MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Modular Deep Learning Framework for Vapor-Compression HVAC Systems: Scalable, Efficient, and Physically Consistent Modeling

Vaziri, Ali; Qiao, Hongtao; Laughman, Christopher R.; Fang, Huazhen

TR2026-058 May 28, 2026

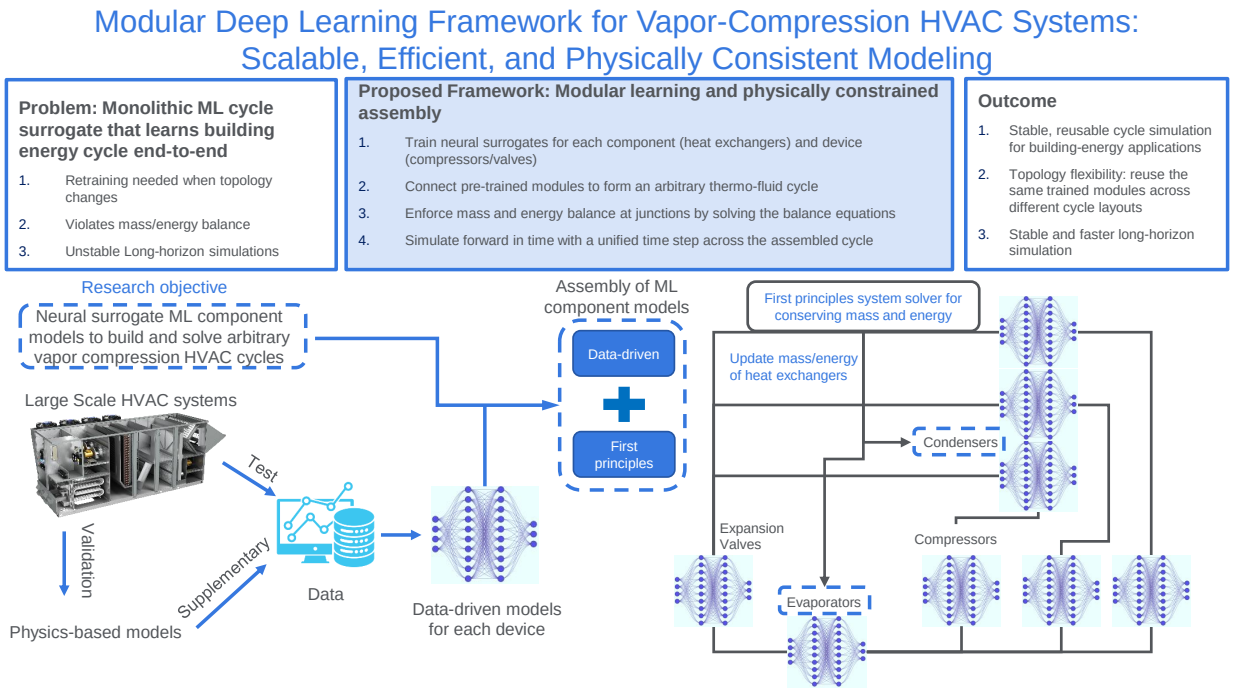
Abstract

Machine learning (ML) is promising for heating, ventilation, and air conditioning (HVAC) modeling because it can provide fast surrogate models that capture complex nonlinear behavior. Most existing ML-based vapor-compression system (VCS) models, however, adopt a monolithic formulation that learns cycle-level behavior directly. Such models are tied to a fixed system topology, require retraining when the refrigerant circuit changes, and may violate mass and energy conservation, leading to drift and instability in long-horizon simulations. We propose a modular, deep learning framework in which individual VCS components are learned independently during their respective training phases and then assembled into a cycle-level simulator. The resulting deep learning framework is modular, allowing various VCS cycle configurations to be constructed from pretrained component models. Within this framework, dynamic components are represented in continuous time using neural ordinary differential equations. To ensure physical consistency, we enforce mass and energy conservation at component interconnections as exact algebraic constraints during system assembly, rather than as penalty terms during training. This design en- sures that the assembled cycle satisfies fundamental conservation laws and results in a numerically stable cycle-level simulation over long time horizons, even when individual component models are imperfect. We evaluate the framework on two air-source heat pump configurations with different topologies (single- and dual- compressor) using a high-fidelity Modelica reference. Across thermodynamic and air-side variables, the assembled cycle achieves a maximum mean absolute percentage error of 2.15% while satisfying mass and energy conservation by construction. The proposed simulator runs 8.7× faster (single-compressor) and 5.54× faster (dual-compressor) than the Modelica baseline, enabling stable long-horizon rollout suitable for control-oriented simulation and topology variation studies.

Applied Thermal Engineering 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Graphical Abstract



Modular Deep Learning Framework for Vapor-Compression HVAC Systems: Scalable, Efficient, and Physically Consistent Modeling

Ali Vaziri[a], Hongtao Qiao[b,*], Christopher R. Laughman[b], Huazhen Fang[a]

[a]*Department of Mechanical Engineering, Michigan State University, East Lansing, MI, USA*
[b]*Mitsubishi Electric Research Laboratories, Cambridge, MA, USA*

Abstract

Machine learning (ML) is promising for heating, ventilation, and air conditioning (HVAC) modeling because it can provide fast surrogate models that capture complex nonlinear behavior. Most existing ML-based vapor-compression system (VCS) models, however, adopt a monolithic formulation that learns cycle-level behavior directly. Such models are tied to a fixed system topology, require retraining when the refrigerant circuit changes, and may violate mass and energy conservation, leading to drift and instability in long-horizon simulations.

We propose a modular, deep learning framework in which individual VCS components are learned independently during their respective training phases and then assembled into a cycle-level simulator. The resulting deep learning framework is modular, allowing various VCS cycle configurations to be constructed from pretrained component models. Within this framework, dynamic components are represented in continuous time using neural ordinary differential equations. To ensure physical consistency, we enforce mass and energy conservation at component interconnections as exact algebraic constraints during system assembly, rather than as penalty terms during training. This design ensures that the assembled cycle satisfies fundamental conservation laws and results in a numerically stable cycle-level simulation over long time horizons, even when individual component models are imperfect.

We evaluate the framework on two air-source heat pump configurations with different topologies (single- and dual-compressor) using a high-fidelity Modelica reference. Across thermodynamic and air-side variables, the assembled cycle achieves a maximum mean absolute percentage error of 2.15% while satisfying mass and energy conservation by construction. The proposed simulator runs 8.7× faster (single-compressor) and 5.54× faster (dual-compressor) than the Modelica baseline, enabling stable long-horizon rollout suitable for control-oriented simulation and topology variation studies.

Keywords: Modular modeling, vapor-compression cycle, heat pump, neural ODE, physics constraints, long-horizon simulation, HVAC

Nomenclature

Acronyms

ASHP	Air-source heat pump
DAE	Differential–algebraic equation
FNN	Feed-forward neural network
GRU	Gated recurrent unit
NODE	Neural ordinary differential equation

*Corresponding author.
Email address: qiao@merl.com (Hongtao Qiao)

ODE	Ordinary differential equation
PINN	Physics-informed neural network
VCS	Vapor compression system

Greek symbols

η	Generic efficiency
η_v	Volumetric efficiency (compressor)
η_{is}	Isentropic efficiency (compressor)

Roman symbols

Δt	Time step [s]
δ	Local error tolerance
$\dot{m}$	Mass flow rate [$\mathrm{kg\,s^{-1}}$]
$\dot{Q}$	Heat transfer rate [W]
$\dot{Q}_{\mathrm{loss}}$	Compressor heat loss [W]
ω	Compressor speed [Hz]
ϕ	Valve opening (0–1)
π	Vector of junction pressures [Pa]
ρ	Density [$\mathrm{kg\,m^{-3}}$]
A	Area [$\mathrm{m^2}$]
E	Energy [J]
E_{sys}	System internal energy [J]
h	Specific enthalpy [$\mathrm{J\,kg^{-1}}$]
K, α	Pressure-drop coeff./exponent
M	Mass (charge) [kg]
p	Pressure [Pa]
P_{tot}	Total system power [W]
P_W	Input power [W]
R	Residual of algebraic constraints
T	Temperature [K]
t	Time [s]

Subscripts

a	Air side
amb	Ambient

bk	Back (downstream) condition
cond	Condenser
dis	Discharge
e	Near exit (edge) state
evap	Evaporator
i	Near inlet (edge) state / component index
in	Inlet
is	Isentropic
lat	Latent
loss	Loss term
out	Outlet
r	Refrigerant
suc	Suction
tot	Total

1. Introduction

A substantial proportion of total building energy demand is attributable to heating, ventilation, and air conditioning (HVAC) systems, with reported estimates typically around 40% of total consumption. [1]. Engineers therefore rely on accurate HVAC models to quantify and reduce energy use, optimize component sizing and refrigerant circuitry, and design supervisory and embedded control strategies that maintain comfort and equipment safety under transients [2, 3, 4]. Vapor-compression systems (VCS) dominate HVAC deployments because they deliver efficient heat pumping across a wide operating envelope. Yet VCS dynamics exhibit strong nonlinearities and multi-scale behavior driven by refrigerant thermodynamics, heat exchanger inertia, and flow/pressure interactions. These effects make high-fidelity modeling difficult and often too slow for real-time design sweeps, optimization, and control in building HVAC applications.

Machine learning (ML) offers a practical path to fast surrogates. The community has successfully learned steady-state performance maps for compressors and chillers and has advanced transient prediction for building loads and equipment cycling with recurrent and probabilistic sequence models [5, 6, 7, 8, 9, 10, 11, 12, 13]. However, most current ML-based cycle simulators adopt a *monolithic* strategy: they train a single network to predict cycle-level evolution or outputs directly from aggregated inputs. This monolithic formulation couples the learned surrogate to one specific topology and operating configuration, forces retraining or extensive recalibration when engineers modify the refrigerant circuit (e.g., add a compressor, split a condenser, change a valve strategy), and obscures failure modes because the model entangles component behaviors and interconnection physics. Monolithic models also typically impose physics through soft penalties, so the model can violate mass and energy balances when it encounters conditions that differ from the training data distribution. These violations matter operationally: small balance errors accumulate across time and across junctions, drift pressures and enthalpies, and destabilize long-horizon cycle simulation and closed-loop control [14, 15].

This work targets scalability and reuse across cycle topologies. We replace the monolithic surrogate with a *modular* scheme that learns component models independently and then assembles them to form a cycle simulator. Modularization mirrors HVAC engineering practice: designers change only a subset of components when they alter a topology, and a reusable library of component surrogates should transfer across architectures with minimal additional

identification. A modular strategy also supports systematic validation and targeted retraining because each component model exposes its own errors and operating envelope.

A modular cycle simulator must also resolve a fundamental time-modeling issue. Many data-driven component models use discrete-time models (e.g., RNNs or fixed-step one-step transition maps), and experiments often have different components at different rates due to sensor constraints and test design. Discrete-time component models therefore inherit *time asynchrony*: each component implicitly learns dynamics tied to its own sampling period, Δt, and the assembled system lacks a single global Δt. System-level operation, however, requires components to exchange port quantities at the same physical time so that junction balances hold. Interconnecting heterogeneous discrete-time models forces ad hoc interpolation/extrapolation, injects numerical inconsistency into nodal constraints, and degrades stability during long-horizon simulations. Using a single global Δt for all components avoids interpolation, but it either oversamples slow components or undersamples fast ones, which wastes computation or reduces accuracy and stability.

We address these limitations by constructing the framework in continuous time. We represent dynamic components with neural ordinary differential equation (NODE) vector fields and allow the deployed integrator to choose its own accepted time steps. This continuous-time formulation decouples training data timestamps from cycle simulation time stepping during the test: components can learn from irregular and component-specific sampling without defining a shared grid, and the assembled cycle advances on a single adaptive time grid chosen by the solver. The solver provides the unified time instances at which all components exchange their shared variables. This enables consistent system-level coupling and supports multi-rate dynamics without sacrificing numerical accuracy [16, 17, 18].

Modularity alone does not guarantee stable assembly when learned component models remain imperfect. In practice, component surrogates cannot achieve perfect accuracy because data remain finite and operating envelopes remain broad. If engineers connect imperfect component models naively, small prediction errors in mass flow rate, enthalpy, or pressure propagate through the network, prevent junction residuals from closing, and can cause the simulated cycle to "blow up" over time. We therefore assemble components through *hard* conservation constraints. Specifically, we enforce mass and energy conservation at steady regimes and at junction interconnections by solving the algebraic coupling constraints that define nodal pressures and balance conditions. This strategy makes the assembly robust to component-level approximation error: the cycle remains physically consistent because the interconnection explicitly corrects incompatible port predictions by selecting junction variables that satisfy conservation. The simulator therefore preserves the physics that must hold for a connected thermo-fluid network, which enables stable long-horizon simulation and reliable reuse of components across topologies.

1.1. Contributions

This paper makes the following contributions:

1. **Hard-constrained, mass- and energy-conserving modular deep-learning framework.** We learn component surrogates and assemble them at the system level by solving the interconnection constraints as hard equations. This assembly enforces mass and energy conservation at junctions and steady regimes, which stabilizes cycle simulation under finite-data component errors and enables direct reuse of trained component models across alternative refrigerant circuit topologies.
2. **Continuous-time component modeling for scalable cycle assembly.** We model dynamic components as NODE vector fields, which eliminates the need for a single fixed global Δt and supports time-asynchronous training data across components. An adaptive ODE solver provides a unified accepted time grid for components, which enables consistent constraint satisfaction and efficient multi-rate simulation.
3. **Cycle-level validation on multiple topologies with fast, accurate, and stable simulation.** We validate the proposed framework on two distinct air-source heat pump configurations (single-compressor and dual-compressor). The results demonstrate substantial simulation speedups relative to high-fidelity Modelica baselines while maintaining low prediction error. Our results also show that modular reuse reduces retraining effort when the topology changes, enabling control-oriented simulation and topology studies for building HVAC applications.

1.2. Literature Review

Deep learning dynamical modeling of VCS sits at the intersection of three active lines of research: (i) black–box and hybrid neural–network models for HVAC and VCS equipment, (ii) discrete–time deep learning architectures for building and equipment dynamics, and (iii) continuous–time neural differential–equation models. In what follows, we briefly review each theme, emphasizing their treatment of physical constraints, modularity, and multi–rate behavior, and highlight the gap that motivates the present work.

1.2.1. Neural–Network Modeling of Vapor–Compression and HVAC systems

Early applications of artificial neural networks (ANNs) to HVAC and VCS focused on replacing steady–state performance maps of chillers, compressors, and heat pumps with data–driven surrogates that map boundary conditions to capacity, COP, and power [5, 6, 7, 8, 19]. These black–box models offer fast evaluations and have been embedded in supervisory control and optimization frameworks such as predictive controllers and learning–based controllers [4, 20, 9].

However, pure ANN models require extensive, carefully designed datasets to interpolate reliably across seasons, load conditions, and equipment operating envelopes. Extrapolation outside the training range often leads to nonphysical predictions (e.g., inconsistent refrigerant mass flow or energy outputs), because conservation laws and thermodynamic relationships are not enforced by construction [14, 15]. This lack of interpretability and physics fidelity has motivated "grey–box" and physics–guided approaches that mix mechanistic structure with learned corrections or regularization terms [9, 21, 22, 23, 24]. In most cases, physical constraints (e.g., energy balance, bounds on states) are incorporated as soft penalties in the loss function rather than as hard equalities, so conservation is only approximately satisfied and can drift over long horizons or in closed–loop.

Another limitation is that many ANN models are built at the system level for a single VCS configuration. When the refrigerant circuit is modified (e.g., adding a subcooler, parallel condenser, or second compressor), substantial retraining is typically required, and model reuse at the component level is limited [25, 26, 27, 28]. This motivates modular, component–oriented data–driven formulations in which heat exchangers, compressors, and valves are modeled separately and later assembled under physically consistent system constraints.

1.2.2. Discrete–Time Deep Learning Models for HVAC Dynamical System

To address transient behavior, discrete–time recurrent neural networks (RNNs) and their variants (LSTM, GRU, encoder–decoder / seq2seq) have been widely adopted for building loads, zone temperatures, and equipment cycling [9, 10, 11, 12, 13, 29]. These models capture long–range temporal dependencies and can deliver accurate short–term forecasts for control and energy management.

Despite their success, discrete–time architectures present several structural drawbacks for VCS modeling. First, they operate on fixed time steps, which assumes uniformly sampled data and prevents the use of adaptive time–stepping that is standard in stiff thermo–fluid simulations. Real HVAC data are often irregular due to asynchronous logging and sensor outages; handling this typically requires ad–hoc interpolation, which may introduce artifacts. Second, the learned transition maps approximate finite–difference updates rather than the underlying continuous–time dynamics. When multiple components with different intrinsic time scales (e.g., heat exchangers vs. valves) are interconnected, enforcing system–level mass and energy balances requires synchronizing all components to a single discrete grid or interpolating between grids, which can degrade numerical consistency and conservation [25, 30, 31, 32, 33, 34, 35]. Third, physics constraints are usually added as soft penalties during training; discrete–time conservation may hold approximately at a chosen sampling period but is not guaranteed under time–rescaling or long–horizon rollout.

Recent hybrid GRU–based models for VCS [25] address some of these challenges by combining moving–boundary heat–exchanger models with GRU surrogates for fast components and enforcing discrete–time conservation laws at each simulation step. While this improves physical consistency and modularity compared to purely black–box approaches, the formulation remains tied to a global time step and does not leverage continuous–time integration or hard system–level constraints. In particular, the hybrid GRU-based framework of [25] addresses several limitations of purely black-box models by combining heat-exchanger models with GRU surrogates for fast components and enforcing discrete-time conservation laws at each simulation step. While effective for their targeted system, such discrete-time formulations introduce a fundamental difficulty when one wishes to assemble multiple learned components into a closed refrigerant cycle: the algebraic mass- and energy-balance constraints must be enforced at a single

global sampling rate. This global step may not coincide with the natural time scales of individual components, forcing either unnecessarily small steps for slowly varying elements or accepting larger errors at junctions and cycle balances.

Our work is complementary to and distinct from this line of research in three key aspects:

- *Time representation.* Hybrid-GRU models operate on a fixed discrete-time grid, whereas our framework learns continuous-time dynamics via NODEs and integrates them with an adaptive ODE solver. This allows us to naturally handle irregular sampling and to adjust the step size to local dynamics, avoiding the need to impose a single global time step on all components when closing the refrigerant cycle.
- *Constraint enforcement.* Prior hybrid models usually enforce conservation through discrete-time updates or soft penalties, while our approach solves cycle-level algebraic equations, therefore guaranteeing mass and energy conservation at all times during the steady state regime.
- *Modularity and topology transfer.* Existing GRU-based formulations are typically engineered for a specific circuit configuration. By contrast, our component-based design explicitly targets reuse: once a library of trained component models is available, new cycle topologies can be assembled by remapping components and lightly retuning only those whose local operating conditions change significantly.

1.2.3. Continuous–Time Neural Differential Equation Models

NODEs and related continuous–time models offer an appealing alternative for dynamical systems with multi–scale behavior and irregular sampling [36, 37, 38, 39, 40, 41, 42]. In NODEs, the state derivative is parameterized by a neural network and integrated using an ODE solver, which naturally accommodates variable step sizes and nonuniform observation times. Extensions such as latent ODEs, ODE–RNNs, and GRU–ODE–Bayes [43, 44, 45, 46, 47] have been proposed for irregularly sampled time series, while Neural SDEs and Bayesian NODEs incorporate stochasticity and uncertainty quantification [48, 49, 50, 51, 52, 53].

For HVAC and building energy applications, continuous–time learning is particularly relevant because coupled building–equipment models exhibit time scales spanning many orders of magnitude and can be numerically stiff [15, 54]. Adaptive integrators can take small steps during fast transients (compressor cycling, valve changes) and large steps near steady operation, improving both accuracy and computational efficiency. Recent work has begun to explore NODE–type models for building and HVAC dynamics [55, 56, 57], but existing studies generally (i) treat the system as a single monolithic NODE, which limits modularity and reuse, and/or (ii) incorporate physics via soft penalties or partial structures rather than enforcing system–level conservation with algebraic constraints and junction solves.

In summary, the literature suggests that: (i) black–box and hybrid neural models can approximate VCS behavior but often lack strict conservation and are tied to specific topologies; (ii) discrete–time RNN frameworks face fundamental limitations in handling multi–rate components and enforcing conservation under time–rescaling; and (iii) continuous–time NODEs provide the right mathematical substrate for stiff, irregular, multi–scale HVAC dynamics, yet have not been combined with a physics–guaranteed, modular assembly strategy for vapor–compression cycles.

The present work addresses this gap by (a) learning continuous–time component models (NODE heat exchangers and FNN mass–flow devices), (b) assembling them through system–level mass and energy balances solved as hard algebraic constraints. This combination yields a physically consistent, modular, and computationally efficient framework for VCS dynamical modeling that, to the best of our knowledge, is not available in the existing related literature.

2. Data-Driven Modeling of VCS Components

2.1. Heat Exchanger Modeling

A NODE is written as

$$\frac{dx_t}{dt} = f_\theta\left(t, x_t, u_t\right), \tag{1}$$

where f_θ is a neural network with parameters θ, $x(t) \in \mathbb{R}^{n_x}$ is the state, and $u \in \mathbb{R}^{n_u}$ denotes the input. To find the state at an arbitrary time t_1, we can integrate f_θ to get

$$x_{t_1} = x_{t_0} + \int_{t_0}^{t_1} f_\theta(x_t, u_t)dt = \text{ODESolve}(f_\theta, x, u, t_0, t_1). \tag{2}$$

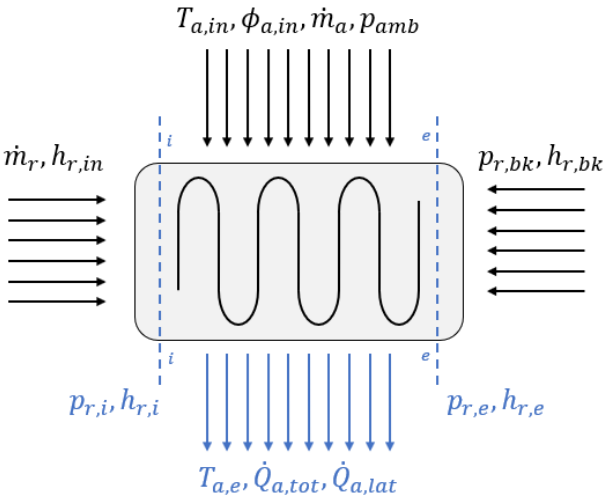


Figure 1: Typical input-output flow of heat exchanger models commonly used in the literature without considering mass and energy.

In NODE-based representations of heat exchangers, the first step is to define the system inputs and outputs in a way that reflects the underlying thermofluid physics. As depicted in Fig. 1, the refrigerant-side inputs consist of the inlet mass flow rate ($\dot{m}_{r,\text{in}}$) and specific enthalpy ($h_{r,\text{in}}$), together with the outlet back pressure ($p_{r,\text{bk}}$). When flow reversal occurs, the specific enthalpy upstream of the outlet ($h_{r,\text{bk}}$) is also included. On the air side, the model receives the inlet air temperature ($T_{a,\text{in}}$), relative humidity ($\phi_{a,\text{in}}$), air mass flow rate ($\dot{m}_{a,\text{in}}$), and the surrounding pressure (p_{amb}). The outputs of the NODE model are defined to summarize the dominant dynamic effects of the heat exchanger. These outputs include refrigerant state variables distributed along the inlet and outlet boundaries of the exchanger, along with the air-side outlet temperature and the effective thermal capacity of the air stream.

We define the following inputs and outputs for the heat exchanger NODE models:

$$x = \begin{bmatrix} p_{r,i} & h_{r,i} & p_{r,e} & h_{r,e} & T_{a,e} & \dot{Q}_{a,\text{tot}} & \dot{Q}_{a,\text{lat}} \end{bmatrix}^\top, \tag{3}$$

$$u = \begin{bmatrix} T_{a,\text{in}} & \phi_{a,\text{in}} & \dot{m}_a & p_{\text{amb}} & \dot{m}_{r,\text{in}} & h_{r,\text{in}} & h_{r,\text{bk}} & p_{r,\text{bk}} & M_r & E_{\text{hx}} \end{bmatrix}^\top. \tag{4}$$

This formulation can be used more generally to construct data-driven heat exchanger models that enforce mass and energy conservation, regardless of the particular prediction framework employed [25]. The key aspect of this approach is treating mass and energy as inputs to the NODEs, ensuring that these quantities are consistently updated at the system level. By doing so, this strategy enforces physical conservation laws while also preventing the accumulation of numerical errors that could arise if mass and energy were instead treated as outputs. This not only enhances the stability and reliability of the model but also ensures its applicability to real-world dynamic system simulations.

For n_{hx} heat exchangers and n_{mf} mass-flow components, we can write

$$\begin{cases} \text{Const} = \displaystyle\sum_{n=1}^{n_{\text{hx}}} M_n, \\ \dfrac{dE_{\text{sys}}}{dt} = \displaystyle\sum_{n=1}^{n_{\text{hx}}} \dot{Q}_n + \sum_{n=1}^{n_{\text{mf}}} \left(P_n - \dot{Q}_{n,\text{loss}} \right). \end{cases} \tag{5}$$

Here, M_n and $\dot{Q}_n$ denote the refrigerant mass and the air-side thermal capacity associated with the nth heat exchanger, respectively, where positive values correspond to inflow and negative values to outflow. The total internal energy

of the ASHP system is represented by E_{sys}. For the nth mass-flow device, P_n denotes the supplied mechanical or electrical power, while $\dot{Q}_{n,\text{loss}}$ accounts for the corresponding heat losses.

Since thermal capacities, power inputs, and heat losses are produced by separate component-level predictors, consistency at steady-state operation is not inherently enforced and can be disrupted by model errors. To mitigate this issue, the refrigerant mass and the internal energy associated with each heat exchanger are introduced as explicit model inputs and are updated in time using a forward-difference discretization.

Refrigerant mass and the internal energy of each heat exchanger are updated using a forward difference scheme

$$\begin{cases} M_{n,t+\Delta t} = M_{n,t} + \left(\dot{m}_{n,\text{in}} - \dot{m}_{n,\text{out}}\right)\Delta t, \quad n = 1, 2, \ldots, n_{\text{hx}}, \\ E_{n,t+\Delta t} = E_{n,t} + \left(\dot{m}_{n,\text{in}} h_{n,\text{in}} - \dot{m}_{n,\text{out}} h_{n,\text{out}} + \dot{Q}_n\right)\Delta t, \quad n = 1, 2, \ldots, n_{\text{hx}}. \end{cases} \tag{6}$$

In this expression, $h_{n,\text{in}}$ and $h_{n,\text{out}}$ denote the specific enthalpy of the refrigerant at the inlet and outlet of the nth heat exchanger, while $\dot{m}_{n,\text{in}}$ and $\dot{m}_{n,\text{out}}$ represent the corresponding mass flow rates. For air–to–refrigerant heat exchangers, the storage of mass and energy on the air side is assumed to be negligible. Consequently, the internal energy of each heat exchanger is attributed solely to the refrigerant inventory and the thermal storage of the metal walls.

In our work, mass and energy storage are assumed to be negligible for mass-flow devices. By enforcing energy balance, it becomes evident that the change in refrigerant enthalpy flow rate across these devices directly corresponds to the net energy inflow. This relationship can be expressed as

$$\begin{cases} M = 0, \\ \dot{m}_{\text{out}} h_{\text{out}} - \dot{m}_{\text{in}} h_{\text{in}} = P_W - \dot{Q}_{\text{loss}}. \end{cases} \tag{7}$$

When summing the mass and energy balances across all components, it becomes evident that the overall mass and energy conservation for the entire system is inherently maintained. This occurs naturally as inter-component mass and energy exchanges cancel out, ensuring that system-level balances hold without requiring additional constraints. As a result, the overall mass of the refrigerant is conserved at each time step, and variations in the system energy arise from the heat exchangers.

$$\begin{cases} \Delta E_{\text{tot}} = \displaystyle\sum_{n=1}^{n_{\text{hx}}} \left(\dot{m}_{n,\text{in}} h_{n,\text{in}} - \dot{m}_{n,\text{out}} h_{n,\text{out}} + \dot{Q}_n\right)\Delta t, \\ 0 = \displaystyle\sum_{n=1}^{n_{\text{hx}}} M_{n,t+\Delta t} - \sum_{n=1}^{n_{\text{hx}}} M_{n,t}. \end{cases} \tag{8}$$

Under steady-state conditions, the models evolve such that $E_{n,t+1} = E_{n,t}$, and $M_{n,t+1} = M_{n,t}$, indicating that internal energy and refrigerant mass remain constant over time. By enforcing mass and energy conservation through the proposed approach, the model ensures physical consistency and reliability, enabling accurate representation of real-world system behavior.

2.1.1. Structure Design: Seq2seq GRU-ODE

For the choice of forecasting model, we leverage the neural ODE-GRU method developed in [44] because they are particularly adept at learning long-term dependencies in sequential data, making them highly suitable for identifying nonlinear dynamical systems. We follow the derivation in [44] as follows: Let r_t, z_t, and g_t represent the reset gate, update gate, and update vector of the GRU, respectively. With $y = \begin{bmatrix} x^\top & u^\top \end{bmatrix}^\top$, these gates are defined as

$$\begin{aligned} r_t &= \sigma\left(W_r y_t + U_r h_{t-1} + b_r\right), \\ z_t &= \sigma\left(W_z y_t + U_z h_{t-1} + b_z\right), \\ g_t &= \tanh\left(W_h y_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right), \end{aligned}$$

where $\odot$ denotes the element-wise product. The hidden state h_t is updated as

$$\frac{dh_t}{dt} = (1 - z_t) \odot (g_t - h_t). \tag{9}$$

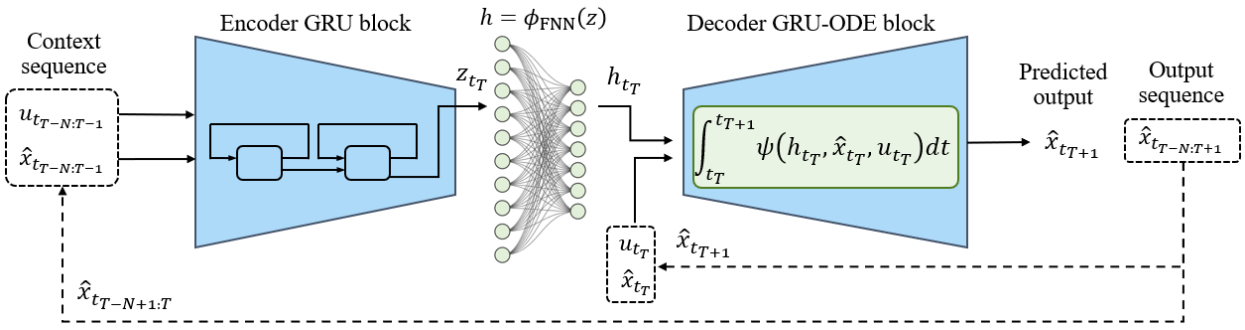


Figure 2: Seq2seq learning architecture for NODE-based heat exchanger modeling. The encoder GRU block processes input and state sequences to produce a latent representation z, which is transformed into h using ϕ_{FNN}. The decoder GRU-ODE block predicts the system dynamics using the ODE in (10).

Further, to increase the long-term prediction capability of the neural networks we utilize the encoder-decoder seq2seq architecture. We collect sequences of input variables and their corresponding outputs over a look-back window of length N. This window of variables contain previous states and inputs from time t_{T-N} to time t_{T-1}, augmented as a context sequence $x_{t_{T-N:T-1}}, u_{t_{T-N:T-1}}$. This input is passed through the encoder GRU, and then a decoder GRU–ODE generates the output.

The structure of the seq2seq NODE model is illustrated in Fig. 2. The inputs to the encoder, denoted as $x_{t_{T-N:T-1}}, u_{t_{T-N:T-1}}$, correspond to the states of the heat exchanger model representing the internal variables characterizing the dynamic behavior of the system and the input variables such as boundary conditions (e.g., mass flow rate, pressure, and enthalpy). These sequences are fed into the encoder GRU block. The GRU layers within the encoder are designed to learn long-term dependencies by iteratively updating a hidden state, z, which encapsulates the temporal information from the input sequence. Once the input sequence has been processed, the final hidden state of the encoder, z, is transformed into a latent representation $h \in \mathbb{R}^{n_x}$ through a feedforward neural network, denoted as $h_{t_T} = \phi_{\text{FNN}}(z_t)$. This latent state h is crucial for mapping the encoder's learned representation to the decoder for subsequent prediction.

The decoder block integrates the dynamics of the system using a GRU-ODE function, ψ, which operates over the latent state h_t, the current states x_t, and the inputs u_t. The latent dynamics are governed by

$$\frac{dh_t}{dt} = \psi(h_t, x_t, u_t), \tag{10}$$

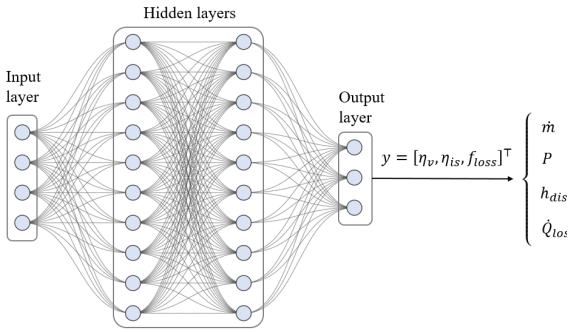
and the output sequence is generated by integrating this ODE over the desired time horizon

$$h_{t_{T+1}} = h_{t_T} + \int_{t_T}^{t_{T+1}} \psi(h_{t_T}, x_{t_T}, u_{t_T})\, dt. \tag{11}$$

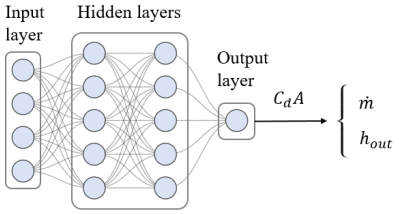
The decoder uses this integrated state $h_{t_{T+1}}$ to predict the output states $\hat{x}_{t_{T+1}}$, which are subsequently compared with the ground truth for training. The ϕ_{FNN} plays a pivotal role in ensuring that the latent state is mapped to the same dimensional space as the heat exchanger states x. By applying this mapping before the adaptive integration scheme in Section 3.2, the computational overhead of solving the ODE at the system level is significantly reduced. This efficiency is achieved without compromising the fidelity of the learned representations. Additionally, we implemented the linear mapping after the GRU-ODE decoder, following the standard practice in the machine learning literature, and observed comparable performance. In further trials, we intentionally moved this mapping before the final decoder to reduce the size of the state vectors being integrated. This adjustment helps minimize computational load when multiple components work together, particularly in scenarios involving numerous components and integrations.

2.2. Compressor and Expansion Valve Modeling

Traditional black-box models for mass-flow devices like compressors and expansion valves, such as ten-coefficient polynomials, often require extensive datasets and exhibit poor extrapolation performance [58, 59]. They have limited



(a) Neural network model of the compressors, followed by conventional standard calculation of physical properties as in (14).



(b) Neural network model of the expansion valves, followed by conventional standard calculation of physical properties as in Eqs. (17)-(18).

Figure 3: Structures employed for the data-driven modeling of (a) the compressor and (b) the expansion valve. Additionally, the neural network predictions are used to calculate the physical properties of the mass flow devices.

physical interpretability, which can result in unphysical responses and unstable system simulations when the operating regime differs from the training conditions.

To address these limitations, we adopt feed-forward neural networks integrating physical knowledge into the modeling process to enhance reliability and generalizability [25]. This approach involves mapping inputs to physically meaningful intermediate variables, such as efficiencies and coefficients, rather than directly predicting outputs like mass flow rates and power consumption.

We use different model classes for heat exchangers and mass-flow devices because the underlying physics play different roles in the cycle dynamics. Heat exchangers exhibit dominant dynamic storage effects through refrigerant inventory and wall thermal mass, so their behavior is naturally represented by continuous-time state evolution and is therefore modeled with NODEs. By contrast, compressors and expansion valves are treated here as quasi-static mass-flow devices with negligible internal mass and energy storage relative to the heat exchangers. For these components, the primary task is to map instantaneous thermodynamic conditions and actuation signals to physically meaningful flow quantities, efficiencies, and enthalpy changes, for which a compact physics-guided FNN is sufficient and computationally more efficient, and there is no need for complicated neural networks (see Fig. 3) .

For compressors, the neural network predicts dimensionless efficiencies that characterize the compression process. The input and output vectors are

$$x = \begin{bmatrix} \omega & p_{\text{suc}} & h_{\text{suc}} & p_{\text{dis}} \end{bmatrix}^\top, \tag{12}$$

$$y = \begin{bmatrix} \eta_v & \eta_{is} & f_{\text{loss}} \end{bmatrix}^\top, \tag{13}$$

In this formulation, we constrain the compressor model outputs to remain physically admissible by enforcing bounds on the predicted efficiencies. Imposing these limits is crucial for maintaining stable and reliable system-level simulations when the compressor model is coupled with other component models. Using the predicted efficiencies, we can evaluate the following properties

$$\begin{cases} \dot{m} = \omega \eta_v \rho_{\text{suc}} V_s, & \text{Mass flow rate} \\ P_W = \dfrac{\dot{m}(h_{\text{dis,is}} - h_{\text{suc}})}{\eta_{is}}, & \text{Power consumption} \\ h_{\text{dis}} = h_{\text{suc}} + \dfrac{P_W - \dot{Q}_{\text{loss}}}{\dot{m}}, & \text{Actual discharge enthalpy} \\ \dot{Q}_{\text{loss}} = f_{\text{loss}} P_W. & \text{Heat loss} \end{cases} \tag{14}$$

This structure is advantageous as it allows for the calculation of the physical properties of the compressor based on dimensionless efficiencies. This implies that the compressor model becomes applicable to different refrigerants

without requiring re-training. Bounding the outputs within physically plausible ranges ensures stability when integrating the compressor model with other system components. Additionally, the discharge enthalpy calculation strictly adheres to energy balance, ensuring integration with the heat exchanger model while maintaining system-wide energy conservation and preventing artificial discrepancies.

For expansion valves, we also integrate physical principles into the neural network model. We define the input and output as

$$x = \begin{bmatrix} p_{\text{in}} & h_{\text{in}} & p_{\text{out}} & \phi \end{bmatrix}^\top, \tag{15}$$

$$y = C_d A = C_d A_{\min} + \phi(A_{\max} - A_{\min}), \tag{16}$$

which allows us to calculate the mass flow rate as

$$\dot{m} = y\sqrt{2\rho_{\text{in}}(p_{\text{in}} - p_{\text{out}})}, \tag{17}$$

where ρ_{in} is the inlet refrigerant density. Assuming an isenthalpic expansion process, the outlet enthalpy equals the inlet enthalpy

$$h_{\text{out}} = h_{\text{in}}. \tag{18}$$

It is important to note that when these neural network-based component models are integrated into system-level simulations, numerical iterations may be necessary at each time step to resolve intermediate variables, such as pressures and enthalpy, ensuring a physically consistent and accurate representation of the system.

3. System Assembly of Data-Driven Component Models

3.1. System Coupling with Mass and Energy Conservation

We assemble the independently trained component models by enforcing mass continuity, momentum closure, and energy consistency at the cycle level. Let

$$X = \text{col}(x_{\text{cond},1}, \ldots, x_{\text{cond},n_c}, x_{\text{evap}}), \qquad \pi \in \mathbb{R}^{n_j}.$$

Denote, respectively, the concatenated heat exchanger states and the vector of unknown junction pressures (and, when needed, mixed enthalpies). NODEs provide the differential part,

$$\frac{dX_t}{dt} = F\left(X_t, U_t, \pi_t\right),$$

while interconnections impose algebraic constraints through residuals

$$0 = R\left(X_t, U_t, \pi_t\right) \in \mathbb{R}^{n_j},$$

which collect (i) mass continuity at junctions and (ii) momentum closure across connecting segments.

For each control volume (or terminal unit of a coil path), we have

$$\Delta p \;=\; K\,\Delta p_0 \left(\frac{\dot{m}}{\dot{m}_0}\right)^{\alpha}, \tag{19}$$

where Δp_0 is the drop at a nominal mass flow $\dot{m}_0$, and K, α are hyperparameters that can be fit to the specific operating conditions and two-phase effects. When a heat exchanger model is coupled to mass–flow devices, outlet flow $\dot{m}_{\text{out}}$ is computed from the local back pressure via (19); The back pressure is iteratively adjusted at each time step to enforce continuity. Equation (19) is employed as a quasi-steady closure for momentum balance in the connecting segments and junctions. Accordingly, transient inertial terms in the momentum equation are neglected, and the pressure-drop relation is assumed to adjust instantaneously relative to the dominant thermal and refrigerant-charge dynamics captured by the heat-exchanger NODEs. This approximation is appropriate for the cycle-level HVAC transients considered in this work, where energy-transfer dynamics are of primary interest and momentum/acoustic transients are much faster. Prior analysis of one-dimensional compressible flow [60] has shown that retaining dynamic momentum introduces acoustic-wave propagation and sonic characteristic speeds that can significantly restrict the allowable integration step size, whereas neglecting these fast momentum effects can greatly improve computational efficiency with limited loss of accuracy for energy-oriented simulations.

3.1.1. Junction mass–balance residuals

Let $\mathcal{J}$ denote the set of refrigerant junctions. For each junction $j \in \mathcal{J}$, continuity gives

$$R_j(X_t, U_t, \pi_t) \;=\; \sum_{i \in \mathrm{in}(j)} \dot{m}_{i\to j}(X_t, U_t, \pi_t) \;-\; \sum_{k \in \mathrm{out}(j)} \dot{m}_{j\to k}(X_t, U_t, \pi_t) \;=\; 0, \tag{20}$$

where the branch flows $\dot{m}$ come from heat exchanger outlets via (19) and from mass–flow devices.

In the single-compressor case, we require that the following manifold junction residuals:

$$\begin{pmatrix} \dot{m}_1(p_a) - \dot{m}_2(p_a) \\ \dot{m}_3(p_b) - \dot{m}_4(p_b) \end{pmatrix} = \mathbf{0}. \tag{21}$$

Fig. 4a illustrates the single-compressor nodal connections of data-driven models. In the dual-compressor ASHP configuration, two junction pressures p_a, p_b are introduced to enforce mass continuity across the network (see Fig. 4b). The residual system

$$\begin{pmatrix} \dot{m}_6(p_a) - \dot{m}_1(p_a) - \dot{m}_2(p_a) \\ \dot{m}_5(p_b) - \dot{m}_3(p_b) - \dot{m}_4(p_b) \end{pmatrix} = \mathbf{0} \tag{22}$$

solves for (p_a, p_b) and enforces mass balances. These equations are decoupled in this particular topology but may be coupled in general. During simulation, systems (20)–(22) are solved by multivariate root finding (e.g., Powell's method [61]).

We can now update the refrigerant mass M_n and internal energy E_n of each heat exchanger NODEs using algebraically consistent port flows/enthalpies via the forward-difference updates

$$\begin{cases} M_{n,t+\Delta t} = M_{n,t} + (\dot{m}_{n,\mathrm{in}} - \dot{m}_{n,\mathrm{out}})\,\Delta t, \quad n = 1, \ldots, n_{\mathrm{hx}}, \\ E_{n,t+\Delta t} = E_{n,t} + (\dot{m}_{n,\mathrm{in}} h_{n,\mathrm{in}} - \dot{m}_{n,\mathrm{out}} h_{n,\mathrm{out}} + \dot{Q}_n)\,\Delta t, \end{cases} \tag{23}$$

where $h_{n,\mathrm{in/out}}$ and $\dot{m}_{n,\mathrm{in/out}}$ are obtained *after* solving $R = 0$ at that step, and air-side storage is neglected so E_n accounts for refrigerant and metal wall only. Mass-flow devices (compressor, valve) are modeled with negligible storage; their energy jump is enforced by

$$M = 0, \qquad \dot{m}_{\mathrm{out}} h_{\mathrm{out}} - \dot{m}_{\mathrm{in}} h_{\mathrm{in}} = P_W - \dot{Q}_{\mathrm{loss}}.$$

Summing component balances shows intercomponent exchanges cancel, yielding system-level conservation:

$$\sum_{n=1}^{n_{\mathrm{hx}}} M_{n,t+\Delta t} = \sum_{n=1}^{n_{\mathrm{hx}}} M_{n,t}, \qquad \Delta E_{\mathrm{tot}} = \sum_{n=1}^{n_{\mathrm{hx}}} (\dot{m}_{n,\mathrm{in}} h_{n,\mathrm{in}} - \dot{m}_{n,\mathrm{out}} h_{n,\mathrm{out}} + \dot{Q}_n)\Delta t.$$

Under steady conditions the model naturally satisfies $M_{n,t+\Delta t} = M_{n,t}$ and $E_{n,t+\Delta t} = E_{n,t}$.

3.2. Determining a Unified Δt in Assembled Cycle Simulations

At each time step we must first choose a step size Δt, then, for that step size, solve the junction residuals $R(X, U, \pi) = 0$ to obtain algebraically consistent junction variables P. Only after this solve is successful do we update every heat exchanger's mass and energy with the same, unified step size by substituting Δt into (23) for all $n = 1, \ldots, n_{\mathrm{hx}}$. Because (23) uses a single Δt across all heat exchanger NODEs, we need a principled way to determine the best step size that admits a successful solution of $R(X, U, \pi) = 0$ and controls the local truncation error of the NODE dynamics.

Adaptive integration scheme tries to take large time steps when the cycle evolves slowly (near quasi-steady operation) and smaller steps when fast transients or stiff dynamics arise. At each candidate time step, we evaluate the NODE models for the heat exchangers. An embedded ODE method provides two approximations of different orders; their difference yields a local error estimate. If this estimate is below a user-specified tolerance, the step is accepted and the time step is slightly increased; otherwise, the step is rejected and the time step is reduced. In this way, the integrator automatically concentrates computational effort where dynamics are fast, while avoiding unnecessary function evaluations in slowly varying regimes.

In this regard, we augment the state derivatives of all the heat exchangers to form

$$\begin{cases} x_{sys} = \begin{bmatrix} x_1 & \dots & x_{n_{hx}} \end{bmatrix}^\top, \\ u_{sys} = \begin{bmatrix} u_1 & \dots & u_{n_{hx}} \end{bmatrix}^\top, \\ \dfrac{dx_{sys}}{dt} = \begin{bmatrix} \left(\frac{dx}{dt}\right)_{\text{cond}} \\ \left(\frac{dx}{dt}\right)_{\text{evap}} \end{bmatrix} = \begin{bmatrix} f_{\text{cond}}(t, x_{\text{cond}}, u_{\text{cond}}) \\ f_{\text{evap}}(t, x_{\text{evap}}, u_{\text{evap}}) \end{bmatrix} = f(t, x_{sys}, u_{sys}), \end{cases} \tag{24}$$

where n_{hx} is the number of heat exchangers and x_{sys} is formed by the concatenation of all the heat exchanger states.

The adaptive integration scheme is implemented using a variable-step Runge-Kutta method [62][Table III Formula 2] which computes intermediate steps $k_1, k_2, \dots, k_6$ to estimate the solution at the next time step. For the sake of clarity and completeness, the governing equations are listed below:

$$\begin{aligned} k_1 &= f\left(t_k, , x_{sys,k}, u_{sys}\right)\Delta t, \\ k_2 &= f\left(t_k + \frac{1}{4}\Delta t, x_{sys,k} + \frac{1}{4}k_1, u_{sys}\right)\Delta t, \\ k_3 &= f\left(t_k + \frac{3}{8}\Delta t, x_{sys,k} + \frac{3}{32}k_1 + \frac{9}{32}k_2, u_{sys}\right)\Delta t, \\ k_4 &= f\left(t_k + \frac{12}{13}\Delta t, x_{sys,k} + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3, u_{sys}\right)\Delta t, \\ k_5 &= f\left(t_k + \Delta t, x_{sys,k} + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4, u_{sys}\right)\Delta t, \\ k_6 &= f\left(t_k + \frac{1}{2}\Delta t, x_{sys,k} - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5, u_{sys}\right)\Delta t. \end{aligned}$$

During the integration the inputs are assumed to be time invariant. Then, we calculate

$$\begin{aligned} \tilde{x}_{sys,k+1} &= x_{sys,k} + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4101}k_4 - \frac{1}{5}k_5, \\ \hat{x}_{sys,k+1} &= x_{sys,k} + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6. \end{aligned}$$

The difference between them gives the error estimate, which is used to adjust the next step size Δt_{k+1} based on error tolerance δ

$$\Delta t_{k+1} = \left(\frac{\delta \Delta t_k}{2\|\tilde{x}_{sys,k+1} - \hat{x}_{sys,k+1}\|_2}\right)^{1/4} \Delta t_k. \tag{25}$$

This step-size adjustment ensures that integration errors remain within the prescribed tolerance while minimizing computational steps. Once an appropriate time step is determined, the mass and energy inputs of the condensers and evaporators are updated using (23). An alternative integration strategy involves separately integrating the derivatives for each component to determine individual time steps Δt_{cond} and Δt_{evap}. The smaller of the two time steps is then selected to maintain accuracy across all components.

At each time step, the adaptive integrator computes the current state update of the heat-exchanger NODEs and uses the difference between the embedded formulas to estimate the local truncation error. This error estimate is then used in (25) to determine the step size for the next integration step. After the state update is obtained, the algebraic system $R(X, U, \pi) = 0$ is solved for the resulting state to determine the junction variables and enforce cycle-level consistency.

This strategy is computationally advantageous, as it employs larger integration time steps during slowly varying dynamics and smaller ones during rapid transients, thereby improving simulation efficiency without sacrificing accuracy. The adaptive integration scheme enables the solver to dynamically adjust step sizes based on local error estimates, maintaining numerical precision while reducing computational overhead.

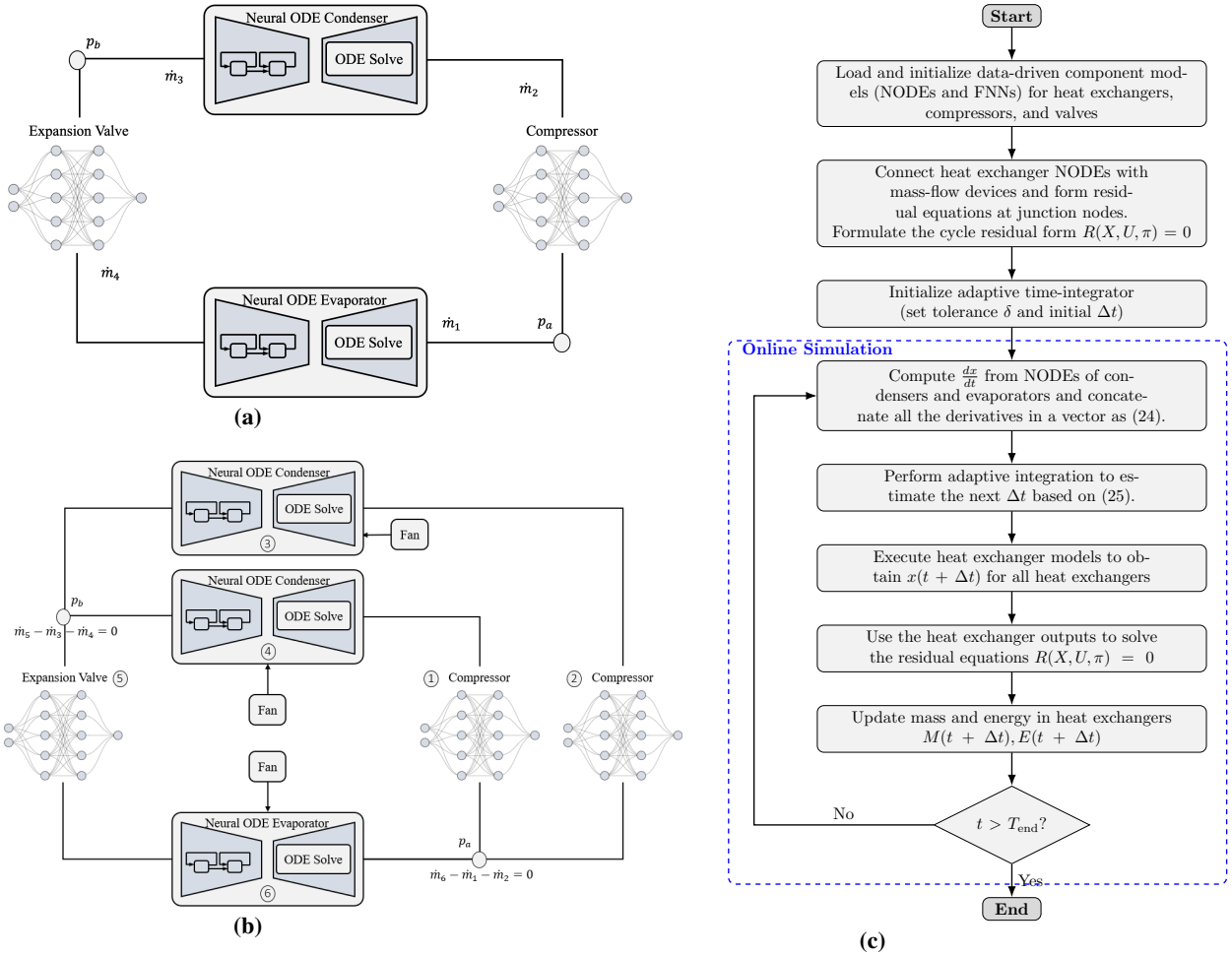


Figure 4: (a) Schematic of the single-compressor and (b) the dual-compressor ASHP system, and (c) flowchart of the assembly of deep learning component models and physically consistent cycle simulation.

3.3. Adaptation Workflow for Arbitrary Cycle Configurations

It is important to emphasize that physical consistency in the proposed framework is enforced at the cycle-assembly level, not guaranteed by component training alone. Each component surrogate is trained independently, but during cycle assembly, the shared junction variables are determined by solving the algebraic coupling equations so that the interconnected cycle satisfies mass continuity and energy consistency at the system level. Consequently, local prediction errors in individual component models do not directly appear as unconstrained balance violations; instead, the assembly step projects the interconnected cycle onto a physically admissible manifold defined by the conservation constraints.

This mechanism still relies on the component surrogates providing sufficiently accurate and stable local predictions over the operating range of interest. If the operating conditions move far outside the training-data distribution, component-level errors may increase substantially and can eventually compromise simulation robustness. To address such cases, we use the following adaptation procedure for arbitrary cycle configurations and shifted operating envelopes.

Given trained component models for a source cycle and a target cycle with modified interconnections (e.g., added subcooler, dual compressor, or re-routed refrigerant paths), we adapt as follows:

1. **Topology mapping:** Define the target graph (components, ports, junctions) and port compatibilities. Map

existing trained components to target positions whenever their local operating conditions remain within the previously trained range.

2. **Interconnection initialization:** Initialize junction pressures, mass flows, and port enthalpies based on the target topology and expected operating conditions. This provides a consistent starting point for the cycle- level algebraic solve.
3. **Targeted retraining:** For each mapped component, compare the target operating envelope against the source training range. If the target inputs remain within the source data range, the pretrained model is reused directly. Otherwise, initialize the target model with the source weights, and fine-tune it on target-topology data for additional epochs. In this way, retraining is localized to the affected modules rather than repeated for the entire cycle.
4. **Cycle assembly and constraint solve:** Assemble the cycle by coupling component surrogates through the algebraic constraints $R(X, U, \pi) = 0$ and solve for junction variables at each time step. This step ensures that mass continuity and energy consistency are enforced globally across the modified configuration.
5. **Simulation:** Start the simulation of the VCS cycle using the flowchart in Fig. 4c.

This workflow allows the same library of NODE and FNN component models to be reused across multiple refrigerant circuit designs, reducing the amount of new data and training effort required when adapting to new equipment configurations.

For more complex cycle topologies, the main additional computational burden lies in the size and coupling structure of the algebraic system, since the number of unknown junction variables and residual equations generally increases with the number of manifolds and interconnections. By contrast, the data requirement does not necessarily grow cycle-wide: new data are mainly required for components whose local operating envelopes shift outside the range represented in the existing component library. Thus, the framework keeps the neural-model adaptation efficient while the dominant additional runtime is associated with the larger root-finding problem for the junction variables.

4. Simulation Setup

We evaluate the framework on two ASHP configurations assembled *from independently trained component models*: (i) a single-compressor cycle (one compressor, one condenser, an expansion valve, and one evaporator in series), and (ii) a dual-compressor cycle with two compressors and two condensers arranged in parallel. In both cases, the trained neural component models (NODE heat exchangers; FNN mass-flow devices) are connected at the system level, and the cycle is closed by enforcing mass continuity and energy consistency.

To benchmark accuracy, we compare against a high-fidelity physics-based reference generated in Dymola [63] following [64]. The refrigerant-side equations (mass, momentum, energy) are discretized on a staggered grid with pressure and enthalpy as states; heat-exchanger metals are lumped with 1D conduction; the air side uses a quasi-steady 3D discretization. Heating transients are induced by step changes in compressor speeds and valve openings; ambient/fan conditions are held fixed. Condenser inlet air is set to 20°C, 60% RH, and 0.4 $m^3\ s^{-1}$; evaporator air is 7°C, 87% RH, and 0.72 $m^3\ s^{-1}$.

We validate both the fidelity of the component reuse under re-assembly and the precision of the adaptive integration scheme in choosing discrete-time steps that preserve cycle-level mass and energy.

4.1. Training Details for Component Models

In this section, we explain the data generation and optimization details to train each component model.

4.1.1. Data Generation for Training

In practice, the detailed Modelica models provide a rich set of internal variables (on the order of a few dozen physical quantities per component) that are difficult or impossible to measure reliably in hardware. For this reason, we base the training primarily on simulation data rather than experimental data. Although simulating the complete cycle can be expensive because of the high-dimensional differential-algebraic equations (DAEs) and the coupling relations between components, running each component with prescribed boundary conditions is inexpensive and well-suited for generating large training data sets.

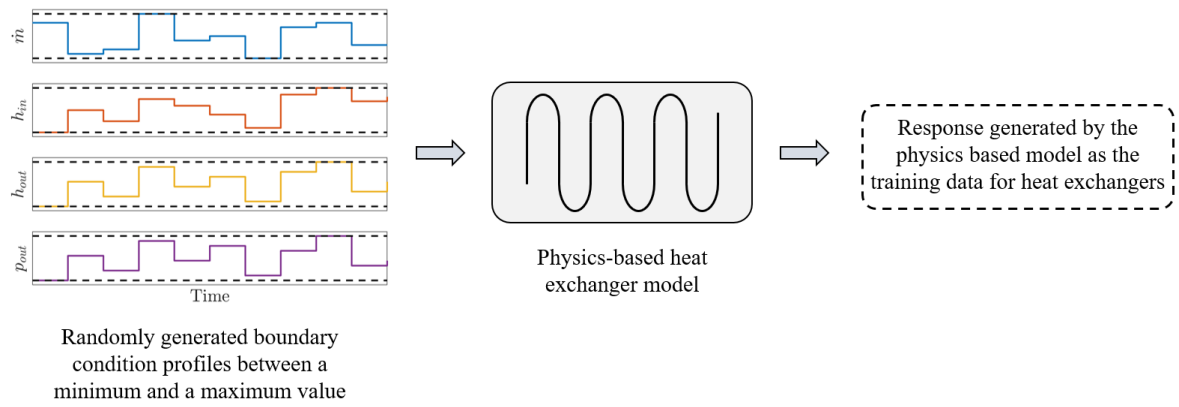


Figure 5: Training data generation for the heat exchanger NODEs.

For each heat exchanger (condenser and evaporator), we drive the physics-based component model with a sequence of boundary conditions and record trajectories of the model predictions. The input profiles are designed to capture both transient and steady-state behavior. To that end, we construct profiles as a sequence of step segments: each boundary condition is held constant for a fixed step length, chosen long enough for the component to approach steady state, and then perturbed again. In our simulations, we carry out a random walk with 500 steps and use a step duration of 30 s, which provides a good compromise between covering a wide range of operating conditions and allowing the model to relax between perturbations (to reach steady state). Figure 5 summarizes this workflow.

To ensure that the component-level trajectories remain consistent with the system-level operating envelope, we define lower and upper bounds for each input variable,

$$\tilde{u}_{\max} = \begin{bmatrix} \dot{m}_{\max} & h^{\text{in}}_{\max} & h^{\text{bk}}_{\max} & p^{\text{bk}}_{\max} \end{bmatrix}^{\top}, \tag{26}$$

$$\tilde{u}_{\min} = \begin{bmatrix} \dot{m}_{\min} & h^{\text{in}}_{\min} & h^{\text{bk}}_{\min} & p^{\text{bk}}_{\min} \end{bmatrix}^{\top}. \tag{27}$$

Starting from a feasible initial input $\tilde{u}_0 \in [\tilde{u}_{\min}, \tilde{u}_{\max}]$, we generate a random-walk sequence while enforcing these bounds. At each time step Δt,

$$\tilde{u}_{t+\Delta t} = \text{clip}(\tilde{u}_t + \text{diag}(a_1, a_2, a_3, a_4)\,\xi_t;\ \tilde{u}_{\min}, \tilde{u}_{\max}), \qquad \xi_t \sim \mathcal{N}(0, I), \tag{28}$$

where ξ_t is a vector of independent standard Gaussian variables and $a_1, \ldots, a_4$ set the step size of each component. Here, $\text{clip}(\cdot; \tilde{u}_{\min}, \tilde{u}_{\max})$ denotes componentwise clamping to the admissible interval: if a component is below its minimum it is set to that minimum, if it is above its maximum it is set to that maximum, and otherwise it is left unchanged. In practice, we choose a_i proportional to $(\tilde{u}_{\max,i} - \tilde{u}_{\min,i})$ so that each update changes the input by only a modest fraction of its full range. The resulting random-walk path is then converted into a piecewise-constant profile by holding each value for a duration of 30 s and clipping any overshoot back into the interval $[\tilde{u}_{\min}, \tilde{u}_{\max}]$. The time-series predictions of refrigerant charge, coil internal energy, and other state variables are sampled at a fine temporal resolution and augmented with the corresponding input profiles to form the training input sequences. The target outputs are the remaining refrigerant-side and air-side variables of interest for each component. Time-series predictions were generated using non-uniform sampling intervals, with the resulting input–output trajectories stored and normalized in the dataset. The sampling time intervals from the training data were also incorporated into the integration process for each component. During system-level testing, no prior information on sampling time is provided: the system must intelligently determine the appropriate integration time intervals.

For the overall dataset, we generate time series data per component from the high-fidelity Modelica models under the random-walk boundary conditions described above. We then perform a trajectory-level split of 80%/10%/10% into training, validation, and test sets, respectively.

4.1.2. Optimization Details

All neural networks are implemented in PyTorch and trained on a standard desktop CPU using the Adam optimizer. Unless otherwise specified, we use an initial learning rate of 10^{-4} and an ℓ_2 weight decay rate of 0.9, with Xavier initialization for all weight matrices. Before training, every dataset feature (inputs and outputs) is linearly scaled to the range $[0, 1]$.

When developing the NODE models for the condenser and evaporator, the choice of input sequence context length is critical, as it balances predictive power against computational cost. Longer sequences typically improve accuracy by exposing the model to more temporal context, but this comes at the price of higher run time and more complicated model initialization. After an iterative trial-and-error study, we found that a context length of two provides a good compromise for this work. Concretely, the NODE receives the input features from the current and the preceding time, and uses these two inputs to predict the outputs at the next time.

The NODE architecture uses two layers with a hidden size of 256 units for encoder and decoder, and the ϕ_{FNN} consists of two linear layers of the same size that map the hidden state back to the physical state variables. For these models, we use an initial learning rate of 1×10^{-4}, a batch size of 16, and train for 100 epochs. Training stability of the heat-exchanger NODE models was supported by several implementation choices. To reduce the risk of vanishing or exploding gradients, we intentionally used a relatively shallow neural-network architecture rather than a very deep one. In addition, all inputs and outputs were normalized to $[0, 1]$ to reduce scale disparities across thermodynamic variables, and Xavier initialization. In the experiments, these choices were sufficient to avoid persistent training instabilities.

For components such as the compressor and expansion valve, we generate training data simply by sampling inputs uniformly from the admissible ranges defined by (27), without performing a random walk. The compressor is modeled by a feedforward neural network with two hidden layers of ten neurons. We use 1000 epochs with a batch size of 128 to train the neural networks. On a held-out test set, this model achieves root-mean-squared errors of approximately 7×10^{-6} kg/s for mass flow rate, 1.6 W for compressor power, and 167 J/kg for discharge enthalpy.

The expansion valve model follows the same training procedure, but a smaller feedforward network with two hidden layers of five neurons per layer is sufficient for predicting the valve mass flow rate. The resulting surrogate attains an RMSE of about 3×10^{-5} kg/s on the test data. Together, these static surrogates and the NODE heat exchanger models form the component-level library used in the system-level simulations. The compressor and valve component models were trained over the actuation ranges $\omega \in [20, 60]$ Hz and $\phi \in [200, 300]$, respectively.

4.2. Test Setup: Cycle Simulations with Trained Component Models

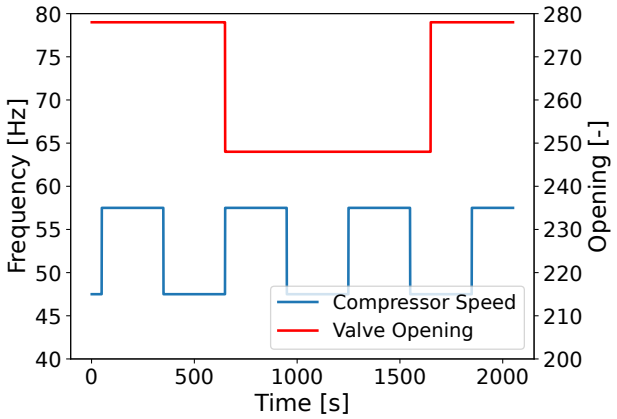
After training each component model separately, we evaluate two assembled system configurations: a single-compressor ASHP cycle and a dual-compressor ASHP cycle (parallel compression/condensation). The compressor-speed and valve-opening actuation profiles used to excite both fast and slow transients—step changes in compressor speed and electronic expansion valve opening—are shown in Fig. 6. These profiles drive the cycles through a sequence of lower- and higher-load operating points so that both steady and transient regimes are exercised during the simulations.

To ensure good predictive performance over a broad operating envelope, rather than only for the specific excitations used during training, we validate the assembled cycles across a range of operating conditions. Fig. 6 illustrates representative actuation programs for the single- and dual-compressor ASHP case studies that span this envelope.

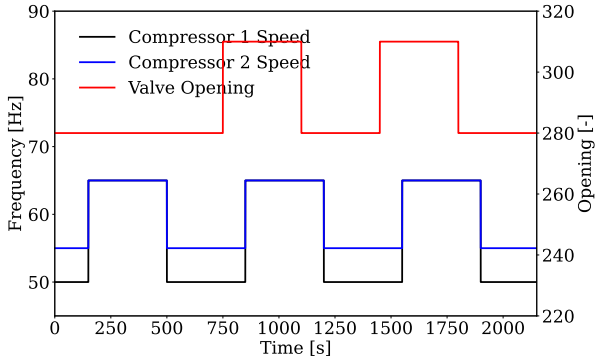
Based on the actuation ranges of the compressors and valves in Fig. 6, the single-compressor case study in Fig. 6a primarily evaluates *interpolation* generalization, since the applied actuation profile remains within the component-level training envelope. By contrast, the dual-compressor case study in Fig. 6b evaluates *extrapolation* capability at the assembled-cycle level: although each individual compressor and valve operates within its admissible component range, the parallel-compressor / parallel-condenser topology creates coupled junction conditions, flow-sharing behavior, and system-level operating regimes that are not present in the single-compressor source configuration.

The prediction accuracy of the assembled components is evaluated with the cycle output using the mean absolute percentage errors (MAPE), which is defined as

$$\text{MAPE}(y_{1:n}, \hat{y}_{1:n}) = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{\hat{y}_t} \right| \times 100\%, \tag{29}$$



(a) Single-compressor actuation: compressor speed and valve opening.



(b) Dual-compressor actuation: compressor speeds and valve opening.

Figure 6: Actuation programs used to probe both fast and slow transients in the two case studies.

where $y_{1:n}$ represents predictions from the VCS cycle surrogate, and $\hat{y}_{1:n}$ are the corresponding results from physics-based cycle model.

To verify the mass and energy conservation, energy inflow and outflow trajectories can be used at steady-state conditions. For our experiments, the energy inflow to the cycle is the sum of the total compressor power and the heat transferred from the ambient to the evaporator. On the other hand, energy leaving the system consists of the heat rejected by the condensers together with the thermal losses from the compressor. These contributions are written as follows:

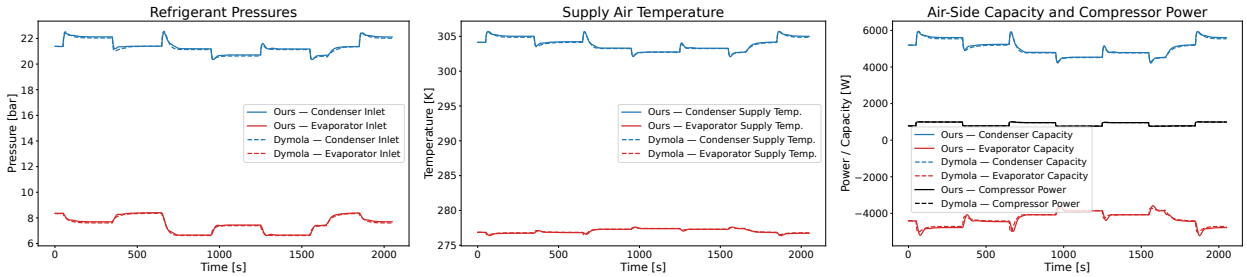
$$\begin{cases} E_{\text{in}} = P_{\text{tot}} + \dot{Q}_{a,\text{evap}}, \\ E_{\text{out}} = \dot{Q}_{a,\text{cond}} + \dot{Q}_{\text{loss}}, \end{cases} \tag{30}$$

where P_{tot} is the power consumption of the compressors.

4.3. Experimental Results

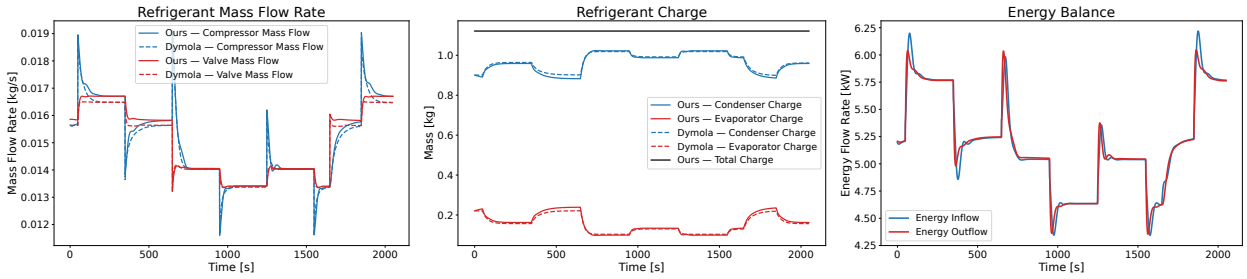
In this section, we answer the following questions:

- **Q1 (Modularity and accuracy):** How accurately does the physics-constrained NODE framework reproduce the thermodynamic and air-side behavior of the two VCS cycles (single- and dual-compressor cycles) relative to a high-fidelity Modelica reference?



(a) Refrigerant pressures. **(b)** Supply air temperature. **(c)** Heating capacity and compressor power.

Figure 7: Thermodynamic and air-side comparisons for the single-compressor cycle. NODE (ours) closely matches the high-fidelity reference across step-induced transients and steady plateaus.



(a) Refrigerant mass flow rate. **(b)** Refrigerant charge distribution. **(c)** Energy balance: inflow vs. outflow.

Figure 8: Mass and energy comparisons for the single-compressor cycle. Charge redistributes with actuation while total charge remains constant; energy inflow/outflow converges at the steady state.

- **Q2 (Physical consistency):** Does the assembled cycle satisfy mass and energy conservation?
- **Q3 (Computational efficiency):** What computational speedups does the proposed framework achieve compared to physics-based Modelica models?
- **Q4 (Comparison with neural-network surrogates):** What computational speedups does the proposed framework achieve compared with state-of-the-art neural-network baselines?
- **Q5 (Tolerance trade-off in adaptive integration scheme):** How does the adaptive step-size tolerance the integrator affect the trade-off between prediction accuracy and runtime?

The remainder of this section is organized as follows. We first examine Q1–Q4 on a single-compressor ASHP cycle in Section 4.4, followed by a dual-compressor ASHP cycle in Section 4.5. Finally, Section 4.6 addresses Q5 by varying the local error tolerance of the adaptive integrator.

4.4. Single-Compressor Cycle Case Study

We first address Q1–Q4 on a single-compressor ASHP cycle assembled from the independently trained component models.

4.4.1. Thermodynamic and Air-Side Responses (Q1)

With components connected in a serial loop, step changes in valve opening and compressor speed produce the expected thermodynamic trends. Opening the valve increases liquid throughput and lifts the evaporator pressure; raising the compressor speed depresses suction pressure and elevates discharge pressure as depicted in Fig. 7a. These shifts propagate to the air side in Fig. 7b: the supply-air temperature tracks the valve/speed steps with magnitudes that

scale with the refrigerant mass flow and the enthalpy lift across the machine. The time-aligned comparisons in Fig. 7c show the heating capacity and compressor power model closely agree the high-fidelity reference during the initial fast excursions and the subsequent approach to steady plateaus. Across pressures, supply-air temperature, capacity/power, mass flow, charge, and energy-flow panels (Figs. 7–8), the predictions exhibit uniformly low error. The summary in Fig. 9 reports a maximum MAPE of 1.57% for evaporator capacity, with all other reported variables below 1.08%.

4.4.2. Physical Consistency: Mass and Energy Conservation (Q2)

Beyond achieving high prediction accuracy, it is essential that the data-driven system model strictly preserves physical conservation laws.

Fig. 8a illustrates that the liquid mass flow rate increases with higher compressor speeds or wider valve openings. Throughout most transient and steady-state phases, the performance of the compressors and valve aligns closely with that of the high-fidelity Modelica model. These results indicate that our deep learning models follows the system dynamics, although small steady-state deviations appear at the beginning and end of the simulation, likely due to residual prediction errors in the mass-flow device models.

Fig. 8b presents the refrigerant charge distribution across each heat exchanger. To verify mass conservation within the system, the total refrigerant charge—computed as the sum of all heat exchangers—is also shown. The results show that the charge is fixed during model initialization and remains unchanged over the course of the simulation, in agreement with the mass conservation structure of the model.

Predictions of the system's energy inflow and outflow are depicted in Fig. 8c to validate compliance with energy conservation under steady-state conditions. As expected, discrepancies between inflow and outflow energy occur during transient periods due to temporary energy storage within the system. However, at steady state, these quantities converge, confirming that the proposed framework preserves energy balance across all operating conditions and upholds the physical conservation principles by design.

Our results underscore the framework capability and potential in both accuracy and physically consistent predictions using individually trained component models. The predictions align exceptionally well with those of the Modelica physics-based cycle model. Additionally, tracking the mass and energy flows into and out of the system verifies adherence to conservation principles under steady-state conditions.

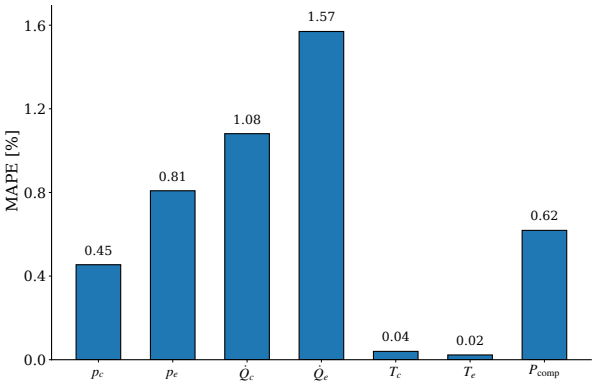
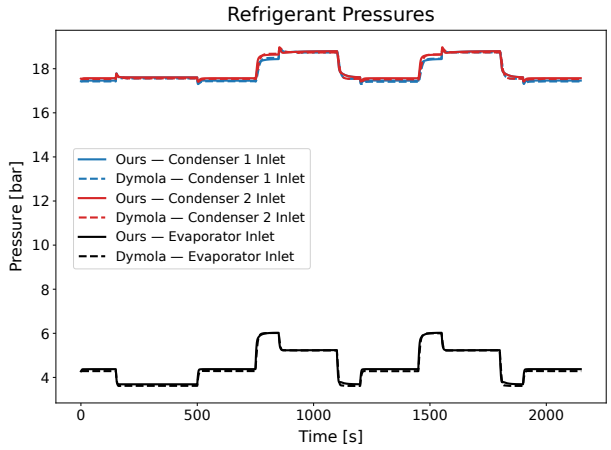


Figure 9: MAPE value for each variable in the single-compressor cycle. Maximum MAPE is 1.57% (for evaporator capacity); all other variables remain below 1.08%.

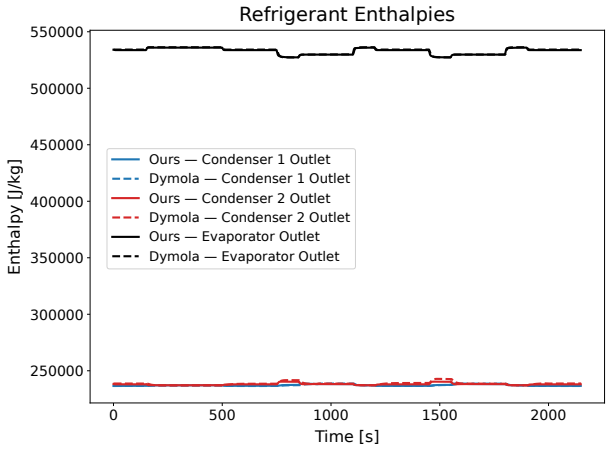
4.4.3. Computational Efficiency and Comparison with Neural-Network Surrogates (Q3 & Q4)

To provide a state-of-the-art hybrid deep-learning baseline, we also evaluate the physics-constrained GRU framework of [25], denoted "Hybrid-GRU" in Table 1. In this model, discrete-time conservation laws are enforced at each simulation step on a fixed time grid. Comparing against this Hybrid-GRU baseline allows us to isolate the impact of continuous-time NODE modeling and the adaptive integration scheme on both runtime and accuracy. For a 2000 s single-compressor simulation, our framework completes in 9.3 s, compared with 24.6 s for the Hybrid-GRU surrogate and 80.8 s for the physics-based Modelica model (Table 1). This corresponds to speedups of roughly 2.6× over the neural-network baseline and 8.7× over the physics-based reference. These results show that the proposed data-driven framework achieves predictive fidelity comparable to the high-fidelity model while delivering substantially lower runtimes than both Modelica (Q3) and the state-of-the-art neural-network surrogate (Q4).

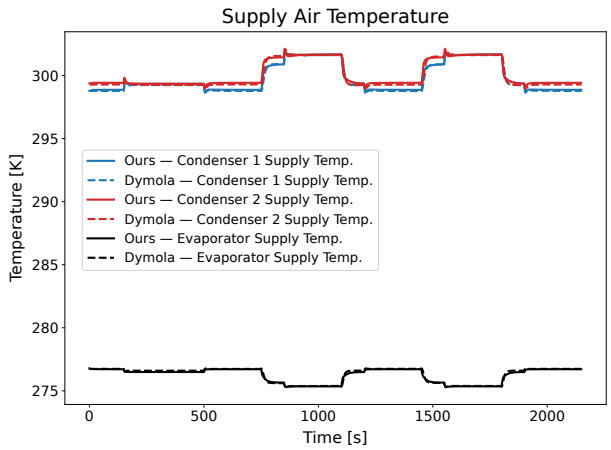
The efficiency gains of our method stem from the adaptive integration scheme, which uses larger time steps in quasi-steady periods and smaller ones during transients. As shown in Fig. 13, this strategy avoids unnecessary function evaluations in steady-state regimes while maintaining numerical accuracy during rapid changes, yielding faster yet precise simulations.



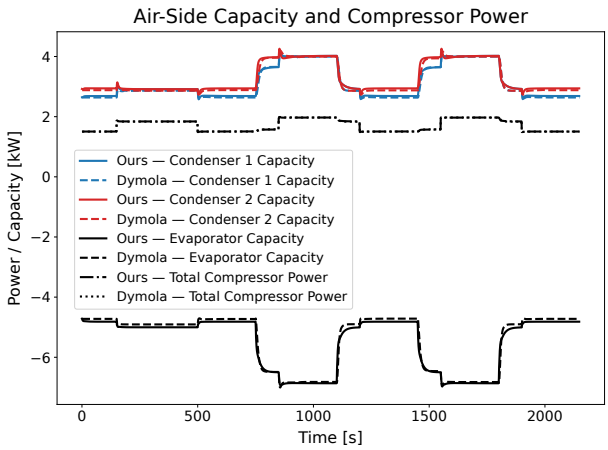
(a) Refrigerant pressures (dual-compressor).



(b) Refrigerant enthalpies (dual-compressor).



(c) Supply air temperature (dual-compressor).



(d) Heating capacity and compressor power (dual-compressor).

Figure 10: Thermodynamic and air-side comparisons for the dual-compressor cycle. Manifold junction solves keep shared ports synchronized during transients and at steady state.

4.5. Dual-Compressor Cycle Case Study

We next revisit Q1–Q4 on a more complex dual-compressor ASHP cycle, where the component models are reassembled into a parallel compression/condensation cycle topology.

4.5.1. Thermodynamic and Air-Side Responses (Q1)

Reassembling the components with two compressors and two condensers in parallel introduces suction and discharge manifolds. At each time step, the solver iteratively adjusts the junction pressures (p_a, p_b) to ensure that the manifold mass-residuals close. As shown in Fig. 10, larger valve openings and higher compressor speeds increase refrigerant throughput, which in turn lowers the evaporator-side air temperature and raises the condenser-side exit temperature. The capacity and power responses closely follow the commanded compressor speeds, particularly when high-speed operation coincides with a wider valve opening.

Fig. 10a demonstrates that compressor speed predominantly governs the refrigerant throughput and pressure ratio, decreasing p_{evap} and increasing p_{cond} at higher speeds. Valve actuation redistributes the refrigerant charge, leading to simultaneous increases in evaporator and condenser pressures as the valve opens. The system exhibits consistent, reversible input–output behavior, indicative of stable and physically coherent dynamics. Similar agreement is observed

across other thermodynamic variables, as shown in Figs. 10b–10c. Fig. 10d shows that the compressor power closely follows commanded speed variations, while the heat exchanger capacities reflect the influence of refrigerant pressures and mass flow rates. Across pressures, supply-air temperature, and capacity/power panels, the NODE-based cycle model exhibits uniformly low errors relative to the high-fidelity Modelica reference. The summary in Fig. 11 reports a maximum MAPE of 2.15% across all variables. The system model captures these behaviors accurately, showing excellent agreement with the Modelica reference. Overall, these results align closely with the high-fidelity Modelica reference.

4.5.2. Physical Consistency: Mass and Energy Conservation (Q2)

Beyond reproducing accurate system responses, it is equally critical that the proposed framework maintains strict adherence to physical conservation laws across the dual-compressor configuration.

Fig. 12a shows that the liquid mass flow rate increases with compressor speed and valve opening, as expected. Vapor-side flows exhibit more complex behavior: while total vapor flow rises with compressor speed, its distribution between the two compressors varies at steady state. At equal speeds, refrigerant redistribution shifts flow from compressor 2 to compressor 1, whereas at lower total flow rates, the faster compressor carries a larger share. The proposed data-driven model captures these trends and closely matches the high-fidelity Modelica results, with only small steady-state discrepancies due to minor mass-flow prediction errors.

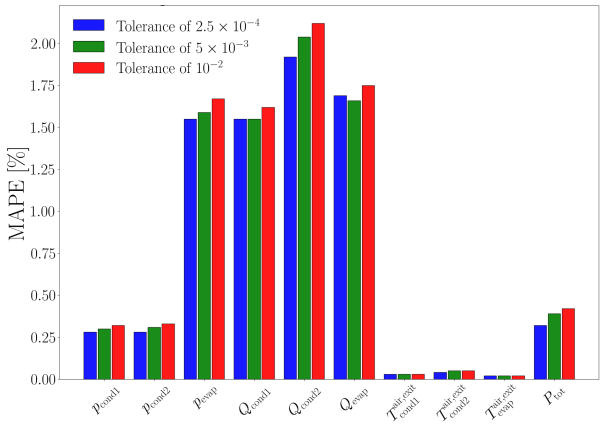
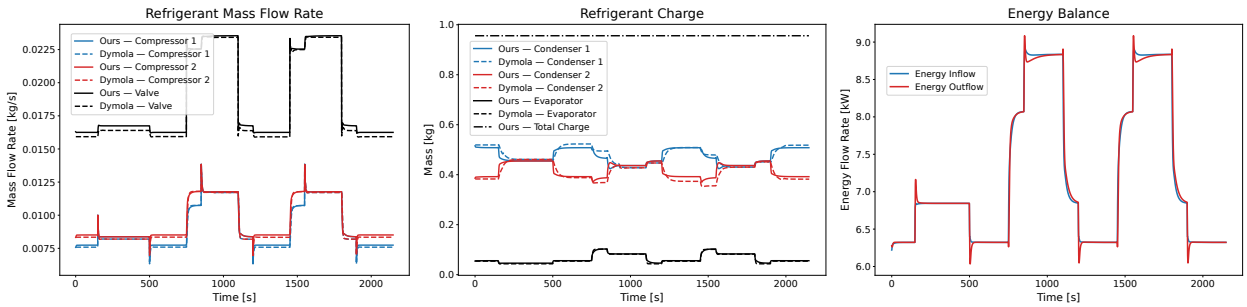


Figure 11: MAPE for various tolerances across reported variables. Errors remain below 2.15% in all cases.

Fig. 12b depicts the refrigerant charge. To verify mass conservation, the total refrigerant charge, computed as the sum of the three heat exchangers, is shown to remain constant throughout the simulation. This invariance, determined entirely at model initialization, confirms strict mass conservation. Notably, both the data-driven and Modelica models predict identical total charge values due to shared initial conditions.



(a) Refrigerant mass flow rate (dual-compressor).

(b) Refrigerant charge distribution (dual-compressor).

(c) Energy balance: inflow vs. outflow (dual-compressor).

Figure 12: Mass and energy comparisons for the dual-compressor cycle. Charge redistributes with manifold flow sharing; energy inflow/outflow converges at the steady state.

Finally, energy inflow and outflow trajectories are tracked to verify energy conservation at steady-state conditions. As shown in Fig. 12c, transient discrepancies arise due to temporary energy storage within the system; however, under steady-state conditions, the inflow and outflow energy match exactly. This confirms the framework's strong adherence to energy conservation principles.

The strong agreement with the high-fidelity reference in the dual-compressor results therefore demonstrates not

Table 1: Average simulation time for the high-fidelity Modelica model, the Hybrid-GRU surrogate, and the proposed NODE-based surrogate. The reduction columns report the percentage runtime reduction achieved by the proposed method relative to the baseline model listed in each row.

Model	CPU Time (s)		Runtime Reduction of Ours (%)	
	Single Comp.	Dual Comp.	Single Comp.	Dual Comp.
Ours	9.3	26.5	–	–
Hybrid-GRU [25]	24.6	40.4	62.2	34.4
Modelica (Dymola)	80.8	146.8	88.5	81.9

only interpolation accuracy of the component models, but also good topology-transfer and out-of-distribution generalization of the modular assembly framework.

4.5.3. Computational Efficiency and Comparison with Neural-Network Surrogates (Q3 & Q4).

For a 2150 s dual-compressor cycle simulation, our framework completes in 26.5 s, compared to 40.4 s for the Hybrid-GRU surrogate in [25] and 146.8 s for the high-fidelity Modelica model, corresponding to a 5.54× speedup over Modelica and a 34.4% reduction in runtime relative to the GRU baseline. Table 1 summarizes average wall-clock simulation times for the single- and dual-compressor configurations. These results directly address Q3 and Q4: the physics-constrained NODE framework achieves substantial computational savings compared with both physics-based and state-of-the-art neural-network surrogates while maintaining high predictive accuracy.

The superior efficiency of our approach arises from the adaptive integration scheme, which dynamically selects larger time steps during quasi-steady periods and finer ones during transients. As illustrated in Fig. 13, this mechanism avoids unnecessary function evaluations in steady-state regimes while preserving numerical accuracy during fast transients, thereby enabling faster yet precise simulations.

4.6. Effect of Adaptive Integration Scheme on the Accuracy and Runtime (Q5)

To address Q5, we investigated the trade-off between prediction accuracy and computational cost as a function of the local error tolerance δ used by the adaptive integrator. We considered three representative tolerances: $\delta \in \{10^{-2}, 5 \times 10^{-3}, 2.5 \times 10^{-4}\}$.

Figure 13 shows that larger tolerances lead to larger adaptive time steps, improving computational efficiency, whereas smaller tolerances result in shorter time steps and hence higher predictive accuracy. Figure 11 quantifies this trade-off by reporting the MAPE across all variables together with the corresponding average simulation times. In all cases, MAPE remains below 2.15%, indicating that the data-driven models retain accuracy comparable to the high-fidelity simulations even at relatively loose tolerances.

In our experiments, a tolerance of $\delta = 2.5 \times 10^{-4}$ offered a good compromise between accuracy and speed: it kept errors at or below 2.15% while still delivering substantial runtime savings relative to both the Hybrid-GRU surrogate [25] and the Modelica model. These findings confirm that the proposed physics-constrained data-driven framework not only captures system dynamics with high fidelity but also allows practitioners to tune the accuracy–efficiency trade-off via a single integrator tolerance parameter.

To make the efficiency gain more interpretable, we also compare the effective time discretization used by the two data-driven simulators in the dual-compressor case. The Hybrid-GRU baseline is a discrete-time model and therefore advances on a fixed step size of 1 s. In contrast, the proposed NODE framework uses adaptive integration, and Table 2 indicates an average time-step size of approximately 2.7 s for the dual-compressor simulation. Thus, our method requires substantially fewer system-level updates than the Hybrid-GRU baseline, while still shrinking the step size during rapid transients and increasing it during quasi-steady periods. This adaptive behavior is a key reason why the proposed framework achieves lower runtime than the fixed-step GRU surrogate.

4.7. Algebraic Solver Performance for Cycle Simulations

To make the role of the nonlinear algebraic coupling step more transparent, we instrumented the root-finding solver and recorded its convergence and runtime statistics for the two assembled-cycle case studies. At each candidate time step, the nonlinear algebraic solve was initialized with a zero vector.

Table 2: Effective time-discretization comparison for the dual-compressor case study.

Method	Time discretization	Effective step size
Hybrid-GRU [25]	Fixed	1.0 s
Ours (NODE)	Adaptive	≈ 2.7 s

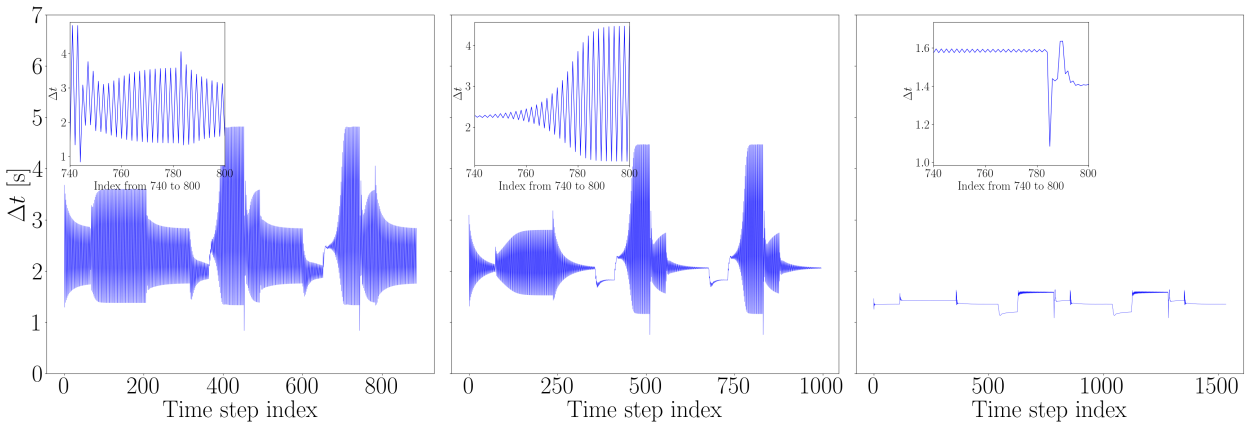


Figure 13: Effect of adaptive step-size tolerance on integration (left to right: 10^{-2}, 5×10^{-3}, 2.5×10^{-4}). Larger tolerances increase step size (improving the computation time) while tighter tolerances decrease step size (improving the system-level accuracy).

Table 3 shows that the nonlinear algebraic solve remains well behaved in both topologies, with modest average and maximum iteration counts. Notably, the dual-compressor case does not require more iterations than the single-compressor case, which indicates that the added topology complexity does not significantly worsen the convergence behavior of the root-finding iterations themselves.

By contrast, the total computation time of the algebraic solve increases from 4.39 s in the single-compressor case to 13.20 s in the dual-compressor case. This indicates that the main computational effect of increasing topology complexity is the higher cost of solving a larger coupled algebraic system.

The runtime shares of 47.21% and 49.83% further show that the algebraic coupling step accounts for roughly half of the total simulation cost in both case studies. Thus, while the neural component models remain efficient, the nonlinear junction solve is a major contributor to runtime and becomes the main source of added computational cost as the cycle topology grows.

5. Discussions and Limitations

In this work, the decision to develop individual component models rather than a single cycle-level model was driven by the need for flexibility and reusability. A monolithic black-box representation of the entire VCS cycle would tightly couple all operating conditions and design parameters into one model [26, 27, 28], limiting adaptability to new system configurations or modifications, such as changes in the number or arrangement of heat exchangers. In contrast, a modular framework with independently trained component models allows for reconfiguration of the VCS cycle topology.

We explored the use of physics-informed ML techniques for training component models [65, 27] to enforce physical laws also in the component models. Although these models exhibited high accuracy when tested individually, their integration into the system still led to mass and energy imbalances, ultimately compromising the robustness of the overall system model. These inconsistencies suggest that enforcing physical constraints solely at the component level is insufficient for maintaining global conservation laws, reinforcing the necessity of system-level constraints for stable and reliable simulations.

Table 3: Convergence and computational statistics of the nonlinear algebraic solver for the two assembled-cycle case studies. The solver was initialized with a zero vector at each candidate time step.

Case	Avg. iteration	Max. iteration	Solver time (s)	Runtime share (%)
Single-compressor	6.68	19	4.39	47.21
Dual-compressor	6.04	17	13.20	49.83

The proposed framework has achieved meaningful speedups over high-fidelity physical models, though these gains are not unlimited. The system solver must still update junction pressures and mass flows at each time step, often requiring iterative routines. Additionally, the adaptive integration scheme used in NODEs introduces computational overhead when detecting rapid transients or stiff dynamics, reducing overall computational acceleration. Furthermore, state-of-the-art physics-based models are already highly optimized in certain toolchains, making further speedups incremental in some cases.

Several avenues exist for further strengthening the proposed framework. First, while some component inputs may not be directly measurable in practical deployments, this naturally motivates the integration of state-estimation techniques or targeted sensing strategies, enabling the framework to function as a full-fidelity digital twin [66, 67, 68]. Second, predictive performance benefits from training data that span a wide range of operating conditions. As system behavior evolves due to factors such as changing ambient climates, control policies, or equipment aging, the modular structure readily supports augmentation of the component library through additional training trajectories or adaptive online and periodic recalibration. Third, although the current implementation combines ODE solvers for NODE-based heat-exchanger models with algebraic FNN representations for compressors and expansion devices, the resulting cycle-level dynamics are inherently differential–algebraic. A dedicated DAE solver [69] could further enhance the proposed framework. In contrast to the current implementation, which advances the differential heat-exchanger states and then solves the algebraic coupling equations separately, a DAE formulation would treat the differential and algebraic variables within one unified numerical scheme. This would provide a more natural representation of tightly coupled component interactions and would also facilitate the inclusion of additional algebraic or quasi-static relations, such as more detailed momentum closures beyond the present quasi-steady approximation. In larger HVAC systems with many interconnections, such a formulation may also improve numerical robustness and reduce repeated solver overhead by handling the coupled system implicitly.

A remaining practical consideration is the integration of the proposed framework into fully end-to-end differentiable optimization pipelines, such as gradient-based model predictive control or trajectory optimization. While the present formulation is designed primarily for fast and physically consistent forward simulation, such an extension would require sensitivity propagation not only through the NODE dynamics, but also through the nonlinear algebraic coupling solve used to enforce cycle-level consistency. In particular, gradients would need to be computed through the implicit solution of $R(X, U, \pi) = 0$, for example, via implicit differentiation or related adjoint-based techniques. Moreover, the adaptive-step integration strategy may introduce additional implementation complexity for gradient computation and numerical conditioning. These issues do not diminish the utility of the present framework for forward simulation, but point to an important and promising direction for future work.

6. Conclusions

This paper presents a modular, continuous-time data-driven framework for dynamic modeling of vapor-compression HVAC systems. The framework learns component-level surrogate models and assembles them into a cycle simulator by enforcing mass and energy conservation at component interconnections through algebraic coupling equations. By imposing these conservation laws as exact constraints during assembly, the simulator maintains physical consistency at the system level even when individual component models exhibit approximation errors.

The framework represents dynamic components using neural ordinary differential equations. The continuous-time formulation allows components trained on heterogeneous and irregularly sampled data to interact without requiring a common discrete time step. A single adaptive integration scheme advances the entire cycle and defines consistent time instances at which all components exchange port variables. This approach supports stable multi-rate simulation and avoids the numerical inconsistencies that arise when we interconnect discrete-time component models with

mismatched sampling periods.

Case studies on single- and dual-compressor air-source heat pump configurations show that the framework accurately reproduces thermodynamic states and air-side behavior relative to a high-fidelity Modelica reference, with a maximum MAPE of 2.15%. The assembled cycles satisfy mass and energy conservation at all times, which confirms the effectiveness of the hard-constrained interconnection strategy. The modular structure enables direct reuse of trained component models across different refrigerant circuit topologies and requires only limited retuning when we modify the system configuration. Comparisons with physics-based and neural-network baselines show that the framework reduces computational cost while preserving accuracy, and an analysis of integration tolerances clarifies the trade-off between runtime and prediction fidelity.

These results demonstrate that enforcing physical consistency at the system level, rather than learning cycle behavior through monolithic surrogates, provides a robust and scalable approach for ML-based HVAC simulation. This capability supports computationally efficient equipment models that can be embedded in building-level supervisory control, diagnostic routines, and fast design studies requiring repeated long-horizon rollouts. Future work will extend the framework to differential–algebraic formulations using neural DAE models and will integrate state estimation methods to support deployment with experimental and field data.

References

[1] R. Energy, Energy efficiency trends in residential and commercial buildings (2010).

[2] B. P. Rasmussen, Dynamic modeling for vapor compression systems—part i: Literature review, HVAC&R Research 18 (5) (2012) 934–955.

[3] P. Li, H. Qiao, Y. Li, J. E. Seem, J. Winkler, X. Li, Recent advances in dynamic modeling of hvac equipment. part 1: Equipment modeling, HVAC&R Research 20 (1) (2014) 136–149.

[4] G. Zhang, H. Xiao, P. Zhang, B. Wang, X. Li, W. Shi, Y. Cao, Review on recent developments of variable refrigerant flow systems since 2015, Energy and Buildings 198 (2019) 444–466.

[5] D. Swider, M. Browne, P. Bansal, V. Kecman, Modelling of vapour-compression liquid chillers with neural networks, Applied thermal engineering 21 (3) (2001) 311–329.

[6] M. Kim, S. H. Yoon, W. V. Payne, P. A. Domanski, Development of the reference model for a residential heat pump system for cooling mode fault detection and diagnosis, Journal of mechanical science and technology 24 (7) (2010) 1481–1489.

[7] J. Wang, X. Lu, V. Adetola, E. Louie, Modeling variable refrigerant flow (vrf) systems in building applications: A comprehensive review, Energy and Buildings 311 (2024) 114128.

[8] S. Yousaf, C. R. Bradshaw, R. Kamalapurkar, O. San, Investigating critical model input features for unitary air conditioning equipment, Energy and Buildings 284 (2023) 112823.

[9] P.-C. Hsu, L. Gao, Y. Hwang, R. Radermacher, A review of the state-of-the-art data-driven modeling of building hvac systems, Energy and Buildings (2025) 115881.

[10] F. Mtibaa, K.-K. Nguyen, M. Azam, A. Papachristou, J.-S. Venne, M. Cheriet, Lstm-based indoor air temperature prediction framework for hvac systems in smart buildings, Neural Computing and Applications 32 (23) (2020) 17569–17585.

[11] E. Hannula, A. Häkkinen, A. Solonen, F. Uribe, J. de Wiljes, L. Roininen, Bayesian lstm for indoor temperature modeling, arXiv preprint arXiv:2504.03350 (2025).

[12] M. J. Ellis, V. Chinde, An encoder–decoder lstm-based empc framework applied to a building hvac system, Chemical Engineering Research and Design 160 (2020) 508–520.

[13] A. Aswani, N. Master, J. Taneja, D. Culler, C. Tomlin, Reducing transient and steady state electricity consumption in hvac using learning-based model-predictive control, Proceedings of the IEEE 100 (1) (2011) 240–253.

[14] C. R. Laughman, H. Qiao, On the influence of state selection on mass conservation in dynamic vapour compression cycle models, Mathematical and Computer Modelling of Dynamical Systems 23 (3) (2017) 262–283.

[15] C. R. Laughman, V. M. Deshpande, H. Qiao, S. A. Bortoff, A. Chakrabarty, Digital twins of vapor compression cycles: Challenges and opportunities, in: Proc. Int. Congress of Refrigeration (ICR), 2023.

[16] C. Runge, Über die numerische auflösung von differentialgleichungen, Mathematische Annalen 46 (2) (1895) 167–178.

[17] W. Kutta, Beitrag zur näherungsweisen Integration totaler Differentialgleichungen, Teubner, 1901.

[18] E. Hairer, S. P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems: With 105 Figures, Springer-Verlag, 1987.

[19] A. Afram, F. Janabi-Sharifi, Review of modeling methods for hvac systems, Applied Thermal Engineering 67 (1) (2014) 507–519.

[20] I. Kang, K. H. Lee, J. H. Lee, J. W. Moon, Artificial neural network–based control of a variable refrigerant flow system in the cooling season, Energies 11 (7) (2018) 1643.

[21] S. Zhou, A. Shah, P. Leung, X. Zhu, Q. Liao, A comprehensive review of the applications of machine learning for hvac, DeCarbon 2 (2023) 100023.

[22] Z. Jiang, X. Wang, H. Li, T. Hong, F. You, J. Drgoňa, D. Vrabie, B. Dong, Physics-informed machine learning for building performance simulation-a review of a nascent field, Advances in Applied Energy (2025) 100223.

[23] S. B. Padmanabhan, M. T. Mabrouk, B. Lacarrière, Neural-accelerated dynamic modeling of heat pumps, Applied Thermal Engineering 274 (2025) 126653.

[24] H. Tu, S. Moura, Y. Wang, H. Fang, Integrating physics-based modeling with machine learning for lithium-ion batteries, Applied energy 329 (2023) 120289.

[25] J. Ma, Y. Dong, H. Qiao, C. R. Laughman, A physics-constrained deep learning framework for dynamic modeling of vapor compression systems, Applied Thermal Engineering 254 (2024) 123734.
[26] J. Drgoňa, A. R. Tuor, V. Chandan, D. L. Vrabie, Physics-constrained deep learning of multi-zone building thermal dynamics, Energy and Buildings 243 (2021) 110992.
[27] Z. Chen, F. Xiao, F. Guo, J. Yan, Interpretable machine learning for building energy management: A state-of-the-art review, Advances in Applied Energy 9 (2023) 100123.
[28] Z. Ma, G. Jiang, Y. Hu, J. Chen, A review of physics-informed machine learning for building energy modeling, Applied Energy 381 (2025) 125169.
[29] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, Advances in neural information processing systems 27 (2014).
[30] H. P. Das, Y.-W. Lin, U. Agwan, L. Spangher, A. Devonport, Y. Yang, J. Drgoňa, A. Chong, S. Schiavon, C. J. Spanos, Machine learning for smart and energy-efficient buildings, Environmental Data Science 3 (2024) e1.
[31] J. Drgoňa, J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Ollé, J. Oravec, M. Wetter, D. L. Vrabie, et al., All you need to know about model predictive control for buildings, Annual reviews in control 50 (2020) 190–232.
[32] A. Afram, F. Janabi-Sharifi, Review of modeling methods for hvac systems, Applied thermal engineering 67 (1-2) (2014) 507–519.
[33] B. Jiang, H. Gong, H. Qin, M. Zhu, Attention-lstm architecture combined with bayesian hyperparameter optimization for indoor temperature prediction, Building and Environment 224 (2022) 109536.
[34] Z. Fang, N. Crimier, L. Scanu, A. Midelet, A. Alyafi, B. Delinchant, Multi-zone indoor temperature prediction with lstm-based sequence to sequence model, Energy and Buildings 245 (2021) 111053.
[35] F. Elmaz, R. Eyckerman, W. Casteels, S. Latré, P. Hellinckx, Cnn-lstm architecture for predictive indoor temperature modeling, Building and Environment 206 (2021) 108327.
[36] R. T. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, Advances in neural information processing systems 31 (2018).
[37] Y. Rubanova, R. T. Chen, D. K. Duvenaud, Latent ordinary differential equations for irregularly-sampled time series, Advances in neural information processing systems 32 (2019).
[38] M. Poli, S. Massaroli, J. Park, A. Yamashita, H. Asama, J. Park, Graph neural ordinary differential equations, arXiv preprint arXiv:1911.07532 (2019).
[39] S. Massaroli, M. Poli, J. Park, A. Yamashita, H. Asama, Dissecting neural odes, Advances in Neural Information Processing Systems 33 (2020) 3952–3963.
[40] V. M. M. Alvarez, R. Roşca, C. G. Fălcuţescu, Dynode: Neural ordinary differential equations for dynamics modeling in continuous control, arXiv preprint arXiv:2009.04278 (2020).
[41] Y. D. Zhong, B. Dey, A. Chakraborty, Symplectic ode-net: Learning hamiltonian dynamics with control, in: International Conference on Learning Representations.
[42] Y. Oh, S. Kam, J. Lee, D.-Y. Lim, S. Kim, A. Bui, Comprehensive review of neural differential equations for time series analysis, arXiv preprint arXiv:2502.09885 (2025).
[43] Y. Rubanova, R. T. Chen, D. K. Duvenaud, Latent ordinary differential equations for irregularly-sampled time series, Advances in neural information processing systems 32 (2019).
[44] E. De Brouwer, J. Simm, A. Arany, Y. Moreau, Gru-ode-bayes: Continuous modeling of sporadically-observed time series, Advances in neural information processing systems 32 (2019).
[45] P. Kidger, J. Morrill, J. Foster, T. Lyons, Neural controlled differential equations for irregular time series, Advances in Neural Information Processing Systems 33 (2020) 6696–6707.
[46] S. Y. Jhin, J. Lee, M. Jo, S. Kook, J. Jeon, J. Hyeong, J. Kim, N. Park, Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 3102–3112.
[47] C. Herrera, F. Krach, J. Teichmann, Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering, arXiv preprint arXiv:2006.04727 (2020).
[48] B. Tzen, M. Raginsky, Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit, arXiv preprint arXiv:1905.09883 (2019).
[49] P. Kidger, On neural differential equations, arXiv preprint arXiv:2202.02435 (2022).
[50] R. Dandekar, K. Chung, V. Dixit, M. Tarek, A. Garcia-Valadez, K. V. Vemula, C. Rackauckas, Bayesian neural ordinary differential equations, arXiv preprint arXiv:2012.07244 (2020).
[51] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, C.-J. Hsieh, Neural sde: Stabilizing neural ode networks with stochastic noise, arXiv preprint arXiv:1906.02355 (2019).
[52] P. Gao, X. Yang, R. Zhang, K. Huang, J. Y. Goulermas, Explainable tensorized neural ordinary differential equations for arbitrary-step time series prediction, IEEE Transactions on Knowledge and Data Engineering 35 (6) (2022) 5837–5850.
[53] Y. Chen, K. Ren, Y. Wang, Y. Fang, W. Sun, D. Li, Contiformer: Continuous-time transformer for irregular time series modeling, Advances in Neural Information Processing Systems 36 (2024).
[54] F. E. Cellier, E. Kofman, Continuous system simulation, Springer, 2006.
[55] V. Taboga, C. Gehring, M. Le Cam, H. Dagdougui, P.-L. Bacon, Neural differential equations for temperature control in buildings under demand response programs, Applied Energy 368 (2024) 123433.
[56] B. He, N. Zhang, C. Fang, Y. Su, Y. Wang, Flexible building energy management with neural odes-based model predictive control, IEEE Transactions on Smart Grid 15 (5) (2024) 4690–4704.
[57] Z. Ma, G. Jiang, J. Chen, Neural ordinary differential equations-based approach for enhanced building energy modeling on small datasets, in: Building Simulation, Springer, 2025, pp. 1–20.
[58] S. Shao, W. Shi, X. Li, H. Chen, Performance representation of variable-speed compressor for inverter air conditioners based on experimental data, International Journal of Refrigeration 27 (8) (2004) 805–815.

[59] L. Yang, L.-X. Zhao, C.-L. Zhang, B. Gu, Loss-efficiency model of single and variable-speed compressors using neural networks, International Journal of Refrigeration 32 (6) (2009) 1423–1432.
[60] H. Qiao, T. Kobayashi, C. Laughman, S. Bortoff, On the characteristics of one-dimensional compressible flow, in: The 15th International Modelica Conference, 2023.
[61] M. Kochenderfer, Algorithms for Optimization, The MIT Press Cambridge, 2019.
[62] E. Fehlberg, Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems, Vol. 315, National aeronautics and space administration, 1969.
[63] Dymola 2023x, Dassault Systèmes, 2023.
[64] H. Qiao, V. Aute, R. Radermacher, Transient modeling of a flash tank vapor injection heat pump system – part i: Model development, International Journal of Refrigeration 49 (2015) 169–182.
[65] A. Sholokhov, Y. Liu, H. Mansour, S. Nabi, Physics-informed neural ode (pinode): embedding physics into models using collocation points, Scientific Reports 13 (1) (2023) 10166.
[66] V. M. Deshpande, C. R. Laughman, Y. Ma, C. Rackauckas, Constrained smoothers for state estimation of vapor compression cycles, in: 2022 American Control Conference (ACC), IEEE, 2022, pp. 2333–2340.
[67] V. M. Deshpande, A. Chakrabarty, A. P. Vinod, C. R. Laughman, Physics-constrained deep autoencoded kalman filters for estimating vapor compression system states, IEEE Control Systems Letters 7 (2023) 3483–3488.
[68] V. M. Deshpande, C. R. Laughman, Multi-pass extended kalman smoother with partially-known constraints for estimation of vapor compression cycles, IFAC-PapersOnLine 56 (2) (2023) 6751–6758.
[69] K. E. Brenan, S. Campbell, L. R. Petzold, Numerical solution of initial-valued problems in differential-algebraic equations, Elsevier Science Publishing Co., Inc, 1989.