Neural Network–Assisted Gradient Approximation and Swarm-Based Optimization for Predictive Control of a Nonlinear Copolymerization Reactor

Rami Boumegoura*, **, Riad Bendib*, **, Kamel Menighed, El-Arkam Mechhoud*, Abdulaziz Alshareef ***, Hafidha Boudouaia*, ****

**Automatic Laboratory of Skikda (LAS), University of 20 August 1955, Skikda 21000, Algeria (e-mails: r.boumegoura@univ-skikda.dz, r.bendib@univ-skikda.dz, el-arkam.mechhoud@univ-skikda.dz)*
*** Department of Petrochemicals, University of 20 August 1955, Skikda 21000, Algeria (e-mail: k.menighed@univ-skikda.dz, h.boudouaia@univ-skikda.dz)*
**** Department of Electrical & Computer Engineering King Abdulaziz University Djeddah KSA (e-mail: alshareef1379@yahoo.com)*

Abstract: Copolymerization reactors are crucial for polymer synthesis but remain difficult to regulate due to nonlinear kinetics, heat sensitivity, and disturbances such as inhibitor impurities. Traditional approaches like Proportional-Integral-Derivative (PID) controllers and linear Model Predictive Control (MPC) sometimes struggle with computational efficiency and accuracy trade-offs. This study presents a hybrid framework incorporating MPC, Neural Network (NN)-based gradient approximation, and adaptive Particle Swarm Optimization (PSO). The NN accelerates gradient computation to bypass iterative quadratic programming, while PSO enhances global search capabilities under actuator constraints. Validated on a nonlinear copolymerization reactor model, the hybrid method displays enhanced setpoint tracking and disturbance rejection compared to standard MPC and standalone PSO, with maintained computational feasibility. Robustness is emphasized by effective management of unmeasured inhibitor perturbations, achieving stable input adjustments and precise output regulation.

Keywords: Model Predictive Control (MPC), MIMO systems, Copolymerization reactor, Neural network (NN) gradient approximation, Particle Swarm Optimization (PSO), Swarm diversity adaptation.

1. INTRODUCTION

Copolymerization reactors are central to producing high-value polymers used in petrochemical industries, but their control remains a significant challenge due to nonlinear reaction kinetics, thermal sensitivities, and unpredictable disturbances (e.g., inhibitor impurities in feed streams; (Congalidis et al., 1989)). These reactors demand precise regulation of monomer feed ratios, temperature, and molecular weight to ensure consistent product quality and safe operation. Traditional control strategies, such as PID or linear Model Predictive Control (MPC), often struggle to manage these complexities, leading to off-spec production, energy inefficiency, or even reactor instability during transitions (Abbas et al., 2015).

Initial efforts in polymerization reactor control centered on analytical methods to address nonlinear kinetics and thermal constraints. Pontryagin's Maximum Principle (PM) became a cornerstone for deriving optimal temperature and initiator policies in batch systems, particularly for managing catalyst decay and reaction exothermicity (Hicks et al., 1969; Osakada and Fan, 1970). Early applications by (Chen and Jeng, 1978) demonstrated truncated S-shaped temperature trajectories for styrene polymerization, while (Farber and Laurence, 1986) optimized minimum-time policies under initiator constraints. Concurrently, Dynamic Programming (DP) enabled bang-bang control strategies for polycondensation reactors, balancing molecular weight targets with reaction rates (Hicks et al., 1969; Sacks et al., 1972). These analytical strategies offer precise, theoretically optimal policies for well-characterized systems and are valuable for understanding fundamental reactor dynamics. However, they typically rely on simplified models, cannot easily handle operational constraints, and are not scalable to high-dimensional, time-varying industrial processes.

The advent of computational optimization introduced Genetic Algorithms and Dynamic Matrix Control (DMC) to handle multi-objective challenges. (Tsoukas et al., 1982) leveraged Genetic Algorithms for copolymer composition control, minimizing molecular weight drift in semi-batch reactors. (Özkan et al., 2003, 2006) advanced multi-model MPC by combining piecewise linear models with Lyapunov-based stability constraints, achieving smooth grade transitions in copolymerization. Parallel work by (Lima et al., 2013) integrated DMC with feedforward compensation to reject inhibitor disturbances, demonstrating precise servo control in continuous reactors. Linear Parameter-Varying (LPV) frameworks emerged to address the computational complexity of high-dimensional scheduling variables in copolymerization systems. (Rahme et al., 2015) reduced model conservatism through parameter set mapping (PSM) and principal component analysis (PCA), enabling efficient synthesis of robust controllers. (Abbas et al., 2015) further refined this approach using linear fractional transformation (LFT) to preserve critical input-output dynamics while truncating redundant states. These methods provided a structured pathway to handle nonlinearities without relying on exhaustive linearization, bridging the gap between traditional MPC and adaptive control paradigms. MPC methods excel at managing multivariable interactions and enforcing input/output

constraints in real time, making them attractive for copolymerization control. Nonetheless, their reliance on iterative optimization at each sampling step can lead to high computational cost for nonlinear models, limiting real-time feasibility in industrial-scale applications.

Modern AI-driven strategies harnessed neural networks (NNs) for real-time adaptability. Early applications by (Shahrokhi and Pishvaie, 1998) employed feedforward NNs to estimate reaction heat generation in batch polymerization, enabling near-perfect feedforward compensation for nonlinear exothermic dynamics. (Zhang, 2004) expanded this paradigm with Bootstrap Aggregate Neural Networks (BANN) to predict molecular weight distributions, circumventing reliance on mechanistic models. (Tian et al., 2001, 2004) hybridized simplified first-principles models with stacked recurrent NNs, capturing gel effects and enabling online reoptimization under disturbances like reactive impurities. Further innovations integrated NNs with fuzzy logic, as seen in (Vosough et al., 2004), where adaptive fuzzy controllers leveraged NN approximations to dynamically adjust exothermic reaction parameters under trajectory uncertainty. (Mahmoud et al., 2021) enhanced robustness via neuro-adaptive controllers, combining backstepping with terminal sliding modes to ensure finite-time convergence under parametric uncertainties. Recent advancements by (Maaruf et al., 2023) fused NN-based dynamics approximation with metaheuristic optimization, addressing input dead zones and asymmetric disturbances in continuous reactors through hybrid AI-control architectures. AI-based approximators such as neural networks can capture complex nonlinearities and adapt online, while metaheuristic optimizers like PSO provide global search capability in challenging cost landscapes. Still, NNs may lack formal stability guarantees and PSO may converge slowly, especially when used without problem-specific guidance or acceleration techniques.

Recent innovations fused metaheuristics with machine learning for global optimization. (Maaruf et al., 2023) combined Particle Swarm Optimization (PSO) with neural approximations to address input dead zones in continuous reactors, achieving sub-second settling times. (Singh and Kodamana, 2020) deployed Reinforcement Learning (RL) for PMMA batch control, using risk-sensitive policies to prevent unsafe states. These hybrid frameworks highlight the shift toward balancing computational efficiency (via NN acceleration) with global search capabilities (via PSO/RL; (Maaruf et al., 2023; Singh and Kodamana, 2020; Tian et al., 2004).

This work targets the trade-off between computational efficiency and closed-loop robustness in MPC for nonlinear, constrained MIMO copolymerization reactors. We propose a hybrid MPC–NN–PSO strategy that retains MPC's constraint handling while enhancing performance in nonlinear and disturbed regimes. The method combines: (1) MPC for constraint-aware optimization, (2) an offline-trained NN to approximate cost gradients, and (3) adaptive PSO for global search. The hybridization aims to accelerate convergence and improve robustness without sacrificing feasibility. Comparative simulations with QP–MPC and standalone PSO highlight its advantages in setpoint tracking and disturbance rejection. Hereafter, we refer to this hybrid framework as the NNGA-PSO framework (Neural Network–based Gradient Approximation with Particle Swarm Optimization).

This paper is organized into the following sections: Section 2 describes the dynamical model of the copolymerization reactor, including the reactor's physical setup and nonlinear dynamic modeling. Section 3 outlines the methodology for designing the hybrid control strategy, incorporating MPC, NN-based gradient approximation, and PSO. Section 4 presents the simulation results, comparing the proposed hybrid strategy's performance with conventional methods. Finally, Section 5 concludes the paper by summarizing key findings and discussing potential directions for future research.

2. COPOLYMERIZATION REACTOR DYNAMICAL MODEL

2.1 Overview of the Reactor System

The copolymerization reactor system under study is a solution-based polymerization setup adapted from the model described by (Congalidis et al., 1989). The system facilitates controlled polymerization of two monomers, methyl methacrylate (Monomer *a*) and vinyl acetate (Monomer *b*), in the presence of various other agents. The reactor setup includes a continuous feed of monomers, initiator (*i*), solvent (*s*), chain transfer agent (*t*), and a potential inhibitor (*z*), all introduced through the fresh feed. Benzene serves as the solvent, azobisisobutyronitrile as the initiator, and acetaldehyde as the chain transfer agent, with m-dinitrobenzene acting as an impurity inhibitor in the feed stream.

Figure 1 provides a schematic of this setup, showing the arrangement of continuous feed lines, the mixing tank, and the cooling jacket—components essential for managing the nonlinear behavior and thermal sensitivity characteristic of free-radical polymerization. The reactor system consists of a jacketed, well-mixed tank designed to ensure uniform composition and temperature, allowing for effective heat management to counteract the exothermic nature of the reactions and maintain stable operation.

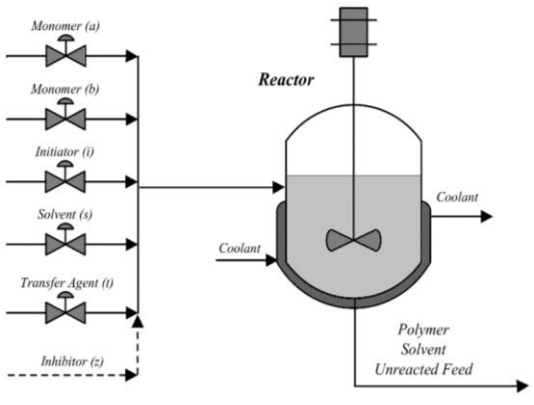

Fig. 1. Flow diagram of the copolymerization reactor system.

The system operates on the basis of a free-radical polymerization mechanism, consisting of 27 individual reactions that govern the formation and growth of polymer chains. This complexity necessitates advanced control strategies to manage the nonlinear behavior and thermal sensitivities inherent in the process.

2.2 Nonlinear Dynamic Model of the Process

The nonlinear dynamic model for the copolymerization reactor is formulated based on mass and energy balances. The reactor's dynamic behavior is described by a set of differential equations that capture the evolution of key state variables over time, including the concentrations of each component, the reactor temperature, the composition of the polymer, and molecular weight distribution moments. The following equations outline the core model dynamics (Abbas et al., 2015; Congalidis et al., 1989):

Component Concentrations:

The rate of change in concentration, Ck, of each component (monomers, initiator, solvent, chain transfer agent, and inhibitor) in the reactor is given by Eqs. (1-6):

$$\frac{\mathrm{d}C_k}{\mathrm{d}t} = -\varphi_l C_k + \frac{F_k}{M_k V} - \frac{\sum_k F_k}{V\rho} C_k \tag{1-6}$$

Reactor Temperature:

The temperature dynamics, T_r described by Eq. (7), incorporate the effects of heat generation from reactions, heat removal via the cooling jacket, and the inflow of reactants at feed temperature T_{rf}:

$$\frac{\mathrm{d}T_r}{\mathrm{d}t} = \frac{\varphi_7}{c\rho} C_a + \frac{\varphi_8}{c\rho} C_b - \frac{SU}{Vc\rho}(T_r - T_j) + \frac{\sum_k F_k}{V\rho}(T_{rf} - T_r) \tag{7}$$

where S and U denote the surface area and overall heat transfer coefficient, respectively, c the heat capacity, and T_j the jacket temperature. The heat generation terms, φ_7 and φ_8, depend on exothermic reaction heats and specific reaction rates of the monomers.

Polymer Composition:

The molar concentrations λ_a and λ_b of monomer units in the polymer matrix are governed by Eq. (8) and (9), which capture how monomers A and B contribute to the polymer composition within the reactor.

$$\frac{\mathrm{d}\lambda_a}{\mathrm{d}t} = \varphi_1 C_b - \frac{\sum_k F_k}{V\rho} \lambda_a \tag{8}$$

$$\frac{\mathrm{d}\lambda_b}{\mathrm{d}t} = \varphi_2 C_b - \frac{\sum_k F_k}{V\rho} \lambda_b \tag{9}$$

Molecular Weight Distribution Moments:

The zeroth, first, and second moments of the molecular weight distribution, ψ_0, ψ_1, and ψ_2, evolve according to Eqs. (10), (11) and (12):

$$\frac{\mathrm{d}\psi_0}{\mathrm{d}t} = \varphi_9 C_a + \varphi_{10} C_b - \frac{\sum_k F_k}{V\rho} \psi_0 \tag{10}$$

$$\frac{\mathrm{d}\psi_1}{\mathrm{d}t} = \varphi_{11} C_a + \varphi_{12} C_b - \frac{\sum_k F_k}{V\rho} \psi_1 \tag{11}$$

$$\frac{\mathrm{d}\psi_2}{\mathrm{d}t} = \varphi_{13} C_a + \varphi_{14} C_b - \frac{\sum_k F_k}{V\rho} \psi_2 \tag{12}$$

where the terms φ_9 through φ_{14} are reaction-dependent factors linked to the molecular weight distribution, with detailed functional forms provided in Appendix A.

Output Variables for Control

The main outputs of interest are the polymer production rate G_{pi}, the mole fraction Y_{ap} of monomer A in the polymer, and the weight-average molecular weight M_{pw}. These outputs are defined by Eqs. (13), (14) and (15):

$$G_{pi} = M_A V \varphi_1 C_A + M_B V \varphi_2 C_B \tag{13}$$

$$Y_{ap} = \varphi_{15} \lambda_a \tag{14}$$

$$M_{pw} = \varphi_{16} \psi_2 \tag{15}$$

where the coefficients φ_{15} and φ_{16} are also discussed in Appendix A. Table 1 summarizes the steady-state operating conditions and key parameters for the copolymerization reactor system. These input parameters play a crucial role in maintaining the desired polymer properties and ensuring stable, efficient reactor performance.

Table 1. Steady-State Operating Conditions and Inputs.

Category	Symbol	Value
Inputs		
Monomer A feed rate	F_a	18 *kg/h*
Monomer B feed rate	F_b	90 *kg/h*
Initiator (*i*) feed rate	F_i	0.18 *kg/h*
Solvent (*s*) feed rate	F_s	36 *kg/h*
Transfer Agent (*t*) feed rate	F_t	2.1 *kg/h*
Inhibitor (*z*) feed rate	F_z	0.05 *kg/h*
Reactor jacket temperature	T_j	336.15 *K*
Reactor feed temperature	T_{rf}	353.15 *K*
Reactor Parameters		
Reactor Volume	V	1.0 m^3
Heat Transfer Area	S	4.6 m^2
Overall heat transfer	U	216.0 $J / (m^2 . s . K)$
Heat capacity	e	2.01 $kJ / (kg . K)$
Density	ρ	879.0 kg/m^3

3. METHODOLOGY

In this section, we briefly describe the methodology used to develop and implement a hybrid control strategy that combines MPC with NN-based gradient approximation and PSO. The approach aims to enhance computational efficiency and optimize control performance for constrained, nonlinear systems.

3.1 MPC: Problem Formulation

MPC is a widely used advanced control strategy that leverages system dynamics to predict and optimize future behavior over

a finite prediction horizon, while ensuring compliance with operational constraints. The mathematical formulation of MPC typically begins with the discrete-time state-space representation of the system (Rawlings et al., 2017), given by Eq. (16):

$$\begin{aligned} x[k+1] &= Ax[k] + Bu[k] \\ y[k] &= Cx[k] + Du[k] \end{aligned} \tag{16}$$

Here $x[k] \in \mathbb{R}^n$ is the state vector, $u[k] \in \mathbb{R}^m$ is the control input vector, and $y[k] \in \mathbb{R}^p$ is the output vector at time step k. The matrices A, B, C, and D describe the system dynamics, mapping inputs and states to outputs.

To predict future states and outputs over a prediction horizon N_p, these dynamics are reformulated in matrix form, as shown in Eq. (17):

$$\begin{aligned} X &= \Phi x[k] + \Gamma U \\ Y &= \Psi x[k] + \Theta U \end{aligned} \tag{17}$$

where $U = [u[k]^\top, u[k+1]^\top, \ldots, u[k+N_c-1]^\top]^\top$ represents the sequence of control inputs over the control horizon N_c, with $N_c \leq N_p$. This results in a column vector $U \in \mathbb{R}^{mN_c \times 1}$, representing the control input trajectory over the control horizon. The matrices Φ, Γ, Ψ, and Θ are constructed to represent the evolution of states and outputs over the prediction horizon. In the MPC formulation, $x[k]$ represents the known plant state at time step k, either measured or estimated, and serves as the initial condition for the prediction model. At each sampling time, the prediction sequence X is initialized with the current value of $x[k]$, and the optimization determines the control sequence U accordingly. The state prediction matrix Φ and input prediction matrix Γ are defined as block-Toeplitz matrices to capture the temporal propagation of states and inputs:

$$\Phi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}, \quad \Gamma = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B & A^{N_p-2}B & \cdots & B \end{bmatrix} \tag{17a}$$

where N_p is the prediction horizon, and the lower-triangular structure of Γ reflects the causal influence of inputs on future states (Rawlings et al., 2017). Similarly, the output prediction matrices Ψ and Θ are constructed from the output equation in (16):

$$\Psi = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N_p} \end{bmatrix}, \quad \Theta = \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & D \end{bmatrix} \tag{17b}$$

Here, Θ inherits the Toeplitz structure of Γ, mapping the input sequence U to the output trajectory Y, The MPC optimization problem aims to minimize a cost function that balances output tracking and control effort as shown in Eq. (18):

$$J = \sum_{i=1}^{N_p} \| y[k+i] - r[k+i] \|_Q^2 + \sum_{i=1}^{N_c} \| u[k+i-1] \|_R^2 \tag{18}$$

Here $r[k+i]$ is the reference trajectory, Q is the output weighting matrix, and R is the control weighting matrix. Substituting the predictive model $(Y = \Psi x[k] + \Theta)$ into the cost function gives:

$$J = (\Psi x[k] + \Theta U - R)^\top Q (\Psi x[k] + \Theta U - R) + U^\top R U \tag{19}$$

This can be expressed in quadratic form as:

$$J = \frac{1}{2} U^\top H U + f^\top U + \text{constant terms}, \tag{20}$$

The Hessian matrix H and gradient vector f are defined as:

$$\begin{aligned} H &= 2(\Theta^\top Q \Theta + R) \\ f &= 2\Theta^\top Q (\Psi x[k] - R) \end{aligned} \tag{21}$$

Here, $H \in \mathbb{R}^{mN_c \times mN_c}$ is the Hessian matrix and $f \in \mathbb{R}^{mN_c}$. is the gradient vector with respect to the stacked control sequence U. Constraints on inputs, outputs, and states ensure system feasibility:

$$\begin{aligned} &u_{\min} \leq u[k+i] \leq u_{\max}, \ \forall i \in [0, N_c - 1], \\ &y_{\min} \leq y[k+i] \leq y_{\max}, \ \forall i \in [1, N_p], \\ &x_{\min} \leq x[k+i] \leq x_{\max}, \ \forall i \in [1, N_p]. \end{aligned} \tag{22}$$

These are reformulated compactly as $GU \leq W$, where G and W encapsulate the constraints over the prediction and control horizons. Bringing together the objective and constraints, the MPC design problem is formulated as the following constrained QP problem:

$$\begin{aligned} &\min_{U \in \mathbb{R}^{mN_c}} \quad J(U) = \frac{1}{2} U^\top H U + f^\top U \\ &\text{subject to} \quad GU \leq W \end{aligned} \tag{23}$$

where $U = [u[k]^\top, u[k+1]^\top, \ldots, u[k+N_c-1]^\top]^\top$, is the stacked control input vector over the control horizon. The cost matrices, H and f are defined in Eq. (21). The constraint matrices $G \in \mathbb{R}^{n_c \times mN_c}$ and $W \in \mathbb{R}^{n_c}$ are constructed by stacking the inequality constraints from Eq. (22), projected using the prediction model in Eq. (17).

Stability is important in predictive control, particularly for nonlinear reactors. Classical MPC typically relies on terminal ingredients (Rawlings et al., 2017), while our hybrid scheme achieves practical stability through the prediction horizon, constraint handling, and the structure of the cost. Input and state limits prevent divergence, and the PSO search—assisted by NN-based gradient estimates—guides the solution toward feasible control moves. The simulations in Sections 4.1 and 4.2 show that the closed-loop response stays bounded under nominal and disturbed conditions. The NN–PSO combination

also improves optimization efficiency by pairing fast gradient surrogates with global search behavior.

3.2 Neural Network for Gradient Approximation

The first component of the hybrid strategy uses a neural network (NN) to approximate the gradient of the MPC cost. Computing this gradient analytically or through numerical differentiation can be expensive for nonlinear or high-dimensional systems, especially in real time. The NN provides a faster surrogate by learning the mapping between $(x[k], r[k], U)$ and the corresponding gradient $\partial J / \partial U$ (Amos et al., 2016). For reference, the gradient of the cost with respect to the control sequence and the state is defined in Eq. (24):

$$\nabla J(u,x) = \left[\nabla_u J(u,x) \quad \nabla_x J(u,x)\right] \tag{24}$$

where U is the stacked input vector and $x[k]$ the current state. The gradient indicates how variations in the decision variables influence the cost function, which is the core mechanism used in optimization updates (Boyd and Vandenberghe, 2004). When analytical differentiation is difficult, numerical approximations such as finite differences may be used, as in Eq. (25):

$$\frac{\partial J}{\partial u} \approx \frac{J(u+\epsilon, x) - J(u,x)}{\epsilon} \tag{25}$$

These methods, although valid (Nocedal and Wright, 2006), can be computationally demanding for online MPC, motivating the use of NN-based gradient approximation.

Neural Network-Based Gradient Approximation

To overcome the computational challenges of traditional gradient computation, a neural network is employed to approximate the gradient of the MPC cost function $\nabla_U J$. The gradient is analytically defined as:

$$\nabla_U J = HU + f \tag{26}$$

The NN is trained as follows:

- *Data Generation:* A dataset is created by solving the QP problem for a range of scenarios, using various system states $x[k]$, reference trajectories $r[k]$, and constraints G and W. For each case, the gradient $\nabla_U J$ is computed using the analytical expression $HU + f$.
- *Network Architecture:* The architecture of the neural network is illustrated in Figure 2, the input to the neural network consists of the system state $x[k]$, the reference trajectory $r[k]$, and the current control input guess U_0. The output of the neural network is the predicted gradient, which approximates the true gradient.
- *Training Objective:* The neural network is trained by minimizing the error between the predicted gradient $\hat{\nabla}_U J$ and the true gradient $\nabla_U J$, typically using the mean squared error (MSE) loss function:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N} \| \hat{\nabla}_U J - \nabla_U J \|^2 \tag{27}$$

The initial guess U_0 is included as part of the neural network input and represents the current candidate control sequence. It is typically initialized using a feasible prior input or the optimizer's last solution and provides context for the gradient approximation.

After sufficient training, the neural network is capable of providing accurate approximations of the gradient in real-time, significantly reducing the computational burden of repeatedly solving for gradients using traditional methods (Chen et al., 2019).

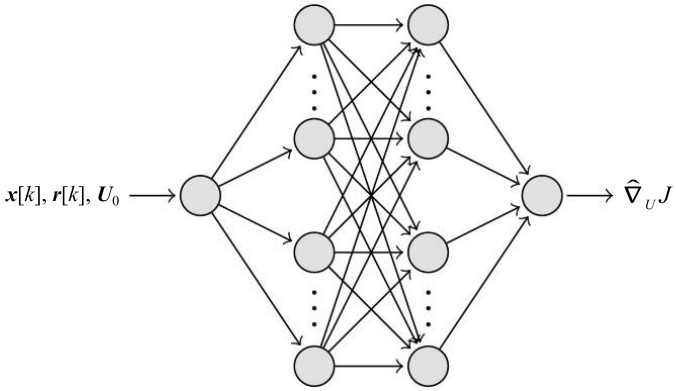

Fig. 2. Schematic of the Neural Network for Gradient Approximation.

3.3 Particle Swarm Optimization (PSO) with Adaptive Mechanisms

To enhance robustness and avoid the limitations of deterministic solvers, the MPC problem (Eq. 23) is also addressed using PSO. At each sampling instant *k*, the decision vector is

$$U = [\Delta u_k^\top \; \Delta u_{k+1}^\top \ldots \Delta u_{k+N_c-1}^\top]^\top \in \mathbb{R}^{mN_c}$$

PSO is applied as an alternative solver for the quadratic program in Eq. (23). Each particle encodes a candidate U, and the fitness is the penalized cost

$$\begin{aligned} &\min_U \tfrac{1}{2}U^\top HU + f^\top U + \rho \left\| [GU - W]_+ \right\|_2^2 \\ &\text{s.t.} \quad \Delta U_{\min} \le U \le \Delta U_{\max}. \end{aligned} \tag{23a}$$

with penalty weight $\rho \ge 0$, After convergence to U^*, the receding-horizon control applies the first m elements,

$$\Delta u_k = (U^*)_{1:m}, \qquad u[k] = \text{sat}\left(u[k-1] + \Delta u_k\right).$$

The velocity update follows (28)–(29) with the NN-gradient pull. A swarm of *P* particles, each particle *i* having an initial position $U_i^{(0)}$ and velocity $V_i^{(0)}$. At each iteration *t*, positions and velocities are updated based on the particle's personal best P_i, the global best G, and the NN-approximated gradient. The velocity of particle *i* is updated as in Eq. (28):

$$V_i^{(t+1)} = \omega V_i^{(t)} + c_1 r_1 (P_i - U_i^{(t)}) + c_2 r_2 (G - U_i^{(t)}) - \alpha \nabla_U J, \tag{28}$$

Where ω is the inertia weight, The coefficients c_1 and c_2 are the cognitive and social factors. The random numbers r_1 and r_2 are uniformly distributed in the interval [0,1]; $\nabla_U J$ is the

gradient term approximated by the neural network, which guides the particles toward promising regions of the solution space; and α is a scaling factor controlling the influence of the gradient term.

After updating the velocity, the position of each particle is updated according to:

$$U_i^{(t+1)} = U_i^{(t)} + V_i^{(t+1)}. \tag{29}$$

Swarm diversity helps balance exploration and exploitation in population-based optimization. Premature convergence to local minima often arises from insufficient diversity, while excessive dispersion impedes efficient refinement. To address this challenge, we propose a dynamic adaptation mechanism governed by a mathematically rigorous diversity metric.

The swarm diversity at iteration t is quantified as the normalized average Euclidean distance between particles:

$$\text{Diversity}(t) = \frac{1}{N_p\sqrt{D}} \sum_{i=1}^{N_p} \sqrt{\sum_{j=1}^{D} \left(U_{ij}^{(t)} - \bar{U}_j^{(t)}\right)^2}, \tag{30}$$

$$\bar{U}j^{(t)} = \frac{1}{N_p} \sum_{i=1}^{N_p} U_{ij}^{(t)} \tag{31}$$

where N_p denotes the number of particles, D is the dimensionality of the control sequence, and Eq. 31 represents the mean position of the swarm in dimension j. The normalization factor $\sqrt{D}$ ensures $\text{Diversity}(t) \in [0,1]$ for hypercube search spaces, where values near 1 indicate global exploration and values near 0 signal local convergence. The inertia weight $\omega(t)$ and gradient scaling factor $\alpha(t)$ are dynamically adjusted to respond to swarm diversity:

$$\omega(t) = \omega_{\max} - (\omega_{\max} - \omega_{\min}) \cdot \frac{\text{Diversity}(t)}{\text{Diversity}_{\max}} \tag{32}$$

$$\alpha(t) = \alpha_{\max} \cdot \left(1 - \frac{\text{Diversity}(t)}{\text{Diversity}_{\max}}\right) \tag{33}$$

where $\text{Diversity}_{\max}$ is the maximum observed diversity during initialization. When diversity is high ($\text{Diversity}(t) \approx \text{Diversity}_{\max}$), the inertia weight $\omega(t)$ approaches $\omega_{\max}$, reducing particle momentum to encourage exploration, while the gradient scaling factor $\alpha(t)$ approaches zero, prioritizing stochastic swarm search. Conversely, low diversity ($\text{Diversity}(t) \approx 0$) drives $\omega(t)$ toward $\omega_{\max}$, preserving particle momentum for local refinement, and $\alpha(t)$ toward $\alpha_{\max}$, activating gradient descent for accelerated convergence. This adaptation mechanism aligns with principles from dynamic swarm optimization (Eberhart and Kennedy, 1995), hybridizing stochastic exploration with deterministic exploitation to mitigate local minima traps (Shi and Eberhart, 1998). To ensure that all particles remain within the feasible set defined by the constraints $GU \leq W$, constraint handling techniques such as penalty methods or projection techniques are employed. These methods ensure that the updated positions of the particles respect the system's operational limits, maintaining physical and practical feasibility (Coello Coello, 2002).

3.4 Hybrid Neural Network-PSO Strategy with Adaptive Mechanisms

The proposed hybrid approach combines a neural network (NN) for real-time gradient approximation with PSO to iteratively solve the constrained QP problem. The NN replaces explicit gradient evaluations with a fast feedforward inference, thereby reducing computational complexity during online execution. PSO explores the decision space by leveraging the approximate gradient to guide its population updates toward optimal solutions within the feasible region. The control policy follows a receding horizon strategy: at each sampling time k, the optimizer computes an optimal control input sequence U, from which only the first input $u[k]$ is applied to the system. The optimization is then repeated at the next time step with updated state measurements. Although an explicit analytical control law is not derived, the controller effectively operates as a state-feedback policy $u[k] = \pi(x[k], r[k])$, where $\pi(\cdot)$ denotes the implicit closed-loop mapping defined by the NN-PSO optimization process. The workflow of the hybrid strategy is summarized in Algorithm 1.

In the full closed-loop simulation, Algorithm 1 is executed at each sampling instant. At time step k, the plant state $x[k]$ and reference $r[k]$ are measured and passed to the controller, which invokes the hybrid NNGA-PSO solver to compute the optimal control sequence U^*. Only the first control input $u[k]$ is applied to the system. The plant model is then simulated forward by one sampling period, and the process repeats at the next time step with updated state measurements. This receding horizon strategy ensures that the controller continuously adapts to the system's evolution and enforces constraints.

Algorithm 1 Hybrid MPC Solver with NNGA & Adaptive PSO

Require: $x[k]$, $r[k]$, constraints (22)
Ensure: Optimal control sequence $U*$

Offline:
1. Train NN to approximate $\nabla_U J$ x using (27)

Online:
1. Initialize swarm within (22) bounds
2. Initialize P_i, G with best particle
while not converged **do**
  for each particle i **do**
    $\hat{\nabla}_U J \leftarrow \text{NN}(x[k], r[k], U_i^{(t)})$
    Update velocity via (28)
    Update position via (29)
    Enforce constraints (22)
    Update P_i if improved
  end for
  Update G
  Adapt ω, α using (32-33)
end while
return $U* \leftarrow G$

4. SIMULATION AND RESULTS

4.1 Performance Evaluation

The goal of the simulation was to evaluate the performance of the proposed hybrid NNGA-PSO framework in achieving a smooth and rapid transition from the first operating point (OP1) to the second operating point (OP2), as detailed in Table 2. This transition required adjusting the monomer feed ratio

(F_a /F_b) from 0.2 to 0.45 while maintaining F_b constant, alongside changes in production rate (G_{pi}), polymer composition (Y_{ap}), and molecular weight (M_{pw}) (Özkan et al. 2003). The dynamics of the copolymerization reactor, previously discussed in Section 2, were central to modeling the process and defining the inputs and outputs used in the simulation. Simulations begin from the steady-state corresponding to OP1, with the initial condition $x[0] = [23.35$ kg/h, 0.56, 3.4x10^4 kg/mol, 353.06K]. which is also used to initialize the MPC prediction model at time $k = 0$ and reflects nominal operating conditions prior to the control-induced transition toward OP2. The reference outputs were fixed at the OP2 values throughout the simulation, while the cost function was evaluated dynamically over the closed-loop trajectory from OP1 to OP2.

Table 2. Operating Conditions for the Process.

Parameters	OP1	OP2
G_{pi} (kg/h)	23.35	24.9
Y_{ap}	0.56	0.64
M_{pw} (10^4 kg/kmol)	3.4	3.9
T_r (K)	353.06	353.3

The simulations were conducted in MATLAB, with the MPC framework designed to achieve specific control objectives, including minimizing tracking errors (evaluated using MAE, RMSE, IAE, and ISE), ensuring rapid settling time, and avoiding overshoot. The framework's performance was compared across three optimization strategies: the conventional QP solver, standalone PSO, and the hybrid NNGA-PSO approach, with computation time and control accuracy as key performance indicators.

The MPC strategy employed a prediction horizon (N_p) of 13 steps and a control horizon (N_c) of 5 steps, with a sampling time (T_s) of 1 hour. Boundary conditions and input constraints were incorporated into the MPC framework to ensure realistic operation of the reactor. The input constraints were:

- Monomer A feed flow rate (F_a): 12 kg/h $\le F_a \le$ 24 kg/h
- Monomer B feed flow rate (F_b): 80 kg/h $\le F_b \le$ 94 kg/h
- Transfer agent feed flow rate (F_t): 0.2 kg/h $\le F_t \le$ 5 kg/h
- Reactor jacket temperature (T_j): 330 K $\le T_j \le$ 340 K

The neural network used to approximate the cost gradient $\nabla_U J$, (see Section 3.2) featured a feedforward architecture with two hidden layers (16 and 8 neurons), ReLU activation functions, and a linear output layer. Inputs were the current state vector x[k], the reference $r[k]$, and a candidate control input sequence U; the output was the estimated gradient vector. The training dataset was generated offline using analytical gradients computed from Eq. (26) for randomized scenarios across the operating range. The network was trained for 8000 epochs using the Adam optimizer with a learning rate of 0.001 and batch size of 64, minimizing the MSE loss. The data were split 80% for training, 10% for validation, and 10% for testing.

PSO was used at each control step to solve the MPC optimization problem defined in Section 3.1, using the neural network–based gradient approximation described in Section 3.2. The swarm was configured with 30 particles, cognitive and social coefficients (c_1=1.9, c_2=1.9), an inertia weight (ω=0.5) and a maximum of 50 iterations.

To assess the effectiveness of the proposed framework, a comparative study was conducted between:

1. The conventional Quadratic Programming (QP) solver used in standard MPC.
2. A PSO-based optimization approach for solving the MPC cost function.
3. The NNGA-PSO hybrid strategy.

The numerical results summarized in Table 3 provide a quantitative basis for comparing the three optimization strategies. These values highlight differences in tracking accuracy, settling time, and overshoot, which are further discussed below in relation to the time-domain responses.

We compare three solvers that address the same MPC objective under identical input constraints: a conventional QP baseline, a standalone PSO optimizer, and the proposed NNGA-PSO (PSO guided by a neural gradient). Evaluation uses OP2 as the reference and computes all metrics from the OP1/OP2 step onward, matching the conventions used in

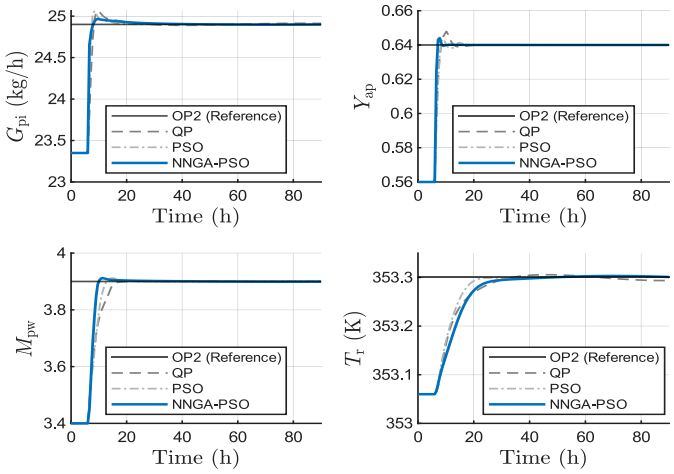

Fig. 3. Time evolution of controlled outputs (G_{pi}, Y_{ap}, M_{pw}, T_r) under nominal transition from OP1 to OP2 using QP, PSO, and NNGA-PSO.

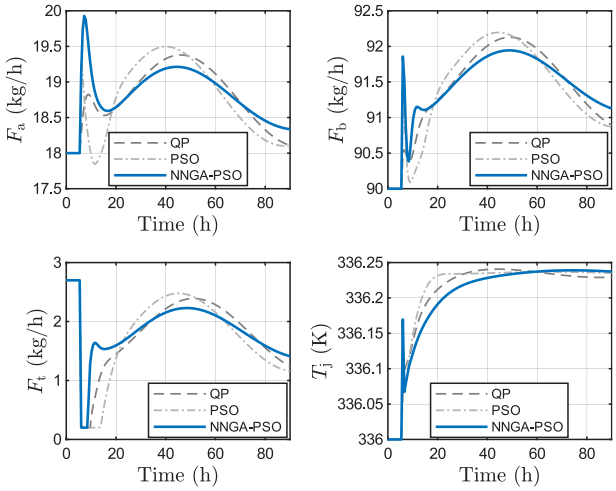

Fig. 4. Control input trajectories (F_a, F_b, F_t, T_j) under nominal conditions using QP, PSO, and NNGA-PSO.

Table 3. Performance Metrics Operating Conditions.

Method	Output	MAE	RMSE	IAE	ISE	Settling Time (h)	Overshoot (%)
Conventional QP	G_{pi} (kg/h)	0.0418	0.1699	3.0843	1.7218	7.29	0.663
	Y_{ap}	0.0013	0.0079	0.0927	0.0034	7.52	1.239
	M_{pw} (10^4 kg/kmol)	0.0198	0.0751	1.5327	0.4027	11.96	0.026
	T_r (K)	0.0228	0.0529	1.8626	0.2204	16.83	0.001
Standalone PSO	G_{pi} (kg/h)	0.0251	0.1463	1.6607	1.0901	6.82	0.639
	Y_{ap}	0.0013	0.0080	0.0864	0.0038	7.54	0.632
	M_{pw} (10^4 kg/kmol)	0.0180	0.0737	1.3740	0.3848	10.45	0.303
	T_r (K)	0.0168	0.0493	1.3563	0.1889	18.7	0.001
Hybrid NNGA-PSO	G_{pi} (kg/h)	0.0241	0.1337	1.5791	0.7918	6.48	0.268
	Y_{ap}	0.0007	0.0068	0.0395	0.0020	6.69	0.543
	M_{pw} (10^4 kg/kmol)	0.0134	0.0665	0.9854	0.2996	8.79	0.301
	T_r (K)	0.0231	0.0564	1.8882	0.2520	20.8	0.001

Table 3 and the trajectories shown in Figs. 3–4. On the controlled outputs (Fig. 3), NNGA-PSO attains equal or lower MAE/RMSE on Y_{ap} and M_{pw} relative to QP and standalone PSO, while remaining comparable on G_{pi} and T_r. Overshoot remains small and of the same order across methods, and the hybrid search does not introduce additional oscillations. Settling-time trends are consistent with the error metrics: NNGA-PSO improves or matches the baselines on Y_{ap} / M_{pw} and shows similar performance on G_{pi} and T_r. For the manipulated inputs (Fig. 4), the hybrid method produces smooth, constraint-respecting trajectories, indicating that the gradient-guided swarm maintains feasible input behavior without introducing irregular oscillations.

With respect to computational cost, Table 4 summarizes the average time required per control step. As anticipated, NNGA-PSO requires more evaluations than the conventional QP solver, yet its runtime remains within a practical range for the selected hourly sampling interval. Standalone PSO is the most computationally intensive, reflecting the absence of gradient guidance.

Table 4. Average Computation Time per Control Step.

Method	Average Computation Time (seconds/step)
Conventional QP	0.0234
Standalone PSO	0.5889
Hybrid NNGA-PSO	0.1065

Overall, under nominal conditions the hybrid NNGA-PSO achieves superior accuracy for key polymer quality indicators (Y_{ap}, M_{pw}), preserves smooth and feasible input behavior, and offers a favorable balance between control performance and computational effort. These results establish it as an effective alternative to both conventional QP and standalone PSO approaches.

4.2 Robustness to Unmeasured Disturbances

To assess robustness, we injected an unmeasured inhibitor in the fresh feed (4 parts per 1000, mole basis) over 1.5–3.0 h during the OP1/OP2 transition (Abbas et al., 2015; Özkan et al., 2003). This disturbance represents a realistic scenario where unexpected impurities enter the reactor, potentially affecting the polymerization process. All three solvers optimize the same MPC objective under identical input constraints. Evaluation follows the same conventions as in the nominal case: OP2 is the reference, and all metrics in Table 5 are computed from the step onward.

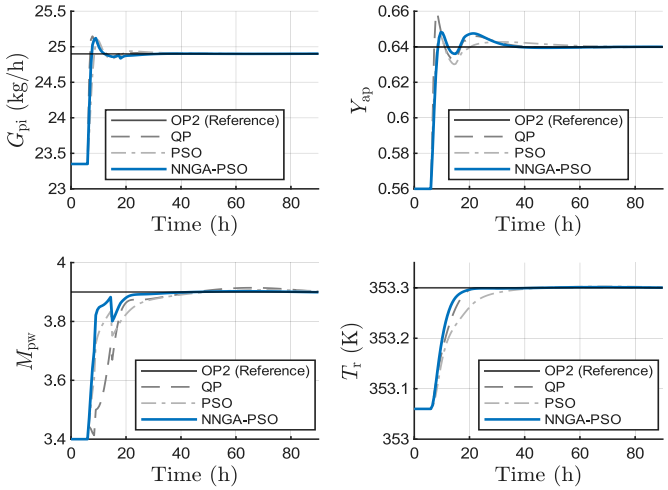

Fig. 5. Time evolution of controlled outputs (G_{pi}, Y_{ap}, M_{pw}, T_r) under unmeasured inhibitor disturbance using QP, PSO, and NNGA-PSO.

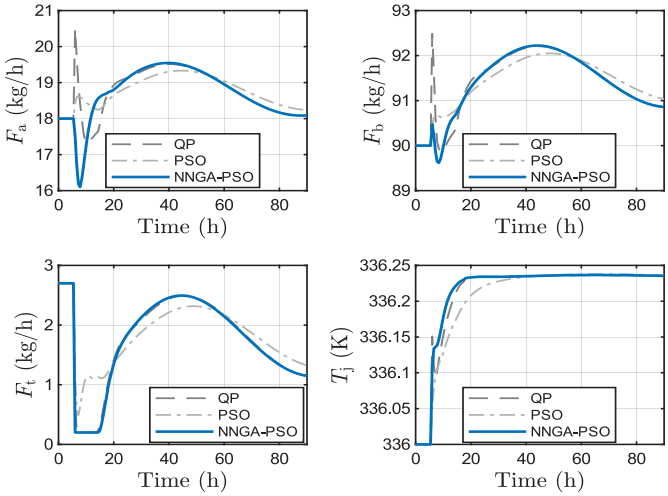

Fig. 6. Control input responses (F_a, F_b, F_t, T_j) during disturbance rejection using QP, PSO, and NNGA-PSO.

Table 5. Performance Metrics Operating Conditions (Under Disturbance).

Method	Output	MAE	RMSE	IAE	ISE	Settling Time (h)	Overshoot (%)
Conventional QP	G_{pi} (kg/h)	0.0271	0.1392	1.8338	0.9194	6.50	0.998
	Y_{ap}	0.0027	0.0084	0.2038	0.0041	9.10	2.925
	M_{pw} (10^4 kg/kmol)	0.0565	0.1302	4.6369	1.3603	17.7	0.369
	T_r (K)	0.0158	0.0480	1.2700	0.1781	21.8	0.003
Standalone PSO	G_{pi} (kg/h)	0.0414	0.1775	3.0387	1.9454	7.47	0.809
	Y_{ap}	0.0027	0.0086	0.2068	0.0043	7.66	0.968
	M_{pw} (10^4 kg/kmol)	0.0389	0.0881	3.1434	0.5825	19.8	0.192
	T_r (K)	0.0238	0.0553	1.9457	0.2416	40.7	0.001
Hybrid NNGA-PSO	G_{pi} (kg/h)	0.0300	0.1515	2.0775	1.2222	6.89	0.891
	Y_{ap}	0.0028	0.0094	0.2140	0.0055	7.95	1.279
	M_{pw} (10^4 kg/kmol)	0.0211	0.0710	1.6421	0.3526	16.0	0.064
	T_r (K)	0.0137	0.0450	1.0906	0.1542	21.7	0.001

As shown in Fig. 5–6 and summarized in Table 5, the hybrid NNGA-PSO maintains stable disturbance rejection while preserving feasibility. On the controlled outputs (Fig. 5), NNGA-PSO achieves equal or lower MAE/RMSE on Y_{ap} and M_{pw} relative to QP and standalone PSO, and remains comparable on G_{pi} and T_r; overshoot stays small and of the same order across methods. Settling-time trends are consistent with the error metrics: NNGA-PSO improves or matches the baselines on Y_{ap}/M_{pw} and is broadly comparable on G_{pi} and T_r. On the manipulated inputs (Fig. 6), NNGA-PSO produces smooth, constraint-respecting trajectories, indicating that feasible input behavior is preserved under perturbations.

Regarding computation, Table 6 reports the average Computation Time per control move under disturbance. As expected, QP remains the fastest, standalone PSO the heaviest, and NNGA-PSO sits in between. Despite the additional particle evaluations, NNGA-PSO retains practical runtimes while delivering competitive tracking quality under unmeasured inputs.

Table 6. Average Computation Time per Control Step (Under Disturbance).

Method	Average Computation Time (seconds/step)
Conventional QP	0.0251
Standalone PSO	0.6942
Hybrid NNGA-PSO	0.1596

In summary, the disturbance study demonstrates that NNGA-PSO preserves closed-loop stability and robustness against unexpected feed impurities. It balances accuracy and feasibility with manageable computational cost, reinforcing its suitability as a reliable nonlinear MPC solver for sensitive copolymerization processes.

5. CONCLUSIONS

We presented a hybrid MPC strategy that couples a neural network–based gradient surrogate with an adaptive PSO search to address the nonconvex, MIMO nature of the copolymerization reactor. The NN provides fast directional information, while diversity-adapted PSO explores feasible sequences under the same objective and identical input constraints. In simulations using OP2 as the reference and post-step evaluation, the hybrid matched or improved MAE/RMSE on Y_{ap} and M_{pw} and showed similar performance on G_{pi} and T_r, with low overshoot, preserved constraint satisfaction, and practical runtimes. Under an unmeasured inhibitor disturbance, the method maintained stable disturbance rejection and preserved constraint feasibility under perturbations. These results suggest that gradient-guided swarm search is a viable path to reliable, real-time nonlinear MPC on sensitive thermal/kinetic processes. For instance, (Wen and Wang, 2024) proposed a Lyapunov-based cascade control strategy for unstable nonlinear systems, which aligns in spirit with the objectives of our predictive framework in terms of handling nonlinearities and ensuring closed-loop performance. Future work may incorporate terminal constraints or Lyapunov-based ingredients to provide formal guarantees of closed-loop stability, especially under uncertainty or in the presence of model mismatch.

REFERENCES

Abbas, Hossam S., Sandy Rahme, Nader Meskin, Christian Hoffmann, Roland Tóth, and Javad Mohammadpour. 2015. “Linear Parameter-Varying Control of a Copolymerization Reactor.” *IFAC-PapersOnLine* 48(26):200–206. doi: 10.1016/j.ifacol.2015.11.137.

Amos, Brandon, Lei Xu, and J. Zico Kolter. 2016. “Input Convex Neural Networks.”

Boyd, Stephen P., and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge, UK ; New York: Cambridge University Press.

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2019. “Neural Ordinary Differential Equations.”

Chen, Show-An, and Wen-Fong Jeng. 1978. “Minimum End Time Policies for Batchwise Radical Chain Polymerization.” *Chemical Engineering Science* 33(6):735–43. doi: 10.1016/0009-2509(78)80051-7.

Coello Coello, Carlos A. 2002. “Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art.” *Computer Methods in Applied Mechanics and Engineering* 191(11–12):1245–87. doi: 10.1016/S0045-7825(01)00323-1.

Congalidis, John P., John R. Richards, and W. Harmon Ray. 1989. “Feedforward and Feedback Control of a Solution

Copolymerization Reactor." *AIChE Journal* 35(6):891–907. doi: 10.1002/aic.690350603.

Eberhart, R., and J. Kennedy. 1995. "A New Optimizer Using Particle Swarm Theory." Pp. 39–43 in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Nagoya, Japan: IEEE.

Farber, Jorge N., and Robert L. Laurence. 1986. "THE MINIMUM TIME PROBLEM IN BATCH RADICAL POLYMERIZATION: A COMPARISON OF TWO POLICIES." *Chemical Engineering Communications* 46(4–6):347–64. doi: 10.1080/00986448608911416.

Hicks, James, Amar Mohan, and W. Harmon Ray. 1969. "The Optimal Control of Polymerisation Reactors." *The Canadian Journal of Chemical Engineering* 47(6):590–97. doi: 10.1002/cjce.5450470619.

Lima, Nádson M. N., Lamia Zuniga Linan, Flavio Manenti, Rubens Maciel Filho, Marcelo Embiruçu, and Maria R. Wolf Maciel. 2013. "Novel Two-Steps Optimal Control of Batch Polymerization Reactors and Application to PMMA Production for the Fabrication of Artificial Bone Tissue." Pp. 163–68 in *Computer Aided Chemical Engineering*. Vol. 32. Elsevier.

Maaruf, Muhammad, Mohammed Mohammed Ali, and Fouad M. Al-Sunni. 2023. "Artificial Intelligence-Based Control of Continuous Polymerization Reactor with Input Dead-Zone." *International Journal of Dynamics and Control* 11(3):1153–65. doi: 10.1007/s40435-022-01038-9.

Mahmoud, Magdi S., Muhammad Maaruf, and Sami El-Ferik. 2021. "Neuro-Adaptive Output Feedback Control of the Continuous Polymerization Reactor Subjected to Parametric Uncertainties and External Disturbances." *ISA Transactions* 112:1–11. doi: 10.1016/j.isatra.2020.11.026.

Nocedal, Jorge, and Stephen J. Wright. 2006. *Numerical Optimization*. Second edition. New York, NY: Springer.

Osakada, K., and L. T. Fan. 1970. "Computation of Near-optimal Control Policies for Free-radical Polymerization Reactors." *Journal of Applied Polymer Science* 14(12):3065–82. doi: 10.1002/app.1970.070141210.

Özkan, Leyla, and Mayuresh V. Kothare. 2006. "Stability Analysis of a Multi-Model Predictive Control Algorithm with Application to Control of Chemical Reactors." *Journal of Process Control* 16(2):81–90. doi: 10.1016/j.jprocont.2005.06.013.

Özkan, Leyla, Mayuresh V. Kothare, and Christos Georgakis. 2003. "Control of a Solution Copolymerization Reactor Using Multi-Model Predictive Control." *Chemical Engineering Science* 58(7):1207–21. doi: 10.1016/S0009-2509(02)00559-6.

Rahme, Sandy, Hossam S. Abbas, Nader Meskin, Roland Toth, and Javad Mohammadpour. 2015. "Reduced LPV Model Development and Control of a Solution Copolymerization Reactor." Pp. 1044–50 in *2015 IEEE Conference on Control Applications (CCA)*. Sydney, Australia: IEEE.

Rawlings, James Blake, David Q. Mayne, and Moritz Diehl. 2017. *Model Predictive Control: Theory, Computation, and Design*. 2nd edition. Madison, Wisconsin: Nob Hill Publishing.

Sacks, Martin E., Soo-Il Lee, and Joseph A. Biesenberger. 1972. "Optimal Policies for Batch, Chain Addition Polymerizations." *Chemical Engineering Science* 27(12):2281–89. doi: 10.1016/0009-2509(72)85105-4.

Shahrokhi, N., and M. R. Pishvaie. 1998. "Artificial Neural Network Feedforward/Feedback Control of a Batch Polymerization Reactor." Pp. 3391–95 vol.6 in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*. Philadelphia, PA, USA: IEEE.

Shi, Y., and R. Eberhart. 1998. "A Modified Particle Swarm Optimizer." Pp. 69–73 in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. Anchorage, AK, USA: IEEE.

Singh, Vikas, and Hariprasad Kodamana. 2020. "Reinforcement Learning Based Control of Batch Polymerisation Processes." *IFAC-PapersOnLine* 53(1):667–72. doi: 10.1016/j.ifacol.2020.06.111.

Tian, Y., J. Zhang, and J. Morris. 2004. "Dynamic On-Line Reoptimization Control of a Batch MMA Polymerization Reactor Using Hybrid Neural Network Models." *Chemical Engineering & Technology* 27(9):1030–38. doi: 10.1002/ceat.200402068.

Tian, Yuan, Jie Zhang, and Julian Morris. 2001. "Modeling and Optimal Control of a Batch Polymerization Reactor Using a Hybrid Stacked Recurrent Neural Network Model." *Industrial & Engineering Chemistry Research* 40(21):4525–35. doi: 10.1021/ie0010565.

Tsoukas, Athanasios, Matthew Tirrell, and George Stephanopoulos. 1982. "Multiobjective Dynamic Optimization of Semibatch Copolymerization Reactors." *Chemical Engineering Science* 37(12):1785–95. doi: 10.1016/0009-2509(82)80050-X.

Vosough, R., M. Rafizadeh, and R. Solgi. 2004. "Adaptive Fuzzy Control of MMA Batch Polymerization Reactor Based on Fuzzy Trajectory Definition." Pp. 1097–1102 vol.2 in *Proceedings of the 2004 American Control Conference*. Boston, MA, USA: IEEE.

Zhang, Jie. 2004. "A Reliable Neural Network Model Based Optimal Control Strategy for a Batch Polymerization Reactor." *Industrial & Engineering Chemistry Research* 43(4):1030–38. doi: 10.1021/ie034136s.

Wen, Jie, and Fangling Wang. 2024. "Stable Levitation of Single-point Levitation Systems for Maglev Trains by Improved Cascade Control." Romanian Journal of Information Science and Technology 27(3–4):348–361. doi: 10.59277/ROMJIST.2024.3-4.08.

Appendix A. FIRST APPENDIX

The nonlinear functions described are adapted from the work of (Abbas et al., 2015), which builds upon the model initially proposed by (Congalidis et al., 1989). The nonlinear functions φ in the dynamic model described in Section 2 are as follows:

$$\varphi_l = \frac{R_k}{C_k} \tag{A.1}$$

$$\varphi_7 = \left(-\Delta H_{\text{paa}}\right) k_{\text{paa}} C_a^{\cdot} + \left(-\Delta H_{\text{pba}}\right) k_{\text{pba}} C_b^{\cdot} \tag{A.2}$$

$$\varphi_8 = \left(-\Delta H_{\text{pab}}\right) k_{\text{pab}} C_a^{\cdot} + \left(-\Delta H_{\text{pbb}}\right) k_{\text{pbb}} C_b^{\cdot} \tag{A.3}$$

$$\varphi_9 = \frac{k_{\text{caa}} \left(\psi_0^{\text{a}\cdot}\right)^2 + k_{\text{cab}} \psi_0^{\text{a}\cdot} \psi_0^{\text{b}\cdot} + L_1 \psi_0^{\text{a}\cdot}}{2C_a} \tag{A.4}$$

$$\varphi_{10} = \frac{k_{\text{cbb}} \left(\psi_0^{\text{b}\cdot}\right)^2 + L_2 \psi_0^{\text{b}} \cdot}{2C_b} \tag{A.5}$$

$$\varphi_{11} = \frac{k_{\text{caa}} \psi_0^{\text{a}\cdot} \psi_1^{\text{a}\cdot} + k_{\text{cab}} \left(\psi_0^{\text{b}\cdot} \psi_1^{\text{a}\cdot}\right) + L_1 \psi_1^{\text{a}\cdot}}{C_a} \tag{A.6}$$

$$\varphi_{12} = \frac{k_{\text{cab}} \left(\psi_0^{\text{a}\cdot} \psi_1^{\text{b}\cdot}\right) + k_{\text{cbb}} \psi_0^{\text{b}\cdot} \psi_1^{\text{b}\cdot} + L_2 \psi_1^{\text{b}}}{C_b} \tag{A.7}$$

$$\varphi_{13} = \frac{k_{\text{caa}} \left(\left(\psi_1^{\text{a}}\right)^2 + \psi_0^{\text{a}\cdot} \psi_2^{\text{a}\cdot}\right) + k_{\text{cab}} \left(2\psi_1^{\text{a}\cdot} \psi_1^{\text{b}\cdot} + \psi_2^{\text{b}} \psi_0^{\text{a}\cdot}\right) + L_1 \psi_2^{\text{a}\cdot}}{C_a}$$

(A.8)

$$\varphi_{14} = \frac{k_{\text{cab}} \left(\psi_2^{\text{a}\cdot} \psi_0^{\text{b}\cdot}\right) + k_{\text{cbb}} \left(\left(\psi_1^{\text{b}\cdot}\right)^2 + \psi_0^{\text{b}\cdot} \psi_2^{\text{b}\cdot}\right) + L_2 \psi_2^{\text{b}\cdot}}{C_b} \tag{A.9}$$

$$\varphi_{15} = \frac{1}{\lambda_a + \lambda_b} \tag{A.10}$$

$$\varphi_{16} = \frac{1}{\psi_1} \tag{A.11}$$

The term R_k represents the reaction rates, which are specified in Equations (4–9) of (Congalidis et al., 1989). The definitions for the variables $C_a^{\cdot}$, $C_b^{\cdot}$, $\psi_0^{a\bullet}$, $\psi_1^{a\bullet}$, $\psi_2^{a\bullet}$, $\psi_0^{b\bullet}$, $\psi_1^{b\bullet}$ and $\psi_2^{b\bullet}$ are provided in Equations (11), (12), and (31–36) of the same reference. along with the full parameters