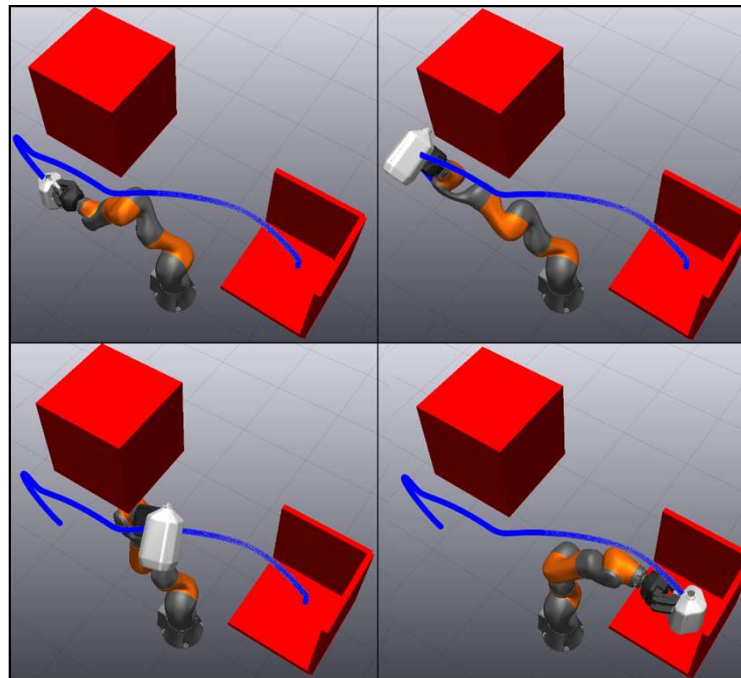


Constrained Unscented Dynamic Programming



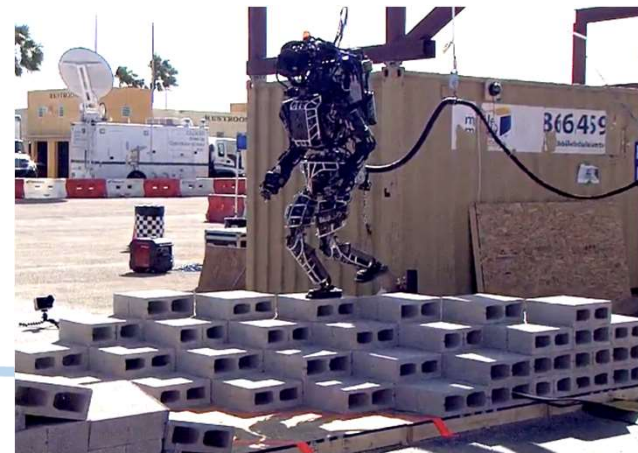
Brian Plancher, Zac Manchester and Scott Kuindersma
Harvard Agile Robotics Lab



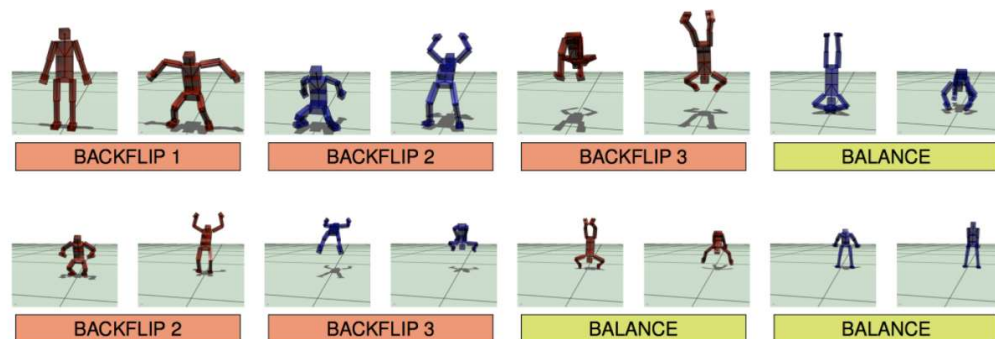
Trajectory Optimization synthesizes dynamic motions for complex robotic systems



[Foehn RSS 2017]



[DARPA Robotics Challenge 2015]



[AI Borno TVCG 2013]

Trajectory optimization minimizes a discrete time cost function subject to dynamics constraints

$$\min_{x,u} l_f(x_N) + \sum_{k=1}^{N-1} l(x_k, u_k)$$

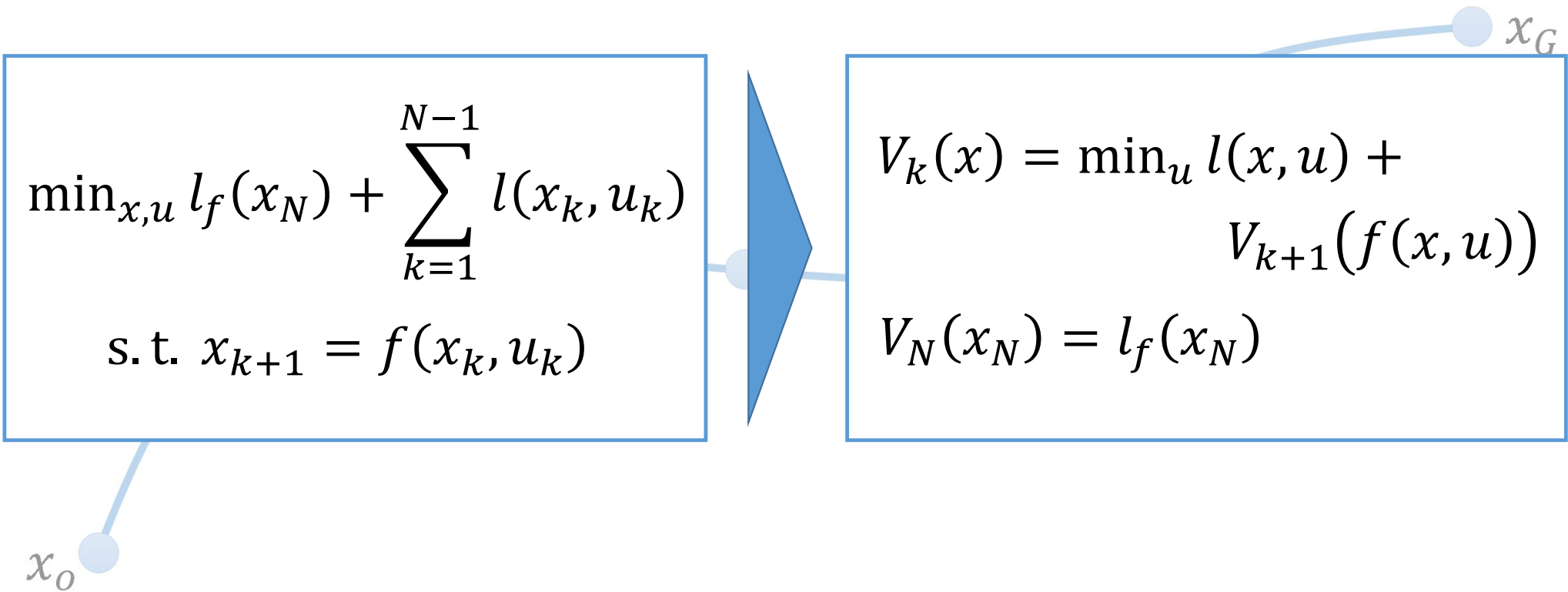
$$\text{s. t. } x_{k+1} = f(x_k, u_k)$$

x_o

x_G



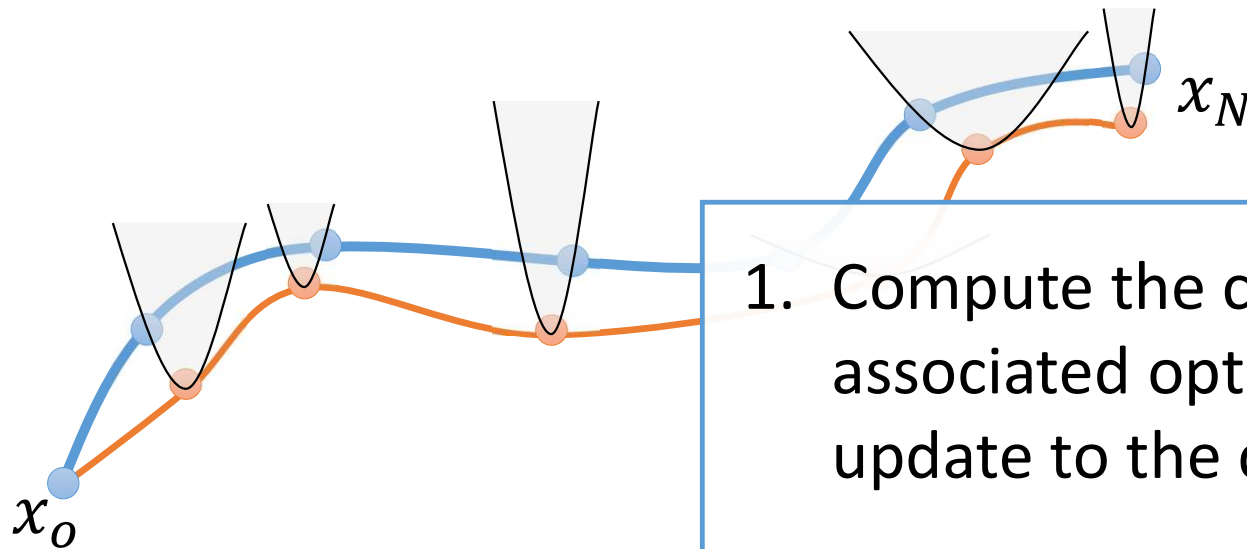
Dynamic Programming solves this problem through the recursive Bellman equation


$$\min_{x,u} l_f(x_N) + \sum_{k=1}^{N-1} l(x_k, u_k)$$

s. t. $x_{k+1} = f(x_k, u_k)$

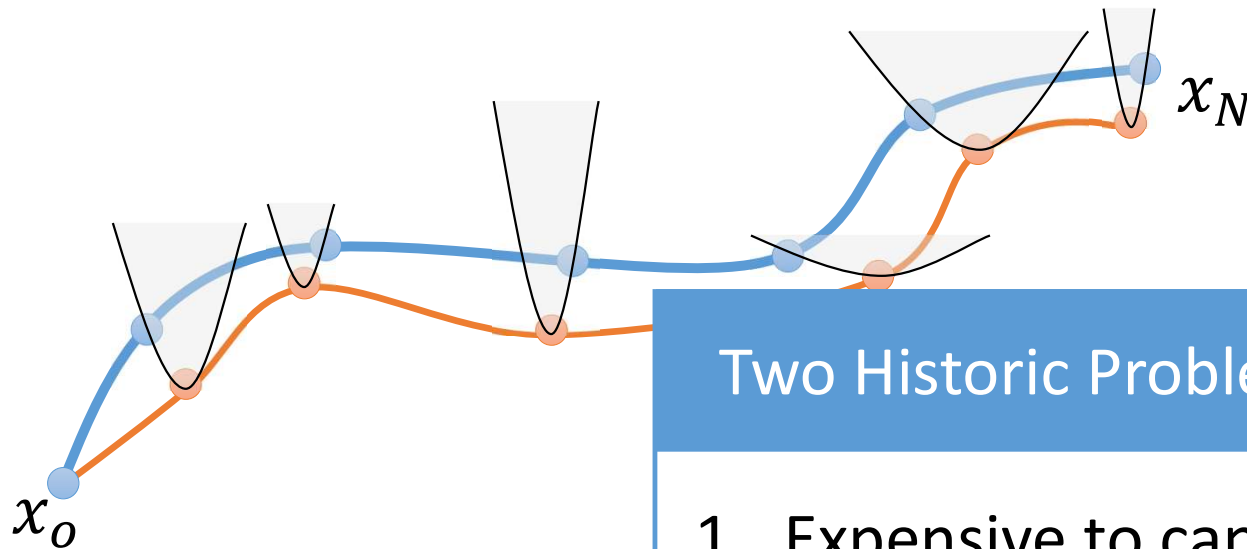
$$V_k(x) = \min_u l(x, u) + V_{k+1}(f(x, u))$$
$$V_N(x_N) = l_f(x_N)$$

DP methods (DDP/SLQ/iLQR) use quadratic approximations around a nominal trajectory



1. Compute the cost-to-go and the associated optimal feedback control update to the controls **backward** in time
2. Simulate the system **forward** in time to create a new nominal trajectory
3. Repeat this process until convergence

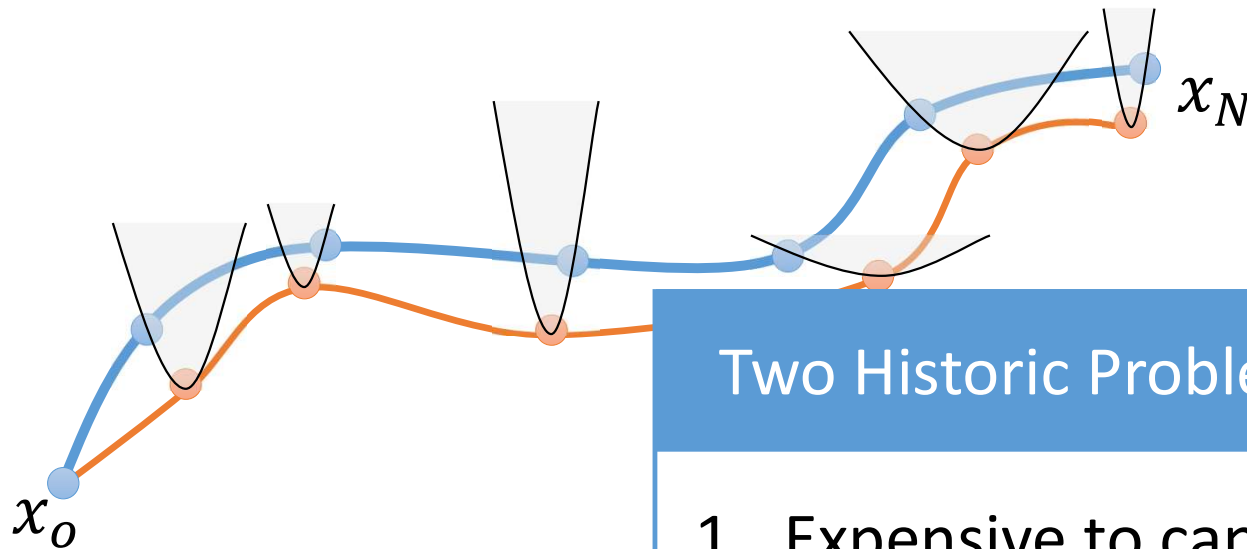
DP methods (DDP/SLQ/iLQR) use quadratic approximations around a nominal trajectory



Two Historic Problems with DP Algorithms

1. Expensive to capture the 2nd order information
2. Hard to enforce constraints

DP methods (DDP/SLQ/iLQR) use quadratic approximations around a nominal trajectory



Two Historic Problems with DP Algorithms

1. Expensive to capture the 2nd order information
2. **Hard to enforce constraints**

Recent research into adding constraints to DP like algorithms has taken two general paths

QP Methods

$$\min_{x,u} l_f(x_N) + \sum_{k=1}^{N-1} l(x_k, u_k) \quad \text{s.t. } x_{k+1} = f(x_k, u_k) \quad b_l \leq u \leq b_u$$

[Tassa ICRA 2014]
[Xie ICRA 2017]
[Farshidian ICRA 2017]

Penalty Methods

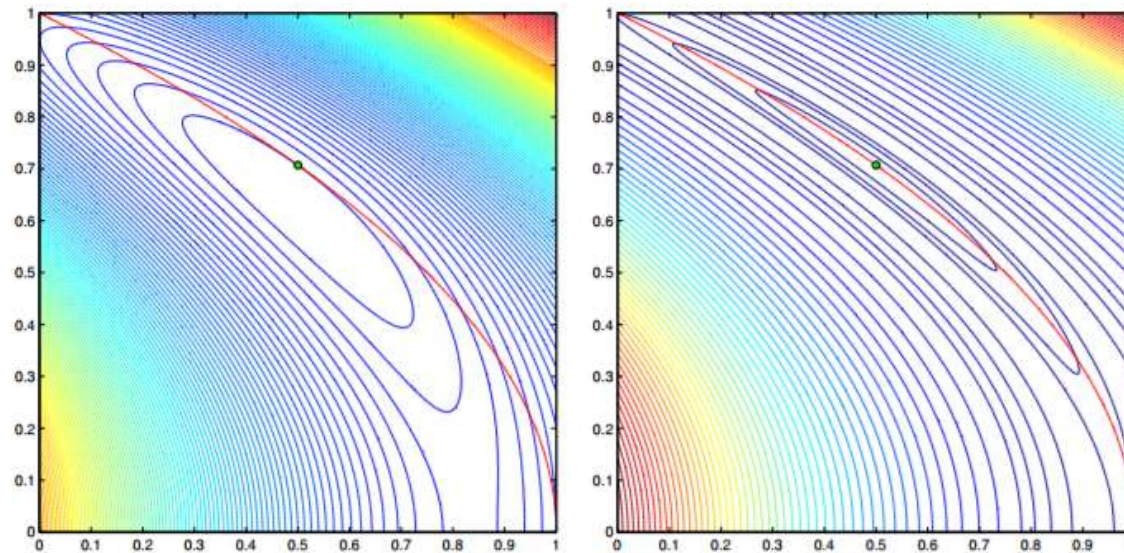
$$\min_{x,u} l_f(x_N) + \sum_{k=1}^{N-1} l(x_k, u_k) + \mu \|\phi(x_k, u_k)\|^2 \quad \text{s.t. } x_{k+1} = f(x_k, u_k)$$

[van den Berg ACC 2014]
[Farshidian ICRA 2017]
[Neunert RAL 2017]

Quadratic penalty methods are popular but can lead to numerical ill conditioning

Penalty Methods

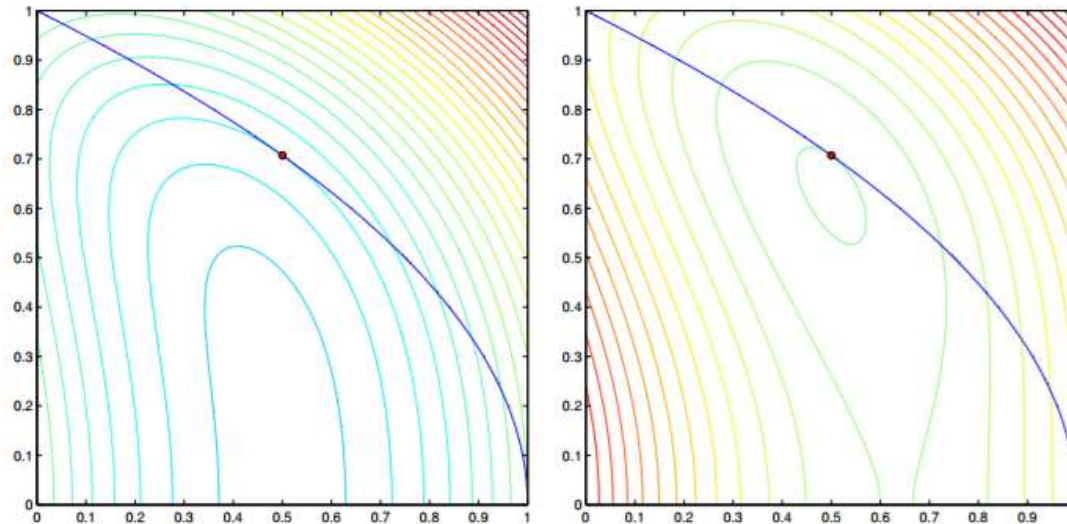
$$\min_{x,u} l_f(x_N) + \sum_{k=1}^{N-1} l(x_k, u_k) + \mu \|\phi(x_k, u_k)\|^2 \quad \text{s.t. } x_{k+1} = f(x_k, u_k)$$



Augmented Lagrangian methods show promise for trajectory optimization problems

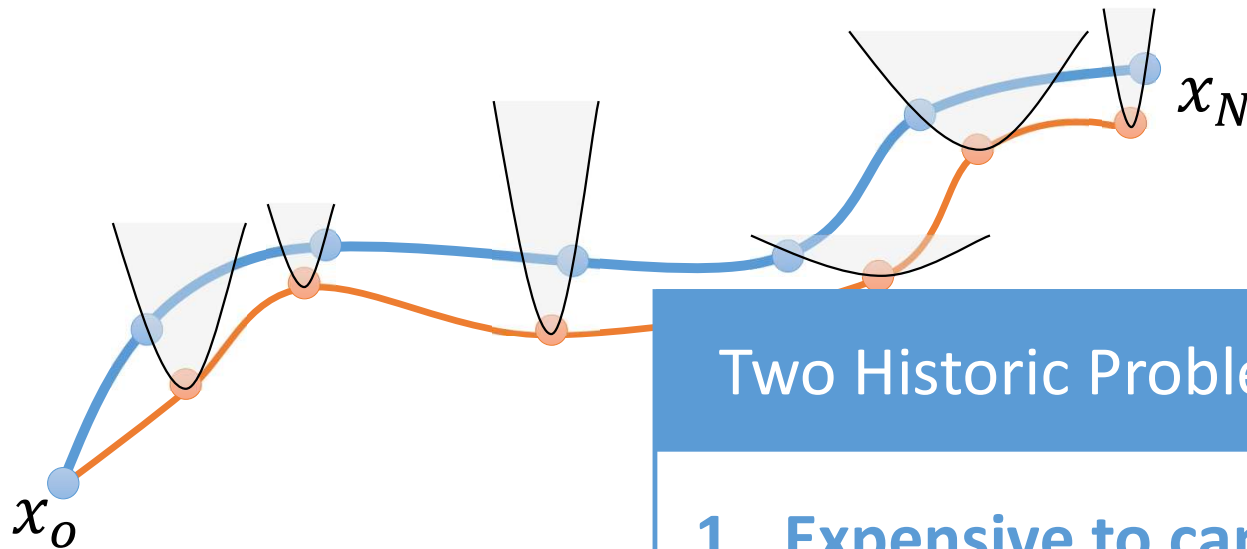
Augmented Lagrangian Methods

$$\min_{x,u} l_f(x_N) + \sum_{k=1}^{N-1} l(x_k, u_k) + \mu \|\phi(x_k, u_k)\|^2 + \lambda^T g(x_k, u_k) \quad \text{s.t. } x_{k+1} = f(x_k, u_k)$$



[Gould 2006]

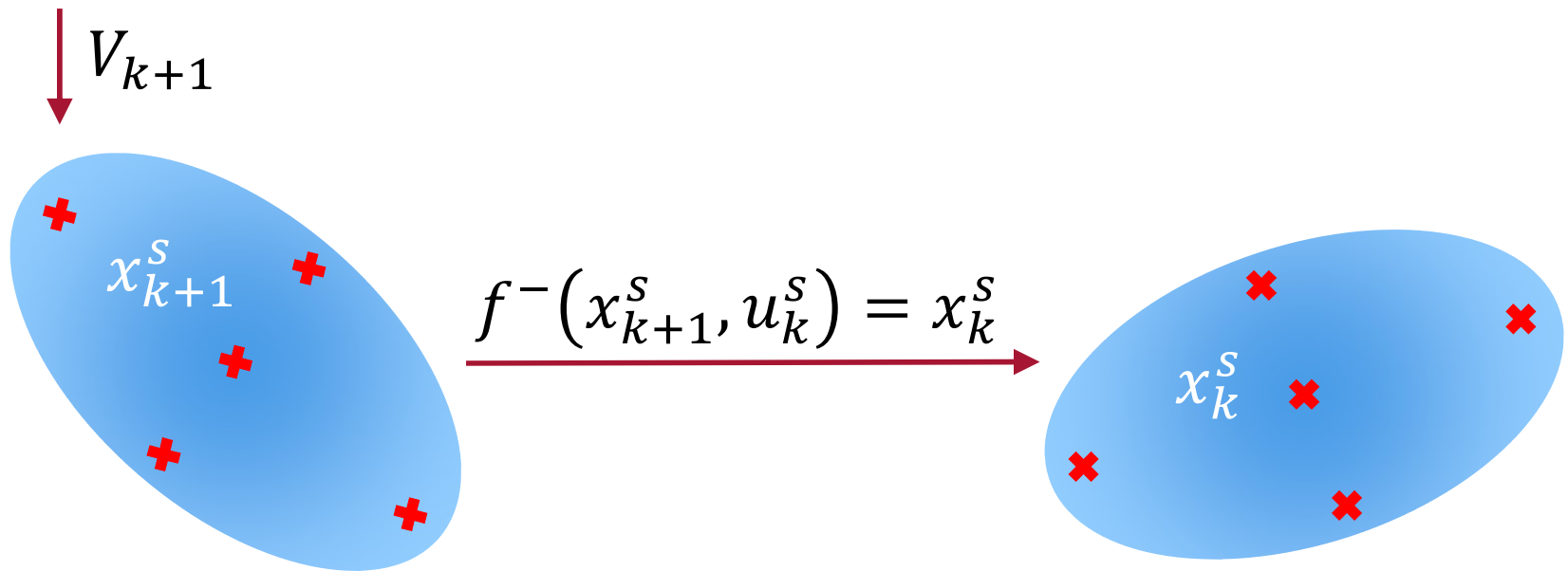
DP methods (DDP/SLQ/iLQR) use quadratic approximations around a nominal trajectory



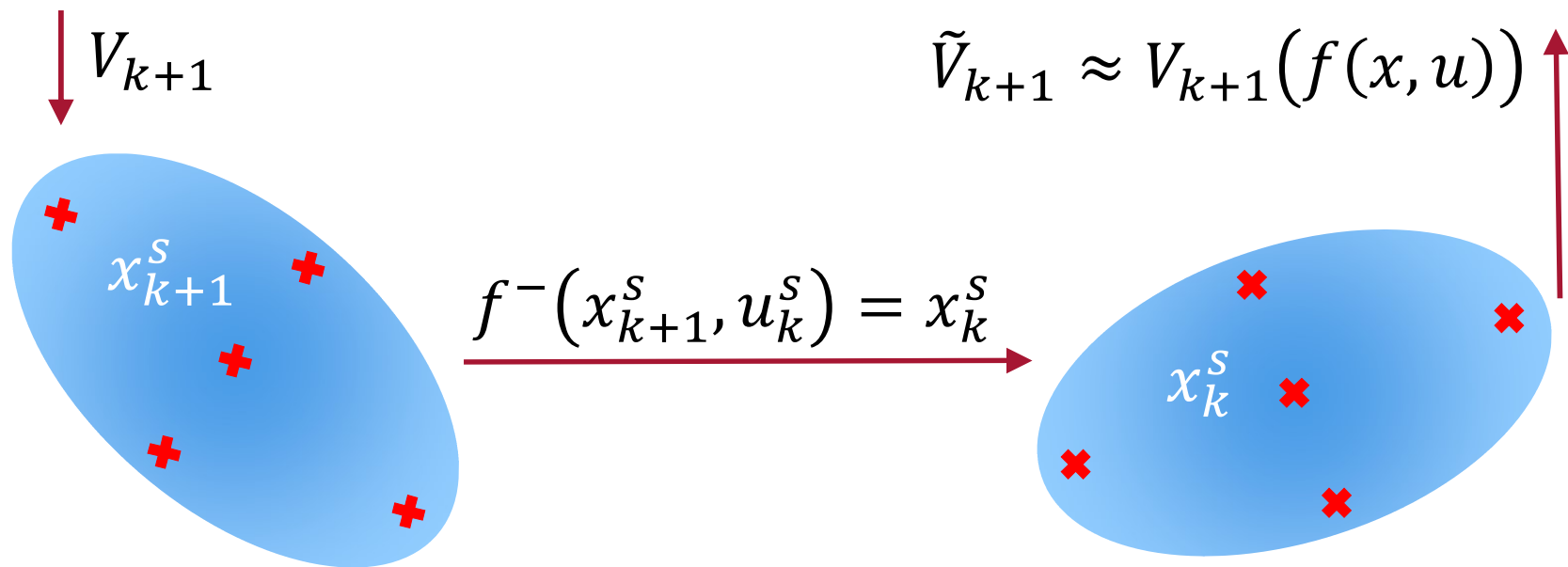
Two Historic Problems with DP Algorithms

1. **Expensive to capture the 2nd order information**
2. Hard to enforce constraints

UDP takes inspiration from the Unscented Kalman Filter to approximate the Hessian



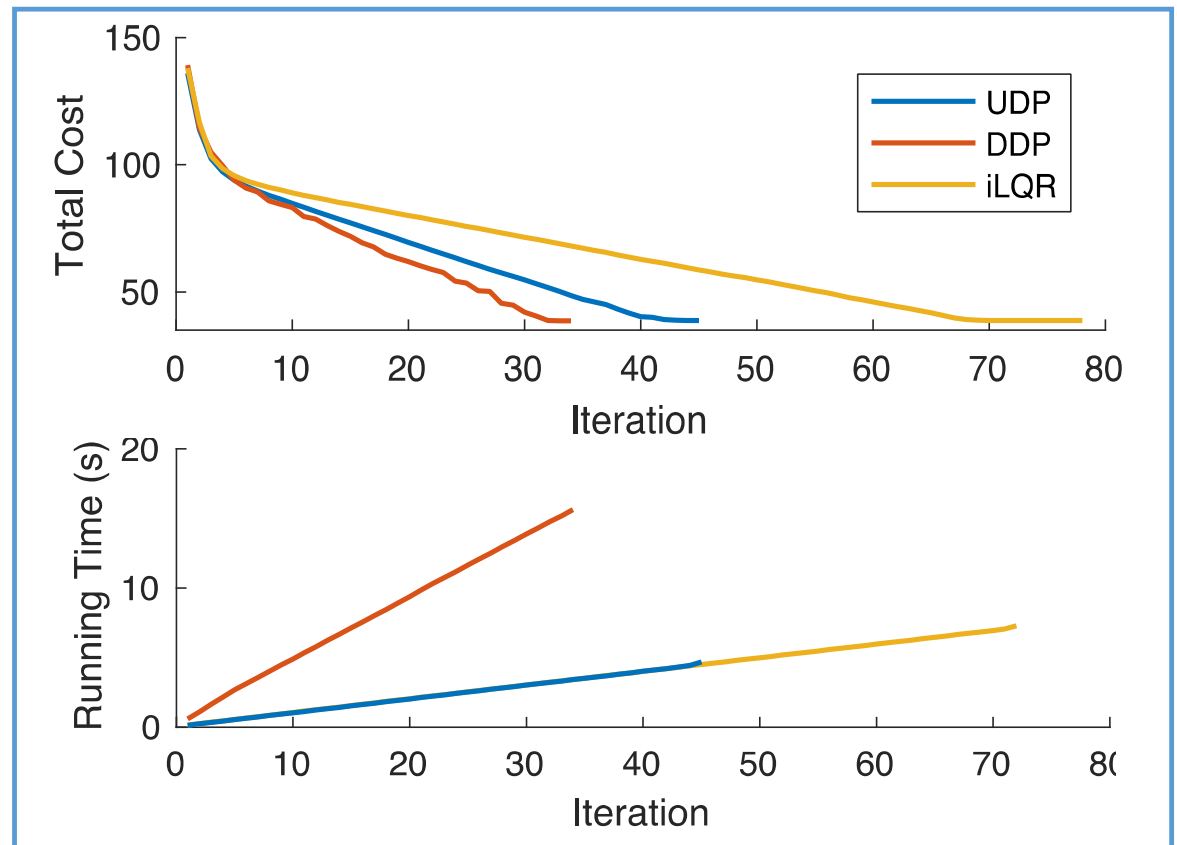
UDP takes inspiration from the Unscented Kalman Filter to approximate the Hessian



$$V_k(x) \approx \min_u l(x, u) + \tilde{V}_{k+1}$$

Experimentally UDP captures 2nd order information with first order per-iteration cost

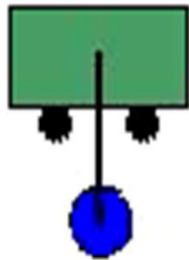
Pendulum Swing Up Task



Constrained Unscented Dynamic Programming (CUDP)

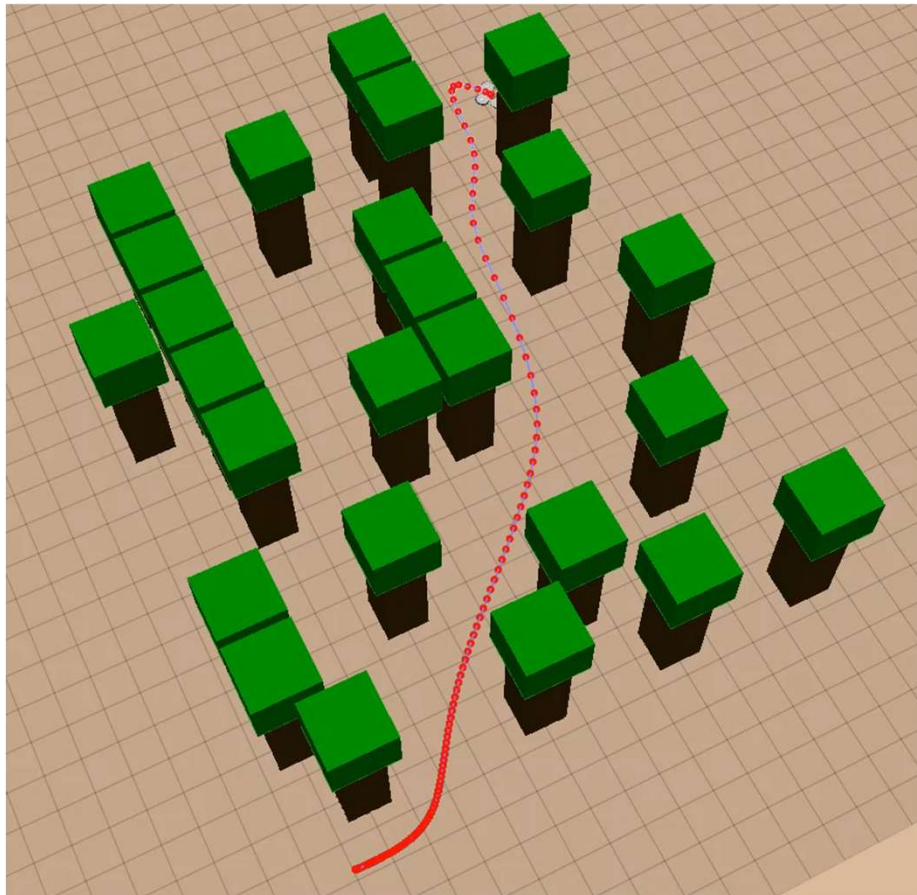
1. Compute the cost-to-go **(including constraint costs)** and the associated optimal feedback control update to the controls **backward** in time **using the unscented transform**
 2. Simulate the system **forward** in time to create a new nominal trajectory
 3. Repeat this process until convergence
 4. **At convergence test for constraint satisfaction and if not update μ, λ and go back to step 1**
-

Precise constraint satisfaction requires both the unscented transform and augmented Lagrangian



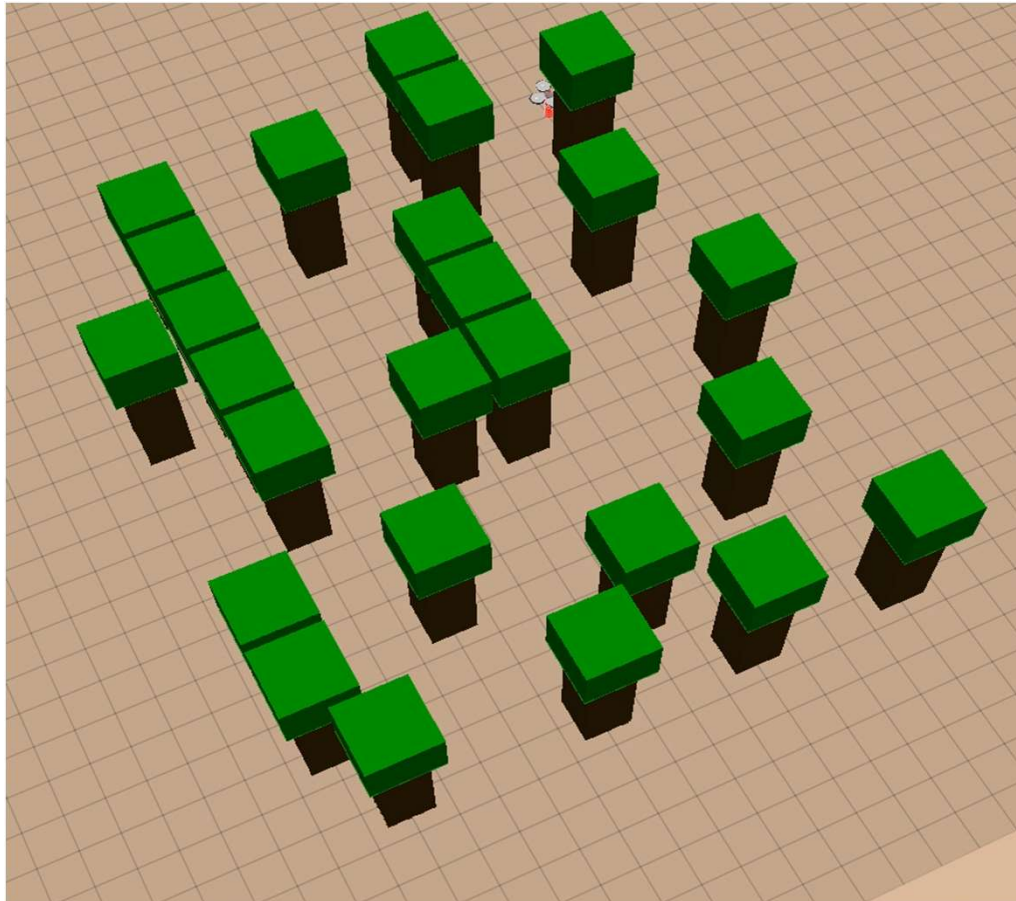
	$\Phi < 1e-2$	$\Phi < 1e-4$	$\Phi < 5e-7$
Penalty iLQR	✓	✗	✗
Penalty UDP	✓	✗	✗
AL iLQR	✓	✓	✗
AL UDP (CUDP)	✓	✓	✓
Constraints	<ul style="list-style-type: none">• Torque Limit on motor• Final state position and velocity constraint		

Precise constraint satisfaction requires both the unscented transform and augmented Lagrangian

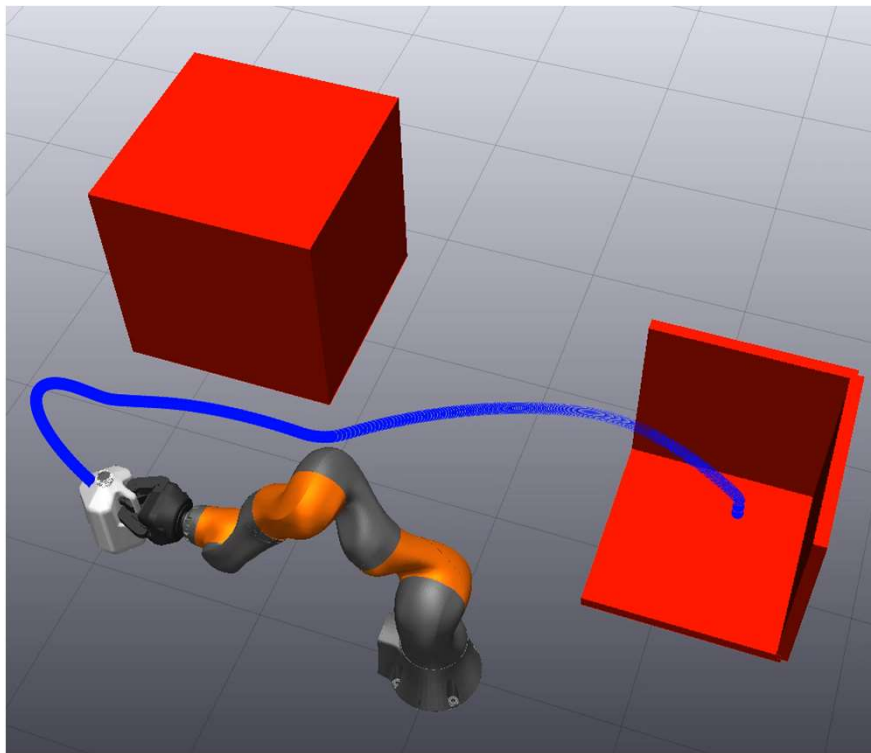


	$\Phi < 1e-2$	$\Phi < 1e-4$	$\Phi < 1e-6$
Penalty iLQR	✓	✗	✗
Penalty UDP	✓	✓	✗
AL iLQR	✓	✗	✗
AL UDP (CUDP)	✓	✓	✓
Constraints	<ul style="list-style-type: none">• Torque Limits on motors• No-contact constraints with trees• Final state position and velocity constraint		

CUDP can pass through constraint boundaries during early major iterations



Precise constraint satisfaction requires both the unscented transform and augmented Lagrangian



	5e-1 Precision	1e-2 Precision	5e-3 Precision
Penalty iLQR	✓	✗	✗
Penalty UDP	✓	✗	✗
AL iLQR	✓	✓	✗
AL UDP (CUDP)	✓	✓	✓
Constraints	<ul style="list-style-type: none">• Torque Limits on motors• No-contact constraints with block and shelf• Final state position and velocity constraint		

Constrained Unscented Dynamic Programming

- A **derivative-free** DDP/iLQR algorithm inspired by the Unscented Kalman Filter
- Uses **augmented Lagrangian** to handle nonlinear state and input constraints
- Provides **faster convergence** and **higher constraint precision** vs iLQR and penalty methods

agile.seas.harvard.edu

brian_plancher@g.harvard.edu

