NAVAL
POSTGRADUATE
SCHOOL

Data Development and Deep Learning for HJB Equations

Wei Kang

Department of Applied Mathematics
Naval Postgraduate School, Monterey, CA

IPAM Workshop
March, 2020

NAVAL
POSTGRADUATE
SCHOOL

1. W. Kang, Q. Gong, T. Nakamura-Zimmerer, *Algorithms of Data Development For Deep Learning and Feedback Design*, arXiv:1912.00492, 2019.
2. T. Nakamura-Zimmerer, Q. Gong, W. Kang, *Adaptive Deep Learning for High-Dimensional Hamilton-Jacobi-Bellman Equations*, arXiv:1907.05317, 2019..

Feedback Design

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
  - ✓ ......

Feedback Design

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
  - ✓ ......

Feedback Design

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?

Feedback Design

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

$$\begin{cases} \underset{\boldsymbol{u}\in\mathcal{U}}{\text{minimize}} & \displaystyle\int_{t_0}^{t_f} L(t,\boldsymbol{x},\boldsymbol{u})dt + \psi(\boldsymbol{x}(t_f)), \\ \text{subject to} & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{u}), \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0. \end{cases}$$

The goal of control design is to find a feedback law

$$\boldsymbol{u} = \boldsymbol{u}^*(t,\boldsymbol{x})$$

that minimizes the cost.

Optimal Controller Design

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.

Optimal Controller Design

Controller Design based on the HJB equation

Define the Hamiltonian

$$H(t, \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) = L(t, \boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}).$$

Solve the HJB equation to find the optimal feedback law

$$\begin{cases} V_t(t, \boldsymbol{x}) + \min_{\boldsymbol{u} \in \mathcal{U}} H\left(t, \boldsymbol{x}, V_{\boldsymbol{x}}, \boldsymbol{u}\right) = 0, \\ V(t_f, \boldsymbol{x}) = \psi(\boldsymbol{x}), \end{cases}$$

$$\boldsymbol{u}^*(t, \boldsymbol{x}) = \arg\min_{\boldsymbol{u} \in \mathcal{U}} H\left(t, \boldsymbol{x}, V_{\boldsymbol{x}}, \boldsymbol{u}\right).$$

The complexity of solving the HJB equation increases exponentially with dimension - the curse-of-dimensionality.

Data-Driven & Model-Based Design

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 $V(t, \boldsymbol{x})$ at random points in a given region.
2. *Training:* Given this data set, a neural network, $V^{NN}(t, \boldsymbol{x})$, 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*:

$$\boldsymbol{u}^*(t, \boldsymbol{x}) = \arg\min_{\boldsymbol{u} \in \mathcal{U}} H\left(t, \boldsymbol{x}, V_{\boldsymbol{x}}^{NN}, \boldsymbol{u}\right).$$

An example of optimal attitude control

State variables:

$$\boldsymbol{v} = \begin{pmatrix}\phi & \theta & \psi\end{pmatrix}^T \quad \text{Euler angles}$$
$$\boldsymbol{\omega} = \begin{pmatrix}\omega_1 & \omega_2 & \omega_3\end{pmatrix}^T \quad \text{angular velocity}$$

Define

$$\boldsymbol{E}(\boldsymbol{v}) := \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{pmatrix}, \qquad \boldsymbol{S}(\boldsymbol{\omega}) := \begin{pmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{pmatrix},$$

$$\boldsymbol{R}(\boldsymbol{v}) := \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\theta\sin\phi \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\theta\cos\phi \end{pmatrix}.$$

$$\boldsymbol{B} = \begin{pmatrix} 1 & 1/20 & 1/10 \\ 1/15 & 1 & 1/10 \\ 1/10 & 1/15 & 1 \end{pmatrix}, \qquad \boldsymbol{J} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Data-Driven & Model-Based Design

The optimal control problem is

$$\begin{cases} \underset{\boldsymbol{u}(\cdot)}{\text{minimize}} & \int_t^{t_f} L(\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{u})d\tau + \frac{W_4}{2}\|\boldsymbol{v}(t_f)\|^2 + \frac{W_5}{2}\|\boldsymbol{\omega}(t_f)\|^2, \\ \text{subject to} & \dot{\boldsymbol{v}} = \boldsymbol{E}(\boldsymbol{v})\boldsymbol{\omega}, \\ & \boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}(\boldsymbol{v})\boldsymbol{h} + \boldsymbol{B}\boldsymbol{u}. \end{cases} \tag{1}$$

Here

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

and

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

Data-Driven & Model-Based Design

Domain in State Space

$$\mathcal{X}_0 = \left\{ \boldsymbol{v}, \boldsymbol{\omega} \in \mathbb{R}^3 \middle| -\frac{\pi}{3} \leqslant \phi, \theta, \psi \leqslant \frac{\pi}{3} \text{ and } -\frac{\pi}{4} \leqslant \omega_1, \omega_2, \omega_3 \leqslant \frac{\pi}{4} \right\},$$

Initial Data Set: Solving TPBVP using time-marching

$$\begin{aligned} &\boldsymbol{x}^{(i)} = (\boldsymbol{v}^{(i)}, \boldsymbol{\omega}^{(i)}) \\ &V(0, \boldsymbol{x}^{(i)}), \qquad\qquad \text{for } i = 1, 2, \cdots, N_d = 64. \\ &\boldsymbol{\lambda}^{(i)}(0), \end{aligned}$$

Loss function:

$$\mathcal{L} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left[V^{(i)} - V^{NN}(t^{(i)}, \boldsymbol{x}^{(i)}; \theta) \right]^2 + \frac{\mu}{N_d} \sum_{i=1}^{N_d} \left\| \boldsymbol{\lambda}^{(i)} - V_{\boldsymbol{x}}^{NN}(t^{(i)}, \boldsymbol{x}^{(i)}; \theta) \right\|^2,$$

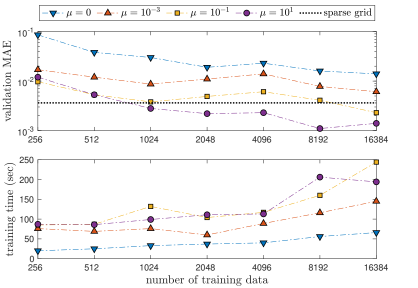
Data-Driven & Model-Based Design

First Neural Network: Train a neural network,

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

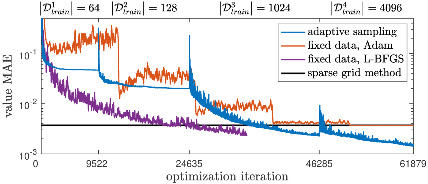
Additional data set: Warm start using $V^{NN}(0, x)$ to speed up data generation.

Validation: Generating more data for accuracy verification.

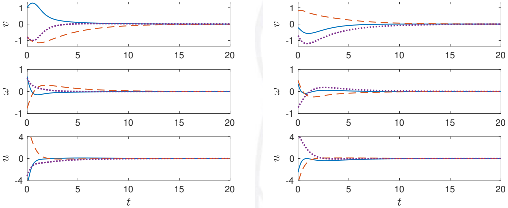


Data-Driven & Model-Based Design

One more reason of generating data - adaptive sampling



Closed-loop: sample trajectories



Methods of Generating Data

DATA IS ESSENTIAL!

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

- An algorithm is causality-free if the value of $V(t, \boldsymbol{x})$ 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

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

$$\begin{cases} \underset{\boldsymbol{u}\in\mathcal{U}}{\text{minimize}} & \displaystyle\int_{t_0}^{t_f} L(t,\boldsymbol{x},\boldsymbol{u})dt + \psi(\boldsymbol{x}(t_f)), \\ \text{subject to} & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{u}), \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0. \end{cases}$$

The goal of control design is to find a feedback law

$$\boldsymbol{u} = \boldsymbol{u}^*(t,\boldsymbol{x})$$

that minimizes the cost.

The Pontryagin Maximum Principle

Optimal Control

$$\boldsymbol{u}^*(t,\boldsymbol{x}) = \arg\min_{\boldsymbol{u}\in\mathcal{U}} H\left(t,\boldsymbol{x},V_{\boldsymbol{x}},\boldsymbol{u}\right)$$

Two Point Boundary Value Problem (TPBVP)

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{u}^*(t,\boldsymbol{x},\boldsymbol{\lambda})), & \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \dot{\boldsymbol{\lambda}}(t) = -\dfrac{\partial H}{\partial \boldsymbol{x}}(t,\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{u}^*(t,\boldsymbol{x},\boldsymbol{\lambda})), & \boldsymbol{\lambda}(t_f) = \dfrac{\partial \psi}{\partial \boldsymbol{x}}(t_f), \\ \dot{V}(t) = L(t,\boldsymbol{x},\boldsymbol{u}^*(t,\boldsymbol{x},\boldsymbol{\lambda})), & V(t_f) = \psi(\boldsymbol{x}(t_f)). \end{cases}$$

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

$$(\boldsymbol{x}^1(t), \boldsymbol{\lambda}^1(t)).$$

Characteristic Methods

3. Extending the trajectory to $[t_0, t_2]$.

$$x_0^2(t) = \begin{cases} x^1(t), & \text{if } t_0 \leqslant t \leqslant t_1, \\ x^1(t_1), & \text{if } t_1 < t \leqslant t_2, \end{cases}$$

Or

$$x_0^2(t) = x^1\left(t_0 + \frac{t_1 - t_0}{t_2 - t_0}(t - t_0)\right), \quad \text{for } t_0 \leqslant t \leqslant t_2.$$

$\lambda_0^2(t)$ is similarly defined.

4. $(x_0^2(t), \lambda_0^2(t))$ is used as initial guess to solve the TPBVP over $[0, t_2]$.
5. ......
6. Repeating the process until $t_K = t_f$

Characteristic Methods

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.

[1] W. Kang and L. Wilcox. *A causality free computational method for HJB equations with application to rigid body satellites*. Proceedings of AIAA GNC Conference, 2015.

[2] W. Kang and L. Wilcox. *Mitigating the curse of dimensionality: sparse grid characteristic method for optimal feedback control and HJB equations*. Comput. Optim. Appl., 2017.

Characteristic Methods

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 $V^{NN}(t, \boldsymbol{x})$.
3. Generate more data using warm start

$$\lambda_0(t) = V_{\boldsymbol{x}}^{NN}(t, \boldsymbol{x}).$$

[1] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations. arXiv:1907.05317, 2019.

Characteristic Methods

Example: Rigid body optimal attitude control

Time-marching method

K	% BVP convergence	mean integration time
1	0.3%	0.37 s
2	38.7%	0.44 s
3	76.2%	0.40 s
4	92.9%	0.45 s
8	98.4%	0.53 s

Neural network warm start

μ	% BVP convergence	mean integration time
0	90%	0.44 s
10^{-3}	99.6%	0.41 s
10^{1}	100%	0.40 s

Characteristic Methods

Backward propagation

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

$$
\begin{cases}
\dot{\boldsymbol{x}}(t) = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda})), & \boldsymbol{x}(t_f) = \boldsymbol{x}_f + \delta \boldsymbol{x}_f, \\
\dot{\boldsymbol{\lambda}}(t) = -\dfrac{\partial H}{\partial \boldsymbol{x}}(t, \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda})), & \boldsymbol{\lambda}(t_f) = \dfrac{\partial \psi}{\partial \boldsymbol{x}}(t_f), \\
\dot{V}(t) = L(t, \boldsymbol{x}, \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda})), & V(t_f) = \psi(\boldsymbol{x}(t_f)).
\end{cases}
$$

Then, the data is used to train a neural network

$$V^{NN}(t, \boldsymbol{x}) \approx V(t, \boldsymbol{x}).$$

[1] D. Izzo, E. Öztürk and M. Märtens, Interplanetary transfers via deep representations of the optimal policy and/or of the value function. arXiv:1904.08809, 2019.

Characteristic Methods

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

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

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

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

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

The Hopf formula

$$V(t,\boldsymbol{x}) = (\psi^* + tH)^*(\boldsymbol{x})$$

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

$$\begin{cases} \dot{\boldsymbol{x}}(s) = \boldsymbol{f}(\boldsymbol{u}(s)), \\ \boldsymbol{x}(t) = \boldsymbol{x}, \\ \min J(\boldsymbol{x}, t; \boldsymbol{u}) = \int_t^T L(\boldsymbol{u}(s))ds + \psi(\boldsymbol{x}(T)), \end{cases}$$

- It can be extended to another family of systems

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

- The method is applicable to the Eikonal equation.

[1] J. Darbon and S. Osher, Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere. Res. Math. Sci., 3(1), 2016.

Minimization-Based Methods

Minimization along characteristics

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda})), & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \\ \dot{\boldsymbol{\lambda}}(t) = -\dfrac{\partial H}{\partial \boldsymbol{x}}(t, \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda})), & \boldsymbol{\lambda}(t_0) = \boldsymbol{\lambda}_0, \\ \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda}) = \arg\min_{\boldsymbol{u} \in \mathcal{U}} H(t, \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}). \end{cases}$$

For fixed initial state $\boldsymbol{x}_0$, the cost is a function of $\boldsymbol{\lambda}_0$,

$$J(t_0, \boldsymbol{x}_0, \boldsymbol{\lambda}_0) = \int_{t_0}^{t_f} L(t, \boldsymbol{x}, \boldsymbol{u}^*(t, \boldsymbol{x}, \boldsymbol{\lambda}))dt + \psi(t_f).$$

Then the solution, $V(t_0, \boldsymbol{x}_0)$, of the HJB equation is

$$V(t_0, \boldsymbol{x}_0) = \min_{\boldsymbol{\lambda}_0} J(t_0, \boldsymbol{x}_0, \boldsymbol{\lambda}_0).$$

Minimization-Based Methods

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].

[1] Y. T. Chow, J. Darbon, S. Osher, and W. Yin. Algorithm for overcoming the curse of dimensionality for state-dependent Hamilton-Jacobi equations. J. Comput. Phys., 387:376-409, 2019.

[2] I. Yegorov and P. M. Dower. Perspectives on characteristics based curse-of-dimensionality-free numerical approaches for solving Hamilton-Jacobi equations. Appl. Math. Optim., 2018.

Direct Methods

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

An incomplete list

- J. Betts. Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadelphia, 2001.
- G. Elnagar, M. A. Kazemi, and M. Razzaghi. The pseudospectral legendre method for discretizing optimal control problems. IEEE Trans. Autom. Control, 40(10):1793-1796, 1995.
- F. Fahroo and I. M. Ross. Costate estimation by a legendre pseudospectral method. J. Guid. Control Dyn., 24(2):270-277, 2001.
- Q. Gong, W. Kang and I. M. Ross. A pseudospectral method for the optimal control of constrained feedback linearizable systems. IEEE Trans. Automat. Control, 51(7):1115-1129, 2006.
- A. L. Dontchev and W. W. Hager. The Euler approximation in state constrained optimal control. Math. Comput., 70:173-203, 2001.
- W. W. Hager. Runge-kutta methods in optimal control and the transformed adjoint system. Numer. Math., 87(2):247-282, 2000.
- I. Chryssoverghi, J. Coletsos, and B. Kokkinis. Discretization methods for optimal control problems with state constraints. J. Comput. Appl. Math., 191:1-31, 2006.
- ....
- ....

A Pseudospectral optimal control

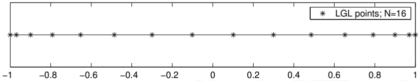
Problem definition:

$$\begin{cases} \min_{\boldsymbol{u}} J = \displaystyle\int_{-1}^{1} L(\boldsymbol{x}(t), \boldsymbol{u}(t))dt + E(\boldsymbol{x}(-1), \boldsymbol{x}(1)), \\ \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \\ \boldsymbol{e}(\boldsymbol{x}(-1), \boldsymbol{x}(1)) = 0, \\ \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leqslant 0. \end{cases}$$

Discretization

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

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



Feasibility and convergence

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

Theorem (Consistent Approximation)
Let $\{(\bar{\boldsymbol{x}}_k^*, \bar{\boldsymbol{u}}_k^*), 0 \leqslant k \leqslant N\}_{N=N_1}^{\infty}$ be a sequence of optimal solutions to the discretized problem. Let $\{t \to (\boldsymbol{x}_N(t), \boldsymbol{u}_N(t))\}_{N=N1}^{\infty}$ be their interpolating function sequence. Assume $\{(\boldsymbol{x}_0^*, \dot{\boldsymbol{x}}^N(\cdot), \boldsymbol{u}^N(\cdot))\}_{N=N_1}^{\infty}$ has a uniform accumulation point with continuous components. Then $t \to \boldsymbol{u}^{\infty}(t)$ is the solution of the original optimal control problem.

Semilinear parabolic PDE

$$\begin{cases} V_t(t,\boldsymbol{x}) + \dfrac{1}{2}\mathrm{Tr}(\sigma\sigma^T \mathrm{Hess}_{\boldsymbol{x}} V)(t,\boldsymbol{x}) + V_{\boldsymbol{x}}(t,\boldsymbol{x})^T \mu(t,\boldsymbol{x}) + \\ \qquad\qquad H(t,\boldsymbol{x},V(t,\boldsymbol{x}),\sigma^T(t,x)V_{\boldsymbol{x}}(t,x)) = 0, \\ V(t_f,\boldsymbol{x}) = \psi(\boldsymbol{x}). \end{cases}$$

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

[1] J. Han and W. E. Deep learning approximation for stochastic control problems. arXiv:1611.07422v1, 2016.

[2] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences, 115(34):8505-8510, 2018.

Stochastic Process

A method of characteristics

1. A forward stochastic differential equation (FSDE)

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \int_0^t \mu(s, \boldsymbol{x}(s))ds + \int_0^t \sigma(s, \boldsymbol{x}(s))d\boldsymbol{W}_s,$$

   $\boldsymbol{W}_t$, $0 \leqslant t \leqslant t_f$, is an n-dimensional Brownian motion.

2. A backward stochastic differential equation (BSDE)

$$\begin{cases} V(t, \boldsymbol{x}(t)) = V(0, \boldsymbol{x}_0) - \displaystyle\int_0^t H\left(s, \boldsymbol{x}(s), V(s, \boldsymbol{x}(s)), \sigma(s, \boldsymbol{x}(s))^T V_{\boldsymbol{x}}(s, \boldsymbol{x}(s))\right) \\ \qquad\qquad + \displaystyle\int_0^t V_{\boldsymbol{x}}(s, \boldsymbol{x}(s))^T \sigma(s, \boldsymbol{x}(s))d\boldsymbol{W}_s, \\ V(t_f, \boldsymbol{x}(t_f)) = \psi(\boldsymbol{x}(t_f)). \end{cases}$$

A method of characteristics (Discretized equations)

1. A forward discrete-time equation

$$\boldsymbol{x}(t_{n+1}) \approx \boldsymbol{x}(t_n) + \mu(t_n, \boldsymbol{x}(t_n))\Delta t_n + \sigma(t_n, \boldsymbol{x}(t_n))(\boldsymbol{W}_{t_{n+1}} - \boldsymbol{W}_{t_n}),$$

   for $n = 0, 1, 2, \cdots, N-1$,

2. A backward discrete-time equation

$$\begin{aligned} &V(t_{n+1}, \boldsymbol{x}(t_{n+1})) \\ &\approx V(t_n, \boldsymbol{x}(t_n)) - H\left(t_n, \boldsymbol{x}(t_n), V(t_n, \boldsymbol{x}(t_n)), \sigma(t_n, \boldsymbol{x}(t_n))^T V_{\boldsymbol{x}}(t_n, \boldsymbol{x}(t_n))\right)\Delta t_n \\ &+ V_{\boldsymbol{x}}(t_n, \boldsymbol{x}(t_n))^T \sigma(t_n, \boldsymbol{x}(t_n))(\boldsymbol{W}_{t_{n+1}} - \boldsymbol{W}_{t_n}), \end{aligned}$$

Stochastic Process

1. Replace $\boldsymbol{x} \to \sigma^T V_{\boldsymbol{x}}(t_n, \boldsymbol{x})$ in the backward discrete-time equation by a neural network with unknown parameters, θ_n.
2. Set up a loss function

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

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

$$\left\{ \{ \boldsymbol{W}^i_{t_{n+1}} - \boldsymbol{W}^i_{t_n} \}_{n=0}^{N-1} \middle|\ i = 1, 2, \cdots, N_s \right\},$$
$$\boldsymbol{W}^i_{t_{n+1}} - \boldsymbol{W}^i_{t_n} \sim N(0, t_{n+1} - t_n)).$$

3. Training step: finding $V(0, \boldsymbol{x})$, $V_{\boldsymbol{x}}(0, \boldsymbol{x})$ and $\{\theta_n\}_{n=1}^{N-1}$ by minimizing the loss function

Summary

Method	Initial Guess Dependent	Suggested Solver or Algorithm	Comments
Time or space-marching	No	BVP solver	Convergence and speed depends on the number of marching steps.
NN warm start	Yes	BVP solve	Convergence and speed depends on the quality of NN initial guess.
Backward propagation	No	ODE solver	No convergence issue. Initial states in data cannot be pre-selected.
Hopf formula	Yes	Bregman algorithm	It works for special types of control systems.
Minimization along characteristics	Yes	Powell's algorithm Coordinate descent	Convergence and speed depends on the quality of initial guess.
Direct methods	Yes	SQP or nonlinear programming	The method is effective for problems with state-control constraints.
Stochastic process	Yes	NN training	The method is applicable to stochastic optimal control.

A hierarchy of grids

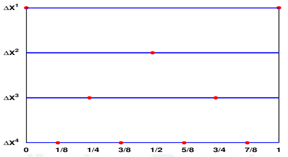
$$X^1 = \{0, 1\}, X^2 = \{0, \tfrac{1}{2}, 1\}, X^3 = \{0, \tfrac{1}{4}, \tfrac{2}{4}, \tfrac{3}{4}, 1\}, \cdots$$
$$X^i = \{\tfrac{k-1}{2^{i-1}}; \quad k = 1, 2, \cdots m_i\}, \quad m_i = 2^{i-1} + 1$$

The sequence has a telescopic structure

$$X^1 \subset X^2 \subset X^3 \subset X^4 \subset \cdots$$

Define

$$\Delta X^1 = X^1, \quad \Delta X^i = X^i \backslash X^{i-1}, i \geqslant 2$$
$$\Delta m_i = |\Delta X^i|$$



ΔX^i for $i = 1, 2, 3, 4$

A hierarchy of grids in $\mathbb{R}$

Vector index

$$\begin{aligned}
&\boldsymbol{i} = \left[\begin{array}{cccc} i_1 & i_2 & \cdots & i_d \end{array}\right], \qquad |\boldsymbol{i}| = i_1 + i_2 + \cdots + i_d \\
&\boldsymbol{j} = \left[\begin{array}{cccc} j_1 & j_2 & \cdots & j_d \end{array}\right] \\
&\Delta X^{\boldsymbol{i}} = \Delta X^{i_1} \times \cdots \times \Delta X^{i_d}, \quad \Delta m^{\boldsymbol{i}} = \left[\begin{array}{ccc} \Delta m^{i_1} & \cdots & \Delta m^{i_d} \end{array}\right] \\
&x_{\boldsymbol{j}}^{\boldsymbol{i}} = (x_{j_1}^{i_1}, \cdots, x_{j_d}^{i_d}) \in \Delta X^{\boldsymbol{i}}
\end{aligned}$$

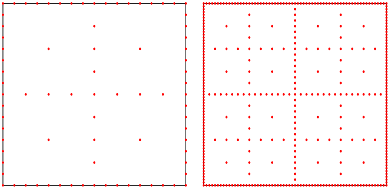
The dense grid is

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

Following Smolyak's approximation algorithm, the sparse grid is

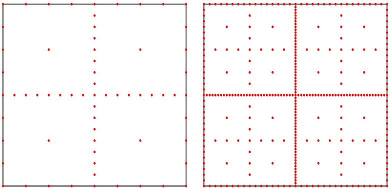
$$G^q_{\text{sparse}} = \bigcup_{|\boldsymbol{i}| \leqslant q} \Delta X^{\boldsymbol{i}}$$

Examples of sparse grids



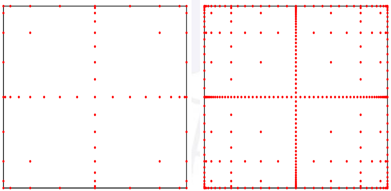
G^q_{sparse} in $[0,1]^2$, $q=6$ and $q=8$

A modified sparse grid



$G^q_{\mathrm{sparse}}(\mathit{Modified})$ $(q=6,\ q=8)$

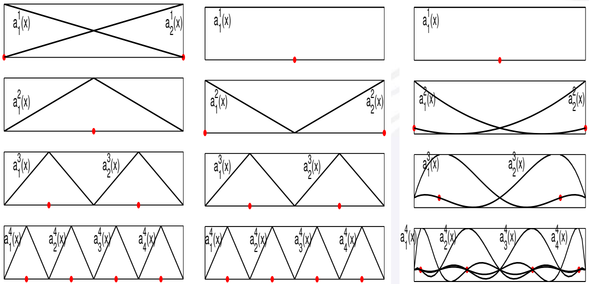
CGL (Chebyshev-Gauss-Lobatto) type



$G^q_{\mathrm{sparse}}(\mathit{CGL})$ $(q=6,\ q=8)$

Interpolation

Basis functions



Hierarchical surpluses

$$I^q(f) = I^{q-1}(f) + \Delta I^q(f), \qquad q \geqslant d$$

$$\Delta I^q(f) = \sum_{|\boldsymbol{i}|=q} \sum_{1 \leqslant \boldsymbol{j} \leqslant \Delta m^{\boldsymbol{i}}} w_{\boldsymbol{j}}^{\boldsymbol{i}} a_{\boldsymbol{j}}^{\boldsymbol{i}}$$

$$w_{\boldsymbol{j}}^{\boldsymbol{i}} = f(x_{\boldsymbol{j}}^{\boldsymbol{i}}) - I^{q-1}(f)(x_{\boldsymbol{j}}^{\boldsymbol{i}})$$

Sparse vs. dense

Grid	Grid size	Interpolation error
Dense	N^d	$O(\frac{1}{N^2})$
Sparse	$O(N(\log N)^{d-1})$	$O(\frac{(\log N)^{d-1}}{N^2})$

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: $O(N(\log N)^{d-1})$.

Error analysis

$$\bar{V}(t,x) = V(t,x) + e_{interp} + e_{BVP}$$

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

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

CGL sparse grid with polynomial interpolation,

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

where ϵ is the error of BVP solver at individual grid points.

[1] W. Kang and L. C. Wilcox, Mitigating the curse of dimensionality: sparse grid characteristic method for optimal feedback control and HJB equations, Comput. Optim. Appl., Vol.68(2), 2017, pp 289-315.

Error analysis - a practical way

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

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

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

[1] W. Kang and L. C. Wilcox, Mitigating the curse of dimensionality: sparse grid characteristic method for optimal feedback control and HJB equations, Comput. Optim. Appl., Vol.68(2), 2017, pp 289-315.

Optimal Attitude Control

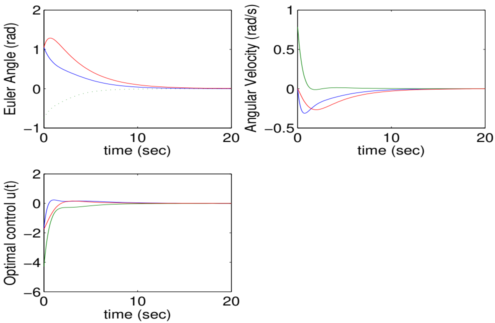
Attitude Control - system and problem definitions are from example 1.

q	$\lvert G^q_{\text{sparse}} \rvert$ CGL	Dense grid size	# of Processors	MAE $N = 1280$ samples
$q = 13$	$44,698$	$> 10^{12}$	512	7.3 e-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



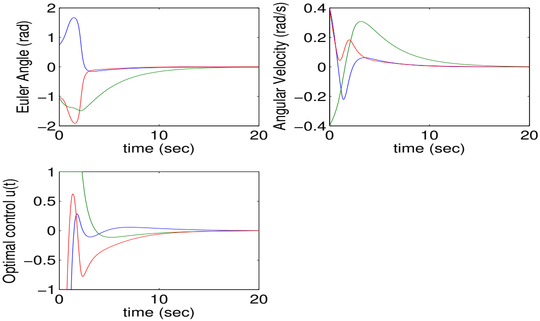
Optimal Attitude Control

Control with saturation $u \leqslant 1$

q	$\lvert G^q_{\text{sparse}} \rvert$ CGL	Dense grid size	# of Processors	MAE $N = 1280$ samples
$q = 13$	$44,698$	$> 10^{12}$	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

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

subject to

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

Parameters

$$B = \begin{bmatrix} 1 & \frac{1}{10} \\ 0 & 1 \\ \frac{1}{12} & 0 \end{bmatrix}, \qquad \begin{array}{l} H = [12\ 12\ 6]^T \\ W_1 = 1, \quad W_2 = 2 \\ W_3 = \frac{1}{2}, \end{array}$$

$$J = \mathrm{diag}(2,3,4),$$

$$-\frac{\pi}{6} \leqslant \phi, \theta, \psi \leqslant \frac{\pi}{6}, \qquad -\frac{\pi}{8} \leqslant \omega_i \leqslant \frac{\pi}{8}$$

Optimal Attitude Control

Equilibrium: $v = v_e(v,\omega), \;\; \omega = 0$

$$\min_{v_e} ||R(v_e) - I||_{\max}$$
$$\text{subject to}$$
$$C^T R(v_e) H = C^T (R(v)H - J\omega)$$

where $C \in \Re^3$ is a constant vector satisfying

$$C^T B = 0$$

The equilibrium, v_e, is computed numerically. The process is equivalent to maximizing trace$(R(v_e))$.

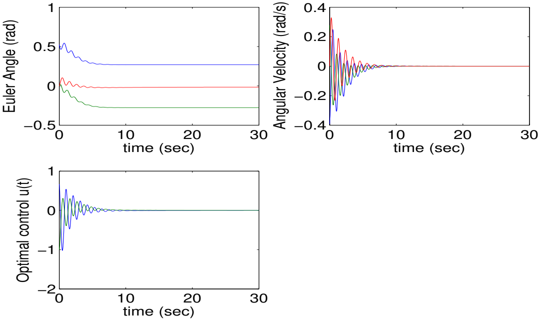
Optimal Attitude Control

Numerical results $(m = 2)$

q	$\lvert G^q_{\text{sparse}} \rvert$	Dense grid size	# of Processors	MAE $N = 1280$ samples
$q = 13$	$44,698$	$> 10^{12}$	512	8.5 e-3

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

An example of optimal trajectory



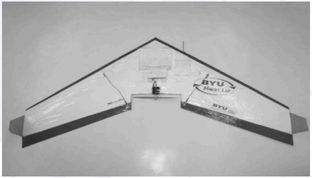
The system model

with Oleg Yakimenko

$$\dot{x}_1 = V\cos\gamma\cos\Psi, \quad \dot{x}_2 = V\cos\gamma\sin\Psi, \quad \dot{x}_3 = -V\sin\gamma$$
$$\dot{V} = \frac{1}{m}T - \frac{C_{D0}\rho S}{2m}V^2 - gA_1 n_z - \frac{2mg^2A_2}{\rho S}\frac{n_z{}^2}{V^2} - g\sin\gamma$$
$$\dot{\gamma} = \frac{g}{V}(n_z\cos\phi - \cos\gamma)$$
$$\dot{\Psi} = \frac{g}{V\cos\gamma}n_z\sin\phi$$
$$\dot{\phi} = u_\phi$$

Parameters adopted from foam Unicorn wing

(x_1, x_2, x_3) - location in NED frame	n_z - vertical lift	ρ - air density
V - speed	T - throttle	$S = 0.321\ m^2$
γ - path angle	u_ϕ - bank angle rate	$mg = 9.34\ N$
ψ - heading		$CD_0 = 0.0213$
ϕ - bank angle		$A_1 = -0.056$
		$A_2 = 0.22$



[1] J. N. Ostler and W. J. Bowman, Flight testing of small, electric powered unmanned aerial vehicles, Proc. U.S. Air Force T&E Days, 2005.

The cost functional:

$$
\begin{aligned}
\mathcal{J} &= \int_0^{t_f} L(V,\gamma,\Psi,\phi,u)dt \\
L(V,\gamma,\Psi,\phi,u) &= \frac{W_1}{2}||V-V^d||^2 + \frac{W_2}{2}||\gamma-\gamma^d||^2 + \frac{W_3}{2}||\Psi-\Psi^d|| \\
&+ \frac{W_4}{2}||\phi-\phi^d||^2 \\
&+ \frac{W_5}{2}||T-T^d||^2 + \frac{W_6}{2}||n_z-n_z^d||^2 + \frac{W_7}{2}||u_\phi-u_\phi^d||^2
\end{aligned}
$$

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_ϕ^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

$$\boxed{\begin{aligned} H(V,\gamma,\Psi,\phi,u,\lambda) &= H_1(V,\gamma,\Psi,\phi,\lambda) + A_T T^2 + B_T T \\ &+ A_{n_z} {n_z}^2 + B_{n_z} n_z + A_{u_\phi} {u_\phi}^2 + B_{u_\phi} u_\phi \end{aligned}}$$

$$A_T = \frac{W_5}{2}, \quad B_T = \lambda_1\alpha_1 - W_5 T^d$$

$$A_{n_z} = \frac{W_6}{2} - \frac{\lambda_1\alpha_4}{V^2}, \quad B_{n_z} = \frac{\lambda_2 g}{V}\cos\phi + \frac{\lambda_3 g}{V\cos\gamma}\sin\phi - \lambda_1\alpha_3 - n_z^d W_6$$

$$A_{u_\phi} = \frac{W_7}{2}, \quad B_{u_\phi} = \lambda_4 - W_7 u_\phi^d$$

$$H_1(V,\gamma,\Psi,\phi,\lambda) = \text{all other terms of states and co-states}$$

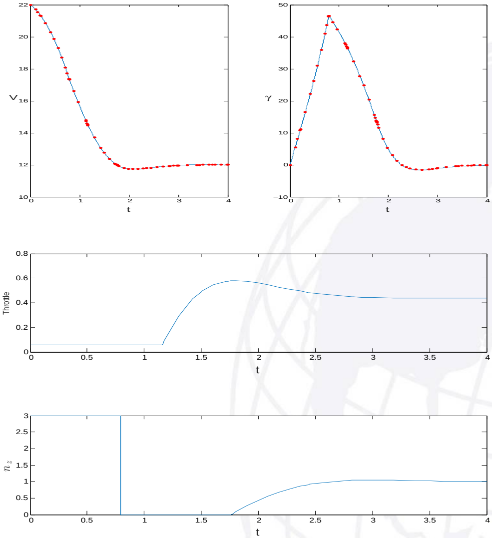
$A > 0$

$$u^* = \begin{cases} -\frac{B}{2A}, & u_{\min} < -\frac{B}{2A} < u_{\max} \\ u_{\min}, & -\frac{B}{2A} \leqslant u_{\min} \\ u_{\max}, & -\frac{B}{2A} \geqslant u_{\max} \end{cases}$$

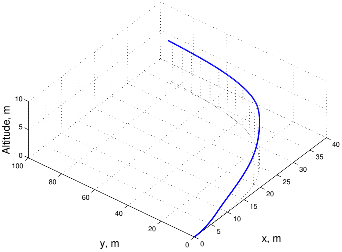
$A < 0$

$$u^* = \begin{cases} u_{\min}, & Au_{\min}^2 + Bu_{\min} \leqslant Au_{\max}^2 + Bu_{\max} \\ u_{\max}, & \text{otherwise} \end{cases}$$

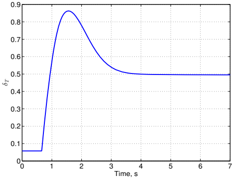
Optimal Trajectories

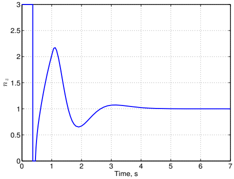


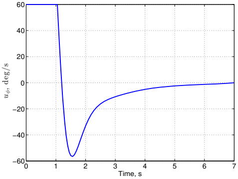
Nominal Trajectory



Optimal control





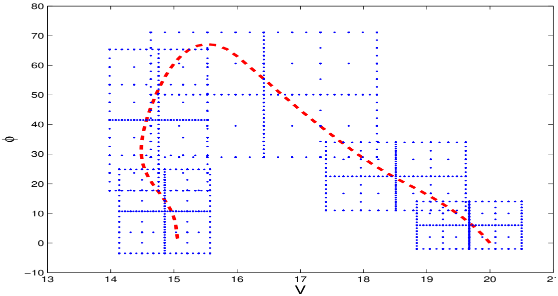


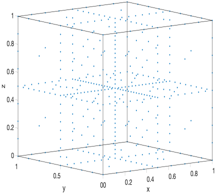
Numerical results - patchy sparse grids

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

Window 1	Window 2	Window 3	Window 4	Window 5
5.8e-5	1.2e-4	5.9e-4	2.0e-4	6.1 e-5

MAE is computed at 1100 random points in each window.

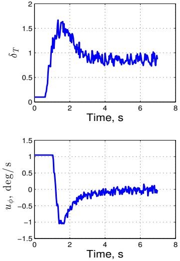




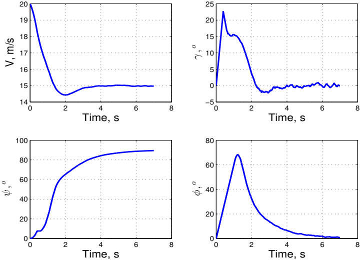
Closed-loop control with saturation

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

Control input



Trajectory



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