Data Development and Deep Learning for HJB Equations

Wei Kang

Department of Applied Mathematics
Naval Postgraduate School, Monterey, CA

IPAM Workshop
March, 2020

In control system design

- Physics laws, first principle models, empirical models are fundamental in engineering designs.
- Design methods with **guaranteed performance or properties** are invaluable.
  - stability
  - minimized cost
  - bounded $L^2$-gain
  - output regulation and tracking
  - ......
- Control theory and mathematical tools have been developed for decades.
  - Lyapunov function
  - Pontryagin maximum principle
  - Riccati equation
  - HJB equation
  - FBI equation
  - feedback linearization and normal form
  - ......
However

- There are obstacles for which existing analysis and numerical methods have been, in general, ineffective.
  - the curse-of-dimensionality in solving the HJB equation
  - finding Lyapunov function for nonlinear high dimensional systems
  - finding the domain of attraction for nonlinear high dimensional systems
  - finding reachable sets
  - ......
However

- There are obstacles for which existing analysis and numerical methods have been, in general, ineffective.
  - the curse-of-dimensionality in solving the HJB equation
  - finding Lyapunov function for nonlinear high dimensional systems
  - finding the domain of attraction for nonlinear high dimensional systems
  - finding reachable sets
  - ....

**Question:** Can we use deep learning to overcome the bottleneck while preserving the guaranteed performance in control theory?
However

- There are obstacles for which existing analysis and numerical methods have been, in general, ineffective.
  - the curse-of-dimensionality in solving the HJB equation
  - finding Lyapunov function for nonlinear high dimensional systems
  - finding the domain of attraction for nonlinear high dimensional systems
  - finding reachable sets

Question: Can we use deep learning to overcome the bottleneck while preserving the guaranteed performance in control theory?

Such approach must be Data-Driven Approach in Model-Based Design
A problem of optimal control

\[
\begin{align*}
\begin{cases}
\text{minimize} & \int_{t_0}^{t_f} L(t, x, u) dt + \psi(x(t_f)), \\
\text{subject to} & \dot{x}(t) = f(t, x, u), \\
& x(t_0) = x_0.
\end{cases}
\end{align*}
\]

The goal of control design is to find a feedback law

\[ u = u^*(t, x) \]

that minimizes the cost.
Online optimization (e.g. some MPC design).

- Direct methods such as pseudospectral optimal control
- Hopf formula
- Minimization along characteristics
- ......

These methods require online optimization that converges in real-time.
Controller Design based on the HJB equation

Define the Hamiltonian

\[ H(t, x, \lambda, u) = L(t, x, u) + \lambda^T f(t, x, u). \]

Solve the HJB equation to find the optimal feedback law

\[
\begin{cases}
V_t(t, x) + \min_{u \in U} H(t, x, V_x, u) = 0, \\
V(t_f, x) = \psi(x),
\end{cases}
\]

\[ u^*(t, x) = \arg \min_{u \in U} H(t, x, V_x, u). \]

The complexity of solving the HJB equation increases exponentially with dimension - the curse-of-dimensionality.
A data-driven method: Design the feedback based on the HJB equation, which is solved using machine learning.

1. Initial data generation: For supervised learning, a data set must be generated. It contains the value of $V(t, x)$ at random points in a given region.

2. Training: Given this data set, a neural network, $V_{NN}(t, x)$, is trained to approximate the value function.

3. Validation: The accuracy of the trained neural network is checked on a new set of validation data computed at Monte Carlo sample points.

4. Feedback control law:

$$u^*(t, x) = \arg\min_{u \in \mathcal{U}} H(t, x, V_{x}^{NN}, u).$$
An example of optimal attitude control

State variables:
\[
\mathbf{v} = \begin{pmatrix} \phi & \theta & \psi \end{pmatrix}^T \quad \text{Euler angles}
\]
\[
\mathbf{\omega} = \begin{pmatrix} \omega_1 & \omega_2 & \omega_3 \end{pmatrix}^T \quad \text{angular velocity}
\]

Define
\[
E(\mathbf{v}) := \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi / \cos \theta & \cos \phi / \cos \theta \end{pmatrix},
\]
\[
S(\mathbf{\omega}) := \begin{pmatrix} 0 & \omega_3 & -\omega_2 \\
-\omega_3 & 0 & \omega_1 \\
\omega_2 & -\omega_1 & 0 \end{pmatrix},
\]
\[
R(\mathbf{v}) := \begin{pmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\
\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \cos \theta \sin \phi \\
\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi & \cos \theta \cos \phi \end{pmatrix}.
\]
\[
B = \begin{pmatrix} 1 & 1/20 & 1/10 \\
1/15 & 1 & 1/10 \\
1/10 & 1/15 & 1 \end{pmatrix},
\]
\[
J = \begin{pmatrix} 2 & 0 & 0 \\
0 & 3 & 0 \\
0 & 0 & 4 \end{pmatrix}, \quad h = \begin{pmatrix} 1 \\
1 \\
1 \end{pmatrix}.
\]
The optimal control problem is

$$\begin{align*}
\text{minimize} & \quad \int_{t}^{t_f} L(v, \omega, u) d\tau + \frac{W_4}{2} \|v(t_f)\|^2 + \frac{W_5}{2} \|\omega(t_f)\|^2, \\
\text{subject to} & \quad \dot{v} = E(v)\omega, \\
& \quad J\dot{\omega} = S(\omega)R(v)h + Bu.
\end{align*}$$

(1)

Here

$$L(v, \omega, u) = \frac{W_1}{2} \|v\|^2 + \frac{W_2}{2} \|\omega\|^2 + \frac{W_3}{2} \|u\|^2,$$

and

$$W_1 = 1, \quad W_2 = 10, \quad W_3 = \frac{1}{2}, \quad W_4 = 1, \quad W_5 = 1, \quad t_f = 20.$$
Domain in State Space

\[ \mathcal{X}_0 = \left\{ \boldsymbol{v}, \boldsymbol{\omega} \in \mathbb{R}^3 \mid -\frac{\pi}{3} \leq \phi, \theta, \psi \leq \frac{\pi}{3} \text{ and } -\frac{\pi}{4} \leq \omega_1, \omega_2, \omega_3 \leq \frac{\pi}{4} \right\}, \]

Initial Data Set: Solving TPBVP using time-marching

\[ \boldsymbol{x}^{(i)} = (\boldsymbol{v}^{(i)}, \boldsymbol{\omega}^{(i)}) \]

\[ V(0, \boldsymbol{x}^{(i)}), \quad \text{for } i = 1, 2, \ldots, N_d = 64. \]

\[ \lambda^{(i)}(0), \]

Loss function:

\[ \mathcal{L} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left[ V^{(i)} - V^{NN}(t^{(i)}, \boldsymbol{x}^{(i)}; \theta) \right]^2 + \frac{\mu}{N_d} \sum_{i=1}^{N_d} \| \lambda^{(i)} - V^{NN}_x(t^{(i)}, \boldsymbol{x}^{(i)}; \theta) \|_2^2, \]
First Neural Network: Train a neural network,

$$V_{NN}(0, x) \approx V(0, x), \quad 3 \text{ hidden layers, 64 neurons}$$

Additional data set: Warm start using $V_{NN}(0, x)$ to speed up data generation.

Validation: Generating more data for accuracy verification.
One more reason of generating data - adaptive sampling

Closed-loop: sample trajectories
**DATA IS ESSENTIAL!**

Causality-free algorithms are ideal for the purpose of generating data.

- An algorithm is causality-free if the value of $V(t, x)$ is computed without using the value of $V$ at any other points.
- A causality-free algorithm does not rely on grids. It avoids the curse-of-dimensionality.
- A causality-free is convenient for generating data in targeted regions, such as in adaptive deep learning.
- Causality-free algorithms do not propagate computational error over a region.
- Many algorithms provide error estimation.
- Causality-free algorithms have perfect parallelism.
- Data generated can be used for both training and validation.
Methods of generating data

• Characteristic methods
  ◇ Time-marching and space-marching
  ◇ Neural network warm start
  ◇ Backward propagation

• Minimization-based methods
  ◇ The Hopf formula
  ◇ Minimization along characteristics

• Direct methods

• Stochastic process
A problem of optimal control

\[
\begin{align*}
\text{minimize} \quad & \int_{t_0}^{t_f} L(t, x, u) \, dt + \psi(x(t_f)), \\
\text{subject to} \quad & \dot{x}(t) = f(t, x, u), \\
& x(t_0) = x_0.
\end{align*}
\]

The goal of control design is to find a feedback law

\[
u = u^*(t, x)
\]

that minimizes the cost.
The Pontryagin Maximum Principle

Optimal Control

\[ u^*(t, x) = \arg \min_{u \in U} H(t, x, V_x, u) \]

Two Point Boundary Value Problem (TPBVP)

\[
\begin{align*}
\dot{x}(t) &= \frac{\partial H}{\partial \lambda} = f(t, x, u^*(t, x, \lambda)), & x(0) = x_0, \\
\dot{\lambda}(t) &= -\frac{\partial H}{\partial x}(t, x, \lambda, u^*(t, x, \lambda)), & \lambda(t_f) = \frac{\partial \psi}{\partial x}(t_f), \\
\dot{V}(t) &= L(t, x, u^*(t, x, \lambda)), & V(t_f) = \psi(x(t_f)).
\end{align*}
\]

The problem may diverge depending on the initial guess.
**Time-marching**

1. Choose a time sequence,
   \[ t_0 < t_1 < t_2 < \cdots < t_K = t_f, \]

2. In \([t_0, t_1]\), solve the TPBVP. If \(t_1\) is small, it converges
   \[(x^1(t), \lambda^1(t)).\]
Extending the trajectory to \([t_0, t_2]\).

\[
x^2_0(t) = \begin{cases} 
x^1(t), & \text{if } t_0 \leq t \leq t_1, \\
x^1(t_1), & \text{if } t_1 < t \leq t_2,
\end{cases}
\]

Or

\[
x^2_0(t) = x^1\left(t_0 + \frac{t_1 - t_0}{t_2 - t_0}(t - t_0)\right), \quad \text{for } t_0 \leq t \leq t_2.
\]

\(\lambda^2_0(t)\) is similarly defined.

\((x^2_0(t), \lambda^2_0(t))\) is used as initial guess to solve the TPBVP over \([0, t_2]\).

\[\ldots\]

Repeating the process until \(t_K = t_f\).
Remarks about time-marching.

- Does not need initial guess.
- Similar idea can be applied to space-marching.
- It is causality-free (does not need a grid, embarrassingly parallel, ...).
- In [1]-[2], the BVP is solved using bvp5c in Matlab based on a four-stage Lobatto IIIa method (Kierzenka-Shampine 2008).
- It can be combined with other algorithms to increase marching step size, e.g. Albrekht’s Method (Krener).
- It can be slow.


Neural network warm start

1. Generate a first data set. It needs algorithms independent of good initial state such as time-marching.
2. Train a neural network $V^{NN}(t, x)$.
3. Generate more data using warm start

$$\lambda_0(t) = V^{NN}_x(t, x).$$

**Example:** Rigid body optimal attitude control

### Time-marching method

<table>
<thead>
<tr>
<th>$K$</th>
<th>% BVP convergence</th>
<th>Mean integration time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3%</td>
<td>0.37 s</td>
</tr>
<tr>
<td>2</td>
<td>38.7%</td>
<td>0.44 s</td>
</tr>
<tr>
<td>3</td>
<td>76.2%</td>
<td>0.40 s</td>
</tr>
<tr>
<td>4</td>
<td>92.9%</td>
<td>0.45 s</td>
</tr>
<tr>
<td>8</td>
<td>98.4%</td>
<td>0.53 s</td>
</tr>
</tbody>
</table>

### Neural network warm start

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>% BVP convergence</th>
<th>Mean integration time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>90%</td>
<td>0.44 s</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>99.6%</td>
<td>0.41 s</td>
</tr>
<tr>
<td>$10^1$</td>
<td>100%</td>
<td>0.40 s</td>
</tr>
</tbody>
</table>
Characteristic Methods

Backward propagation

1. Find a nominal trajectory $x^*(t), \lambda^*(t), u^*(t), V^*(t)$ satisfying the PMP.
2. Perturb the final state; generate characteristic curves backward in time

\[
\begin{align*}
\dot{x}(t) &= \frac{\partial H}{\partial \lambda} = f(t, x, u^*(t, x, \lambda)), \\
\dot{\lambda}(t) &= -\frac{\partial H}{\partial x}(t, x, \lambda, u^*(t, x, \lambda)), \\
\dot{V}(t) &= L(t, x, u^*(t, x, \lambda)),
\end{align*}
\]

Then, the data is used to train a neural network

\[V^{NN}(t, x) \approx V(t, x).\]

Some remarks

- It is independent of initial guess.
- It does not have convergence issue. Any ODE solver is applicable.
- The location of initial state cannot be pre-selected.
The Hopf formula

Consider the HJ equation

\[
\begin{cases}
V_t(t,x) + H(V_x(t,x)) = 0, & \text{in } (0, \infty) \times \mathbb{R}^n, \\
V(0,x) = \psi(x), & x \in \mathbb{R}^n,
\end{cases}
\]

\(H : \mathbb{R}^n \to \mathbb{R}\) is continuous and bounded from below by an affine function, \(\psi : \mathbb{R}^n \to \mathbb{R}\) is convex.

Fenchel-Legendre transform: Given \(f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}\)

\[f^*(z) = \sup_{x \in \mathbb{R}^n} \{x^T z - f(x)\}, \quad z \in \mathbb{R}^n\]

The Hopf formula

\[V(t, x) = (\psi^* + tH)^*(x)\]
Some remarks

- In [1], split Bregman iterative approach is used for the numerical evaluation of the Hopf formula. It converges very fast.
- The method has direct application to a special type of problems

\[
\begin{align*}
\dot{x}(s) &= f(u(s)), \\
x(t) &= x, \\
\min J(x, t; u) &= \int_t^T L(u(s))ds + \psi(x(T)), \\
\end{align*}
\]

- It can be extended to another family of systems

\[
\dot{x}(s) = Ax(s) + B(s)u(s), \quad A, B \in \mathbb{R}^{n \times n}.
\]

- The method is applicable to the Eikonal equation.

Minimization along characteristics

\[
\begin{aligned}
\dot{x}(t) &= \frac{\partial H}{\partial \lambda} = f(t, x, u^*(t, x, \lambda)), \quad x(t_0) = x_0, \\
\dot{\lambda}(t) &= -\frac{\partial H}{\partial x}(t, x, \lambda, u^*(t, x, \lambda)), \quad \lambda(t_0) = \lambda_0, \\
& \\
u^*(t, x, \lambda) &= \arg \min_{u \in U} H(t, x, \lambda, u).
\end{aligned}
\]

For fixed initial state \(x_0\), the cost is a function of \(\lambda_0\),

\[
J(t_0, x_0, \lambda_0) = \int_{t_0}^{t_f} L(t, x, u^*(t, x, \lambda)) dt + \psi(t_f).
\]

Then the solution, \(V(t_0, x_0)\), of the HJB equation is

\[
V(t_0, x_0) = \min_{\lambda_0} J(t_0, x_0, \lambda_0).
\]
Some remarks

- The existence and uniqueness of solutions, under convexity assumptions, can be proved (for instance [1] and [2]).
- Algorithms of unconstrained optimization are applicable to minimize the cost. Coordinate descent is used in [1] and Powell’s algorithm is used in [2].
- Some examples in [1] show fast convergence that may justify real-time computation.
- This approach generalizes Lax/Hopf formula [1].


Direct methods of optimal control: Discretize the optimal control problem and solve the resulting finite-dimensional constrained optimization problem.

An incomplete list

A Pseudospectral optimal control

Problem definition:

\[
\begin{aligned}
\min_u J &= \int_{-1}^{1} L(x(t), u(t)) dt + E(x(-1), x(1)), \\
\dot{x} &= f(x, u), \\
e(x(-1), x(1)) &= 0, \\
h(x(t), u(t)) &\leq 0.
\end{aligned}
\]
Discretization

$$\begin{align*}
\min_{\bar{x}_k, u_k} J^N &= \sum_{k=0}^{N} L(\bar{x}_k, \bar{u}_k) w_k + E(\bar{x}_0, \bar{x}_N), \\
\left\| \sum_{i=0}^{N} \bar{x}_i D_{ki} - f(\bar{x}_k, \bar{u}_k) \right\|_{\infty} &\leq (N - 1)^{1.5-m}, \\
\| e(\bar{x}_0, \bar{x}_N) \|_{\infty} &\leq (N - 1)^{1.5-m}, \\
h(\bar{x}_k, \bar{u}_k) &\leq (N - 1)^{1.5-m} \cdot 1.
\end{align*}$$

where $t_k$ are Legendre-Gauss-Lobatto (LGL) nodes, $\bar{x}_k$ approximates $x(t_k)$, $D = [D_{ki}]$ is the differentiation matrix, $w_k$ are the LGL weights for integration.
Feasibility and convergence

Theorem (Feasibility)
Given any feasible solution, \( t \to (x, u) \), of optimal control, suppose \( x(\cdot) \in W^{m, \infty} \) with \( m \geq 2 \). Then, there exists a positive integer \( N_1 \) such that, for any \( N > N_1 \), there exists a feasible trajectory, \((\tilde{x}_k, \tilde{u}_k)\), that satisfy all the discretized constraints.

Theorem (Consistent Approximation)
Let \( \{(\tilde{x}_k^*, \tilde{u}_k^*), 0 \leq k \leq N\}_{N=N_1}^\infty \) be a sequence of optimal solutions to the discretized problem. Let \( \{t \to (x_N(t), u_N(t))\}_{N=N_1}^\infty \) be their interpolating function sequence. Assume \( \{(x_0^*, \dot{x}^N(\cdot), u^N(\cdot))\}_{N=N_1}^\infty \) has a uniform accumulation point with continuous components. Then \( t \to u^\infty(t) \) is the solution of the original optimal control problem.
Semilinear parabolic PDE

\[
\begin{aligned}
V_t(t, x) + \frac{1}{2} \text{Tr}(\sigma \sigma^T \text{Hess}_x V)(t, x) + V_x(t, x)^T \mu(t, x) + \\
H(t, x, V(t, x), \sigma^T(t, x) V_x(t, x)) &= 0, \\
V(t_f, x) &= \psi(x).
\end{aligned}
\]

\(x \in \mathbb{R}^n,\)
\(\sigma(t, x) \in \mathbb{R}^{n \times n}\) is a matrix valued function,
\(\mu(t, x)\) is a vector valued function,
\(\text{Hess}_x V\) is the Hessian of \(V\) with respect to \(x\),
\(\text{Tr}(\cdot)\) is the trace of matrix.


A method of characteristics

1. A forward stochastic differential equation (FSDE)

\[ x(t) = x_0 + \int_0^t \mu(s, x(s)) ds + \int_0^t \sigma(s, x(s)) dW_s, \]

\( W_t, 0 \leq t \leq t_f, \) is an \( n \)-dimensional Brownian motion.

2. A backward stochastic differential equation (BSDE)

\[
\begin{cases}
V(t, x(t)) = V(0, x_0) - \int_0^t H(s, x(s), V(s, x(s)), \sigma(s, x(s))^T V_x(s, x(s))) ds \\
+ \int_0^t V_x(s, x(s))^T \sigma(s, x(s)) dW_s,
\end{cases}
\]

\[ V(t_f, x(t_f)) = \psi(x(t_f)). \]
A method of characteristics (Discretized equations)

1. A forward discrete-time equation

\[
x(t_{n+1}) \approx x(t_n) + \mu(t_n, x(t_n)) \Delta t_n + \sigma(t_n, x(t_n))(W_{t_{n+1}} - W_{t_n}),
\]

for \( n = 0, 1, 2, \cdots, N - 1, \)

2. A backward discrete-time equation

\[
V(t_{n+1}, x(t_{n+1}))
\]

\[
\approx V(t_n, x(t_n)) - H(t_n, x(t_n), V(t_n, x(t_n)), \sigma(t_n, x(t_n))^T V_x(t_n, x(t_n))) \Delta t_n
\]

\[
+ V_x(t_n, x(t_n))^T \sigma(t_n, x(t_n))(W_{t_{n+1}} - W_{t_n}),
\]

1. Replace $x \rightarrow \sigma^T V_x(t_n, x)$ in the backward discrete-time equation by a neural network with unknown parameters, $\theta_n$.

2. Set up a loss function

$$I(\theta) = \mathbb{E} \left( \left\{ \left| \psi(x^{(i)}(t_N)) - \hat{V}(t_N, x^{(i)}(t_N)) \right|^2 \right| \ 1 \leq i \leq N_s \right) .$$

For each $1 \leq i \leq N_s$, $\{x^{(i)}(t_n)\}_{n=0}^{N}$ and $\{V(t_n, x^{(i)}(t_n))\}_{n=0}^{N}$ are trajectories of the discrete-time equations driven by random numbers

$$\left\{ \left\{ W_{t_{n+1}}^i - W_{t_n}^i \right\}_{n=0}^{N-1} \right| \ i = 1, 2, \cdots , N_s \right\}, \quad W_{t_{n+1}}^i - W_{t_n}^i \sim \mathcal{N}(0, t_{n+1} - t_n).$$

3. Training step: finding $V(0, x)$, $V_x(0, x)$ and $\{\theta_n\}_{n=1}^{N-1}$ by minimizing the loss function
<table>
<thead>
<tr>
<th>Method</th>
<th>Initial Guess Dependent</th>
<th>Suggested Solver or Algorithm</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time or space-marching</td>
<td>No</td>
<td>BVP solver</td>
<td>Convergence and speed depends on the number of marching steps.</td>
</tr>
<tr>
<td>NN warm start</td>
<td>Yes</td>
<td>BVP solve</td>
<td>Convergence and speed depends on the quality of NN initial guess.</td>
</tr>
<tr>
<td>Backward propagation</td>
<td>No</td>
<td>ODE solver</td>
<td>No convergence issue. Initial states in data cannot be pre-selected.</td>
</tr>
<tr>
<td>Hopf formula</td>
<td>Yes</td>
<td>Bregman algorithm</td>
<td>It works for special types of control systems.</td>
</tr>
<tr>
<td>Minimization along characteristics</td>
<td>Yes</td>
<td>Powell’s algorithm Coordinate descent</td>
<td>Convergence and speed depends on the quality of initial guess.</td>
</tr>
<tr>
<td>Direct methods</td>
<td>Yes</td>
<td>SQP or nonlinear programming</td>
<td>The method is effective for problems with state-control constraints.</td>
</tr>
<tr>
<td>Stochastic process</td>
<td>Yes</td>
<td>NN training</td>
<td>The method is applicable to stochastic optimal control.</td>
</tr>
</tbody>
</table>
A hierarchy of grids

\[ X^1 = \{0, 1\}, X^2 = \{0, \frac{1}{2}, 1\}, X^3 = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\}, \ldots \]
\[ X^i = \left\{ \frac{k-1}{2i-1}; \ k = 1, 2, \cdots m_i \right\}, \ m_i = 2^{i-1} + 1 \]

The sequence has a telescopic structure

\[ X^1 \subset X^2 \subset X^3 \subset X^4 \subset \cdots \]

Define

\[ \Delta X^1 = X^1, \ \Delta X^i = X^i \setminus X^{i-1}, i \geq 2 \]
\[ \Delta m_i = |\Delta X^i| \]

\[ \Delta X^i \text{ for } i = 1, 2, 3, 4 \]
A hierarchy of grids in $\mathbb{R}$

Vector index

$$i = \left[ i_1 \ i_2 \ \cdots \ i_d \right], \quad |i| = i_1 + i_2 + \cdots + i_d$$

$$j = \left[ j_1 \ j_2 \ \cdots \ j_d \right]$$

$$\Delta X^i = \Delta X^{i_1} \times \cdots \times \Delta X^{i_d}, \quad \Delta m^i = \left[ \Delta m^{i_1} \ \cdots \ \Delta m^{i_d} \right]$$

$$x^i_j = (x_{j_1}^{i_1}, \ \cdots, \ x_{j_d}^{i_d}) \in \Delta X^i$$

The dense grid is

$$X^q \times \cdots \times X^q = \bigcup_{1 \leq i \leq q} \Delta X^i$$

Following Smolyak’s approximation algorithm, the sparse grid is

$$G^q_{\text{sparse}} = \bigcup_{|i| \leq q} \Delta X^i$$
Examples of sparse grids

$G_{\text{sparse}}^q$ in $[0, 1]^2$, $q = 6$ and $q = 8$

A modified sparse grid  \textbf{CGL} (Chebyshev-Gauss-Lobatto) type

$G_{\text{sparse}}^q(\text{Modified})$ ($q = 6$, $q = 8$)  $G_{\text{sparse}}^q(\text{CGL})$ ($q = 6$, $q = 8$)
Sparse Grids

Interpolation

Basis functions

Hierarchical surpluses

\[ I^q(f) = I^{q-1}(f) + \Delta I^q(f), \quad q \geq d \]

\[ \Delta I^q(f) = \sum_{|i|=q} \sum_{1 \leq j \leq \Delta m^i} w^i_j a^i_j \]

\[ w^i_j = f(x^i_j) - I^{q-1}(f)(x^i_j) \]
Sparse Grids

Sparse vs. dense

<table>
<thead>
<tr>
<th>Grid</th>
<th>Grid size</th>
<th>Interpolation error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense</td>
<td>$N^d$</td>
<td>$O\left( \frac{1}{N^2} \right)$</td>
</tr>
<tr>
<td>Sparse</td>
<td>$O\left( N(\log N)^{d-1} \right)$</td>
<td>$O\left( \frac{(\log N)^{d-1}}{N^2} \right)$</td>
</tr>
</tbody>
</table>

Comparison based upon linear interpolation on standard sparse grids for functions with bounded second order derivatives.

**Remark:** Essentially, we pay the price of $(\log N)^{d-1}$ in accuracy in exchange for the reduction of grid size: $O\left( N(\log N)^{d-1} \right)$. 
Error analysis

\[ \tilde{V}(t, x) = V(t, x) + e_{\text{interp}} + e_{\text{BVP}} \]

where \( \tilde{V}(t, x) \) is the approximate value function. Sparse grid with piecewise linear interpolation,

\[ \frac{\|e_{\text{BVP}}\|_{L^\infty}}{\epsilon} = O \left( (\log N)^{d-1} \right). \]

CGL sparse grid with polynomial interpolation,

\[ \frac{\|e_{\text{BVP}}\|_{L^\infty}}{\epsilon} = O \left( (\log N)^{2d-1} \right). \]

where \( \epsilon \) is the error of BVP solver at individual grid points.

**Error analysis** - a practical way

If the TPBVP solution, \( \tilde{V}(t, x) \), has high accuracy, error can be approximated by

\[
|e_{interp} + e_{BVP}| \approx |\tilde{V}(t, x) - \tilde{V}(t, x)|
\]

on a random set of points. The computation has perfect parallelism.

Attitude Control - system and problem definitions are from example 1.

| $q$ | $|G_q^g|_{\text{sparse}}$ CGL | Dense grid size | # of Processors | MAE $N = 1280$ samples |
|-----|-------------------------------|-----------------|-----------------|------------------------|
| $q = 13$ | $44,698$ | $> 10^{12}$ | $512$ | $7.3 \times 10^{-4}$ |

The Euler angle and angular velocity are bounded by $\pm \frac{\pi}{3}$ and $\pm \frac{\pi}{4}$, resp.

The error at 1280 points are computed in parallel using 128 CPU cores. The error tolerance of $\tilde{V}(t, x)$ is $10^{-9}$.

An example of optimal trajectory
Control with saturation $u \leq 1$

| $q$ | $|G^q_{\text{sparse}}|_{\text{CGL}}$ | Dense grid size | # of Processors | MAE $N = 1280$ samples |
|-----|----------------------------------|-----------------|-----------------|-----------------------|
| 13  | 44,698                           | $> 10^{12}$     | 512             | 2.2 e-2               |

Inner-loop error tolerance = 1e-4; final loop tolerance = 1e-9, MAE is computed in parallel using 128 CPU cores.

An example of optimal trajectory
An uncontrollable example - control with two momentum wheels

\[
\min_u \int_0^{t_f} \left( \frac{W_1}{2} ||v - v_e(v, \omega)||^2 + \frac{W_2}{2} ||\omega||^2 + \frac{W_3}{2} ||u||^2 \right) dt
\]

subject to

\[
\dot{v} = E(v) \omega \\
J\dot{\omega} = S(\omega)R(v)H + Bu
\]

Parameters

\[
B = \begin{bmatrix}
1 & \frac{1}{10} \\
0 & 1 \\
\frac{1}{12} & 0
\end{bmatrix}, \quad H = [12 12 6]^T
\]

\[
W_1 = 1, \quad W_2 = 2 \quad W_3 = \frac{1}{2},
\]

\[
J = \text{diag}(2, 3, 4), \quad \frac{\pi}{6} \leq \phi, \theta, \psi \leq \frac{\pi}{6}, \quad -\frac{\pi}{8} \leq \omega_i \leq \frac{\pi}{8}
\]
Equilibrium: \( \mathbf{v} = \mathbf{v}_e(\mathbf{v}, \omega), \ \omega = 0 \)

\[
\begin{align*}
\min_{\mathbf{v}_e} \| R(\mathbf{v}_e) - \mathbf{I} \| \max \\
\text{subject to} \\
C^T R(\mathbf{v}_e) H = C^T (R(\mathbf{v}) H - J\omega)
\end{align*}
\]

where \( C \in \mathbb{R}^3 \) is a constant vector satisfying

\[ C^T B = 0 \]

The equilibrium, \( \mathbf{v}_e \), is computed numerically. The process is equivalent to maximizing trace(\( R(\mathbf{v}_e) \)).
Numerical results \((m = 2)\)

| \(q\) | \(|G^q_{\text{sparse}}|\) | Dense grid size | \# of Processors | MAE |
|---|---|---|---|---|
| 13 | 44,698 \(> 10^{12}\) | 512 | 8.5 e-3 |

\(N = 1280\) samples

Inner-loop error tolerance = 1e-4; final loop tolerance = 1e-9

An example of optimal trajectory
The system model

\[
\begin{align*}
\dot{x}_1 &= V \cos \gamma \cos \Psi, \\
\dot{x}_2 &= V \cos \gamma \sin \Psi, \\
\dot{x}_3 &= -V \sin \gamma \\
\dot{V} &= \frac{1}{m} \left( T - \frac{C_{D0} \rho S}{2m} V^2 - g A_1 n_z - \frac{2mg^2 A_2}{\rho S} \frac{n_z^2}{V^2} - g \sin \gamma \right) \\
\dot{\gamma} &= \frac{g}{V} (n_z \cos \phi - \cos \gamma) \\
\dot{\psi} &= \frac{g}{V \cos \gamma} n_z \sin \phi \\
\dot{\phi} &= u_\phi
\end{align*}
\]

Parameters adopted from foam \textit{Unicorn} wing

- \((x_1, x_2, x_3)\) - location in NED frame
- \(n_z\) - vertical lift
- \(T\) - throttle
- \(\rho\) - air density
- \(S = 0.321 \, \text{m}^2\)
- \(m g = 9.34 \, \text{N}\)
- \(CD_0 = 0.0213\)
- \(A_1 = -0.056\)
- \(A_2 = 0.22\)

The cost functional:

\[
J = \int_0^{t_f} L(V, \gamma, \Psi, \phi, u) \, dt
\]

\[
L(V, \gamma, \Psi, \phi, u) = \frac{W_1}{2} \| V - V^d \|^2 + \frac{W_2}{2} \| \gamma - \gamma^d \|^2 + \frac{W_3}{2} \| \Psi - \Psi^d \|^2
+ \frac{W_4}{2} \| \phi - \phi^d \|^2
+ \frac{W_5}{2} \| T - T^d \|^2 + \frac{W_6}{2} \| n_z - n_z^d \|^2 + \frac{W_7}{2} \| u_{\phi} - u_{\phi}^d \|^2
\]

\( W_i, i = 1, 2, 3, 4, 5, 6, \) are constant weights, \((V^d, \gamma^d, \Psi^d, \phi^d)\) is the desired target state, \((T^d, n_z^d, u_{\phi}^d)\) makes final state an equilibrium.

The parameters

\[
W_1 = \frac{1}{4}, \ W_2 = 1, \ W_3 = 1, \ W_4 = 1, \ W_5 = 0.2, \ W_6 = 0.2, \ W_7 = 0.2
\]
The Hamiltonian

\[ H(V, \gamma, \psi, \phi, u, \lambda) = H_1(V, \gamma, \psi, \phi, \lambda) + A_T T^2 + B_T T + A_{n_z} n_z^2 + B_{n_z} n_z + A_{u_\phi} u_\phi^2 + B_{u_\phi} u_\phi \]

\[ A_T = \frac{W_5}{2} \], \quad B_T = \lambda_1 \alpha_1 - W_5 T^d \]
\[ A_{n_z} = \frac{W_6}{2} - \frac{\lambda_1 \alpha_4}{V^2} \], \quad B_{n_z} = \frac{\lambda_2 g}{V} \cos \phi + \frac{\lambda_3 g}{V \cos \gamma} \sin \phi - \lambda_1 \alpha_3 - n_z^d W_6 \]
\[ A_{u_\phi} = \frac{W_7}{2} \], \quad B_{u_\phi} = \lambda_4 - W_7 u_\phi^d \]
\[ H_1(V, \gamma, \psi, \phi, \lambda) = \text{all other terms of states and co-states} \]

\[ \begin{align*}
A > 0 & \quad \Rightarrow u^* = \begin{cases} 
- \frac{B}{2A}, & u_{\min} < - \frac{B}{2A} < u_{\max} \\
- \frac{B}{2A} \leq u_{\min} & u_{\min}, \\
- \frac{B}{2A} > u_{\max}, & u_{\max}, \\
\end{cases} \\
A < 0 & \quad \Rightarrow u^* = \begin{cases} 
\frac{A u_{\min}^2 + B u_{\min}}{A u_{\max} + B u_{\max}}, & u_{\min}, \\
u_{\max}, & \text{otherwise} \\
\end{cases}
\end{align*} \]
Optimal Trajectories
Optimal Control of UAVs

Nominal Trajectory

Optimal control
## Numerical results - patchy sparse grids

| $q$ | $|G^q_{\text{sparse}}|$ Linear Interpolation | Dense grid size | # of windows |
|-----|--------------------------------------------|-----------------|--------------|
| $q = 9$ | 1,105                                      | $> 10^6$        | 5            |

<table>
<thead>
<tr>
<th>Window 1</th>
<th>Window 2</th>
<th>Window 3</th>
<th>Window 4</th>
<th>Window 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8e-5</td>
<td>1.2e-4</td>
<td>5.9e-4</td>
<td>2.0e-4</td>
<td>6.1e-5</td>
</tr>
</tbody>
</table>

MAE is computed at 1100 random points in each window.
Closed-loop control with saturation

- Controller: zero-order hold feedback at 30 Hz.
- Sensor error: uniform distribution ($e_V: \pm 0.2 m/s; e_\gamma, e_\psi, e_\phi: \pm 2^\circ$)
- Feedback: interpolation of costates in $u^*(x, \lambda)$.

Control input

![Control input plots]

Trajectory

![Trajectory plots]
Conclusions

Outline

- Data-driven approach in model-based design
- Causality-free algorithms for data generation
- NN training and validation
- Algorithms (characteristic methods, minimization-based methods, direct methods, stochastic process)
- Sparse grid and error analysis
- Examples

THANK YOU