Physics Informed Neural Networks for Optimal Intercept Problem

1Enrico Schiassi,
2Andrea D’Ambrosio, 1Roberto Furfaro, and 2Fabio Curti,

1University of Arizona, USA
2Sapienza University of Rome

IAA/AAS SciTech Forum 2020, Dec. 08-10, 2020,
Moscow, Russia
Contents

• Introduction
  • Overview and Motivations
  • Goals

• Background
  • Physics-Informed Neural Networks and Optimal Control Problems
  • New Methods for Solving DEs

• Extreme Theory of Functional Connections
  • X-TFC approach to solving generic DEs
  • X-TFC approach to solving generic OCPs
  • ELM algorithm

• Problems and Results
  • Feldbaum Problem
  • Minimum Time-Energy Optimal Intercept

• Conclusions and Outlooks
• Optimal intercept problems represent one of the most useful optimization problem in aerospace engineering.

• Optimal intercept problems are mainly related to missile guidance.
  • It is important to have robust algorithms suitable for real-time applications.
Introduction: Goals

• To employ *Physics-Informed Neural Networks (PINN)* to solve Optimal Control Problems (OPCs).

• To develop a new algorithm, suitable for on-board application, based on the newly developed *Physics-Informed Extreme Theory of Functional Connections (X-TFC)* [Schiassi et al. 2020] to compute optimal trajectories for intercept problems.

  • The focus of this talk is to show the effectiveness of our X-TFC based algorithm in solving optimal control problems, focusing particularly to the solution of the optimal intercept problem.
PINN and OCPs

- (Data) + Neural Networks + Physics Laws = Physics-Informed Neural Networks (PINN)
- PINNs are a newly developed framework for solving DEs
  - The physics laws (modeled via DEs), and eventually data, drive the training of the network

\[ f \left( \mathbf{x}, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}, \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots; \lambda \right) = 0, \quad \mathbf{x} \in \Omega, \quad B(u, \mathbf{x}) = 0 \text{ on } \partial \Omega. \]

- Optimal Control Problems (OCPs) are generally hard and computationally expensive. In general, Open-Loop solutions can be found in two ways
  - **Direct Method**: Transform a continuous problem in a finite NLP problems and find the minimum
  - **Indirect Method**: Apply Pontryagin Minimum Principle (PMP) to derive the necessary conditions
    - The solution of the OCP reduces to the solution of a Two Point Boundary Value Problem (TPBVP) that is a systems of ODEs

The Theory of Functional Connections (TFC) [Mortari, 2017] is a recently developed framework for functional interpolation:

- The functions are approximated via a constrained expression:
  - Sum of a free-chosen function and a functional that analytically satisfies the constraints
- TFC is applied to solve parametric DEs:
  - The free-chosen function is an expansion of Chebyshev polynomials
  - The constraints are the Initial/Boundary Conditions (IC or BC)
- The Physics-Informed Neural Network (PINN) Methods are a novel approach, coming from the Machine Learning community:
  - The DE latent solutions are approximated via a (Deep) Neural Network (NN), and the DEs drive the NN training (i.e. it acts as regulator)

Pros | Cons
--- | ---
**TFC** (w/Cheb. Pol.) | **ICs/BCs always analytically satisfied**
- Accurate Solutions
- Low Computational Time
- Non-linear problems can be very sensitive to the initial guesses
- It suffers of the curse of dimensionality when solving ODE/PDE problems

**PINN** | **Many training points required for high accuracy (ICs/BCs not analytically satisfied)**
- Expanding the latent solution via NN allows to apply this method to solve high order PDEs (e.g. no curse of dimensionality)
- Computational expensive when gradient-based methods are used to train the NN

New Methods for Solving DEs
New Methods for Solving DEs (cont’d)

- The Physics-Informed **Extreme Theory of Functional Connections** (X-TFC) is a synergy of the TFC and the standard PINN methods that helps to overcome their limitations for solving DEs
  - X-TFC uses the TFC constrained expression where the free-chosen function is a Single Layer Feedforward NN (SLNN) trained via Extreme Learning Machine (ELM) Algorithm [Huang et al., 2006]
X-TFC approach to solving generic DEs

1) Approximate the Latent Solution(s) with the CE
2) Analytically Satisfy the ICs/BCs
3) Expand with NN
4) Substitute into the (S)DEs
5) Build the Loss(es)
6) Train the NN
7) Build the Approximate Solution(s)
X-TFC approach to solving generic OCPs

1. Optimal Control Problem
   \[ \mathcal{J} = \Phi(x(t_0), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}(x(t), u(t), t) \, dt \]
   Meyer Cost
   \[ \dot{x} = f(x(t), u(t), t) \]
   Lagrange Cost
   \[ \Phi(x(t_0), t_0) = \Phi_0 \]
   \[ \Phi(x(t_f), t_f) = \Phi_f \]

2. Expand with a SLNN

3. Analytically Satisfy the BCs
   \[
   \begin{align*}
   y_j(t_0) &= y_{0j} \\
   y_j(t_f) &= y_{fj} \\
   \dot{y}_j(t_0) &= \dot{y}_{0j} \\
   \dot{y}_j(t_f) &= \dot{y}_{fj}
   \end{align*}
   \]

4. Substitute into

5. Build the Losses
   \[ \mathcal{L}_i = F_i(t, y_j(t), \dot{y}_j(t), y_j(t)) = 0 \]

6. Build the Approximate Solution

7. Train the NN via (Iterative) Least-Squares

8. Build the Approximate Solution

TPBVP
ELM algorithm

- ELM is a training algorithm for SLNN that randomly selects input weights and bias, and computes the output weights via least-square
  - Input weights and bias are not tuned during the training
- The convergence of the ELM algorithm is proved by Huang et al. [2006]
  - The convergence is guaranteed for any input weights and bias randomly chosen to any continuous probability distribution
The Feldbaum Problem is a generic OCP that we have chosen as it has analytical solution. This allowed us to perform sensitivity analysis to check the accuracy and the robustness of the proposed physics-informed algorithm. The OCP is posed as following:

\[ J = \frac{1}{2} \int_0^1 (f^2 + u^2) \, dt \]

subject to

\[ \frac{df}{dt} = -f + u \quad 0 \leq t \leq 1 \]

\[ f(0) = 1 \]

Applying the PMP the TPBVP that we will solve via X-TFC is:

\[
\begin{align*}
\dot{f} &= \frac{\partial H}{\partial \lambda} = -f - \lambda \\
\dot{\lambda} &= -\frac{\partial H}{\partial x} = \lambda - f \\
\end{align*}
\]

s.t. \( f(0) = f_0 = 1 \) \( \lambda(1) = \lambda_f = 0 \) (transversality condition)

The CE and their derivatives are:

\[
\begin{align*}
\dot{f} &= (\mathbf{\sigma} - \Omega_1 \mathbf{h}_0)^T \mathbf{\beta}_f + \Omega_1 f_0 \\
\dot{\lambda} &= (\mathbf{\sigma} - \Omega_1 \mathbf{h}_f)^T \mathbf{\beta}_\lambda + \Omega_1 \lambda_f \\
\end{align*}
\]

The unknowns and the losses are:

\[ \mathbf{\beta} = \{\mathbf{\beta}_f \quad \mathbf{\beta}_\lambda\}^T \]

\[
\begin{align*}
\mathcal{L}_f &= \dot{f} + f + \lambda \\
\mathcal{L}_\lambda &= \dot{\lambda} - \lambda + f \\
\end{align*}
\]
Time evolution of the state and control

Sensitivity analysis: fixed $L = 100$ (left), fixed $N = 100$ (right)

| $N$ | $L$ | CPU time [s] | $\text{mean}(L)$ | $|J - J^*|$ |
|-----|-----|--------------|------------------|----------|
| 10  | 50  | 0.0007       | $4.9 \times 10^{-16}$ | $5.5 \times 10^{-2}$ |
| 10  | 90  | 0.0008       | $5.1 \times 10^{-16}$ | $6.5 \times 10^{-3}$ |
| 20  | 90  | 0.0008       | $5.6 \times 10^{-16}$ | $1.5 \times 10^{-3}$ |
| 50  | 90  | 0.001        | $5.6 \times 10^{-16}$ | $7.3 \times 10^{-8}$ |
| 200 | 90  | 0.004        | $8.3 \times 10^{-15}$ | $1.3 \times 10^{-9}$ |
| 500 | 90  | 0.006        | $8.7 \times 10^{-14}$ | $1.3 \times 10^{-9}$ |
| 100 | 20  | 0.0009       | $1.9 \times 10^{-1}$  | $1.7 \times 10^{-2}$ |
| 100 | 50  | 0.003        | $1.5 \times 10^{-6}$  | $8.9 \times 10^{-9}$ |
| 100 | 100 | 0.004        | $1.4 \times 10^{-15}$ | $1.3 \times 10^{-9}$ |
| 100 | 150 | 0.005        | $8.1 \times 10^{-15}$ | $1.3 \times 10^{-9}$ |

Activation Function: Gaussian
Input weights and bias sampled from: unif [-10,10]
The minimum time-energy optimal intercept problem is posed as following:

\[
\min J = \Gamma t_f + \frac{1}{2} \int_{t_0}^{t_f} (a_M a_M) \, dt
\]

subject to

\[
\begin{align*}
\dot{r} &= v \\
\dot{v} &= a_T - a_M \\
t_0 &\leq t \leq t_f \\
r(t_0) &= r_0 \\
v(t_0) &= v_0 \\
r(t_f) &= 0
\end{align*}
\]

where \( r \) and \( v \) are the relative position and velocity vectors between the target and the interceptor, \( a_T \) and \( a_M \) are the commanded acceleration of the target and the interceptor, respectively.

Applying the PMP (plus transversality conditions), we get the following TPBVP:

\[
\begin{align*}
\dot{r} &= \frac{\partial H}{\partial \lambda_r} = v \\
\dot{v} &= -\frac{\partial H}{\partial \lambda_v} = a_T - \lambda_v \\
\dot{\lambda}_r &= -\frac{\partial H}{\partial r} = 0 \\
\dot{\lambda}_v &= -\frac{\partial H}{\partial v} = -\lambda_r \\
H(t_f) + \Gamma &= 0 \\
r(t_0) &= r_0 \\
v(t_0) &= v_0 \\
r(t_f) &= 0 \\
\lambda_v(t_f) &= 0
\end{align*}
\]

The TPBVP will be solved via X-TFC.
Minimum Time - Energy Optimal Intercept: formulation (cont’d)

- The CEs and their derivatives are:

\[
\begin{align*}
    r_j &= \left( \sigma - \Omega_1 \sigma_0 - \Omega_2 \sigma_f - \Omega_3 \sigma_0' \right)^T \beta_j + \Omega_1 r_{0j} + \Omega_2 r_{fj} + \frac{\Omega_3 v_0_j}{b^2} \\
    v_j &= b^2 \left[ \left( \sigma' - \Omega_1' \sigma_0 - \Omega_2' \sigma_f - \Omega_3' \sigma_0' \right)^T \beta_j + \Omega_1' r_{0j} + \Omega_2' r_{fj} + \frac{\Omega_3' v_0_j}{b^2} \right] \\
    a_j &= b^4 \left[ \left( \sigma'' - \Omega_1'' \sigma_0 - \Omega_2'' \sigma_f - \Omega_3'' \sigma_0'' \right)^T \beta_j + \Omega_1'' r_{0j} + \Omega_2'' r_{fj} + \frac{\Omega_3'' v_0_j}{b^2} \right] \\
    \lambda_{r,j} &= \sigma^T \beta_{r,j} \\
    \dot{\lambda}_{r,j} &= b^2 \sigma^T \beta_{r,j} \\
    \lambda_{v,j} &= \left( \sigma - \sigma_f \right)^T \beta_{v,j} + \lambda_{v,j} \\
    \dot{\lambda}_{v,j} &= b^2 \sigma^T \beta_{v,j}
\end{align*}
\]

- The unknowns and the losses are:

\[
\begin{align*}
    \Xi &= \{ \beta_{r,1}, \beta_{r,2}, \beta_{r,3}, \beta_{\lambda_r,1}, \beta_{\lambda_r,2}, \beta_{\lambda_r,3}, \beta_{\lambda_v,1}, \beta_{\lambda_v,2}, \beta_{\lambda_v,3} \} \\
    \mathcal{L}_{a,j} &= a_j - aT_{r,j} + \lambda_{v,j} \\
    \mathcal{L}_{\lambda_{r,j}} &= \dot{\lambda}_{r,j} \\
    \mathcal{L}_{\lambda_{v,j}} &= \dot{\lambda}_{v,j} + \lambda_{r,j} \\
    \mathcal{L}_{\lambda_H} &= \sum_{j=1}^{3} (\lambda_{r,j} v_j) + \Gamma \\
    b^2 &= c = \frac{z_f - z_0}{t_f - t_0}
\end{align*}
\]

The $\sigma$ are the activation functions of the SLNN that is trained via ELM, where $\beta$’s are the output weights of the network.

The $\Omega$’s are called switching functions, and their expression can be found in the manuscript.
Performances analysis ($\Gamma=1$)

\[
\begin{align*}
 r_0 &= [500, -600, -500] \text{ m} \\
 v_0 &= [-50, 60, 5] \text{ m/s} \\
 a_T &= [1, -2, 0,1] \text{ m/s}^2 \text{ (assumed constant)}
\end{align*}
\]

We compare our results with GPOPS. The results obtained with GPOPS were the following: $t_f = 45.54$, $H(t_f) + \Gamma = 2.35 \times 10^{-6}$, with a CPU time $\approx 1.48$ [s]
Minimum Time – Energy Optimal Intercept: results (cont’d)

Time evolution of the states and control

\[ L_2 = 1.3 \times 10^{-9} \]

Optimal Trajectories

Activation Function: hyperbolic tangent
Input weights and bias sampled from: unif [-1,1]
Number of points: 20
Number of neurons: 16
Conclusions and Outlooks

- We presented a new algorithm based on the newly developed Physics-Informed X-TFC for solving general OPCs.
  - The physics-informed X-TFC framework is used to solve the TPBVP arising from the application of the PMP.
- The algorithm was tested in designing minimum time – energy optimal intercept trajectories.
  - The CPU time, in order of milliseconds, makes the proposed algorithm suitable for on board applications.
  - The performances are comparable with the state-of-the-art software such as GPOPS II.
- Works are in progress to:
  - Employing the physics-informed X-TFC based algorithm to tackle a wide variety of OPCs (especially OPCs for space guidance, navigation, and control).
  - Use the ability of the X-TFC framework in solving PDEs with high accuracy with a low CPU time to perform real-time computation of closed-loop optimal control via the direct solution of the HJB equation.
Thanks for watching =)