Input Convex LSTM for Fast Machine Learning-Based Optimization

Zihao Wang[1] Donghan Yu [1] Zhe Wu [1]

[1]Department of Chemical and Biomolecular Engineering, National University of Singapore

Motivation

- Traditional model-based optimization and control rely on the development of first-principles models, a process that is resource-intensive
- Neural network-based optimization suffers from slow computation times, limiting its applicability in real-time tasks
- Computational efficiency is a critical parameter for real-world and real-time implementation of neural network-based optimization
- The optima of convex optimization problems are easier and faster to obtain than those of non-convex optimization problems

Objective: Proposes an Input Convex Long Short-Term Memory (ICLSTM) neural network for real-time optimization

Full paper and code: Interested readers can search "*Input Convex LSTM*" on Google for the full paper (titled "*Real-Time Machine-Learning-Based Optimization Using Input Convex Long Short-Term Memory Network*") and on GitHub for the source code.

Problem Formulation

- Nonlinear dynamic systems:

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}) \tag{1}$$

  - $\mathbf{x} \in \mathbb{R}^{n_x}$: the state vector
  - $\mathbf{u} \in \mathbb{R}^{n_u}$: the manipulated input
  - $F : D \times U \to \mathbb{R}^{n_x}$: a $\mathcal{C}^1$ function, where $D \subset \mathbb{R}^{n_x}$ and $U \subset \mathbb{R}^{n_u}$ are compact and connected subsets that contain an open neighborhood of the origin, respectively

- Neural network-based optimization (also termed Model Predictive Control (MPC), Fig. 1a):

$$\mathcal{L} = \min_{\mathbf{u} \in S(\Delta)} \int_{t_k}^{t_{k+N}} J(\widetilde{\mathbf{x}}(t), \mathbf{u}(t)) dt \tag{2a}$$

$$\text{s.t. } \dot{\widetilde{\mathbf{x}}}(t) = F_{nn}(\widetilde{\mathbf{x}}(t), \mathbf{u}(t)) \tag{2b}$$

$$\mathbf{u}(t) \in U, \ \forall t \in [t_k, t_{k+N}) \tag{2c}$$

$$\widetilde{\mathbf{x}}(t_k) = \mathbf{x}(t_k) \tag{2d}$$

  - $\widetilde{\mathbf{x}}$: the predicted state trajectory
  - $S(\Delta)$: the set of piecewise constant functions with sampling period Δ
  - N: the number of sampling periods in the prediction horizon
  - The objective function $\mathcal{L}$ in Eq. (2a) incorporates a cost function J in terms of the system states $\mathbf{x}$ and the control actions $\mathbf{u}$
  - The dynamic function $F_{nn}(\widetilde{\mathbf{x}}(t), \mathbf{u}(t))$ in Eq. (2b) is parameterized as neural networks
  - Eq. (2c) is the constraint function U on feasible control actions
  - Eq. (2d) defines the initial condition $\widetilde{\mathbf{x}}(t_k)$ of Eq. (2b), which is the state measurement at $t = t_k$

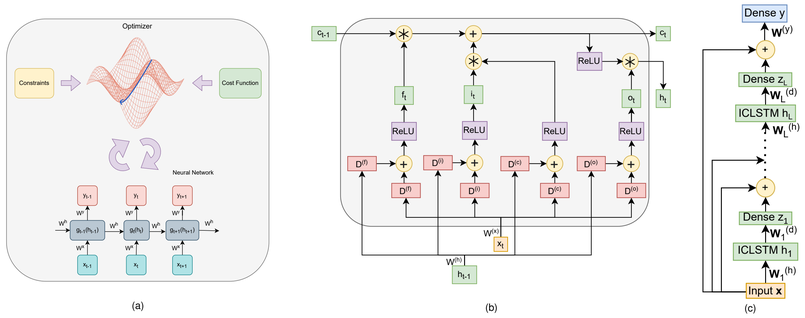

Figure 1 - Architecture diagrams. **(a)** Neural network-based optimization. **(b)** ICLSTM cell. **(c)** L-layer ICLSTM.

Input Convex LSTM

- The output of the ICLSTM cell (Fig. 1b):

$$\mathbf{f}_t = g^{(g)}[\mathbf{D}^{(f)}(\mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\hat{\mathbf{x}}_t) + \mathbf{b}^{(f)}] \tag{3a}$$

$$\mathbf{i}_t = g^{(g)}[\mathbf{D}^{(i)}(\mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\hat{\mathbf{x}}_t) + \mathbf{b}^{(i)}] \tag{3b}$$

$$\mathbf{o}_t = g^{(g)}[\mathbf{D}^{(o)}(\mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\hat{\mathbf{x}}_t) + \mathbf{b}^{(o)}] \tag{3c}$$

$$\widetilde{\mathbf{c}}_t = g^{(c)}[\mathbf{D}^{(c)}(\mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\hat{\mathbf{x}}_t) + \mathbf{b}^{(c)}] \tag{3d}$$

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \widetilde{\mathbf{c}}_t \tag{3e}$$

$$\mathbf{h}_t = \mathbf{o}_t * g^{(c)}(\mathbf{c}_t) \tag{3f}$$

  - $\mathbf{f}$, $\mathbf{i}$, $\mathbf{o}$, $\mathbf{c}$, $\mathbf{h}$: the forget gate, the input gate, the output gate, the cell gate, the hidden state
  - $*$: element-wise multiplication (i.e., Hadamard product)
  - $\mathbf{D}^{(f)}, \mathbf{D}^{(i)}, \mathbf{D}^{(o)}, \mathbf{D}^{(c)} \in \mathbb{R}^{n_h \times n_h}$: diagonal **scaling** matrices with non-negative entries
  - $\mathbf{W}^{(h)} \in \mathbb{R}^{n_h \times n_h}$ and $\mathbf{W}^{(x)} \in \mathbb{R}^{n_h \times n_i}$: non-negative weights (i.e., **sharing weights** across all gates)
  - $\mathbf{b}^{(f)}, \mathbf{b}^{(i)}, \mathbf{b}^{(o)}, \mathbf{b}^{(c)} \in \mathbb{R}^{n_h}$: the bias
  - $g^{(g)}$ and $g^{(c)}$: convex, non-negative, and non-decreasing activation function
  - $\hat{\mathbf{x}}_t = \left[\mathbf{x}_t^\top, -\mathbf{x}_t^\top\right]^\top \in \mathbb{R}^{2n_x}$: the expanded input

- The output of L-layer ICLSTM (Fig. 1c):

$$\mathbf{z}_{t,l} = g^{(d)}[\mathbf{W}_l^{(d)}\mathbf{h}_{t,l} + \mathbf{b}_l^{(d)}] + \hat{\mathbf{x}}_t, \quad l = 1, 2, \ldots, L \tag{4a}$$

$$\mathbf{y}_t = g^{(y)}[\mathbf{W}^{(y)}\mathbf{z}_{t,L} + \mathbf{b}^{(y)}] \tag{4b}$$

  - $\mathbf{W}_l^{(d)} \in \mathbb{R}^{n_i \times n_h}$ and $\mathbf{W}_l^{(y)} \in \mathbb{R}^{n_o \times n_i}$: the non-negative weights
  - $\mathbf{b}_l^{(d)} \in \mathbb{R}^{n_i}$ and $\mathbf{b}^{(y)} \in \mathbb{R}^{n_o}$: the bias
  - $g^{(d)}$: convex, non-negative, and non-decreasing activation function
  - $g^{(y)}$: convex and non-decreasing activation function

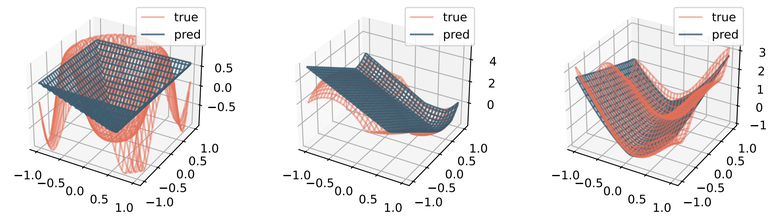

Figure 2 - Surface fitting of non-convex bivariate scalar functions. **(a)** $f_1(x, y) = -\cos(4x^2 + 4y^2)$. **(b)** $f_2(x, y) = \max(\min(x^2 + y^2, (2x-1)^2 + (2y-1)^2 - 2), -(2x+1)^2 - (2y+1)^2 + 4)$. **(c)** $f_3(x, y) = x^2(4 - 2.1x^2 + x^4) - 4y^2(1 - y^2) + xy$.

Theory: preservation of convexity

Theorem 1: Consider the L-layer IC-LSTM as shown in Fig. 1(c). Each element of the output $\mathbf{y}_t$ is a convex, non-decreasing function of the input $\hat{\mathbf{x}}_\tau = \left[\mathbf{x}_\tau^\top, -\mathbf{x}_\tau^\top\right]^\top$ (or just $\mathbf{x}_\tau$) at the time step $\tau = t, t-1, \ldots, 1$, for all $\hat{\mathbf{x}}_\tau \in D \times D$ in a convex feasible space if all of the following conditions are met: (1) All weights are non-negative; (2) All activation functions are convex, non-decreasing, and non-negative (e.g., ReLU), except for the activation function of the output layer which is convex and non-decreasing (e.g., ReLU, Linear, LogSoftmax).

Theorem 2: Consider the neural network-based convex MPC problem in Eq. (2). The problem remains input convex in the face of multi-step ahead prediction (i.e., when the prediction horizon $N > 1$) if and only if the neural network embedded is inherently input convex and non-decreasing (e.g., IC-LSTM) and the neural network output is non-negative for certain objective functions (e.g., quadratic functions and absolute functions).

Application to a CSTR example

- CSTR with second-order, exothermic, irreversible reaction of the form $A \to B$:

$$\frac{dC_A}{dt} = \frac{F}{V_L}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT}} C_A^2 \tag{5a}$$

$$\frac{dT}{dt} = \frac{F}{V_L}(T_0 - T) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT}} C_A^2 + \frac{Q}{\rho_L C_p V_L} \tag{5b}$$

- Control objective: Operates the CSTR at the unstable equilibrium point (C_{As}, T_s) by manipulating $\Delta C_{A0} = C_{A0} - C_{A0_s}$ and $\Delta Q = Q - Q_s$, using Eq. (2) with an additional Lyapunov-Based constraint (i.e., $V(\widetilde{\mathbf{x}}(t)) < V(\mathbf{x}(t_k))$, if $\mathbf{x}(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}}$, $\forall t \in [t_k, t_{k+N})$), and finally reaches the steady state
- ICLSTM-based MPC achieved convergence in 15 initial conditions (i.e., it achieved the fastest convergence in 13 out of 15 different initial conditions), with an average percentage decrease in computational time of 54.4%, 40.0%, and 41.3% compared to plain RNN, plain LSTM, and ICRNN, respectively

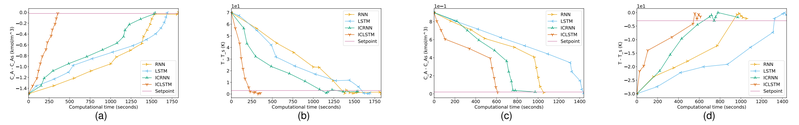

Figure 3 - State trajectories of NN-based MPC with respect to computational time (i.e., the program will terminate immediately upon system convergence). **(a)** $C_A - C_{As}$ vs. computational time for initial condition of (-1.5, 70). **(b)** $T - T_s$ vs. computational time for initial condition of (-1.5, 70). **(c)** $C_A - C_{As}$ vs. computational time for initial condition of (0.9, -30). **(d)** $T - T_s$ vs. computational time for initial condition of (0.9, -30).

Application to a solar PV energy system

- The solar PV system's dynamics are governed by the following ODEs:

$$\frac{dv_{pv}}{dt} = \frac{1}{C}(i_{pv} - i_s u), \ \frac{di_s}{dt} = \frac{1}{L}(-v_b + v_{pv} u), \ \frac{dv_c}{dt} = \frac{1}{C_b}(i_s - i_L) \tag{6}$$

- Control objective: Operates the system using Eq. (2) to finds a sequence of optimal control actions $u_1^*, \ldots, u_N^*$ that maximize energy outputs of the solar system to meet the demands, where only the first value u_1^* is applied to the system, enabling a close tracking of the load
- ICLSTM-based MPC enjoys a faster (at least $\times 4$) solving time compared to LSTM (i.e., for a scaled-up solar PV energy system or a longer prediction horizon, the time discrepancy will be even greater)

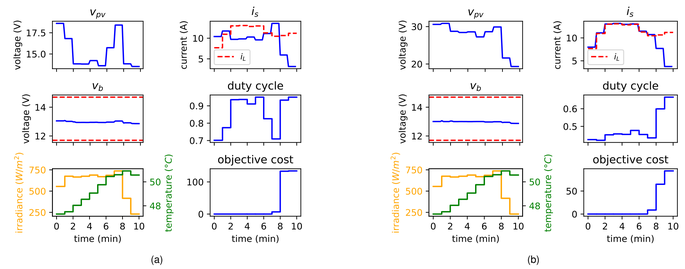

Figure 4 - Comparisons of MPC performances using ICLSTM and LSTM with real irradiance and temperature data on May 5, 2024, 10:00 a.m. at LHT Holdings. Dashed lines represent the demand currents i_L in the top right subplots; the upper and lower bounds of v_b in the middle left subplots. **(a)** ICLSTM. **(b)** LSTM.

Acknowledgments

- This study was supported by A*STAR MTC YIRG 2022 Grant (222K3024) and MOE AcRF Tier 1 FRC Grant (22-5367-A0001).