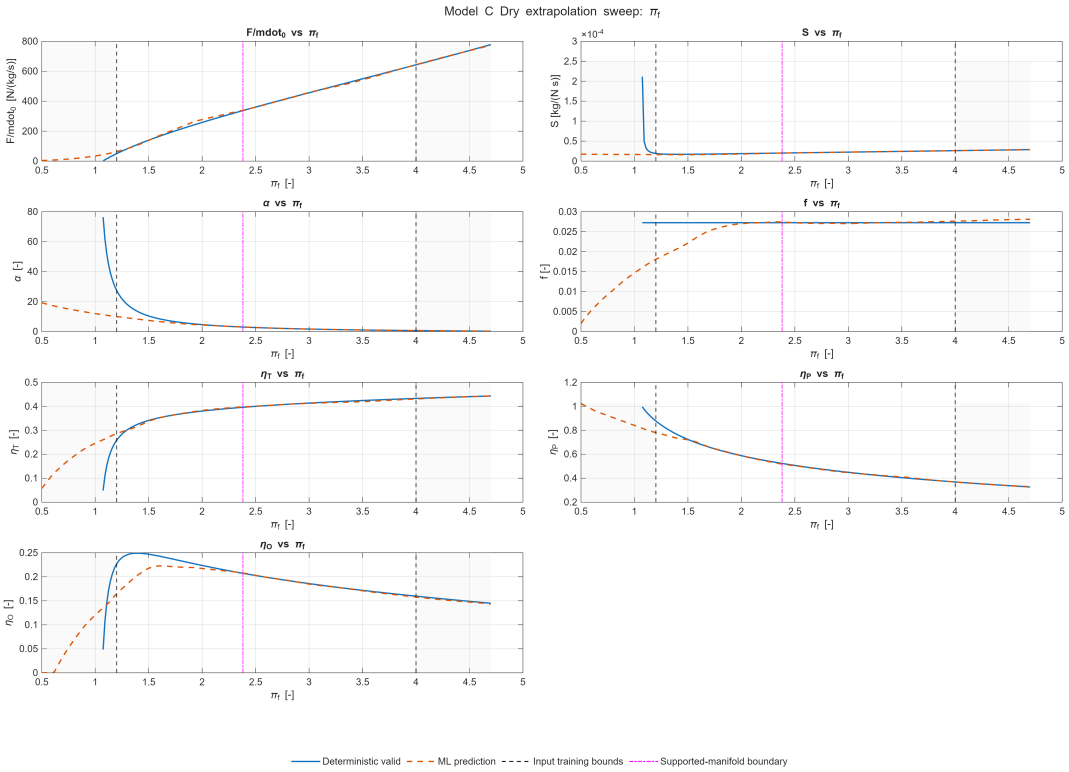


Figure 6.49 – Model C dry-mode extrapolation sweep over fan pressure ratio

The Model C dry burner exit total-temperature extrapolation sweep is shown in Fig. 6.50. This sweep exhibits significant out-of-envelope error. The surrogate does not consistently reproduce the deterministic thrust, TSFC, bypass ratio, fuel-air ratio, and efficiency trends across the extended temperature interval. This is expected to be a difficult direction because T_{t4} affects the main combustor fuel requirement, turbine work balance, mixer state, and downstream nozzle expansion. The T_{t4} sweep direction for the model is already a source of prediction accuracy decay for input within bounds, and as such extrapolation makes the trend inaccuracy error even worse. The result identifies dry-mode thermal loading as one of the most important Model C extrapolation limitations and areas for model improvement.

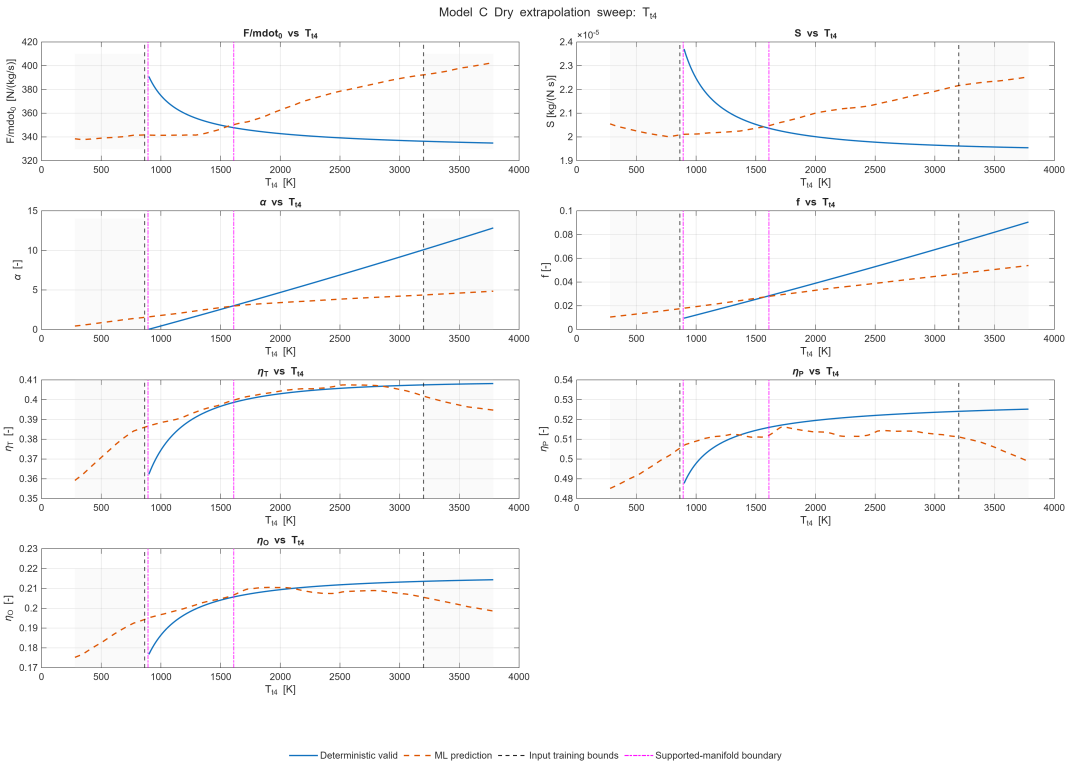


Figure 6.50 – Model C dry-mode extrapolation sweep over burner exit total temperature

The Model C wet Mach number extrapolation sweep is shown in Fig. 6.51. This is one of the better-behaved Model C wet extrapolation cases. The neural network remains close to the deterministic solution over much of the plotted range, with the main trends in thrust, TSFC, fuel-air ratio, afterburner fuel-air ratio, and efficiency quantities preserved. Slight drift can be seen in the thrust curve over the entire range, with the most prominent deviation occurring outside of the dataset boundary at low Mach numbers and past Mach 2. This suggests that the wet expert learned the principal Mach-number response more robustly than several of the component-parameter directions.

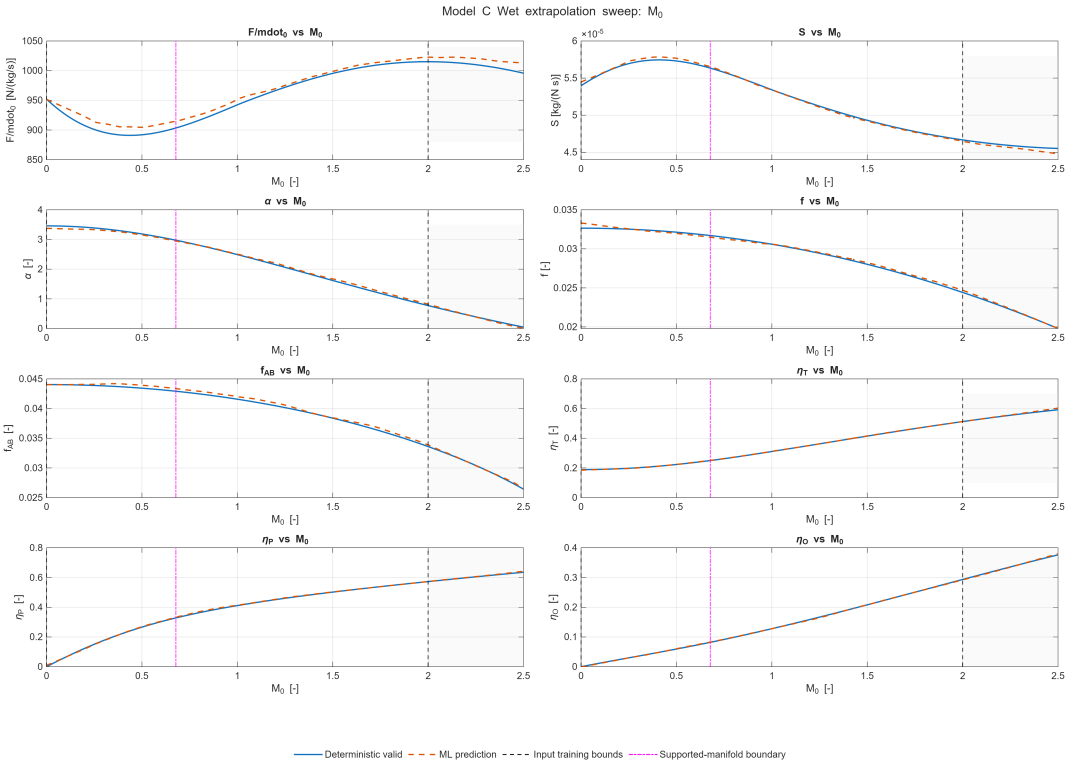


Figure 6.51 – Model C afterburning-mode extrapolation sweep over freestream Mach number

The Model C wet compressor pressure ratio extrapolation sweep is shown in Fig. 6.52. The surrogate reproduces the general pressure-ratio values inside the supported region with some significant error being visible primarily in the thrust alongside thermal and overall efficiency metrics. Local deviations appear in extrapolation region in larger compressor pressure ratio values in several efficiency-related quantities. The machine learning model still predicts the bypass ratio and fuel to air ratios accurately. The afterburning case is more demanding than the dry case because changes in π_c affect the core state entering the mixer, the afterburner fuel requirement, and the final nozzle expansion process. The plot therefore shows reduced accuracy even inside of the supported region alongside reduced reliability when the sweep moves toward unsupported pressure-ratio regions.

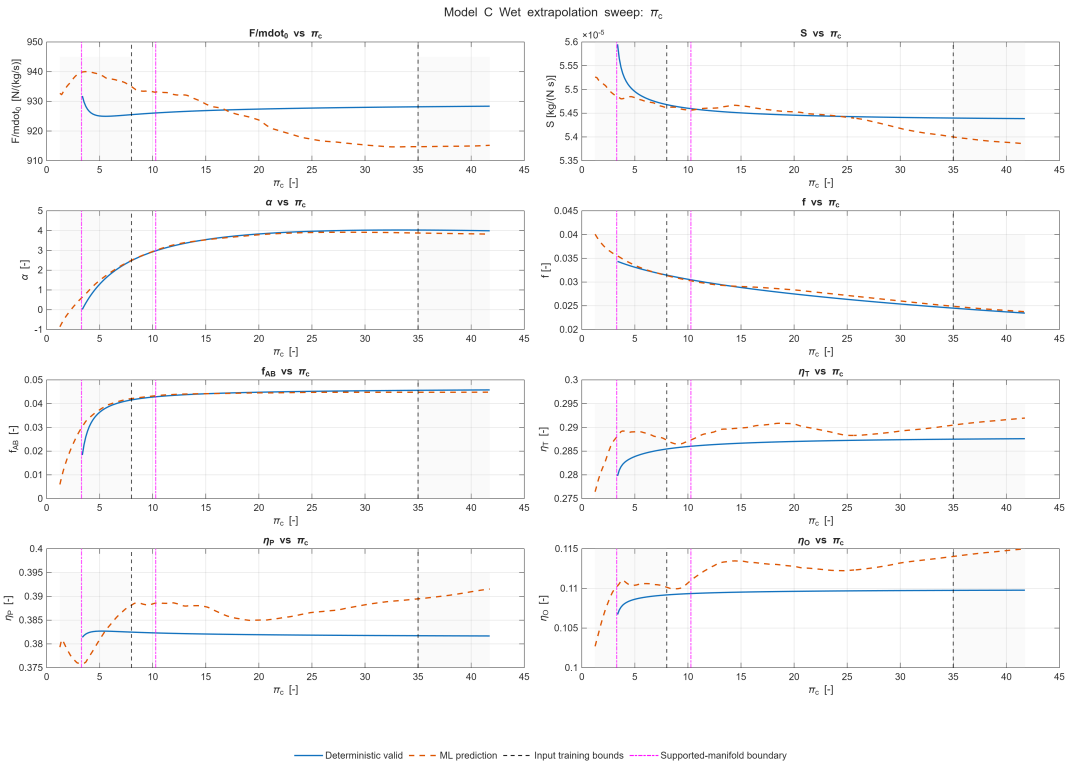


Figure 6.52 – Model C afterburning-mode extrapolation sweep over compressor pressure ratio

The Model C wet fan pressure ratio extrapolation sweep is shown in Fig. 6.53. This sweep shows strong edge-envelope sensitivity. The neural network does not reproduce all of the sharp behavior associated with the deterministic solution near the lower fan pressure ratio region where the expected value of the bypass ratio requirement grows rapidly. Lower values for the fan pressure ratios were not included in the dataset for training due to feasibility constraints, and the model attempts to extrapolate the trend for π_f values lower than 2.4 for this specific anchor point. The model retains accuracy near the supported boundary for a small portion of the lower range, but the prediction drift increases as the sweep targets parameter combinations further away from the initial set. The fan pressure ratio affects the bypass-stream total pressure, mixer state, afterburner inlet condition, and nozzle expansion, so this sweep contains several coupled pathways for error growth. As a result, π_f remains one of the least reliable extrapolation directions for the real-cycle wet expert for low fan pressure ratio values.

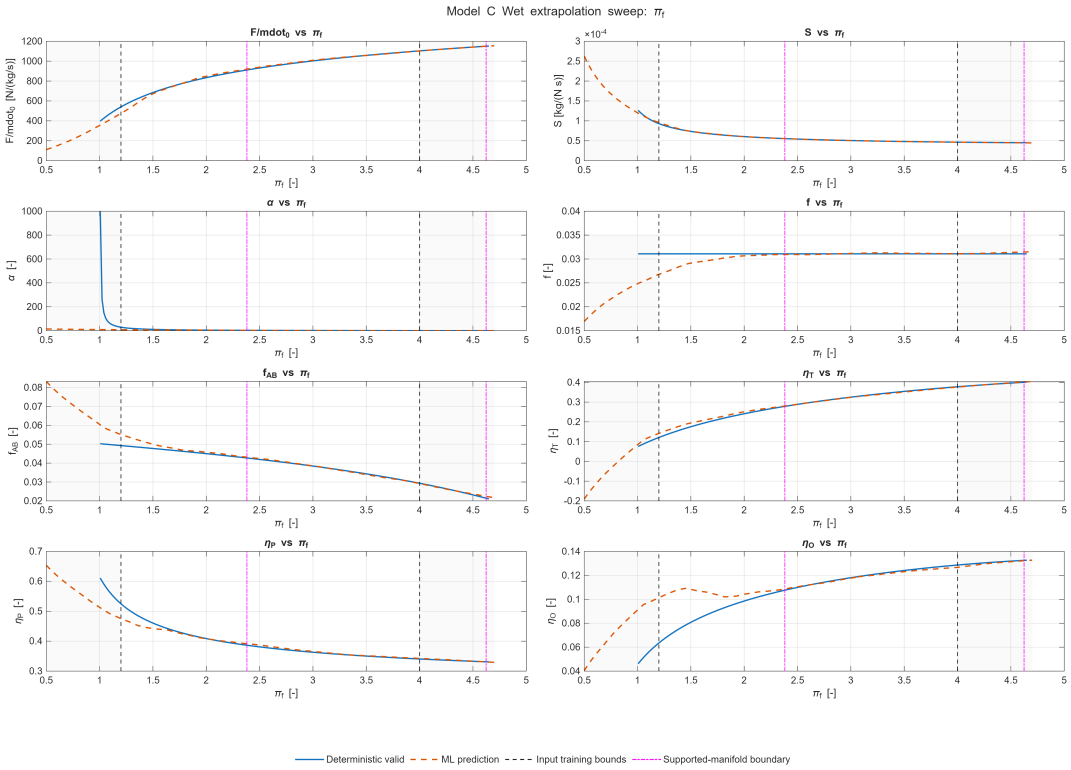


Figure 6.53 – Model C afterburning-mode extrapolation sweep over fan pressure ratio

The Model C wet burner exit total-temperature extrapolation sweep is shown in Fig. 6.54. The surrogate captures only part of the deterministic trend and exhibits substantial deviation in several outputs. The errors are most visible in thrust, TSFC, bypass ratio, and efficiency metrics. Within the supported domain, the thrust metrics do not accurately track the trend but remain within 5% of expected value. As the sweep progresses further away from the supported region to high total temperature values, the model accuracy degrades further as the prediction drifts. Similar behavior can be seen in the efficiency metrics, where inside of the supported region the prediction made by the machine learning model maintains a small error. As the total temperature increases further, the model extrapolates values that are higher error at some regimes as high as 20%. This confirms that T_{t4} is a difficult real-cycle direction because it changes the main burner thermal loading while also altering the state delivered to the turbine, mixer, and afterburner. This subsequently affects all performance metrics and estimated internal states.

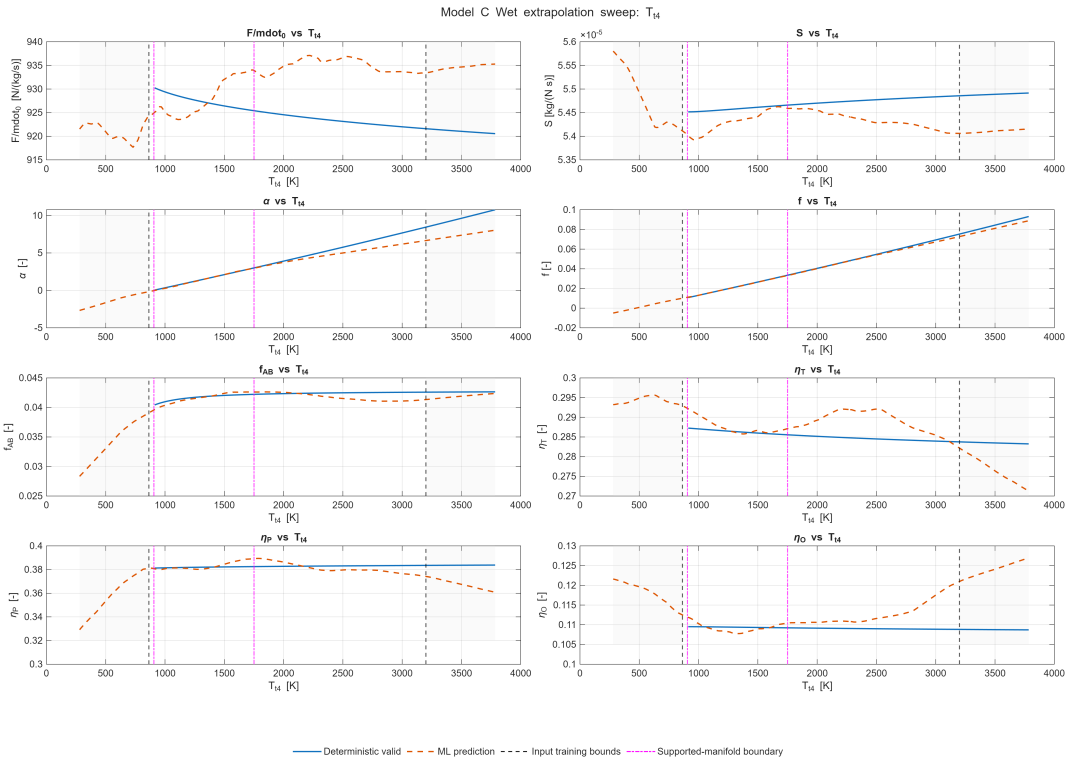


Figure 6.54 – Model C afterburning-mode extrapolation sweep over burner exit total temperature

The Model C wet afterburner exit total-temperature extrapolation sweep is shown in Fig. 6.55. The surrogate captures the broad increase in thrust associated with increasing T_{t7}, but it does not preserve all secondary trends with the same accuracy. The model captures the general range of the expected bypass ratio and main burner fuel to air ratio, but predicts fluctuation that is not present in the deterministic formulation. As the bypass ratio and main burner fuel to air ratio are not dependent on the afterburner total temperature, it is expected they remain constant in this particular sweep. This consequent statistical noise even within the supported boundary is direct evidence of the model not fully capturing this independent relationship and approximating. Efficiency quantities show visible drift outside the supported region, and the predicted afterburning response becomes less reliable toward the high-temperature extension. The deviation seen in the efficiency metrics towards the edge of the high temperature supported boundary is not of incredibly high magnitude but indicates that afterburner thermal loading is learned well enough for in-domain prediction, but not sufficiently constrained for unrestricted extrapolation.

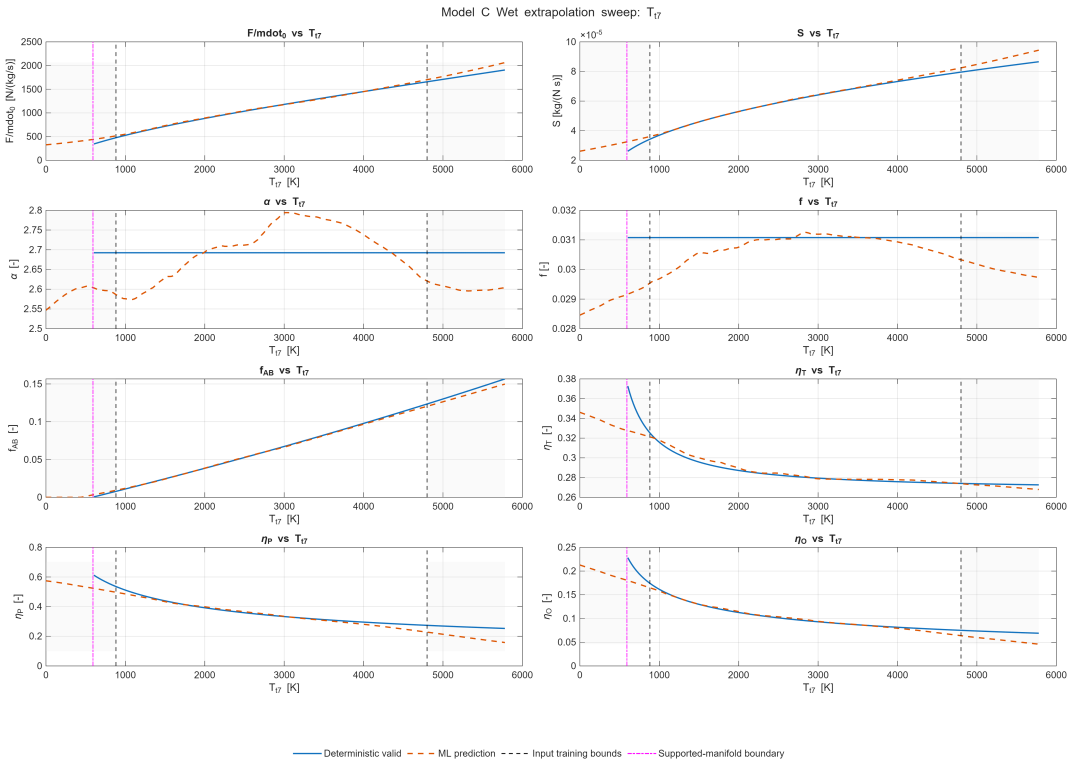


Figure 6.55 – Model C afterburning-mode extrapolation sweep over afterburner exit total temperature

Overall, Model C exhibits the strongest distinction between interpolation and extrapolation performance. The surrogate can reproduce many deterministic trends within the supported envelope, especially for Mach number and portions of the compressor pressure ratio sweeps. However, larger errors appear when the model is pushed toward unsupported low fan pressure ratios, high burner temperatures, and high afterburner temperature regions. These failure modes are expected because the real-cycle mapping is high dimensional and strongly coupled. The Model C results therefore reinforce the central limitation of the surrogate approach: the trained network is useful as a fast in-domain forward model, but out-of-envelope predictions should not be trusted without deterministic PCA verification.

6.5.4 Extrapolation Synthesis and Observed Failure Modes

The extrapolation studies show a clear distinction between interpolation inside the supported training envelope and prediction outside the accepted feasible manifold. Across Models A, B, and C, the neural network predictions generally remain close to the deterministic PCA reference within the supported regions of the sweeps. This is consistent with the baseline validation and parametric sweep results discussed earlier in the chapter. The main loss of reliability appears when the swept variable approaches or passes beyond the region represented by accepted training samples.

The most important observation is that the valid surrogate domain is not defined only

by the raw minimum and maximum bounds of each input variable. The accepted dataset occupies a feasibility manifold inside the high-dimensional input space. A point may remain near the nominal range of one variable while still leaving the supported region of the combined cycle solution because the fixed anchor values and the swept variable no longer produce a well-supported physical state. This is why edge-envelope behavior is more meaningful than simple one-dimensional bound checking.

The first observed failure mode is smooth continuation outside the supported domain. In these cases, the neural network produces continuous numerical outputs even after the deterministic reference solution becomes unsupported, invalid, or no longer representative of the accepted training manifold. These predictions may appear reasonable visually, but they should not be interpreted as physically validated engine behavior. This behavior is expected because the surrogate is a regression model and does not solve the PCA equations during inference.

The second failure mode is boundary smoothing. Several extrapolation plots show sharp deterministic behavior near the edge of the feasible region, while the neural network produces a smoother continuation. This appears most strongly in low fan pressure ratio, low compressor pressure ratio, and low burner temperature ratio regions. These are difficult regions because the cycle solution can change rapidly as fuel-flow, bypass-ratio, turbine work-balance, mixer, or nozzle constraints are approached.

The third failure mode is coupled-output drift. Even when the primary thrust trend remains reasonable, secondary quantities such as TSFC, fuel-air ratios, bypass ratio, and efficiency can drift more noticeably. This is especially important because these quantities are not independent. An apparently acceptable thrust prediction can still be accompanied by a less reliable fuel-consumption or efficiency prediction. For this reason, extrapolated surrogate predictions should be evaluated as full output vectors rather than by specific thrust alone.

Models A and B show the most stable extrapolation behavior because their reference mappings are ideal-cycle formulations with lower dimensionality. Even so, both models still exhibit clear edge failures in low pressure-ratio and low thermal-loading regions, as well as high-Mach extensions outside the supported manifold. The afterburner temperature ratio direction is generally more stable for the ideal-cycle wet experts, suggesting that direct downstream heat addition is a smoother learned direction than several upstream cycle-parameter changes.

Model C exhibits the strongest extrapolation sensitivity. This is expected because the real-cycle surrogate includes component efficiencies, pressure losses, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. These additional variables increase the dimensionality of the feasible manifold and create more coupled pathways for error growth. The most difficult Model C directions are the fan pressure ratio, burner exit total temperature, mixer pressure recovery, and afterburner exit total temperature sweeps.

Overall, the extrapolation study confirms that the trained surrogates should be used primarily as in-domain forward models. They are effective for rapid prediction and trend

analysis within the supported feasible envelope, but they should not be trusted as standalone extrapolation tools outside that region. Any prediction near a feasibility boundary or outside the sampled domain should be verified using the deterministic PCA solver. Future work should therefore include boundary-refined sampling, feasibility classification, and uncertainty quantification so that unsupported predictions can be identified before they are used in design or control studies.

6.6 Runtime Benchmarking

The runtime benchmark evaluates the computational advantage of using the trained neural network surrogates instead of direct deterministic PCA evaluation. This comparison is important because the practical value of the surrogate framework depends not only on prediction accuracy, but also on the ability to evaluate large numbers of engine states efficiently. A surrogate that reproduces the PCA trends but does not reduce computation time would have limited usefulness for large parametric studies, optimization loops, or future control-oriented applications.

The benchmark compares three timing quantities. The first is the deterministic single-point evaluation time, which represents the cost of evaluating one engine state using the PCA solver. The second is the neural network single-point evaluation time, which includes the overhead required to process one input through the trained surrogate. The third is the neural network batch average evaluation time, which measures the effective per-point cost when many input states are evaluated together in a single batch.

This distinction is important because neural networks are not always fastest for isolated single-point evaluations. For small input counts, model-loading, data-formatting, normalization, and prediction overhead can dominate the runtime. For larger batches, however, the fixed overhead is distributed across many points, and the effective per-point neural network cost can decrease substantially. The runtime benchmark therefore evaluates scaling behavior rather than only a single timing value.

The results are organized by model family. Separate runtime plots are presented for Models A, B, and C, with each plot showing dry and afterburning runtime scaling and the corresponding surrogate speedup factors. The section then concludes with the implications of these results for large-scale design studies and future optimization workflows.

6.6.1 Model A Runtime Benchmark

The Model A runtime benchmark is shown in Fig. 6.56. Model A is the lowest-dimensional surrogate family, so this benchmark represents the simplest case for comparing direct PCA evaluation against neural network inference. The dry and wet experts are both evaluated over increasing batch sizes to determine how the runtime scales with the number of requested predictions.

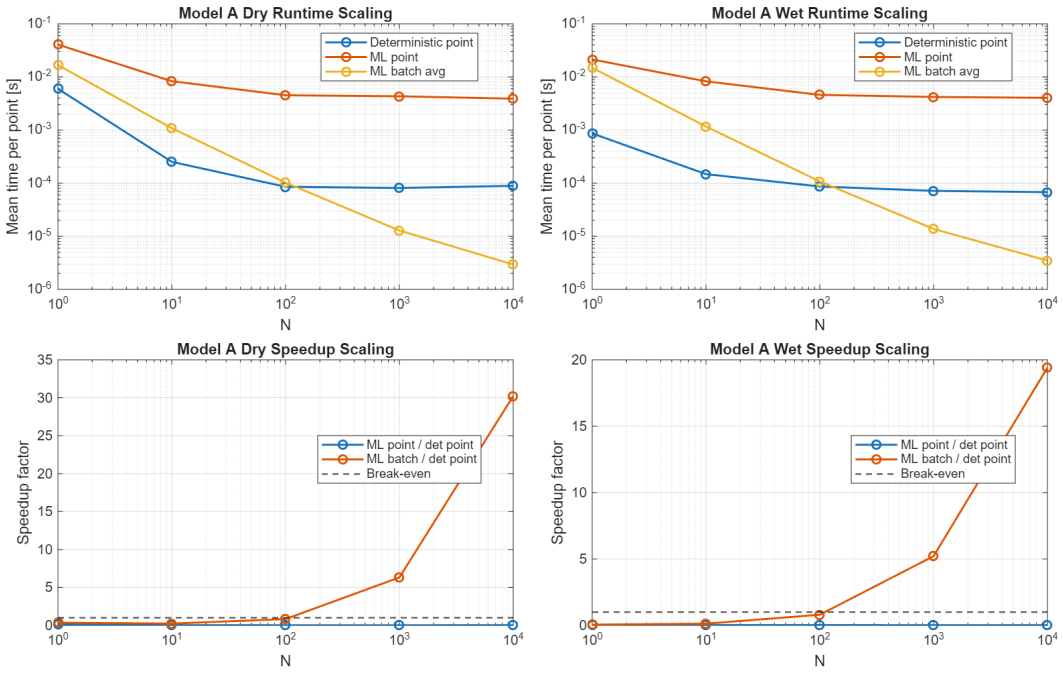


Figure 6.56 – Model A dry and afterburning runtime scaling

For single-point evaluation, the deterministic PCA solver is faster than the neural network surrogate. This occurs because the PCA equations for Model A are relatively compact, while the neural network prediction still requires normalization, data formatting, network execution, and output denormalization. Therefore, Model A does not provide a speed advantage when only one point is evaluated at a time.

The advantage of the surrogate appears during batch evaluation. As the number of evaluated points increases, the effective neural network time per point decreases because the fixed inference overhead is distributed over the full batch. The dry Model A expert reaches the break-even region near the intermediate batch sizes and provides a substantial speedup at the largest tested batch size. The wet expert follows the same trend, although the maximum speedup is slightly lower than the dry case.

The Model A benchmark therefore shows that even the simplest ideal-cycle surrogate is most useful when used for batch prediction rather than isolated single-point calls. For large parametric sweeps, the neural network provides a clear computational advantage over direct deterministic evaluation. For small interactive calculations, however, the direct PCA solver remains more efficient because the overhead of neural network inference is not yet amortized.

6.6.2 Model B Runtime Benchmark

The Model B runtime benchmark is shown in Fig. 6.57. Model B retains the ideal-cycle structure used in Model A, but includes active thermochemical variation in the input space. This makes the learned mapping more complex than the fixed-property case, while the

deterministic PCA equations remain relatively compact.

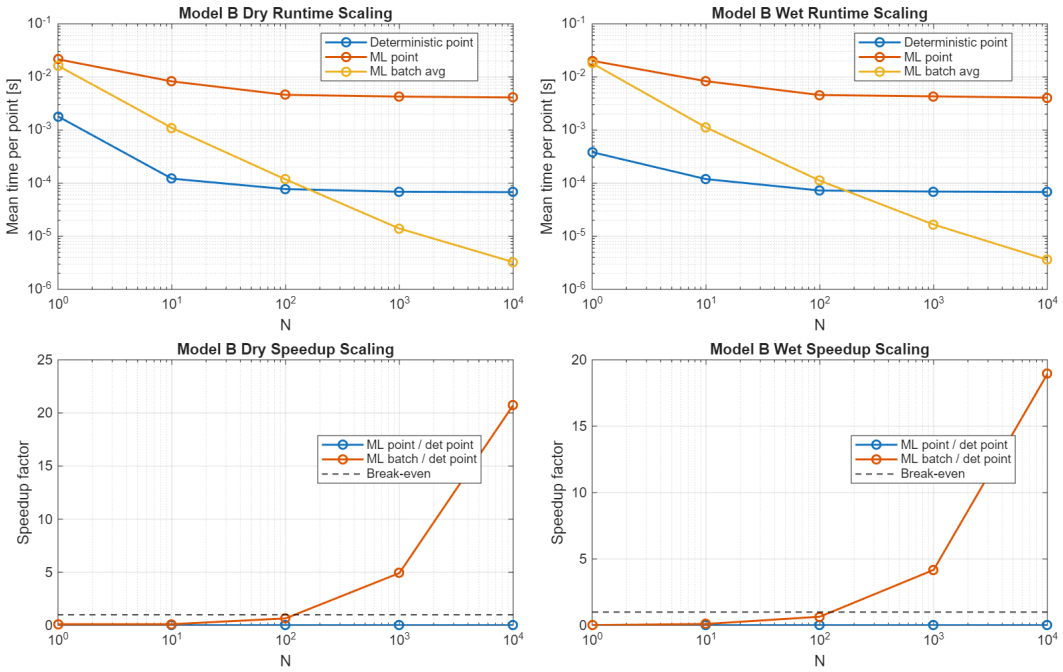


Figure 6.57 – Model B dry and afterburning runtime scaling

As in Model A, the deterministic PCA solver is faster for single-point evaluation. The neural network has a larger fixed overhead associated with input transformation, normalization, network execution, and output reconstruction. For one or only a few evaluated points, this overhead is large relative to the cost of directly evaluating the ideal-cycle equations.

The batch behavior is more favorable for the surrogate. As the batch size increases, the effective neural network time per point decreases rapidly. Both the dry and wet Model B experts cross into the useful speedup region at larger batch sizes, with the largest tested batches producing a clear computational advantage over direct PCA evaluation. This shows that the variable-property ideal-cycle surrogate is beneficial when many predictions are required.

The Model B benchmark confirms the same general runtime pattern observed for Model A. Direct PCA evaluation remains preferable for isolated single-point calculations, but neural network inference becomes advantageous for batch prediction. Since Model B includes additional input variability while still achieving batch speedup, it provides a useful intermediate demonstration that the surrogate approach remains computationally effective as the learned cycle mapping becomes more complex.

6.6.3 Model C Runtime Benchmark

The Model C runtime benchmark is shown in Fig. 6.58. Model C is the highest-dimensional surrogate family and uses the full real-cycle formulation. The deterministic reference eval-

uation is more complex than the ideal-cycle cases because it includes component losses, efficiency terms, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. Therefore, Model C provides the most important runtime comparison for assessing the practical value of the surrogate approach.

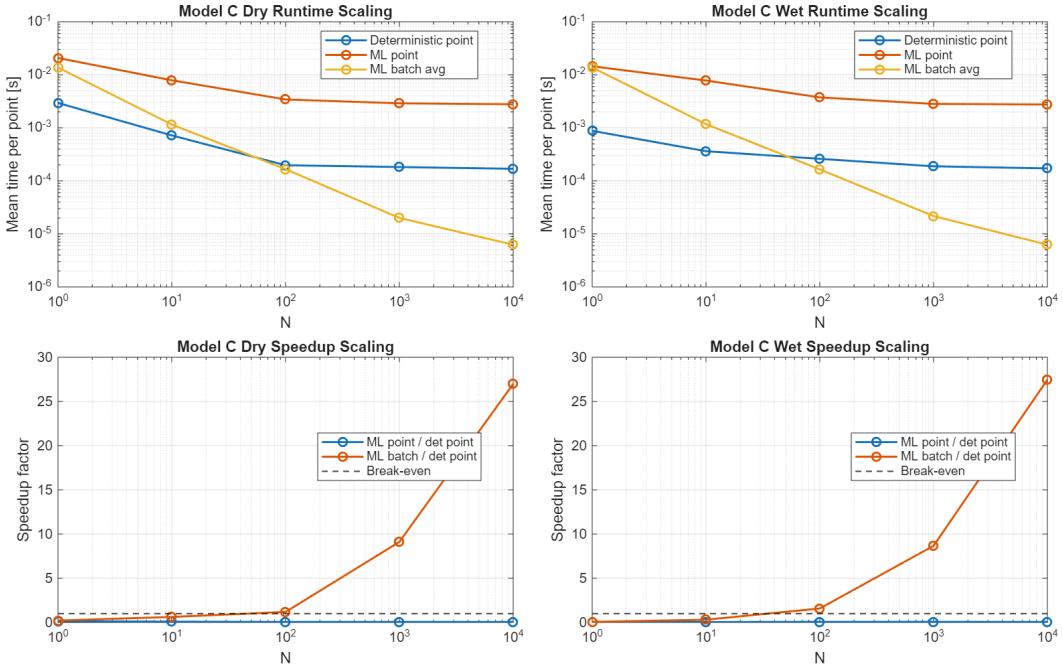


Figure 6.58 – Model C dry and afterburning runtime scaling

For single-point evaluation, the deterministic PCA solver remains faster than the neural network surrogate. This indicates that the overhead associated with network inference, input normalization, output reconstruction, and data handling is still significant when only one point is evaluated. However, the difference is less important for the intended use case of the surrogate, since large design studies and optimization loops typically require many repeated evaluations rather than isolated single-point predictions.

The batch evaluation results show the main advantage of Model C. As the number of evaluated points increases, the effective neural network time per point decreases substantially. At the largest tested batch sizes, both the dry and afterburning Model C experts provide a clear speedup over deterministic cycle evaluation. The wet expert achieves a particularly strong batch speedup, which is valuable because the afterburning real-cycle calculation includes additional coupling through afterburner fuel addition, pressure loss, gas properties, and nozzle expansion.

The Model C benchmark demonstrates that the surrogate framework provides its strongest practical benefit when applied to large batches of real-cycle evaluations. Although direct PCA remains preferable for isolated evaluations, the trained Model C surrogate becomes advantageous for high-throughput prediction tasks. This is the most relevant regime for parametric sweeps, large-scale design-space exploration, and future inverse-control studies

where many candidate engine states must be evaluated efficiently.

6.6.4 Implications for Large-Scale Design Studies

The runtime benchmarks show that the neural network surrogates are most useful in high-throughput evaluation settings. For isolated single-point calculations, the deterministic PCA solvers remain faster because the direct cycle equations are relatively inexpensive and the neural network prediction pipeline has fixed overhead. However, when many points are evaluated together, the surrogate overhead is distributed across the batch and the effective per-point cost decreases substantially.

This behavior is important for large-scale design studies. Parametric sweeps, Monte Carlo studies, optimization loops, sensitivity analyses, and future controller-development workflows all require repeated evaluation of many candidate engine states. In these applications, the surrogate can reduce the computational cost of exploring the design space while preserving the primary cycle trends learned from the deterministic PCA solver.

The batch-speedup behavior is especially relevant for Model C. The real-cycle formulation contains the highest-dimensional input space and the most physically complete set of component loss and efficiency effects. As a result, large Model C studies are the most likely to benefit from surrogate inference. The speedup observed at larger batch sizes indicates that the trained real-cycle surrogate is practical for broad operating-envelope exploration, where direct deterministic evaluation would require many repeated calls to the cycle solver.

The results also clarify how the surrogate should be used. The trained neural network models should not replace the deterministic PCA solver for occasional single-point checks or final verification of critical operating conditions. Instead, the most effective workflow is to use the surrogate for rapid screening, trend exploration, optimization pre-search, or controller iteration, and then use the deterministic PCA solver to verify selected candidate points.

Overall, the runtime study supports the practical value of the surrogate framework for large-scale propulsion analysis. The models provide their largest benefit when used as batch prediction tools, making them suitable for future design-space studies and inverse-control development where thousands of candidate engine states may need to be evaluated repeatedly.

6.7 Model Limitations

The validation studies show that the trained surrogates can reproduce many of the deterministic PCA trends within the supported training envelope. However, the models also have clear limitations that must be recognized before they can be used for design or control studies. These limitations arise from the sampled training domain, the feasibility-filtered nature of the datasets, the complexity of the real-cycle equations, and the fact that the neural

networks are regression models rather than direct solvers of the governing cycle relations.

The purpose of this section is to summarize the main technical limitations observed during training, baseline validation, parametric sweeping, extrapolation testing, and runtime benchmarking. The discussion is focused on the practical interpretation of the trained models: where the surrogates are useful, where their predictions should be treated with caution, and where future improvements should be directed.

6.7.1 Limitations of the Parametric Cycle Reference Model

The deterministic parametric cycle analysis model provides the reference data used for dataset generation, training, validation, and sweep comparison. However, the PCA model is itself an approximate engineering model rather than a full-fidelity representation of a physical engine. Therefore, the neural network surrogates are limited by both their own regression error and the assumptions embedded in the reference cycle model.

A primary limitation is that parametric cycle analysis is a design-point or on-design method. As described by Mattingly, parametric cycle analysis relates engine performance to design choices, design limits, and flight environment, but it does not directly represent the fixed geometry and component operating maps of a specific physical engine [15]. As a result, the sweep results should not be interpreted as the off-design behavior of one manufactured engine. Instead, they represent the behavior of the cycle model under the selected parametric assumptions.

The reference model is also one-dimensional and station based. Flow properties are represented by bulk thermodynamic states at selected engine stations rather than by spatially resolved flowfields. This removes detailed three-dimensional effects such as blade-row aerodynamics, secondary flows, boundary-layer growth, nonuniform combustor exit profiles, local mixing structures, and nozzle plume behavior. The mixer and nozzle are therefore modeled through algebraic relations and loss terms rather than through resolved turbulent mixing or compressible-flow CFD.

For the ideal-cycle models, additional simplifications are present. Models A and B assume ideal component behavior and use simplified thermodynamic relations for the compressor, fan, turbine, mixer, afterburner, and nozzle. Model A further assumes fixed gas properties and fixed fuel heating value. Model B introduces thermochemical variation, but the gas properties are still represented through simplified sampled quantities rather than a full chemical equilibrium or variable-composition gas model. Model C reduces some of these limitations by including component pressure losses, polytropic efficiencies, burner and afterburner efficiencies, mixer pressure recovery, nozzle loss, and mode-dependent gas properties. Even so, these effects are still represented as lumped parameters. The model does not include compressor or turbine maps, surge-margin calculations, spool-speed dynamics, actuator limits, transient thermal lag, combustion instability, emissions, liner cooling, or detailed afterburner flameholding behavior. Consequently, Model C should be interpreted as a higher-fidelity PCA surrogate rather than as a complete engine simulation.

The afterburner model is another important source of limitation. In the reference cycle formulation, afterburning is treated as a steady heat-addition process with prescribed efficiency and pressure loss. This captures the primary effect of raising the mixed-stream total temperature before nozzle expansion, but it does not resolve ignition limits, fuel spray distribution, flameholder blockage, acoustic coupling, combustion rumble, or local hot-streak formation. These effects would be required for a full afterburner operability model.

The feasibility filters used during dataset generation also impose a model boundary. Points that violate thermodynamic, fuel-flow, bypass-ratio, mixer, or nozzle constraints are removed before training. This improves the physical quality of the dataset, but it also means the neural networks learn only the accepted feasible manifold. Predictions near the edge of that manifold, or outside it, may be smooth numerically while still being physically unreliable.

These limitations do not invalidate the surrogate framework, but rather define its scope. The trained networks approximate the PCA model used to generate the data; they do not exceed the physical fidelity of that reference model. Therefore, the results are most appropriate for rapid cycle-level studies, trend exploration, and preliminary design-space analysis. Higher-fidelity component maps, transient engine dynamics, combustion stability models, and CFD-based mixer or nozzle corrections would be required before using the framework as a detailed engine design or control-certification tool.

6.7.2 Limitations of the Neural-Network Surrogate

The neural network models developed in this project are supervised regression surrogates of the deterministic PCA solver. They do not solve the cycle equations directly during inference. As a result, the trained networks are limited by the training data, the selected architecture, the loss formulation, and the input-output transformations used during model development.

The most important limitation is that the surrogates are reliable only within the supported region of the feasible training manifold. The networks were trained on accepted cycle points after feasibility filtering, so they learned the behavior of physically valid samples rather than the full raw input space. When a prediction point lies near the edge of the accepted manifold, or outside it, the network can still return a smooth numerical output even though the prediction is not guaranteed to correspond to a valid cycle solution. This behavior was observed directly in the extrapolation studies.

A second limitation is that feasibility is not enforced as a hard constraint during inference. Models A and B use output transformations, soft feasibility penalties, and algebraic consistency terms to improve physical behavior, but the final prediction is still produced by the neural network rather than by a constrained solver. Model C predicts physical primary and auxiliary quantities, but the explicit algebraic physics-residual terms were not enabled in the final Model C training runs. Therefore, Model C remains physics-guided through the data, feasibility filtering, and supervised physical outputs, but it is not a fully constrained physics-informed model.

The dry and afterburning experts also introduce a mode-separation limitation. The binary expert-gating approach improves training stability by separating dry and wet operation, but it does not learn a continuous transition between non-afterburning and afterburning operation. The dry expert is trained only on $m_{AB} = 0$ samples, and the wet expert is trained only on $m_{AB} = 1$ samples. Therefore, intermediate ignition, light-off, partial scheduling, and mode-transition dynamics are not represented.

The surrogate models are also static mappings. They predict cycle quantities from instantaneous input vectors and do not include time history, spool dynamics, actuator dynamics, combustor delay, thermal lag, or feedback control effects. This is sufficient for steady parametric cycle prediction, but it means the models should not be interpreted as transient engine simulations. A future inverse controller would require additional dynamic modeling or a closed-loop simulation layer before being treated as a control-ready engine model.

Another limitation is that the validation data come from the same generated PCA framework used for training. The held-out test sets, parametric sweeps, and extrapolation studies verify agreement with the deterministic cycle model, but they do not constitute experimental validation against measured engine data. The trained networks therefore demonstrate the ability to reproduce the reference PCA solver, not the ability to predict an actual engine without further calibration.

Model C has additional limitations because of its higher-dimensional real-cycle input space. Although auxiliary outputs such as T_{t6A}, P_{t6A}, M_9, A_9/A_t, and component temperature ratios provide useful internal-state supervision, they do not guarantee that all cycle relations remain exactly satisfied. The multi-output network can approximate the coupled real-cycle mapping, but local errors can still appear in strongly coupled directions such as fan pressure ratio, burner exit total temperature, mixer pressure recovery, and afterburner exit total temperature.

The training implementation also does not include uncertainty quantification. The networks return point predictions only. They do not provide confidence intervals, epistemic uncertainty estimates, or automatic warnings when an input is far from the training distribution. For engineering use, this means that input-bound checks, dataset-support checks, or a separate feasibility and uncertainty model would be needed before relying on the surrogate in automated design or control loops.

Finally, the runtime benchmarks show that the neural network models are most useful for batch evaluation. For isolated single-point predictions, the direct PCA solver can be faster because the neural network pipeline includes input formatting, normalization, inference, denormalization, and output packaging. The surrogate advantage becomes significant when many points are evaluated together, which makes the models better suited for large parametric studies, screening, and optimization workflows than for occasional single-point checks.

These limitations define the correct interpretation of the trained networks. The surrogates are fast approximations of the PCA reference model within the sampled feasible envelope. They are not replacements for the deterministic solver outside the training domain, and they should be verified by PCA evaluation when used near feasibility boundaries or for critical

design decisions.

6.7.3 Limitations of Binary Wet-Dry Gating

The dry and afterburning models were trained using a binary expert-gating strategy. This approach separates the cycle mapping into a dry expert for $m_{AB} = 0$ and a wet expert for $m_{AB} = 1$. The separation improves training stability because the dry and wet regimes have different fuel-flow structure, thermal states, and output sensitivities. However, the same separation also introduces limitations.

The primary limitation is that the gating structure does not represent a continuous transition between dry and afterburning operation. In a physical engine, afterburner operation involves ignition, fuel scheduling, flame stabilization, nozzle-area coordination, and transient adjustment of upstream matching conditions. The binary gate does not model this transition. It only selects between two steady-state mappings that were trained separately.

As a result, the trained model should not be interpreted as an afterburner light-off model or a transient augmentation model. The dry expert predicts steady non-afterburning operation, and the wet expert predicts steady afterburning operation. Intermediate states associated with partial ignition, combustion delay, flame blowout, or unstable afterburner operation are outside the scope of the present surrogate framework.

The binary gate also prevents shared learning between dry and wet operating modes. This is useful for avoiding conflicting gradients during training, but it means that each expert must independently learn the portions of the cycle mapping relevant to its operating mode. If one regime has weaker data support in a particular region of the input space, the other expert cannot directly compensate for it.

Another limitation is that the gate is not uncertainty-aware. The model does not determine whether an input is close to a valid dry or wet operating manifold; it simply routes the input according to the value of m_{AB}. If an input point is physically inconsistent with the selected mode, the routed expert may still return a numerical prediction. This reinforces the need for input-bound checks, feasibility checks, or a separate classifier before using the surrogate in automated design or control loops.

Despite these limitations, binary wet-dry gating is appropriate for the present forward-surrogate study. It provides a clean separation between two distinct steady operating regimes and reduces the training difficulty associated with afterburner activation. Future work should replace or augment the binary gate with a continuous mode-transition model if ignition dynamics, partial afterburning schedules, or closed-loop thrust control are to be represented.

6.7.4 Limitations in Extrapolation

The extrapolation studies show that surrogate accuracy degrades when the models are evaluated outside the supported feasibility manifold. This is an important distinction from

simply leaving the raw input bounds. A point may remain close to the nominal range of an individual input variable while still being poorly supported in the full high-dimensional input space. The accepted training data occupies only the subset of the raw bounds that satisfies the cycle equations, fuel-flow limits, bypass-ratio limits, mixer constraints, nozzle constraints, and thermodynamic feasibility checks.

As a result, the practical validity region of the surrogate is not a rectangular box defined only by minimum and maximum input values. It is a high-dimensional feasible manifold embedded inside those bounds. When a single variable is swept beyond its training range, or even toward the edge of its range while the other variables are held fixed, the resulting input point can leave the region where accepted training samples provide meaningful support. In that case, the neural network may continue to return smooth outputs, but those outputs are not guaranteed to correspond to a physically valid cycle solution.

This behavior was observed most clearly near low fan pressure ratio, low compressor pressure ratio, low burner temperature ratio, high Mach number, and several real-cycle thermal-loading directions. In these regions, the deterministic PCA solution either changed rapidly, became invalid, or left the dataset-supported portion of the feasible manifold. The neural network did not inherently recognize this loss of physical support. Instead, it produced a continuous regression-based continuation of the learned mapping.

The limitation is more severe for Model C than for Models A and B. The real-cycle input space contains additional component efficiencies, pressure losses, gas-property terms, mixer parameters, afterburner quantities, and nozzle loss terms. These variables interact nonlinearly, so the feasible region is more constrained than the raw bounds suggest. A Model C input point can therefore appear numerically reasonable when inspected variable by variable, while still being far from the accepted real-cycle manifold in the combined input space.

For this reason, extrapolated surrogate predictions should not be used as standalone engineering results. They are useful for identifying qualitative failure modes and edge-envelope sensitivity, but they should be verified with the deterministic PCA solver before being interpreted physically. For future automated design or control studies, the surrogate should be paired with explicit input-bound checks, feasibility classifiers, distance-to-training-manifold metrics, or uncertainty estimates so that unsupported predictions can be flagged before they are used.

6.7.5 Unmodeled Combustion and Component Effects

The surrogate framework is limited by the combustion and component effects included in the reference PCA model. The burner and afterburner are modeled through steady heat-addition relations, fuel heating value, efficiency terms, and pressure-loss factors. This captures the main thermodynamic effect of fuel addition, but it does not resolve the detailed combustion physics that would determine operability in a real engine.

The afterburner treatment is especially simplified. Real afterburner behavior depends

on ignition limits, flameholder geometry, fuel spray distribution, local equivalence ratio, liner cooling, flame stability, acoustic coupling, and combustion instability. These effects can strongly affect whether a commanded afterburner state is physically achievable, even if the corresponding cycle-level heat addition appears feasible. The present model therefore predicts steady augmented-cycle performance, not afterburner stability, light-off reliability, rumble, blowout margin, or local thermal loading.

The component models are also represented through lumped parameters rather than detailed maps. Compressor, fan, and turbine behavior are described using pressure ratios and efficiency quantities, but full compressor maps, turbine maps, corrected mass-flow limits, shaft-speed constraints, and surge-margin calculations are not included. As a result, the model can identify thermodynamically feasible cycle points, but it cannot guarantee that those points correspond to stable operation on real component maps.

Similarly, the mixer and nozzle are modeled using cycle-level relations and loss parameters. The mixer does not resolve three-dimensional turbulent mixing, swirl, nonuniform inlet profiles, separation, or detailed total-pressure loss mechanisms. The nozzle model captures the primary expansion behavior, but does not include detailed nozzle geometry, plume structure, overexpanded or underexpanded shock-cell behavior, side loads, or thrust-vector effects.

These omissions are acceptable for the purpose of a cycle-level surrogate, but they limit the physical interpretation of the predictions. The trained models should be understood as fast approximations of the selected PCA formulation, not as full combustor, turbomachinery, mixer, or nozzle simulations. Future extensions would require component maps, combustor stability constraints, afterburner operability limits, cooling-flow models, and higher-fidelity mixer/nozzle corrections to move from cycle-level prediction toward a more complete engine analysis framework.

6.8 Practical and Economic Implications

The trained surrogate models have practical value because they provide a fast approximation of the deterministic PCA framework within the sampled feasible envelope. Although the neural networks do not replace detailed engine analysis, they can reduce the computational cost of repeated cycle evaluation during early-stage design studies. This makes the approach most useful in applications where many candidate engine states must be screened before more expensive verification is performed.

The economic implication is that surrogate modeling can shift part of the design workflow from repeated direct analysis to rapid prediction and filtering. Large parametric sweeps, trade studies, sensitivity analyses, and preliminary optimization loops can be performed more efficiently when the trained model is used as a first-pass evaluator. The deterministic PCA solver can then be reserved for verification of selected candidates, boundary cases, or final reported results.

This section discusses how the surrogate framework could support faster conceptual de-

sign iteration, reduce computational burden in large-scale studies, and provide a foundation for future control-oriented applications. The discussion remains limited to cycle-level analysis, since the present models do not include experimental calibration, transient dynamics, component maps, or detailed combustor and nozzle physics.

6.8.1 Reduced Cost of Repeated Cycle Evaluation

One of the main practical advantages of the surrogate framework is the reduced cost of repeated cycle evaluation. A single deterministic PCA calculation is not computationally expensive by itself, especially for the ideal-cycle models. However, large design studies often require thousands or millions of cycle evaluations. In that setting, the cumulative cost of repeated deterministic evaluation becomes more significant.

The runtime benchmark results show that the neural network models are most effective when used for batch prediction. For isolated single-point calculations, the direct PCA solver can remain faster because the surrogate pipeline includes input formatting, normalization, network inference, denormalization, and output packaging. When many input points are evaluated together, this overhead is distributed across the batch, and the effective per-point prediction cost decreases.

This behavior makes the surrogate useful for high-throughput workflows such as broad parametric sweeps, Monte Carlo sampling, sensitivity studies, and optimization pre-search. Instead of evaluating every candidate point through the deterministic cycle solver, the trained network can be used to rapidly screen large regions of the design space. The PCA solver can then be applied only to the most promising candidates or to points near feasibility boundaries.

The cost reduction is most relevant for the real-cycle surrogate. Model C contains the largest input vector and the most complex reference formulation, so repeated direct evaluation is more expensive than in the ideal-cycle cases. The trained Model C surrogate therefore provides the greatest practical value when many real-cycle predictions are needed in a short time.

In this role, the neural network acts as an acceleration layer rather than a replacement for the deterministic model. The recommended workflow is to use the surrogate for rapid screening and trend exploration, followed by PCA verification for final quantitative results. This preserves the reliability of the physics-based reference model while still gaining the computational benefit of the trained surrogate during repeated evaluation.

6.8.2 Use in Preliminary Engine Design

The surrogate framework is well suited for preliminary engine design because the early design process often requires rapid exploration of many possible cycle configurations. At this stage, the goal is not to finalize a detailed engine geometry, but to identify promising regions of the design space and understand how major cycle variables affect performance.

A trained surrogate can support this process by providing fast estimates of thrust, TSFC, fuel-air ratio, bypass ratio, efficiency, and selected internal cycle states.

For mixed-flow afterburning turbofan concepts, this is especially useful because the design space is highly coupled. Changes in compressor pressure ratio, fan pressure ratio, burner temperature, afterburner temperature, mixer pressure recovery, and nozzle behavior can interact strongly. Directly evaluating all possible combinations with the PCA solver is possible, but becomes increasingly expensive as the number of variables and candidate points grows. The surrogate allows these trends to be explored more quickly before selected cases are checked with the deterministic model.

The surrogate also provides a convenient tool for identifying tradeoffs. For example, a designer can rapidly compare regions with higher specific thrust against regions with lower TSFC, or examine how afterburner heat addition improves thrust while increasing fuel consumption. This type of first-pass screening can help narrow the design space before higher-fidelity analyses are introduced.

The appropriate role of the surrogate in preliminary design is therefore as a screening and trend-analysis tool. It can accelerate broad exploration and help identify candidate operating regions, but it should not be used as the final authority for critical design decisions. Promising points should still be verified using the deterministic PCA solver, and later design stages would require component maps, geometric constraints, structural limits, combustion stability checks, and higher-fidelity flow analysis.

6.8.3 Use in Optimization and Control Studies

The surrogate framework can also support future optimization and control studies. Optimization methods often require repeated evaluations of candidate engine states while searching for combinations of inputs that improve thrust, reduce TSFC, or satisfy operating constraints. Since the trained neural network can evaluate large batches efficiently, it can be used as a fast forward model inside preliminary optimization loops.

For the mixed-flow afterburning turbofan problem, this is relevant because the desired operating condition may require coordinated changes in several inputs. Afterburner fuel addition, burner exit temperature, fan pressure ratio, compressor pressure ratio, mixer pressure recovery, and nozzle behavior can all influence the final thrust and efficiency. A surrogate model allows these coupled effects to be explored rapidly before selected candidates are verified with the deterministic PCA solver.

The same forward surrogate could also be used as the foundation for an inverse-control framework. In that case, the objective would be to determine the engine settings required to achieve a desired thrust while minimizing fuel consumption or avoiding infeasible operating conditions. The forward model could be embedded inside a search routine, gradient-based correction step, or secondary inverse neural network that proposes control inputs for a target performance condition.

However, the present surrogate is not yet a complete control model. It is a steady-state

mapping and does not include actuator dynamics, spool-speed dynamics, transient thermal response, combustion delay, feedback stability, or controller robustness. Therefore, its immediate use is limited to steady optimization and preliminary thrust-targeting studies. A control-ready extension would require dynamic modeling, feasibility monitoring, uncertainty estimation, and deterministic PCA verification of selected operating points.

Even with these limitations, the trained surrogate provides a useful starting point. It can reduce the cost of searching the operating envelope, identify candidate high-performance regions, and support the later development of an inverse model for thrust control and efficiency-aware engine scheduling.

6.8.4 Scaling to Larger Surrogate Models

The surrogate approach can be extended beyond the models developed in this project. Models A, B, and C demonstrate a progression from a fixed-property ideal-cycle mapping to a higher-dimensional real-cycle mapping. This same strategy could be expanded further by increasing the number of inputs, including additional internal-state outputs, and training on higher-fidelity reference data.

A larger surrogate could include component maps, corrected mass-flow parameters, spool-speed constraints, cooling-flow fractions, nozzle geometry variables, and additional mixer or afterburner loss parameters. These inputs would allow the model to represent a broader set of engine behavior than the current PCA-based formulation. The same framework could also be extended to include outputs related to surge margin, corrected flow, exhaust temperature, thermal limits, or infrared-relevant quantities.

Scaling the surrogate would increase the value of the neural network approach because direct evaluation of a higher-fidelity reference model would be more expensive than the present cycle solvers. If the training data were generated from component-map solvers, CFD-corrected mixer models, or coupled cycle-performance tools, the runtime advantage of the trained surrogate would likely become more important for large design studies.

However, larger surrogate models would also require stronger safeguards. Higher-dimensional input spaces are more difficult to sample completely, and the feasible manifold becomes harder to identify. Future large-scale models would therefore need improved sampling strategies, feasibility classifiers, uncertainty estimation, and stronger physics-consistency constraints. Without these additions, the model could produce smooth but unreliable predictions in poorly supported regions of the expanded input space.

Overall, the results of this project suggest that the surrogate methodology can scale to more detailed propulsion models, but only if the dataset-generation and validation process scales with it. Increasing model fidelity should be paired with improved coverage of the feasible envelope, more rigorous out-of-domain detection, and deterministic verification of critical candidate points.

6.9 Recommended Model Improvements

The results of this project show that the surrogate framework can reproduce many of the deterministic PCA trends within the sampled feasible envelope. However, the validation studies also identify several areas where the models could be improved. These improvements are primarily related to dataset coverage, extrapolation safeguards, physics-consistency enforcement, and the transition from steady forward prediction toward optimization or control use.

The recommendations in this section are intended to guide future development rather than revise the scope of the present project. The current models provide a functional forward-surrogate baseline. Future work should build on that baseline by improving physical robustness, expanding the reference model fidelity, and adding mechanisms that detect or prevent unreliable predictions near the edge of the feasible manifold.

6.9.1 Neural Gating Function

A recommended improvement is to replace the fixed binary wet-dry gate with a learned neural gating function. In the present framework, the afterburner mode indicator routes the input directly to either the dry expert or the wet expert. This works well for separating two steady operating regimes, but it does not represent gradual transitions, partial afterburning schedules, or uncertainty near the boundary between operating modes.

A neural gating function would learn a continuous weighting between multiple experts. Instead of selecting only one expert, the model could blend expert predictions according to the operating condition:

$$\hat{\mathbf{y}} = \mathcal{G}(\mathbf{x})\hat{\mathbf{y}}_{wet} + [1 - \mathcal{G}(\mathbf{x})]\,\hat{\mathbf{y}}_{dry} \tag{6.10}$$

where $\mathcal{G}(\mathbf{x})$ is a learned gate output between 0 and 1. Values near zero would favor the dry expert, values near one would favor the wet expert, and intermediate values would represent blended operating behavior.

This would provide a more flexible structure for future afterburner modeling. A learned gate could account for variables such as afterburner temperature ratio, afterburner fuel flow, nozzle area, pressure loss, or commanded thrust. It could also be expanded beyond two experts if additional operating regimes are introduced, such as partial augmentation, high-altitude relight, or separate expert regions for different Mach-number ranges.

The primary benefit of a neural gate is smoother mode handling. The current binary gate is appropriate for steady dry and steady wet prediction, but it creates a hard discontinuity between experts. A learned gate would allow the surrogate to transition more gradually between regimes and could reduce prediction artifacts near operating-mode boundaries.

However, a neural gating function would require additional training data near the transition region. The present datasets were generated as separated dry and wet cases, so they do

not fully define the physics of ignition, partial afterburner operation, or transient light-off behavior. Future implementation would therefore need intermediate augmentation states, mode-transition samples, or a separate dynamic model before the gate could be interpreted as physically representative of real afterburner activation.

For the present project, binary gating was sufficient for the forward-surrogate objective. For future thrust-control work, a neural gating function would be a natural next step because it would allow smoother scheduling between dry and afterburning operation while retaining the advantages of expert-based specialization.

6.9.2 Hyperparameter Optimization

Another recommended improvement is a more systematic hyperparameter optimization study. The architectures and training settings used in this project were selected to provide stable training and sufficient regression capacity for each model stage. This was appropriate for establishing the forward-surrogate framework, but the final models were not the result of a formal hyperparameter search.

Future work should evaluate the effect of network depth, layer width, activation function, batch size, learning-rate schedule, loss weighting, and regularization strength. This is especially important for Model C, where the real-cycle mapping is higher dimensional and more strongly coupled than the ideal-cycle cases. The parametric sweep and extrapolation results suggest that some directions, such as fan pressure ratio, burner exit total temperature, mixer pressure recovery, and afterburner exit total temperature, would benefit from improved capacity or refined output weighting.

A structured hyperparameter study could compare candidate models using a combination of validation loss, test-set error, parametric sweep agreement, and extrapolation robustness. This is preferable to selecting a model using validation loss alone, because a network with low average validation error may still produce poor physical trends in specific regions of the input space. Including sweep-based metrics in the selection process would make the final model more useful for engineering analysis.

The learning-rate schedule should also be treated as an optimization variable. Different model families may benefit from different initial learning rates, decay factors, decay intervals, or adaptive stopping criteria. Model C in particular may benefit from a schedule that allows longer low-learning-rate refinement after the primary loss has plateaued.

Loss weighting is another important target for future tuning. Outputs such as specific thrust, TSFC, bypass ratio, fuel-air ratios, efficiencies, and auxiliary internal states do not contribute equally to engineering usefulness. Increasing the weight on difficult or physically important quantities may improve the accuracy of the surrogate in the sweep directions where the current model shows larger deviations.

Overall, hyperparameter optimization would improve confidence that the final surrogate architecture is not merely functional, but close to optimal for the available dataset and intended validation tasks. This would be especially valuable before using the surrogate as

the basis for inverse design or closed-loop thrust-control studies.

6.9.3 Expanded Dataset Generation

Expanded dataset generation is one of the most important improvements for future surrogate development. The current datasets were large enough to train the forward models and demonstrate the feasibility of the approach, but the validation studies show that some regions of the input space remain more difficult than others. This is especially true near feasibility boundaries and in strongly coupled real-cycle directions.

Future datasets should place additional samples near the regions where the current models show larger sweep and extrapolation errors. For Model C, these regions include fan pressure ratio, burner exit total temperature, mixer pressure recovery, and afterburner exit total temperature. Increasing sample density in these directions would give the network more information about the local curvature of the real-cycle mapping and reduce the tendency to smooth through sharp edge-envelope behavior.

The sampling strategy should also account for the fact that the feasible domain is not a simple rectangular input box. The accepted cycle points form a high-dimensional feasibility manifold inside the raw input bounds. Future dataset generation should therefore focus not only on increasing the total number of samples, but also on improving coverage of the accepted manifold. This could be done through targeted Sobol sampling, adaptive resampling, or active learning based on regions where validation error is largest.

An active learning workflow would be especially useful. A preliminary surrogate could first be trained on an initial dataset, then evaluated over structured sweeps and candidate operating points. Regions with high surrogate-PCA disagreement, high uncertainty, or frequent feasibility-boundary interaction could then be resampled with the deterministic PCA solver and added to the training set. This would concentrate new data where it provides the greatest improvement rather than uniformly increasing the dataset size.

Future datasets should also include more deliberately structured samples for transition and control-relevant behavior. If the surrogate is extended toward inverse thrust control, the dataset should contain dense coverage in regions where thrust changes rapidly with afterburner heat addition, burner temperature, nozzle behavior, and pressure-ratio variables. This would improve the usefulness of the forward model as a foundation for optimization or control studies.

Overall, expanded dataset generation should be treated as a targeted refinement process rather than only a larger random sampling effort. The goal should be to improve coverage of the feasible manifold, strengthen the difficult real-cycle directions, and provide additional training support near the regions where the current models begin to lose physical reliability.

6.9.4 Boundary-Refined Sampling

Boundary-refined sampling is recommended because many surrogate errors occur near the edge of the feasible manifold. The current datasets were generated by sampling broad input ranges and retaining only physically feasible cycle solutions. This produced useful training data, but it did not guarantee dense coverage near the boundaries where the cycle transitions from feasible to infeasible behavior.

Future dataset generation should explicitly target these boundary regions. Instead of only sampling the full input space uniformly, additional candidate points should be generated near known feasibility limits, such as low fan pressure ratio, low compressor pressure ratio, low burner temperature, high-Mach operation, fuel-flow bounds, bypass-ratio bounds, mixer limits, and nozzle choking constraints. These regions are important because the cycle outputs can change rapidly near the boundary, making them difficult for a smooth neural network to approximate without sufficient local data.

A boundary-refined dataset could be constructed by first generating a broad baseline dataset, identifying accepted points near feasibility limits, and then sampling locally around those points. The deterministic PCA solver would be used to classify the new samples as feasible or infeasible. Accepted points would improve regression accuracy near the edge of the valid domain, while rejected points could be used to train a separate feasibility classifier.

This approach could also improve extrapolation safeguards. The surrogate should not only learn output values inside the feasible region; it should also learn where the reliable prediction region ends. A boundary-refined dataset would provide the information needed to estimate distance from the feasible manifold and flag inputs that are likely to produce unreliable predictions.

For Model C, boundary-refined sampling is especially important. The real-cycle feasible region is shaped by coupled effects from component efficiencies, pressure losses, gas properties, mixer pressure recovery, afterburner pressure loss, and nozzle behavior. A point may appear reasonable when each input is examined individually, but still lie near or outside the feasible real-cycle manifold when all inputs are considered together. Additional boundary samples would help the model learn these coupled feasibility limits more effectively.

Overall, boundary-refined sampling would make the surrogate more robust near the edge of the operating envelope. It would reduce sharp sweep errors, improve feasibility awareness, and provide a stronger foundation for future optimization or control workflows where candidate points may naturally move toward performance boundaries.

6.9.5 Feasibility Classification

A future improvement is to train a dedicated feasibility classifier alongside the regression surrogate. The current models predict cycle outputs directly for a given input vector, but they do not independently determine whether the input lies inside the feasible cycle manifold. As a result, the surrogate can return numerical predictions for points that are outside the

region where the deterministic PCA solver would produce a valid solution.

A feasibility classifier would address this limitation by predicting whether an input point is physically admissible before the regression model is used. The classifier could be trained using both accepted and rejected candidate points from the dataset-generation process. Accepted points would be labeled as feasible, while rejected points would be labeled according to the feasibility checks that failed, such as fuel-flow limits, bypass-ratio limits, mixer constraints, nozzle constraints, or invalid thermodynamic states.

The simplest version of this classifier would produce a binary output: feasible or infeasible. A more useful version would produce either a probability of feasibility or a set of failure-mode labels. This would allow the prediction workflow to distinguish between different causes of infeasibility rather than only rejecting the point. For example, one input may fail because the bypass ratio is outside the accepted range, while another may fail because the nozzle state is invalid.

A feasibility classifier would be especially useful for automated design and optimization studies. Optimization algorithms may naturally search near the edge of the operating envelope, where small input changes can move a point outside the feasible manifold. Without a classifier, the regression surrogate may still return smooth but physically meaningless outputs. With a classifier, unsupported candidate points could be rejected or passed to the deterministic PCA solver for verification before being used.

For Model C, feasibility classification would be particularly valuable because the real-cycle input space is high dimensional and strongly coupled. The raw input bounds alone are not sufficient to determine whether a candidate point is physically meaningful. A classifier trained on feasible and infeasible real-cycle samples would provide a practical safeguard against using the surrogate outside the region supported by the cycle equations.

Overall, feasibility classification would make the surrogate framework more robust and more suitable for optimization or control applications. It would separate the question of whether an input is physically valid from the question of what outputs the surrogate predicts, reducing the risk of interpreting out-of-manifold neural network predictions as valid engine states.

6.9.6 Uncertainty Quantification

Uncertainty quantification is recommended before the surrogate framework is used in automated design or control workflows. The current models return point predictions only. They do not provide confidence intervals, prediction uncertainty, or automatic warnings when an input is far from the training distribution. This is a limitation because the neural network can produce a smooth numerical output even when the input lies near or outside the supported feasible manifold.

A useful improvement would be to estimate both prediction uncertainty and out-of-distribution uncertainty. Prediction uncertainty would indicate how confident the surrogate is in a specific output value, while out-of-distribution uncertainty would indicate whether

the input point is poorly supported by the training data. These two quantities are related but not identical. A point may be inside the raw input bounds but still far from the accepted feasible manifold in the full high-dimensional input space.

Several approaches could be used. An ensemble of independently trained surrogates could provide uncertainty estimates based on disagreement between models. Monte Carlo dropout could approximate uncertainty by evaluating the same input multiple times with stochastic dropout active during inference. Another option is to train a separate distance-to-manifold or density model that estimates how close a candidate input is to the accepted training distribution.

Uncertainty estimates would be especially useful for Model C. The real-cycle input space is high dimensional, and the feasible region is shaped by coupled component efficiencies, pressure losses, gas-property variables, mixer relations, afterburner behavior, and nozzle constraints. In this setting, raw input bounds alone are not sufficient to determine whether a prediction is trustworthy. An uncertainty model could flag regions where the surrogate is likely to be extrapolating, even if each individual input appears reasonable.

For future optimization studies, uncertainty quantification could also be used as a penalty term or constraint. Candidate points with high uncertainty could be rejected, resampled with the deterministic PCA solver, or assigned lower priority during design-space exploration. This would reduce the risk of an optimizer exploiting inaccurate surrogate behavior near the edge of the training domain.

Overall, uncertainty quantification would improve the practical reliability of the surrogate framework. It would not replace deterministic PCA verification, but it would provide an additional safeguard by identifying predictions that should be treated with caution before they are used for design decisions or control-oriented studies.

6.10 Chapter Summary

This chapter presented the main results and analysis of the trained surrogate models. The validation process moved beyond random test-set accuracy and examined whether the models preserved physically meaningful cycle behavior under structured changes in engine and flight parameters. The chapter therefore focused on parametric sweep validation, extrapolation behavior, runtime performance, model limitations, practical implications, and recommended improvements.

The parametric sweep studies showed that the trained models generally reproduced the deterministic PCA trends within the supported operating envelope. Models A and B showed strong agreement across the dry and afterburning ideal-cycle sweeps, confirming that the surrogate framework can learn both fixed-property and variable-thermochemistry ideal-cycle behavior. Model C also captured the dominant real-cycle trends, especially in the Mach number and compressor pressure ratio directions, although small deviations appeared in more strongly coupled directions such as fan pressure ratio, burner exit total temperature,

mixer pressure recovery, and afterburner exit total temperature.

The extrapolation studies demonstrated that surrogate reliability decreases significantly outside the accepted feasible manifold. The results showed that the valid prediction region is not defined only by raw input bounds, but by the high-dimensional subset of points that satisfy the cycle equations and feasibility filters. Outside this supported region, the networks often produced smooth numerical continuations that were not physically guaranteed. The most common failure modes were boundary smoothing, coupled-output drift, and loss of accuracy near low pressure-ratio, low thermal-loading, high-Mach, and strongly coupled real-cycle regions.

The runtime benchmarks showed that the neural network surrogates are most valuable for batch prediction. For isolated single-point evaluations, the deterministic PCA solvers were often faster because the neural network pipeline includes input formatting, normalization, inference, denormalization, and output packaging. For larger batches, however, the effective surrogate cost per point decreased substantially, producing useful speedup for large-scale sweeps, design-space exploration, optimization pre-search, and potential control-oriented studies.

The limitations discussion clarified the scope of the present framework. The models are surrogates of the PCA reference formulation and therefore inherit the assumptions of the underlying cycle model. They do not include full component maps, transient dynamics, combustion instability, detailed afterburner operability, resolved mixer flow, or nozzle plume physics. The neural networks also do not enforce feasibility as a hard constraint during inference and should not be treated as reliable outside the sampled feasible envelope without deterministic verification.

The practical implication of the results is that the surrogate framework is most useful as a rapid cycle-level prediction tool. It can accelerate repeated evaluation during preliminary design, broad parametric studies, and early optimization workflows, while the deterministic PCA solver remains necessary for verification of selected points and boundary cases. This establishes a balanced workflow in which the neural network provides speed and the PCA solver provides physical authority.

Finally, several improvements were identified for future work. These include a continuous or learned neural gating function, systematic hyperparameter optimization, expanded dataset generation, boundary-refined sampling, feasibility classification, and uncertainty quantification. Together, these improvements would increase robustness near the edge of the feasible manifold and provide a stronger foundation for future studies.

Chapter 7: Conclusions and Future Work

7.1 Conclusion

This project developed and evaluated a physics-guided neural-network surrogate framework for predicting the performance of mixed-flow afterburning turbofan cycles. The work combined parametric cycle analysis, feasibility-filtered dataset generation, staged neural-network development, and structured validation studies to determine whether machine learning could reproduce the behavior of ideal and real mixed-flow afterburning turbofan models.

The project was organized around a forward surrogate objective. Instead of replacing the propulsion model with an unconstrained black-box predictor, the neural networks were trained to approximate deterministic parametric cycle solutions generated from the governing engine equations. This allowed the surrogate models to remain tied to the thermodynamic and gas-dynamic relationships of the reference cycle while providing faster prediction for large batches of candidate operating points.

Three model families were developed. Model A established the fixed-property ideal-cycle baseline, Model B extended the ideal-cycle framework to include thermochemical variation, and Model C applied the same surrogate strategy to the full real-cycle formulation with component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. Across all three stages, dry and afterburning operation were handled using a binary-gated mixture-of-experts structure.

The results showed that the trained surrogates can reproduce many of the deterministic cycle trends within the sampled feasible operating envelope. Parametric sweep validation confirmed strong agreement for the ideal-cycle models and useful trend agreement for the real-cycle model, especially in the Mach number and compressor pressure ratio directions. The extrapolation studies also showed that model reliability decreases outside the accepted feasible manifold, which defines an important limitation of the current structure.

Overall, the project demonstrates that a physics-guided neural-network surrogate can serve as a practical forward model for mixed-flow afterburning turbofan cycle prediction. The developed framework provides a foundation for future work in larger surrogate models, uncertainty-aware feasibility screening, inverse design, and closed-loop thrust-control studies.

7.1.1 Project Objective Revisited

The objective of this project was to determine whether a machine learning surrogate could accurately predict the performance and internal flow properties of mixed-flow afterburning turbofan cycles over a broad operating envelope. The intended model was not a replacement for propulsion theory, but a faster forward approximation of a parametric cycle analysis framework suitable for large-scale evaluation, trend analysis, and potential control-oriented studies.

This objective was addressed by first developing deterministic ideal and real-cycle formulations for the mixed-flow afterburning turbofan. These relations provided the reference mapping between sampled engine inputs and the resulting thrust, TSFC, fuel-flow quantities, efficiencies, and selected internal states. The neural networks were then trained on datasets generated from this reference model after feasibility filtering was applied to remove nonphysical cycle solutions.

The project objective was also addressed through staged model development. Model A tested the feasibility of the surrogate framework on a fixed-property ideal-cycle problem. Model B increased the difficulty by introducing thermochemical variation while retaining the ideal-cycle structure. Model C then extended the approach to the real-cycle formulation with component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties.

The final results show that the objective was substantially achieved within the scope of the selected PCA reference model. The trained surrogates learned the primary dry and afterburning cycle mappings, reproduced many structured parametric trends, and provided useful batch-evaluation speedup for large prediction sets. The results also clarified the limits of the approach: the models are reliable primarily within the sampled feasible envelope and should not be used as standalone extrapolation tools outside that region.

7.1.2 Summary of Technical Approach

The technical approach used in this project combined deterministic propulsion modeling with supervised neural-network regression. The first step was to formulate the mixed-flow afterburning turbofan as a parametric cycle analysis problem. Ideal and real-cycle equation sets were implemented to map sampled flight conditions, component parameters, thermodynamic properties, and afterburner settings to performance outputs and internal cycle states.

After the cycle equations were established, candidate engine states were generated across prescribed operating bounds. Lower-dimensional ideal-cycle datasets were generated using space-filling sampling methods, while the higher-dimensional real-cycle dataset used Sobol sampling and dataset sharding to make large-scale generation manageable. Each candidate point was evaluated with the deterministic cycle solver and passed through feasibility filters before being accepted into the final training dataset.

The feasibility filters formed an essential part of the technical method. Samples were

rejected when they produced nonphysical thermodynamic states, invalid fuel-air ratios, unacceptable bypass ratios, failed mixer or nozzle solutions, negative or near-zero thrust, invalid TSFC, or non-finite efficiency values. This ensured that the neural networks were trained on physically meaningful cycle solutions rather than numerical artifacts from failed evaluations.

The accepted datasets were then separated into dry and afterburning subsets. Separate neural-network experts were trained for each operating mode using a binary mixture-of-experts structure. Each expert was trained as a multi-output regression model using normalized inputs and outputs, weighted loss functions, validation monitoring, and stored prediction artifacts for later evaluation.

The final step was validation and analysis. The trained models were assessed using test-set parity plots, residual plots, consistency checks, parametric sweeps, extrapolation studies, and runtime benchmarks. This allowed the surrogates to be evaluated not only by statistical error, but also by physical trend agreement and computational usefulness.

7.1.3 Summary of Machine Learning Framework

The machine learning framework was built around a supervised forward-surrogate formulation. Each neural network was trained to approximate the deterministic parametric cycle mapping from a selected input vector to a corresponding output vector of engine performance quantities and internal cycle states. The networks were not trained as independent black-box models; instead, the training labels were generated from the ideal or real mixed-flow afterburning turbofan cycle relations.

The final architecture used a binary-gated mixture-of-experts structure. Dry and afterburning operation were treated as separate regression problems because afterburner activation changes the fuel-flow structure, downstream thermal state, nozzle behavior, and efficiency response of the engine. The dry expert learned the non-afterburning cycle mapping, while the wet expert learned the afterburning mapping. During prediction, the afterburner mode indicator routed each input to the appropriate expert.

The surrogate models were developed in three stages. Model A used the fixed-property ideal-cycle formulation and served as the baseline proof-of-capability model. Model B retained the ideal-cycle structure but introduced thermochemical variation through active gas-property and fuel heating value inputs. Model C extended the framework to the real-cycle formulation and included additional inputs associated with component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties.

All models used normalized inputs and outputs to improve numerical conditioning during training. Weighted regression losses were used to balance the influence of outputs with different physical meanings and numerical scales. For the ideal-cycle models, additional consistency terms helped preserve selected algebraic relationships between fuel-flow and efficiency quantities. For the real-cycle model, physical guidance was provided primarily through feasibility-filtered training data, supervised prediction of primary outputs, and auxiliary

internal-state prediction.

The trained networks were stored with their normalization constants, transformation rules, metadata, and prediction routines so that inference could be performed consistently after training. This made it possible to use the same models for test-set validation, parametric sweeps, extrapolation studies, and runtime benchmarking. Overall, the machine learning framework provided a consistent method for converting the deterministic cycle model into a fast forward surrogate suitable for repeated prediction over the sampled feasible operating envelope.

7.2 Major Accomplishments

The project produced a complete workflow for generating, training, validating, and analyzing neural-network surrogates for mixed-flow afterburning turbofan cycle prediction. The main accomplishments are not limited to the final trained models. They also include the development of the deterministic cycle-analysis framework, the construction of large feasibility-filtered datasets, the staged surrogate hierarchy, and the validation studies used to assess physical trend agreement.

A major result of the project is that the surrogate framework was carried from theory through implementation and validation. The ideal and real parametric cycle models were implemented as dataset generators, the accepted cycle solutions were used to train multiple neural-network experts, and the trained models were tested through both statistical and physical validation studies. This produced a functional end-to-end pipeline for cycle-level surrogate modeling.

The following subsections summarize the main technical accomplishments of the project. These include the parametric cycle analysis implementation, dataset generation and feasibility filtering, development of Models A, B, and C, the mixture-of-experts architecture, validation through parametric sweep studies, and the runtime acceleration demonstrated by batch surrogate inference.

7.2.1 Parametric Cycle Analysis Implementation

A major accomplishment of this project was the implementation of deterministic parametric cycle analysis models for the mixed-flow afterburning turbofan configuration. These models provided the physics-based reference used for all dataset generation, neural-network training, validation, and analysis. Both ideal and real-cycle formulations were implemented, allowing the surrogate framework to be developed in stages rather than applied immediately to the highest-dimensional case.

The ideal-cycle implementation provided a controlled baseline for studying the mixed-flow afterburning turbofan response under simplified assumptions. It included the main thermodynamic relationships needed to compute specific thrust, TSFC, fuel-air ratios, bypass

ratio, and efficiency quantities for dry and afterburning operation. This model served as the foundation for Models A and B.

The real-cycle implementation expanded the reference model by including component losses, polytropic efficiencies, burner and afterburner efficiencies, mixer pressure recovery, nozzle loss, nonideal pressure expansion, and mode-dependent gas properties. This allowed Model C to be trained on a more physically complete cycle mapping while retaining a computationally practical one-dimensional PCA structure.

The PCA implementation also included internal state calculations for quantities such as mixed-stream total temperature, mixed-stream total pressure, nozzle exit Mach number, nozzle area ratio, and component temperature ratios. These quantities were important because they allowed the neural networks to be validated beyond final thrust and fuel consumption alone.

Overall, the parametric cycle analysis implementation established the physical backbone of the project. It converted the mixed-flow afterburning turbofan problem into a repeatable forward mapping from engine inputs to performance and state outputs, which made the later machine learning surrogate development possible.

7.2.2 Dataset Generation and Feasibility Filtering

Another major accomplishment was the generation of large training datasets from the deterministic PCA models. The datasets were constructed by sampling the defined operating envelope, evaluating each candidate point with the appropriate cycle solver, and retaining only samples that satisfied the physical and numerical feasibility requirements. This converted the PCA formulation into supervised learning data suitable for neural-network training.

The dataset-generation process was developed in stages to match the increasing complexity of the surrogate models. The ideal-cycle datasets used lower dimensional sampling for Models A and B, while the real-cycle Model C dataset required larger-scale Sobol sampling and sharded generation to manage the higher-dimensional input space. This allowed the final real-cycle dataset to cover a broader feasible envelope without relying on an impractical structured grid.

Feasibility filtering was essential to the quality of the generated data. Candidate points were rejected when they produced invalid thermodynamic states, nonphysical fuel-air ratios, unacceptable bypass ratios, failed mixer or nozzle solutions, invalid thrust or TSFC values, or non-finite efficiency quantities. These filters prevented the neural networks from being trained on failed cycle evaluations or numerical artifacts.

The final accepted datasets therefore represented feasible subsets of the sampled operating envelope rather than arbitrary collections of input-output pairs. This was particularly important for Model C, where the real-cycle input space contains many coupled variables and the valid operating region is only a portion of the raw input bounds. The feasibility-filtered data provided the physical foundation needed for training a useful surrogate model.

Overall, the dataset-generation and feasibility-filtering workflow established a repeatable method for producing physically meaningful training data. This was one of the key enabling steps of the project because the quality of the surrogate models depended directly on the coverage and physical consistency of the accepted datasets.

7.2.3 Development of Models A, B, and C

A central accomplishment of the project was the development of three progressively more complex surrogate model families. This staged structure allowed the neural-network framework to be tested first on a simplified ideal-cycle problem before being extended to variable thermochemistry and then to the full real-cycle formulation. This reduced development risk and made it possible to isolate the effects of increasing physical complexity.

Model A was developed as the fixed-property ideal-cycle surrogate. In this model, the gas properties and fuel heating value were held constant while the main cycle variables were varied over the sampled operating envelope. Model A served as the first proof-of-capability case because it tested whether the network could learn the basic mixed-flow afterburning turbofan mapping for both dry and afterburning operation.

Model B retained the ideal-cycle equation structure but introduced variation in thermochemical properties. The ratio of specific heats and fuel heating value were treated as active input variables, while the specific heat was computed consistently from the sampled gas properties. This increased the dimensionality of the learned mapping while preserving the same general ideal-cycle framework used by Model A. Model B therefore tested whether the surrogate approach remained effective when the ideal-cycle assumptions were relaxed.

Model C was developed as the real-cycle surrogate. This model expanded the input space to include component efficiencies, pressure losses, mixer pressure recovery, afterburner pressure loss, nozzle loss, non-ideal expansion behavior, and separate gas-property variables for different engine regions. Model C was the most physically complete and highest-dimensional model developed in the project, and it represented the primary production-level surrogate for the real mixed-flow afterburning turbofan cycle formulation.

The development of these three model families demonstrated that the same general surrogate strategy could be applied across increasing levels of cycle complexity. Model A established the baseline learning capability, Model B confirmed that the architecture could accommodate thermochemical variation, and Model C extended the method to a real-cycle formulation with coupled component losses and internal state behavior. This staged progression was one of the main strengths of the project because it provided both validation confidence and a clear development path from simplified theory to the final real-cycle surrogate.

7.2.4 Mixture-of-Experts Surrogate Architecture

Another major accomplishment was the implementation of the wet and dry mixture-of-experts surrogate architecture. This structure allowed non-afterburning and afterburning operation to be modeled as separate but related regression problems. Instead of forcing a single neural network to learn both operating regimes simultaneously, separate dry and wet experts were trained and selected using the afterburner mode indicator.

This architecture was important because afterburner activation changes the structure of the cycle response. In dry operation, the afterburner fuel-air ratio is zero and the nozzle receives the mixed stream produced by the fan and core flow. In afterburning operation, additional fuel is added downstream of the mixer, raising the nozzle-inlet total temperature and changing the thrust, TSFC, total fuel-air ratio, and efficiency behavior. Separating these regimes reduced the difficulty of the learning problem and allowed each expert to focus on a smoother local mapping.

The binary gate provided a simple and interpretable prediction pathway. For a specified dry operating point, the dry expert was evaluated. For a specified afterburning operating point, the wet expert was evaluated. This avoided the need for the network to infer the operating mode internally and preserved the physical distinction between inactive and active augmentation.

The same mixture-of-experts strategy was retained across Models A, B, and C. This provided architectural continuity while the physical complexity of the underlying cycle model increased. Model A and Model B used the architecture for ideal-cycle prediction, while Model C applied the same dry/wet expert structure to the real-cycle surrogate with component losses, mixer behavior, afterburner pressure loss, and nozzle loss included.

Overall, the mixture-of-experts architecture was a key part of making the surrogate framework practical. It improved organization of the training datasets, reduced mode-dependent discontinuities in the learned mappings, and provided a clear route for evaluating dry and afterburning operation within a single prediction framework.

7.2.5 Validation Through Parametric Sweep Studies

A major accomplishment of the project was the use of parametric sweep studies to validate the trained surrogate models beyond random test-set accuracy. The sweep studies provided a structured method for determining whether the neural networks learned physically meaningful cycle trends rather than only producing low average error on held-out samples.

The validation method used anchor points selected from feasible dataset regions. Around each anchor point, one engine or operating variable was varied while the remaining variables were held fixed. The resulting neural-network predictions were compared against the deterministic PCA reference over the same sweep range. This allowed the local response of the surrogate to be checked directly against the underlying cycle model.

Parametric sweeps were performed for both dry and afterburning operation. The dry-mode

sweeps examined the response to variations in quantities such as freestream Mach number, compressor pressure ratio, fan pressure ratio, burner temperature ratio or burner exit total temperature, and mixer pressure recovery for the real-cycle model. The afterburning-mode sweeps extended this analysis to augmented operation, including afterburner temperature ratio or afterburner exit total temperature.

The sweep results showed that Models A and B reproduced the ideal-cycle reference trends with strong agreement across most of the tested directions. This confirmed that the mixture-of-experts surrogate framework could learn both fixed-property and variable-thermochemistry ideal-cycle behavior. Model C also reproduced the dominant real-cycle trends, although larger deviations appeared in more strongly coupled directions such as fan pressure ratio, burner exit total temperature, mixer pressure recovery, and afterburner exit total temperature.

These studies were important because they revealed model behavior that could not be fully assessed from test-set parity plots alone. A model can have low average error while still failing to reproduce the correct local trend in a specific operating direction. The parametric sweeps therefore provided a more engineering-relevant validation method by testing the response of the surrogate under controlled changes in physically meaningful variables.

Overall, the parametric sweep studies demonstrated that the trained surrogates were capable of reproducing the primary PCA trends within the sampled feasible operating envelope. They also identified the regions where the models were most sensitive, providing useful guidance for later recommendations such as boundary-refined sampling, feasibility classification, and uncertainty quantification.

7.2.6 Runtime Acceleration Demonstration

A final major accomplishment was the demonstration of runtime acceleration through neural-network surrogate inference. The runtime benchmarking studies compared the deterministic PCA solvers against the trained surrogate models for single-point and batch prediction cases. This showed where the surrogate framework provides practical computational value and where the deterministic cycle solver remains more efficient.

The benchmark results showed that neural-network inference is not always faster for isolated single-point evaluations. For one input condition, the surrogate prediction workflow includes input formatting, normalization, network evaluation, de-normalization, and output reconstruction. This overhead can exceed the cost of directly evaluating the algebraic PCA equations, especially for the lower-dimensional ideal-cycle models.

The value of the surrogate becomes more significant during batch prediction. When many operating points are evaluated at once, the neural-network overhead is distributed across the full batch, and the average cost per prediction decreases substantially. This makes the trained models useful for large parametric sweeps, design-space exploration, optimization pre-search, and other workflows requiring repeated evaluation of many candidate engine states.

The runtime advantage is most relevant for the real-cycle surrogate. Model C has the highest-dimensional input space and the most complex deterministic reference calculation because it includes component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. As the physical complexity of the cycle model increases, the benefit of replacing repeated deterministic evaluations with batched surrogate inference becomes more meaningful.

The runtime study also clarified the appropriate role of the surrogate. The neural network should not be viewed as a replacement for the PCA solver in all situations. Instead, the surrogate is most useful as a rapid screening and prediction tool for large sets of feasible inputs, while the deterministic PCA solver remains the preferred method for verifying selected points, boundary cases, and extrapolated predictions.

Overall, the runtime acceleration demonstration showed that the surrogate framework can reduce computational cost when used in the proper operating mode. This supports the use of the trained models for repeated cycle-level prediction and provides the computational foundation for future optimization, inverse modeling, and closed-loop control studies.

7.3 Principal Findings

The results of this project produced several findings regarding the use of physics-guided neural-network surrogates for mixed-flow afterburning turbofan cycle prediction. These findings are based on the staged development of Models A, B, and C, the feasibility-filtered training datasets, the baseline validation results, the parametric sweep studies, the extrapolation analysis, and the runtime benchmarks.

The main conclusion is that neural-network surrogates can reproduce the dominant behavior of deterministic parametric cycle models when they are trained on sufficiently consistent and physically filtered data. However, the results also show that surrogate accuracy is strongly tied to the sampled feasible envelope. The models perform best when used for interpolation within well-supported regions of the dataset and become less reliable when evaluated near or beyond the edge of the feasible manifold.

The following subsections summarize the principal findings of the project. These include the effectiveness of neural surrogates for cycle prediction, the importance of separating wet and dry operation, the value of physics-guided dataset construction, the distinction between interpolation and extrapolation behavior, and the computational value of the surrogate framework.

7.3.1 Effectiveness of Neural Surrogates for Cycle Prediction

The results show that neural-network surrogates can effectively approximate parametric cycle analysis mappings for mixed-flow afterburning turbofan engines when the training data are generated from a consistent physical reference model. Across the staged model sequence,

the networks learned the relationship between sampled engine inputs and the corresponding thrust, TSFC, fuel-flow quantities, efficiencies, and selected internal states.

Model A demonstrated that the surrogate framework could reproduce the fixed-property ideal-cycle response. This was the simplest case, but it was an important proof of capability because it confirmed that the network structure, normalization process, training workflow, and dry/wet expert separation could learn the basic mixed-flow afterburning turbofan mapping. The strong agreement observed in the baseline validation and parametric sweep studies showed that the ideal-cycle equations could be approximated accurately over the sampled feasible envelope.

Model B showed that this capability remained effective after thermochemical variation was introduced. Allowing gas-property and fuel-heating-value inputs to vary increased the dimensionality of the mapping, but the surrogate was still able to reproduce the major ideal-cycle trends. This confirmed that the framework was not limited to a single fixed-property approximation and could handle a more general ideal-cycle dataset.

Model C demonstrated that the same general approach could be extended to the real-cycle formulation. The real-cycle surrogate was more difficult to train because the input space included component losses, efficiencies, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. Even with this added complexity, the trained model captured the dominant cycle behavior in many of the validation studies, particularly in well-supported regions of the sampled feasible envelope.

The effectiveness of the surrogates was strongest for interpolation within the accepted training domain. When the input conditions remained inside regions well represented by feasible training samples, the neural-network predictions generally followed the deterministic PCA reference trends. This indicates that the networks learned more than isolated pointwise behavior; they also captured many of the local relationships between engine parameters and performance outputs.

At the same time, the results show that surrogate effectiveness depends on the quality and coverage of the dataset. The neural networks approximate the sampled PCA mapping, but they do not solve the governing cycle equations during inference. Therefore, their predictions are reliable only to the extent that the queried input lies within a region represented by physically consistent training data. This finding reinforces the need to treat the neural surrogate as a fast approximation of the PCA model rather than as an unrestricted engine solver.

7.3.2 Importance of Wet and Dry Expert Separation

The results support the decision to separate dry and afterburning operation into independent neural-network experts. The two operating modes are governed by the same general engine architecture, but the activation of the afterburner changes the structure of the cycle response. In dry operation, the afterburner fuel-air ratio is zero and the nozzle expands the mixed stream formed by the core and bypass flow. In afterburning operation, additional fuel

is added downstream of the mixer, raising the downstream total temperature and changing the thrust, TSFC, fuel-flow, and efficiency behavior.

This difference creates a mode-dependent discontinuity in several outputs. The afterburner fuel-air ratio is identically zero in dry operation but positive in accepted wet samples. The total fuel-air ratio, afterburner exit state, nozzle expansion behavior, and thrust specific fuel consumption also change sharply when augmentation is active. If a single neural network were forced to learn both regimes at once, it would need to represent this abrupt transition internally while also learning the continuous cycle behavior within each regime.

The mixture-of-experts structure reduced this difficulty by allowing each expert to learn a smoother mapping. The dry expert learned the non-afterburning cycle response without being influenced by augmented samples, while the wet expert learned the afterburning response without being diluted by dry samples. This separation improved the organization of the training data and made the prediction workflow easier to interpret.

The staged validation results showed that this approach remained useful across all three model families. For Models A and B, the dry and wet experts both reproduced the ideal-cycle trends with strong agreement. For Model C, the separation was even more important because the real-cycle formulation included additional component losses, gas-property variation, mixer behavior, and afterburner pressure loss. Keeping dry and wet operation separate prevented the highest-dimensional model from having to learn inactive and active augmentation as one combined response surface.

The binary expert gate also provided a clear inference structure. The afterburner mode was specified directly, and the corresponding expert was evaluated. This avoided ambiguity in the prediction process and preserved the physical distinction between dry and augmented operation. The limitation is that the present gate does not model gradual afterburner lightoff, staging, or continuous transition behavior. Future work could improve this by replacing the binary gate with a learned or continuous gating function.

Overall, wet and dry expert separation was one of the most important architectural decisions in the project. It reduced the complexity of the learning problem, preserved the physical distinction between operating modes, and provided a practical structure for extending the surrogate framework toward future adaptive afterburner and thrust-control applications.

7.3.3 Importance of Physics-Guided Dataset Construction

The project results show that the quality of the neural-network surrogate is strongly dependent on the construction of the training dataset. The networks did not learn from arbitrary numerical samples; they learned from accepted cycle solutions generated by the deterministic PCA models. This made the dataset-generation process one of the most important parts of the overall framework.

The physics-guided dataset construction provided two main benefits. First, it anchored the supervised learning problem to the thermodynamic and gas-dynamic structure of the mixed-flow afterburning turbofan cycle. Each accepted label was produced by the same

station-based cycle equations used to define thrust, TSFC, fuel-flow quantities, efficiencies, mixer behavior, and nozzle behavior. As a result, the neural networks were trained to reproduce a physically interpretable reference mapping rather than an unconstrained numerical relationship.

Second, the feasibility filters prevented the networks from learning failed cycle evaluations. Candidate samples were rejected when they produced nonphysical temperatures or pressures, invalid fuel-air ratios, unacceptable bypass ratios, failed mixer or nozzle solutions, invalid thrust or TSFC, or non-finite efficiency values. This was especially important for Model C, because the real-cycle formulation contains a much larger set of coupled variables and therefore a larger number of possible failure modes.

The results of the parametric sweep and extrapolation studies show why this filtering process is necessary. The accepted training data do not fill the raw input bounds uniformly. Instead, they occupy a feasible manifold inside the larger sampled input space. Within this feasible manifold, the surrogate predictions generally followed the deterministic PCA trends. Near the edge of the manifold, or outside it, the models became less reliable even when the queried variable was still near a nominal input bound.

This finding is important because it shows that the operating envelope of the surrogate is not defined only by the prescribed minimum and maximum values of each input. It is defined by the combination of those bounds with the thermodynamic, fuel-flow, mixer, nozzle, and performance constraints used during data acceptance. A surrogate trained on unfiltered samples would likely learn numerical artifacts from invalid cycle solutions, while a surrogate trained on filtered samples learns the physically meaningful subset of the PCA response.

The staged datasets also helped isolate model behavior. Model A established the baseline fixed-property ideal-cycle mapping, Model B introduced thermochemical variation, and Model C expanded the input space to include real-cycle losses and efficiencies. This progression made it possible to increase physical complexity while retaining control over the data-generation process and validation workflow.

Overall, physics-guided dataset construction was essential to the success of the project. The final surrogate performance depended not only on network architecture or training settings, but also on the physical consistency, coverage, and feasibility of the accepted data. This finding supports the use of dataset construction and feasibility filtering as core design elements for future propulsion surrogate models.

7.3.4 Interpolation Capability and Extrapolation Limitations

The validation results show that the surrogate models are most reliable when used for interpolation within the sampled feasible operating envelope. In these regions, the neural networks were able to reproduce the dominant parametric cycle trends because similar physical states were represented in the accepted training data. This was shown in the parametric sweep studies, where the surrogate predictions generally followed the deterministic PCA reference inside the supported portions of the sweep ranges.

The extrapolation studies showed a different behavior. When the models were evaluated near or outside the supported training envelope, prediction quality decreased. The networks often produced smooth numerical continuations beyond the region where the deterministic PCA solution was supported by feasible training data. Although these predictions may appear continuous, they are not guaranteed to represent physically valid engine behavior because the neural networks do not solve the governing cycle equations during inference.

A key finding is that the valid prediction domain is not defined only by the raw minimum and maximum bounds of each input variable. The accepted training samples occupy a feasible manifold inside the full high-dimensional input space. A point can remain near the nominal range of one variable while still leaving the supported region of the combined cycle solution because the fixed values of the other inputs no longer produce a feasible or well-sampled engine state. This distinction is especially important for Model C, where the real-cycle formulation contains many coupled losses, efficiencies, gas properties, mixer quantities, and nozzle quantities.

The main extrapolation failure modes were boundary smoothing, coupled-output drift, and unsupported smooth continuation. Boundary smoothing occurred when the PCA reference changed sharply near a feasibility limit while the neural network produced a smoother response. Coupled-output drift occurred when the primary thrust trend remained reasonable but secondary quantities such as TSFC, fuel-air ratios, bypass ratio, or efficiency deviated more strongly. Unsupported smooth continuation occurred when the network continued to return finite outputs even though the corresponding deterministic cycle behavior was outside the supported feasible region.

Models A and B showed more stable extrapolation behavior than Model C because their ideal-cycle mappings were lower-dimensional and less strongly coupled. However, even these models showed reduced reliability near low pressure-ratio, low thermal-loading, and high-Mach edge cases. Model C showed the greatest extrapolation sensitivity, particularly in directions associated with fan pressure ratio, burner exit total temperature, mixer pressure recovery, and afterburner exit total temperature.

The principal conclusion is that the trained surrogates should be treated as in-domain interpolation models, not unrestricted extrapolation models. Within well-supported regions of the accepted dataset, they can provide rapid and useful cycle predictions. Near feasibility boundaries or outside the sampled domain, predictions should be checked against the deterministic PCA solver. This finding motivates future additions such as feasibility classification, boundary-refined sampling, and uncertainty quantification so that unsupported predictions can be identified before being used in design or control studies.

7.3.5 Computational Value of the Surrogate Model

The runtime benchmarking results show that the computational value of the surrogate model depends on how it is used. For isolated single-point evaluations, the deterministic PCA solver can be faster than the neural network because the cycle equations are algebraic and relatively inexpensive to evaluate. A single surrogate prediction also requires input

formatting, normalization, network inference, output denormalization, and reconstruction of the predicted physical quantities. This overhead reduces the advantage of the neural network for individual points.

The surrogate becomes more valuable when many points are evaluated together. For batch prediction, the preprocessing and inference overhead is distributed across the full set of input samples, reducing the average cost per prediction. This makes the neural-network framework useful for applications that require large numbers of repeated cycle evaluations, such as parametric sweeps, design-space exploration, sensitivity studies, optimization presearch, and future control-oriented simulation.

The computational advantage is especially relevant for the real-cycle model. Model C includes more input variables, more coupled physical behavior, and more extensive feasibility-related state calculations than the ideal-cycle models. As the deterministic reference model becomes more complex, the benefit of replacing repeated cycle evaluations with batched surrogate inference becomes more significant. This suggests that surrogate acceleration becomes increasingly valuable as the underlying propulsion model is expanded.

The results also clarify that the surrogate and PCA solver should be used together rather than treated as competing tools. The neural network is most useful for rapid screening across many feasible operating points, while the deterministic PCA solver remains the preferred tool for verification, boundary-case analysis, and extrapolated predictions. This creates a practical workflow in which the surrogate provides computational speed and the PCA solver provides physical authority.

The principal finding is that the surrogate model provides meaningful computational value for large-scale evaluation tasks. Its greatest advantage is not replacing a single deterministic calculation, but enabling rapid evaluation of many candidate engine states. This supports its use as a forward model for future optimization, inverse prediction, and closed-loop thrust-control studies where repeated model calls would otherwise become a computational bottleneck.

7.4 Recommended Future Work

The results of this project show that physics-guided neural-network surrogates can reproduce mixed-flow afterburning turbofan cycle behavior within a sampled feasible operating envelope. At the same time, the validation and extrapolation studies identified several areas where the framework could be improved. These improvements are directed toward increasing physical robustness, expanding the valid operating domain, improving accuracy near feasibility boundaries, and extending the surrogate from forward prediction toward inverse control.

The most important future work is not simply to train a larger neural network. The present results show that surrogate quality depends on the interaction between dataset construction, feasibility filtering, model architecture, training strategy, and validation method. Future

improvements should therefore focus on strengthening the full modeling pipeline rather than modifying only one part of the framework.

The following subsections summarize the main recommended extensions of the project. These include improved neural gating, refined physics-guided loss functions, systematic hyperparameter and architecture optimization, expanded and boundary-refined datasets, component map integration, uncertainty quantification, feasibility classification, and development of an inverse model for closed-loop thrust control.

7.4.1 Improved Neural Gating Architecture

One recommended extension is to replace the present binary dry/wet gate with a more flexible neural gating architecture. In the current framework, dry and afterburning operation are treated as two discrete operating modes. This was appropriate for the present study because the datasets were generated as separate dry and wet cases, but it does not represent gradual afterburner lightoff, fuel staging, partial augmentation, or smooth transitions between operating regimes.

A future version of the surrogate could use a continuous gating function that blends the dry and afterburning experts according to the augmentation state of the engine. Instead of selecting only one expert, the model could allow both experts to contribute when the engine is operating between fully dry and fully augmented conditions. This would provide a smoother representation of the cycle response and would reduce the discontinuity introduced by the present binary mode selection.

The gating function could be based on afterburner command, afterburner fuel-air ratio, afterburner exit temperature, nozzle-area schedule, or another variable that identifies the degree of augmentation. A learned gate could also use the full input vector to determine how strongly each expert should contribute at a given operating condition. This would allow the model to capture transition behavior without forcing the entire dry and wet response surface into a single network.

This improvement would be especially valuable for future control applications. A thrust controller must be able to move between dry, partially augmented, and fully augmented operation without producing discontinuous commands or unrealistic jumps in predicted engine state. A continuous gating architecture would provide a more suitable forward model for inverse prediction, optimization, and closed-loop thrust-tracking studies.

The main requirement for this improvement is additional training data in the transition region. The present datasets define dry and wet operation as separate modes, so they do not contain enough information to learn true intermediate afterburner behavior. Future work should therefore include partially augmented cases and afterburner scheduling variables directly in the dataset. With this expanded data, a learned gating architecture could provide a more realistic representation of adaptive afterburner operation.

7.4.2 Physics-Guided Loss Function Refinement

Another recommended extension is to refine the physics-guided loss functions used during training. In the present framework, the models are guided primarily through PCA-generated training labels, feasibility-filtered datasets, normalized output prediction, and selected consistency terms. This was sufficient to demonstrate the feasibility of the surrogate approach, but future work could strengthen the connection between the neural-network predictions and the underlying cycle physics.

For the ideal-cycle models, additional loss refinement could focus on preserving known relationships between fuel-flow quantities, efficiency quantities, thrust, and TSFC. These relationships were already partially accounted for through consistency checks, but they could be weighted and tuned more systematically. A refined loss structure could place greater emphasis on outputs that are physically coupled rather than treating every prediction error as an independent quantity.

For the real-cycle model, physics-guided loss refinement would be even more important. Model C includes coupled behavior from component efficiencies, pressure losses, mixer pressure recovery, afterburner pressure loss, nozzle loss, and mode-dependent gas properties. A future loss function could include stronger penalties for predictions that violate expected physical consistency between internal states and final performance metrics. This would help reduce cases where thrust remains accurate while fuel consumption, efficiency, or internal state predictions drift.

Care must be taken when adding physics-based penalties. If the physics terms are weighted too strongly, the model may reduce residual inconsistency while sacrificing primary prediction accuracy. If they are weighted too weakly, they may not meaningfully improve physical behavior. Future work should therefore treat loss weighting as a tunable part of model development, using validation performance, parametric sweeps, and boundary behavior to determine whether the added penalties improve the surrogate in practice.

A useful refinement would be to introduce physics-guided loss terms progressively during training. Early training could focus on learning the primary supervised mapping, while later training could increase the influence of physical consistency penalties after the model has already learned the general response surface. This type of staged training may improve stability and prevent the optimization process from being dominated by difficult physical residuals too early.

Overall, refining the physics-guided loss function would improve the physical robustness of the surrogate models. The goal would not be to make the neural network replace the deterministic PCA solver, but to make its predictions more consistent with the coupled structure of the cycle model, especially near regions where data are sparse or outputs are strongly coupled.

7.4.3 Hyperparameter and Architecture Optimization

Another recommended area of future work is systematic hyperparameter and architecture optimization. The neural-network architectures used in this project were selected to provide stable training and sufficient model capacity for the staged surrogate-development process. While these architectures were effective for demonstrating the feasibility of the approach, they were not the result of an exhaustive optimization study.

Future work should investigate the effect of network depth, layer width, activation function, learning-rate schedule, batch size, regularization strength, residual connections, and output-loss weighting. These choices can strongly influence both prediction accuracy and training stability, particularly for the higher-dimensional real-cycle model. Model C is the most important candidate for this improvement because it contains the largest input space and the strongest coupling between outputs.

Architecture optimization should also consider whether all outputs should be predicted by a single shared network head or separated into specialized output groups. For example, primary performance quantities such as thrust and TSFC may benefit from a different output pathway than internal diagnostic states such as mixer pressure, nozzle Mach number, or area ratio. A multi-head architecture could allow the model to share common cycle features while still specializing the final layers for different classes of outputs.

Additional optimization should examine model capacity relative to dataset size. A network that is too small may underfit the coupled real-cycle response, while a network that is too large may learn local noise or become less stable near sparse regions of the feasible manifold. This balance should be assessed using validation loss, test-set error, parametric sweep agreement, and extrapolation behavior rather than only a single training metric.

Future studies could also compare the current MLP-based structure against alternative regression architectures. Candidate approaches include deeper residual networks, ensemble models, mixture-of-experts variants with learned gates, and models with monotonic or constraint-aware layers for selected outputs. These alternatives may improve accuracy in strongly nonlinear regions without sacrificing interpretability or inference speed.

Overall, hyperparameter and architecture optimization would provide a more systematic way to improve surrogate performance. The present models establish that the approach is viable, but a dedicated optimization study could reduce prediction error, improve trend agreement, and increase robustness for the real-cycle surrogate.

7.4.4 Expanded and Boundary-Refined Datasets

Another important direction for future work is the expansion and refinement of the training datasets. The present datasets were sufficient to train the surrogate models and demonstrate useful prediction capability within the sampled feasible envelope. However, the extrapolation and edge-envelope studies showed that model reliability decreases near sparse regions, feasibility boundaries, and outside the accepted training manifold. This makes dataset im-

provement one of the most direct ways to improve future surrogate performance.

Expanded datasets would increase coverage of the valid operating envelope by adding more accepted samples across the flight condition, component parameter, thermochemical, mixer, afterburner, and nozzle input spaces. This is especially important for Model C because the real-cycle formulation contains the largest number of coupled variables. As the input dimension increases, the accepted feasible points occupy a smaller and more complex portion of the raw sampled domain, requiring more data to represent the cycle response adequately.

Boundary-refined sampling would focus specifically on regions near feasibility limits. The present results showed that several of the largest deviations occurred near low fan pressure ratio, low compressor pressure ratio, low thermal loading, mixer pressure recovery limits, and afterburner temperature limits. These regions are important because the deterministic PCA response can change rapidly as the cycle approaches fuel-flow, turbine work-balance, mixer, nozzle, or performance constraints. Adding targeted samples near these boundaries would help the neural network learn the shape of the response before the solution becomes infeasible.

A refined dataset-generation workflow could use the existing feasibility filters to identify regions where candidate samples frequently fail or where accepted samples cluster near constraint boundaries. Additional samples could then be generated locally in these regions instead of distributing all new samples uniformly across the full input bounds. This would improve sample efficiency because new data would be added where the current surrogate is most sensitive.

Future datasets should also include transition cases if the gating architecture is improved. The current dry and wet datasets are separated into discrete operating modes. If a continuous neural gate is developed, additional partially augmented samples will be required so that the model can learn the intermediate response between fully dry and fully afterburning operation. This would improve the usefulness of the surrogate for adaptive afterburner scheduling and control-oriented applications.

Overall, expanded and boundary-refined datasets would improve both accuracy and robustness. A larger dataset would strengthen global coverage of the operating envelope, while targeted boundary sampling would improve behavior in the regions where the present models are most likely to fail. This would make the surrogate framework more reliable for design-space exploration, optimization, and future closed-loop control studies.

7.4.5 Component Map Integration

A further extension of the project would be to integrate component map behavior into the reference model and surrogate-training workflow. The present real-cycle formulation represents component behavior through pressure ratios, efficiencies, pressure recoveries, and loss parameters. This is appropriate for a cycle-level surrogate, but it does not fully capture the off-design behavior of real turbomachinery components.

In an actual turbofan engine, compressor, fan, and turbine performance depend on cor-

rected mass flow, corrected speed, pressure ratio, efficiency, and operating margin. These quantities are usually represented using component maps. Integrating these maps would allow the model to account for how the fan, compressor, and turbine move across their operating ranges as Mach number, altitude, fuel flow, afterburner state, and nozzle area change. This would make the reference model more representative of realistic engine behavior.

Component map integration would also improve the treatment of matching and operability constraints. In the present framework, feasibility is enforced through algebraic cycle checks and prescribed bounds. With component maps, the model could also evaluate surge margin, choke margin, corrected-speed limits, and turbine operating constraints. These additions would make the feasible envelope more physically meaningful and would provide stronger information for future control-oriented applications.

The fan and compressor maps would be especially important for afterburning operation because nozzle area and afterburner fuel flow affect downstream back-pressure, which can shift upstream component operating points. A map-based model could capture these interactions more directly than the current cycle formulation. This would be useful for studying how variable nozzle scheduling and afterburner heat addition should be coordinated to maintain thrust while avoiding unfavorable compressor or fan operation.

Adding component maps would increase the complexity of the deterministic reference model and the resulting surrogate. The input vector would likely need to include corrected speeds, corrected flow parameters, or schedule variables, and the output vector could include surge margin or map-location diagnostics. The dataset-generation process would also require stronger feasibility filtering because many sampled combinations would not correspond to valid matched engine states.

Despite this added complexity, component map integration would be a major step toward a more realistic propulsion surrogate. It would move the framework from a primarily cycle-level predictor toward a model capable of representing off-design component matching and operability. This would make the surrogate more useful for adaptive engine studies, nozzle scheduling, afterburner control, and future inverse thrust-command models.

7.4.6 Uncertainty Quantification and Feasibility Classification

Another recommended improvement is the addition of uncertainty quantification and feasibility classification. The present surrogate models produce direct point predictions for thrust, TSFC, fuel-flow quantities, efficiencies, and internal states. However, the models do not currently provide an estimate of prediction confidence or an automatic indication that a queried input may lie outside the supported feasible envelope.

Uncertainty quantification would help identify predictions that should not be trusted without additional verification. This is especially important near the edge of the sampled operating envelope, where the extrapolation studies showed that the neural networks can produce smooth outputs even when the deterministic cycle response is unsupported or poorly represented by accepted training data. A future surrogate should therefore return not only a

predicted output vector, but also a confidence measure associated with that prediction.

Several approaches could be used for uncertainty estimation. One practical option is to train an ensemble of surrogate models and use the spread between their predictions as an indication of uncertainty. Another option is to use dropout-based uncertainty estimation or local distance measures in normalized input space. These methods would allow the model to flag regions where the input is far from the accepted training data or where different trained models disagree significantly.

Feasibility classification would provide a complementary safeguard. Instead of relying only on the regression surrogate, a separate classifier could be trained to predict whether an input condition is likely to produce a feasible cycle solution before the forward surrogate is evaluated. This classifier could be trained using both accepted and rejected candidate samples from the dataset-generation process. Accepted samples would define feasible operation, while rejected samples would provide examples of thermodynamic, fuel-flow, mixer, nozzle, and performance failures.

A feasibility classifier would be especially useful for Model C because the real-cycle feasible region is a complex subset of the raw input space. Many input combinations can satisfy the nominal minimum and maximum bounds while still failing the coupled cycle equations. A classifier could identify these unsupported regions before the surrogate returns a misleading prediction. This would reduce the risk of treating a smooth neural-network output as a valid engine state.

Together, uncertainty quantification and feasibility classification would make the surrogate framework more reliable for design and control applications. The uncertainty estimate would indicate how much confidence should be placed in a prediction, while the feasibility classifier would determine whether the input is likely to represent a valid cycle point. Predictions with high uncertainty or low feasibility confidence could then be sent to the deterministic PCA solver for verification.

Overall, these additions would provide an important layer of protection around the neural-network surrogate. They would not replace the need for physical validation, but they would help prevent unsupported predictions from being used blindly in optimization, inverse modeling, or closed-loop thrust-control studies.

7.4.7 Inverse Model and Closed-Loop Control

The final recommended extension is the development of an inverse model and closed-loop thrust-control framework. The present project focused on the forward prediction problem: given flight condition, engine parameters, component behavior, and afterburner state, the surrogate predicts thrust, TSFC, fuel-flow quantities, efficiencies, and internal cycle states. A natural next step is to reverse this workflow and determine what engine settings are required to achieve a desired thrust command.

An inverse model would take a target thrust level and the current operating condition as inputs, then return candidate control settings such as afterburner fuel flow, afterburner exit

temperature, or nozzle-area-related parameters. The objective would not be to meet thrust alone, but to meet the desired thrust while minimizing fuel consumption and remaining inside the feasible operating envelope. This would make the framework more directly useful for adaptive afterburner scheduling and engine control studies.

The forward surrogate developed in this project provides the foundation for this inverse-control extension. Because the forward model can rapidly evaluate many candidate operating points, it could be used inside an optimizer or search routine that adjusts control variables until the predicted thrust matches the desired value. This approach would avoid training the inverse model entirely from scratch and would allow the deterministic PCA solver to remain available for verification of selected control points.

A closed-loop implementation would go one step further by evaluating the predicted engine response after each control update. In this structure, the controller would issue a candidate afterburner or nozzle command, the forward surrogate would predict the resulting thrust and operating state, and the control law would adjust the next command based on thrust error, TSFC, and feasibility constraints. This would allow the system to be tested in simulated transient command sequences even before a higher-fidelity dynamic model is introduced.

Several safeguards would be required before this extension could be used reliably. The controller should include feasibility checks, uncertainty estimation, and deterministic verification near operating boundaries. It should also account for actuator limits, rate limits, mode transitions, and the practical coupling between afterburner fuel addition and nozzle area. Without these safeguards, an inverse model could find mathematically valid commands that are not physically realistic or operationally acceptable.

The inverse-control extension would represent a major step beyond forward cycle prediction. It would move the project from predicting what the engine will do under specified inputs toward determining how the engine should be commanded to meet a performance objective. This aligns directly with the long-term goal of adaptive mixed-flow afterburning turbofan operation, where afterburner scheduling and nozzle control are coordinated to achieve thrust targets efficiently while maintaining feasible engine behavior.

7.5 Final Remarks

This project developed a physics-guided surrogate modeling framework for mixed-flow afterburning turbofan cycle prediction. The work began with deterministic parametric cycle analysis, then used the resulting ideal and real-cycle formulations to generate supervised training data. Feasibility filtering, dry and afterburning expert separation, and staged model development were used to keep the neural-network models tied to physically meaningful cycle behavior. The resulting framework provides a complete forward-modeling pipeline from cycle equations to trained surrogate prediction.

The results show that neural-network surrogates can reproduce the dominant behavior of the reference PCA models when evaluated inside the sampled feasible operating envelope.

The strongest performance was observed in well-supported interpolation regions, where the trained experts generally followed the deterministic reference trends. The runtime benchmarks also showed that the surrogate approach is most valuable for batch prediction, large parametric studies, and repeated design-space evaluation. At the same time, the extrapolation studies confirmed that the models should not be treated as unrestricted engine solvers outside the accepted feasible manifold.

The broader value of the project is that it demonstrates a controlled method for integrating machine learning with propulsion cycle analysis. The neural networks were not trained as isolated black-box predictors, but as fast approximations to a physically defined reference model. This structure makes the approach more suitable for engineering use because the dataset, outputs, constraints, and validation studies all remain connected to the underlying cycle physics. The same framework can be extended toward adaptive afterburner scheduling, inverse thrust prediction, and closed-loop control once additional feasibility and uncertainty safeguards are included.

Future development should focus on improving robustness near the edge of the valid operating envelope and increasing the physical fidelity of the reference model. Component map integration, boundary-refined sampling, uncertainty quantification, feasibility classification, and continuous neural gating would all strengthen the current framework. These improvements would allow the surrogate to move closer to practical adaptive propulsion analysis while preserving the role of the deterministic solver as the source of physical verification. Overall, this project establishes a foundation for using physics-guided machine learning as a computational tool for mixed-flow afterburning turbofan analysis and future control-oriented studies.

References

[1] Marshall, R. L., and Canuel, G. E., "Augmentation Systems for Turbofan Engines," *Combustion in Advanced Gas Turbine Systems*, edited by I. E. Smith, Pergamon, Oxford, 1968, pp. 129-151.
https://doi.org/10.1016/B978-0-08-013275-4.50013-1

[2] Johnsen, R. L., and Cullom, R. R., *Altitude Test of Several Afterburner Configurations on a Turbofan Engine with a Hydrogen Heater to Simulate an Elevated Turbine Discharge Temperature*, NASA Technical Publication 1068, Nov. 1977. (NASA-TP-1068; E-9207.)
https://ntrs.nasa.gov/api/citations/19780003163/downloads/19780003163.pdf

[3] Cullom, R. R., and Johnsen, R. L., *Operating Condition and Geometry Effects on Low-Frequency Afterburner Combustion Instability in a Turbofan at Altitude*, NASA TP-1475, NASA Lewis Research Center, Cleveland, OH, June 1979.
https://ntrs.nasa.gov/api/citations/19790016851/downloads/19790016851.pdf

[4] Piccirillo, G., Gregorio, A., Fusaro, R., Ferretto, D., and Viola, N., "Mixed-Flow Turbofan Engine Model for the Conceptual Design of Sustainable Supersonic Airplanes," *Aerospace*, Vol. 11, No. 9, 2024, Art. 740.
https://www.mdpi.com/2226-4310/11/9/740

[5] Chen, M., Deng, L., Zhang, J., and Tang, H., "Review on the Development Status and Performance Design of Adaptive Cycle Engines," *Propulsion and Power Research*, Vol. 14, 2025.
https://doi.org/10.1016/j.jppr.2025.02.006

[6] Zou, W., Yang, H., Wang, B., Liu, W., Lai, J., Jin, L., and Zheng, X., "A Multi-Fidelity Simulation Approach for Adaptive Cycle Engine Performance Analysis by Integrating 3D Model of Variable Geometry Turbine System," *Applied Thermal Engineering*, Vol. 269, 2025, Art. 125987.
https://doi.org/10.1016/j.applthermaleng.2025.125987

[7] Chen, H., Wang, Y., Chen, S., Cai, C., and Zhang, H., "Performance Analysis of the Adaptive Cycle Engine with Cooling Air: From Propulsion Performance, Thermodynamic Efficiency, Exergy and Infrared Characteristics," *Energy*, Vol. 310, 2024, Art. 133215.
https://doi.org/10.1016/j.energy.2024.133215

[8] Aygun, H., Cilgin, M. E., Ekmekci, I., and Turan, O., "Energy and Performance

Optimization of an Adaptive Cycle Engine for Next Generation Combat Aircraft," *Energy*, Vol. 209, 2020, Art. 118261.
https://doi.org/10.1016/j.energy.2020.118261

[9] Kim, S., Kim, K., and Son, C., "Transient System Simulation for an Aircraft Engine Using a Data-Driven Model," *Energy*, Vol. 196, 2020, Art. 117046.
https://doi.org/10.1016/j.energy.2020.117046

[10] Luan, J., Li, S., Cao, Y., and Gu, C., "Application of a Surrogate Model for Condition Monitoring of a Digital Twin Gas Turbine," *Applied Mathematical Modelling*, Vol. 137, 2025, Art. 115683.
https://doi.org/10.1016/j.apm.2024.115683

[11] Wang, F., Ji, H., Chen, H., Chen, X., Wang, M., and Shi, X., "Neural Network Model for Fast Prediction of Infrared Radiation of Turbofan Engine with Axisymmetric Exhaust System in Preliminary Design Phase," *Case Studies in Thermal Engineering*, Vol. 73, 2025, Art. 106641.
https://doi.org/10.1016/j.csite.2025.106641

[12] Zhu, Y., Pan, M., Zhou, W., and Huang, J., "Intelligent Direct Thrust Control for Multivariable Turbofan Engine Based on Reinforcement and Deep Learning Methods," *Aerospace Science and Technology*, Vol. 120, 2022, Art. 107972.
https://doi.org/10.1016/j.ast.2022.107972

[13] Feng, H., Hao, J., Wang, M., Nie, L., and Song, Z., "Thrust Optimization Control for Gas Turbine Engine Afterburner State via Model-Based Deduction Learning," *Energy*, Vol. 337, 2025, Art. 138621.
https://doi.org/10.1016/j.energy.2025.138621

[14] Pratt & Whitney, "F100 Engine," RTX, Available at: https://www.rtx.com/en/prattwhitney/products/military-engines/f100, Accessed January 5, 2026.

[15] Mattingly, J. D., Heiser, W. H., and Pratt, D. T., *Elements of Propulsion: Gas Turbines and Rockets*, 2nd ed., AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, VA, 2016.

[16] Soares, C., "Accessory Systems," in Gas Turbines: A Handbook of Air, Land and Sea Applications, 2nd ed., Butterworth-Heinemann (Elsevier), Oxford, 2015, pp. 413–484, doi:10.1016/B978-0-12-410461-7.00008-0.

[17] Forrester, A. I. J., Sóbester, A., and Keane, A. J., *Engineering Design via Surrogate Modelling: A Practical Guide*, John Wiley & Sons, Chichester, UK, 2008.
https://doi.org/10.1002/9780470770801

[18] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, Cambridge, MA, 2016.
https://www.deeplearningbook.org

[19] Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V., "Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems," *ACM Com-*

puting Surveys, Vol. 55, No. 4, 2022, Article 66.
https://doi.org/10.1145/3514228

[20] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics*, Vol. 378, 2019, pp. 686–707.
https://doi.org/10.1016/j.jcp.2018.10.045

[21] McKay, M. D., Beckman, R. J., and Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239–245.
https://doi.org/10.1080/00401706.1979.10489755

[22] Sobol', I. M., "On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, Vol. 7, No. 4, 1967.
https://www.mathnet.ru/eng/zvmmf7334

[23] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E., "Adaptive Mixtures of Local Experts," *Neural Computation*, Vol. 3, No. 1, 1991, pp. 79–87.
https://doi.org/10.1162/neco.1991.3.1.79
Available PDF: https://www.cs.toronto.edu/~fritz/absps/jjnh91.pdf

[24] He, K., Zhang, X., Ren, S., and Sun, J., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *arXiv preprint arXiv:1502.01852*, 2015.
https://arxiv.org/abs/1502.01852

[25] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.
https://arxiv.org/abs/1412.6980

Appendices

Appendix A: Code Availability

The complete MATLAB codebase developed for this project is available in the project GitHub repository:

https://github.com/Aylock720/Masters-Project

The repository includes the full set of scripts used for parametric cycle analysis, dataset generation, feasibility filtering, neural-network training, prediction, diagnostic checking, parametric sweep validation, extrapolation studies, and runtime benchmarking. Selected scripts are reproduced in the following appendices to document the main computational workflow without adding the full repository contents to the report body. To avoid adding excessive length to the printed report, the source code is provided through the repository rather than being reproduced directly in the appendix. The repository serves as the archive for the general computational workflow used in the project to generate datasets, train models, produce the validation figures, and perform the benchmark studies.