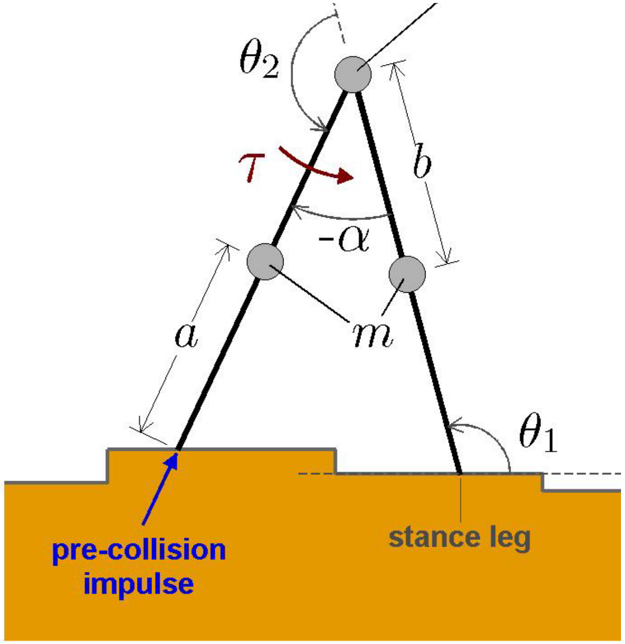
Control of the Compass Gait on Rough Terrain

Katie Byl and Russ Tedrake



Massachusetts Institute of Technology

Motivation

- **How capable can an underactuated, dynamic walking approach be on rough terrain?**
  - **Dynamic walking:**
    - Natural dynamics
    - Likely to be efficient
  - ***But unfortunately…***
    - Notoriously sensitive

- **Long-range goals:**
  - **Implement on real robot**
  - **On-line learning**

Motivation

- **Process toward obtaining underactuated, dynamic walking on rough terrain:**

  1. **Use minimal actuation and control strategies**
     - underactuation at toe

Motivation

- **Process toward obtaining underactuated, dynamic walking on rough terrain:**

1. **Use minimal actuation and control strategies**
   - underactuation at toe

2. **Quantify performance in stochastic environments**

Motivation

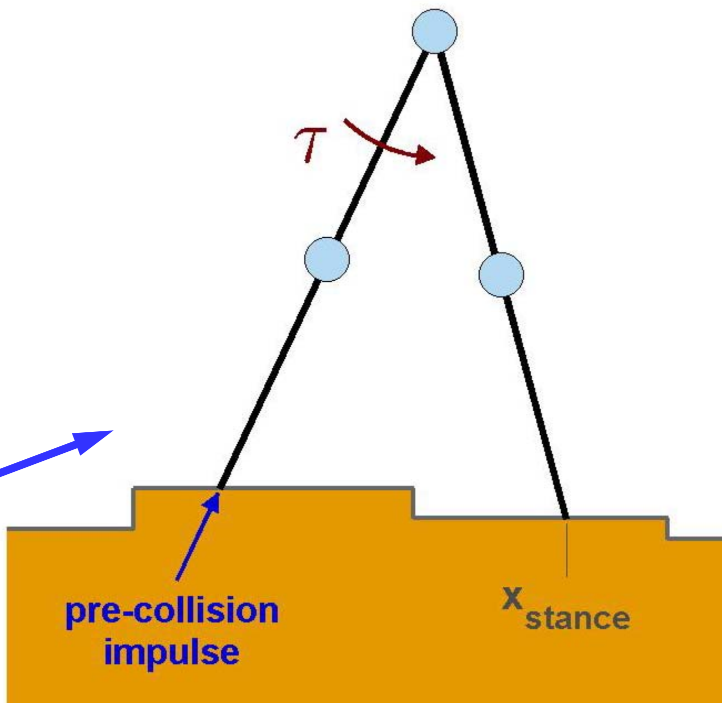
- **Process toward obtaining underactuated, dynamic walking on rough terrain:**

1. **Use minimal actuation and control strategies**
   - underactuation at toe
2. **Quantify performance in stochastic environments**
3. **Iterate to optimize performance**
   - long-living, **metastable** dynamics

Overview

- **Essential model for dynamic walking on rough terrain:**
  - **Hip-actuated compass gait (CG) with leg inertia**
  - **Passive toe pivot**

- **Outline:**
  - **Passive walker example**
  - **Actuated walkers:**
    - Stochastic terrain
    - Known, wrapping terrain

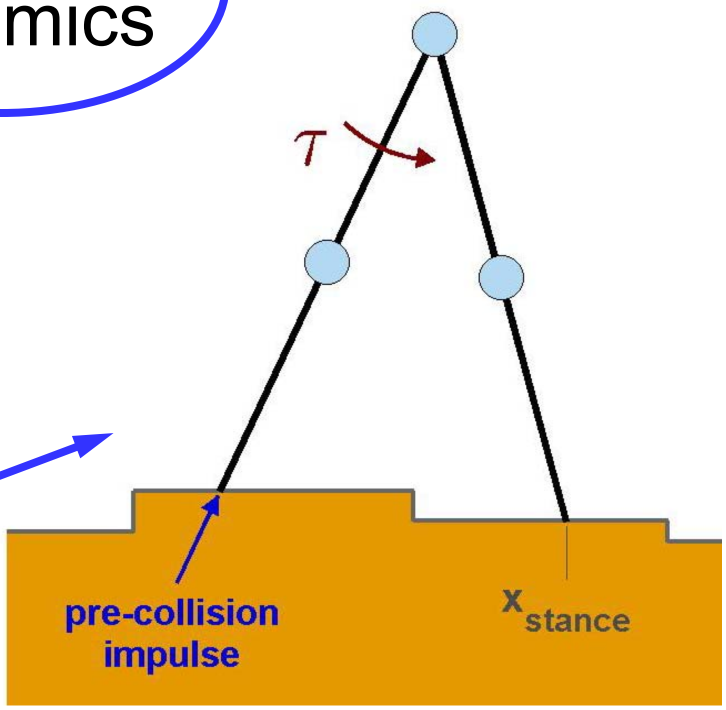


Overview

- Essential model for dynamic walking on rough terrain:
  - **Hip-actuated compass gait (CG) with leg inertia**
  - **Passive toe pivot**

acrobot dynamics

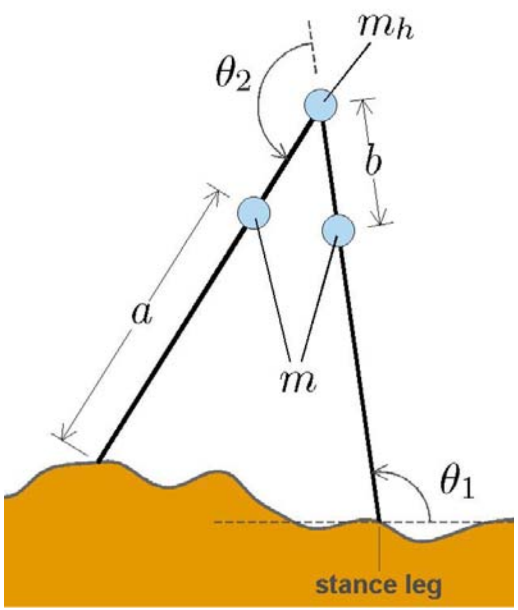
- Outline:
  - Passive walker example
  - Actuated **walkers:**
    - Stochastic terrain
    - Known, wrapping terrain



Passive Walker

- **Unactuated, with stochastic downhill terrain**

$$m = 5 \ \text{kg}$$
$$m_h = 1.5 \ \text{kg}$$
$$a = 0.7 \ \text{m}$$
$$b = 0.3 \ \text{m}$$



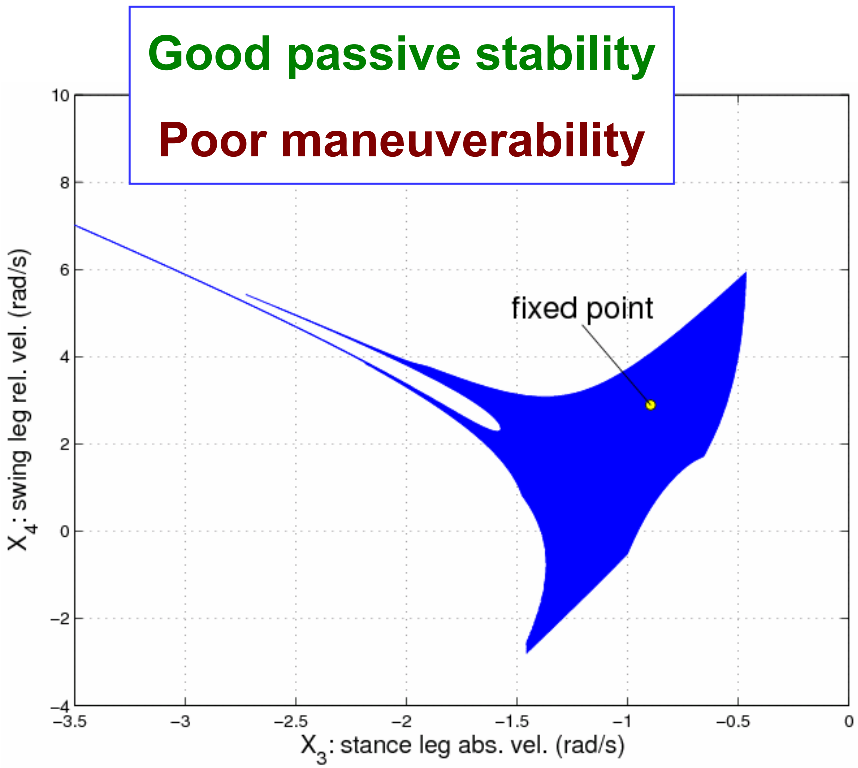
The Compass Gait Walker

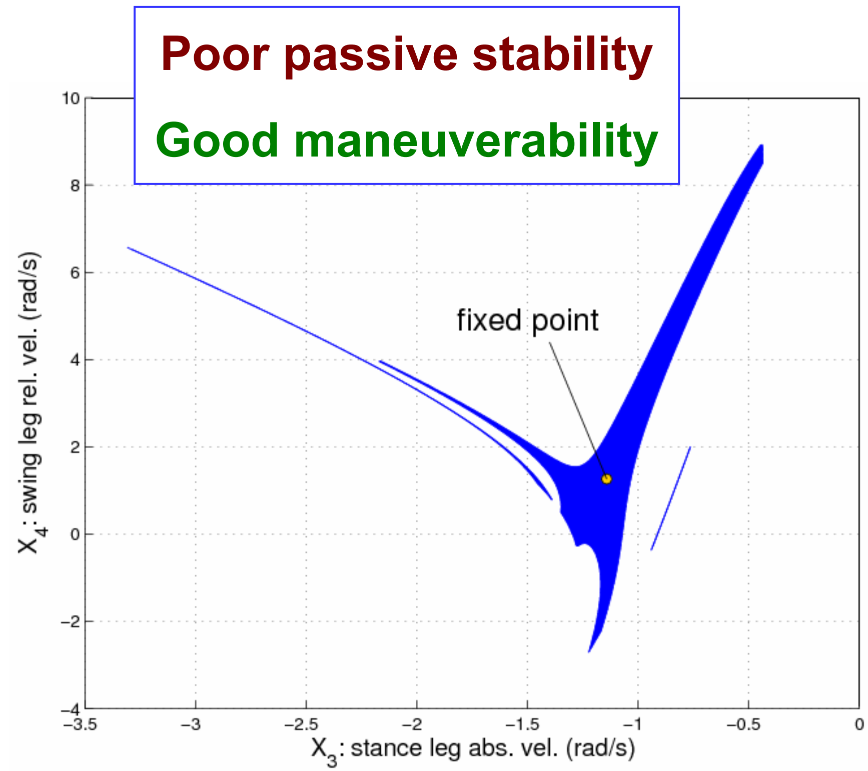
Walker animation: $\gamma_{av} = 4^\circ$, $\sigma = 1.0^\circ$

(Only first 5 sec will be animated...)

Passive Walker

- **Constant 4º** downhill slope (no noise)

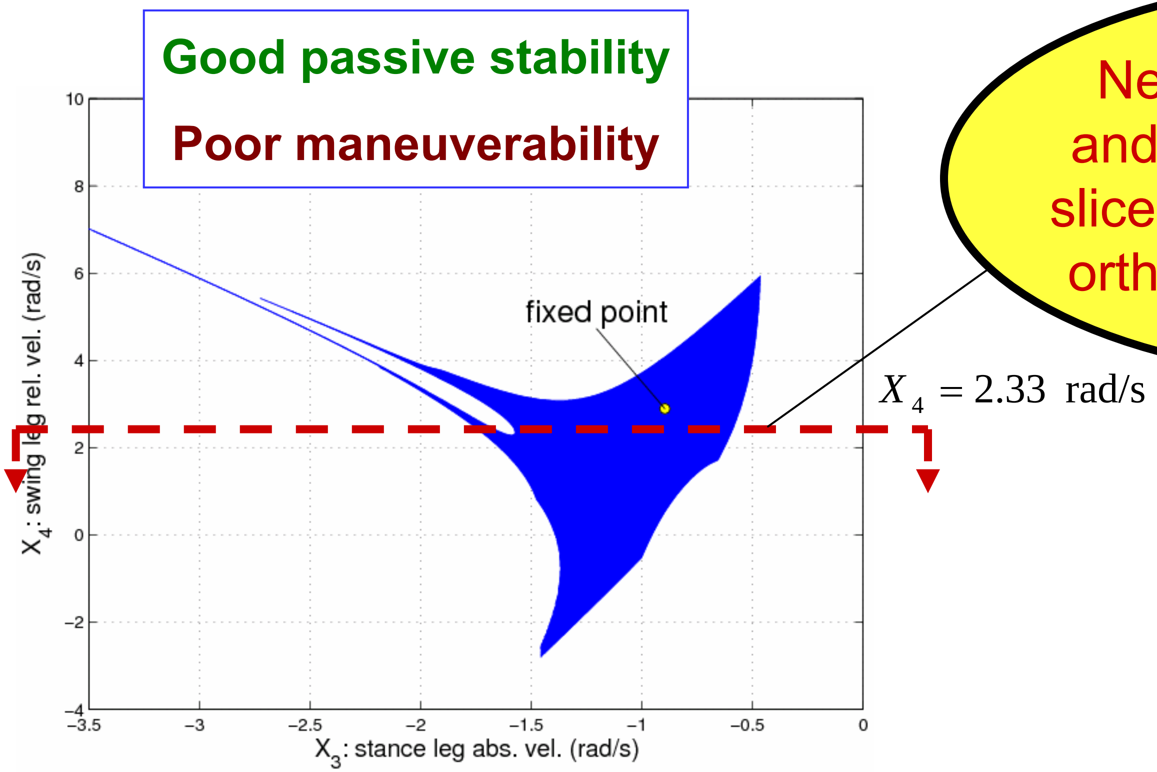




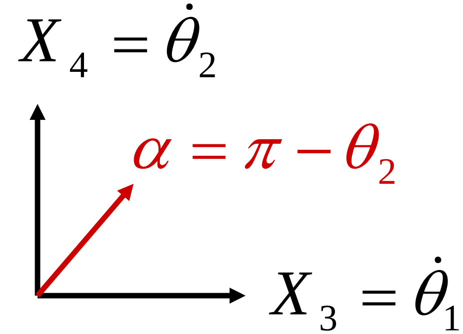
Slices of the deterministic **Basins of Attractions** for the walkers analyzed for **passive** (left) and **controlled** (right) examples throughout.

Passive Walker

- **Constant 4º downhill slope (no noise)**



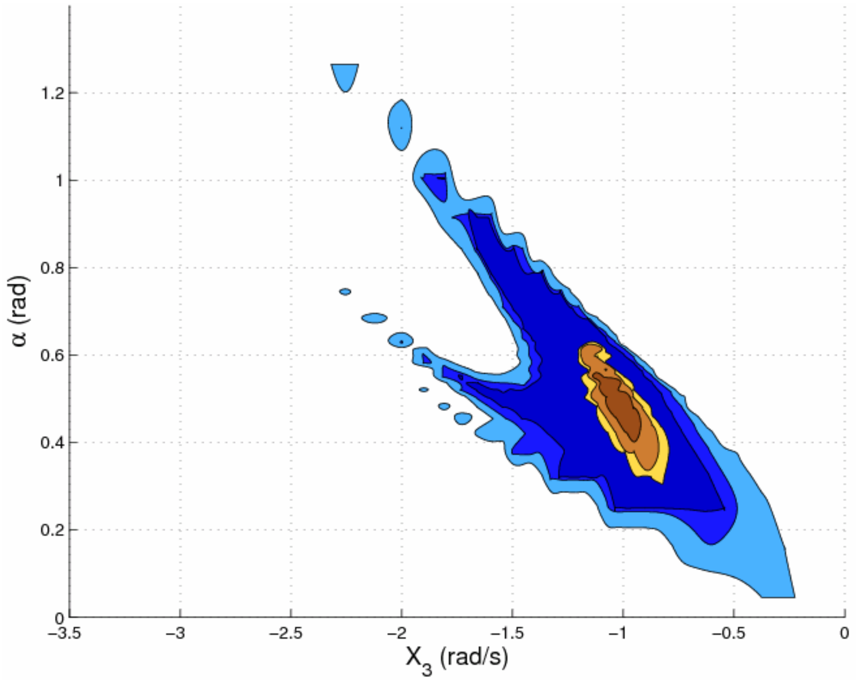
Next, we will add noise and look at a different 2D slice in the 3D state space, orthogonal to this one . . .



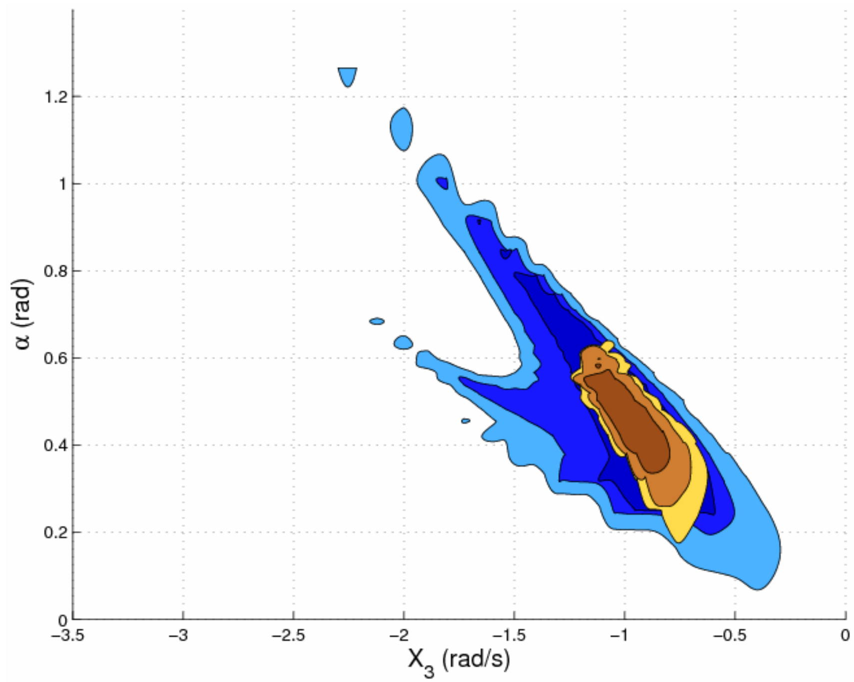
Slice of the deterministic **Basins of Attraction** for the walker analyzed for **passive** examples throughout.

Passive Walker

- **Stochastic downhill terrain, mean slope = 4º**



$$\sigma = 0.5^\circ$$
$$\text{mfpt} \approx 200{,}000$$

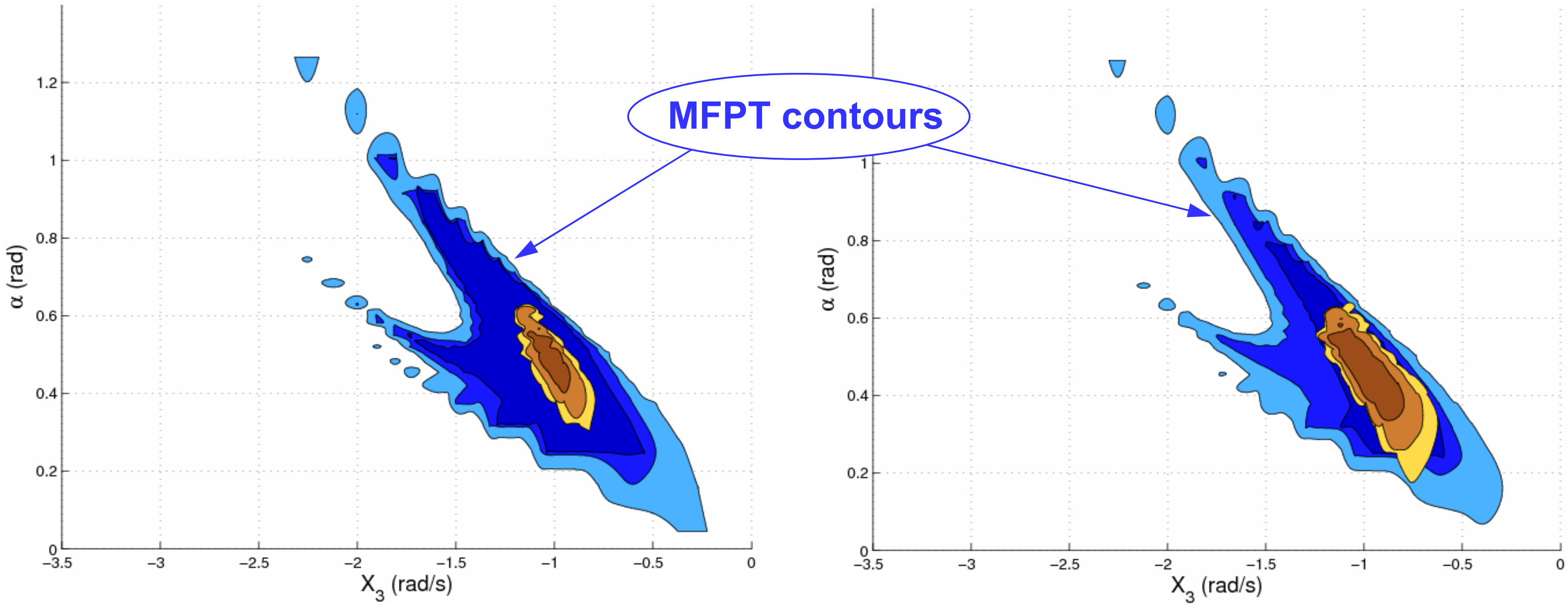


$$\sigma = 1.0^\circ$$
$$\text{mfpt} \approx 300$$

(**mfpt :** mean first-passage time)

Passive Walker

- **Stochastic** downhill terrain, **mean slope = 4º**



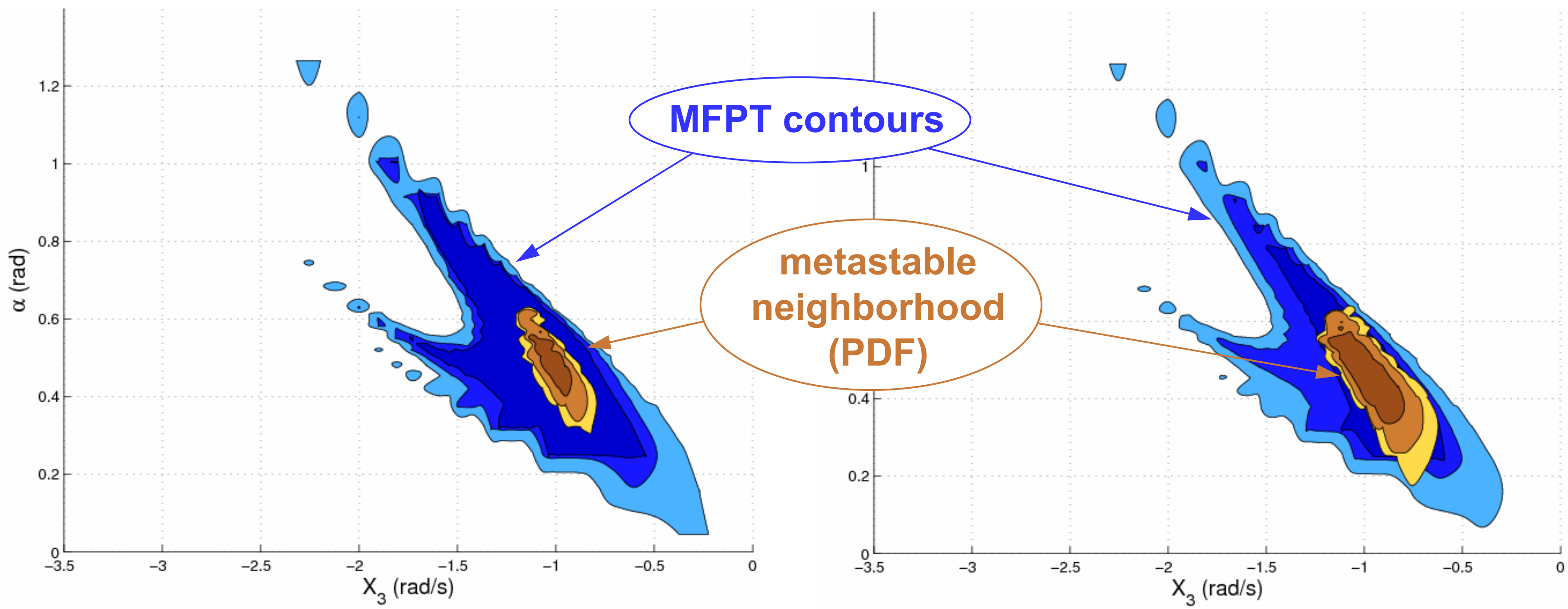
$\sigma = 0.5^\circ$
mfpt ≈ 200,000

$\sigma = 1.0^\circ$
mfpt ≈ 300

(**mfpt :** mean first-passage time)

Passive Walker

- **Stochastic downhill terrain, mean slope = 4º**



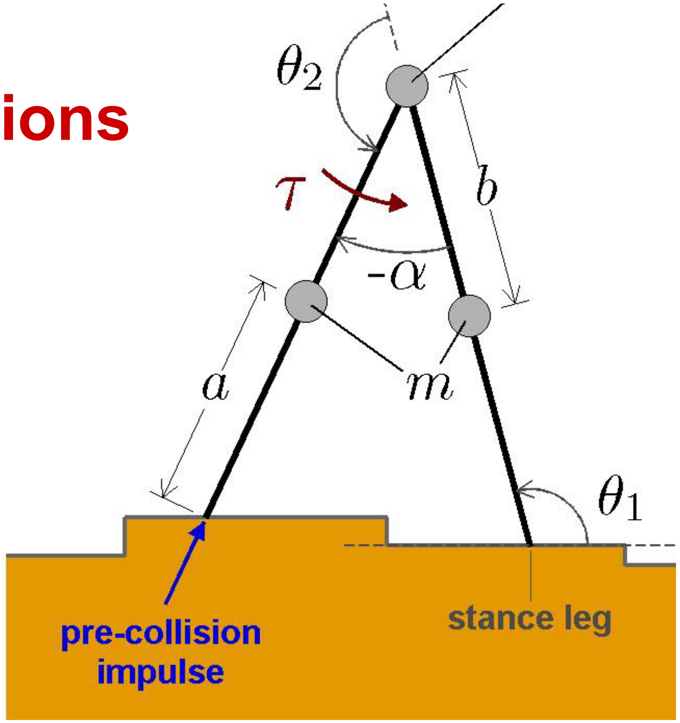
$$\sigma = 0.5^\circ$$
$$\text{mfpt} \approx 200{,}000$$

$$\sigma = 1.0^\circ$$
$$\text{mfpt} \approx 300$$

(**mfpt :** mean first-passage time)

Actuated Walker Models

- **Compass gait (CG)**
  - **Point masses** at hip (m_h) and on each leg (m)
    - $m = m_h = 2\ \text{kg} \quad ; \quad a = b = 0.5\ \text{m}$
  - **Passive pivot** model for "toe" of stance leg
  - **5 States:** $\theta_1,\ \theta_2,\ \dot{\theta}_1,\ \dot{\theta}_2,\ \Delta z$
  - **Instantaneous, inelastic collisions**



- **Actuations**
  - Torque at hip:
    - **+/- 15 N-m limit**
  - Pre-collision impulse:
    - Constant value of **2 kg-m/s**

Methodology

- **Solve iteratively to find optimal policy**
  - **Mesh** state space, using **post-collision** states
  - Define cost function **to reward continuous walking**

Methodology

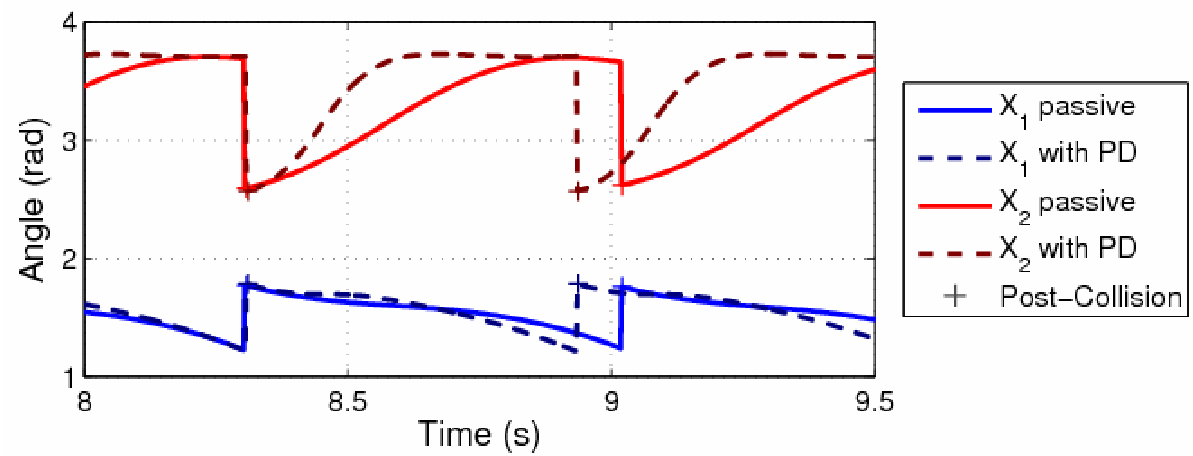
- **Solve iteratively to find optimal policy**
  - **Mesh** state space, using **post-collision** states
  - Define cost function **to reward continuous walking**
- **Hierarchical control**
  - Low-level PD control: $\tau = K_p(\alpha_{des} - \alpha) - K_d\dot{\alpha}$
  - High-level, **once-per-step** selection of $\boldsymbol{\alpha_{des}}$

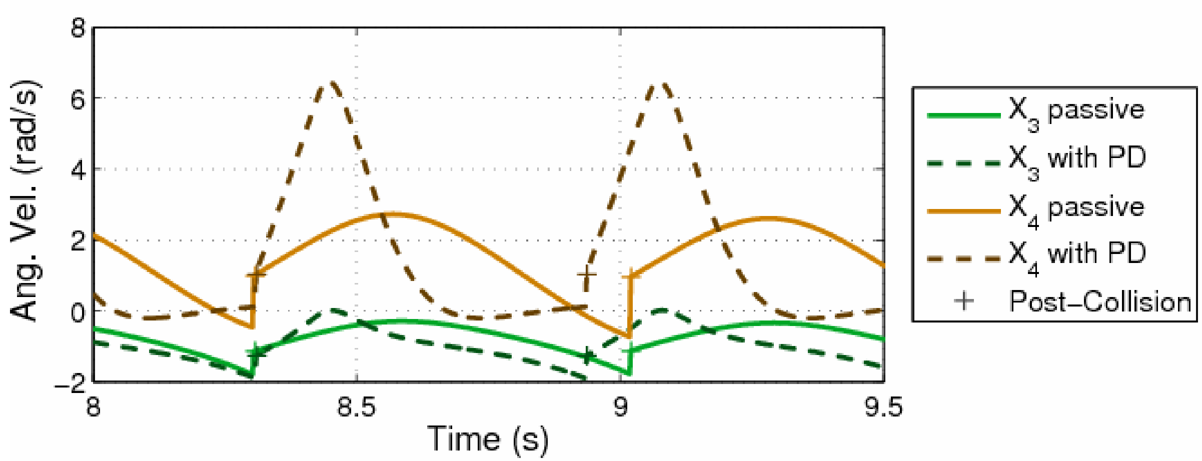
Methodology

- **Solve iteratively to find optimal policy**
  - **Mesh** state space, using **post-collision** states
  - Define cost function **to reward continuous walking**
- **Hierarchical control**
  - Low-level PD control: $\tau = K_p(\alpha_{des} - \alpha) - K_d \dot{\alpha}$
  - High-level, **once-per-step** selection of $\boldsymbol{\alpha_{des}}$
- **Additional Details**
  - Stochastic terrain, **Δz from a Gaussian**
  - Swing toe **retracts** until α is within 10º of $\boldsymbol{\alpha_{des}}$
  - PD controller is always active during step

Low-level PD Control at Hip

- **PD state trajectories versus passive downhill walking**





Note: While positive and negative work is done for active case, overall gait speed is only about 10% faster than passive walker.

PD control only, with no impulsive toe-off: $\alpha_{des} = 35°$

Constant 4° downhill, to compare active with passive

Meshing: stochastic terrain

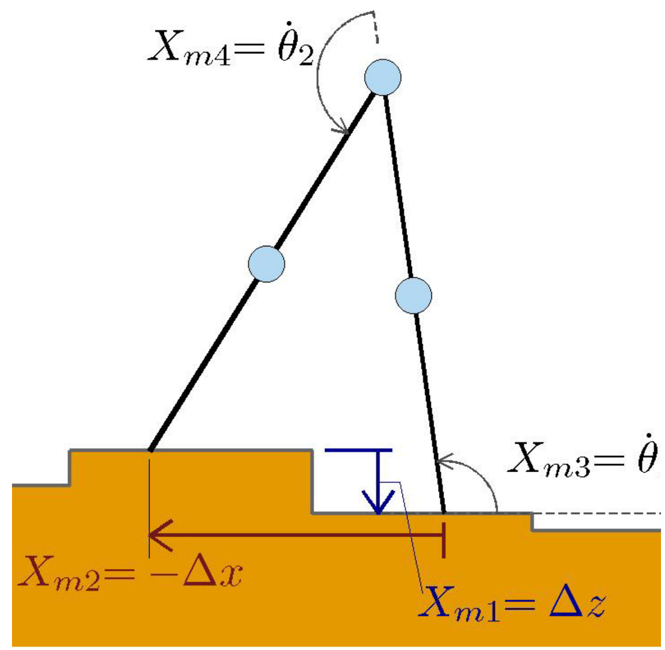
- **Post-collision meshing using 4 state variables**

$$X_{m1} = \Delta z = z_{st} - z_{sw}$$
$$X_{m2} = -\Delta x = x_{sw} - x_{st}$$
$$X_{m3} = \dot{\theta}_1$$
$$X_{m4} = \dot{\theta}_2$$

Including one extra "**fallen**" state, there are **19,001 mesh states**

state	# elem's	min	max	units
X_{m1}	19	-.01	.01	(m)
X_{m2}	10	-0.7	-0.16	(m)
X_{m3}	10	-2.1	-1.1	(rad/s)
X_{m4}	10	-1	1.5	(rad/s)

- **Action**, α_{des} **: 15 - 40 deg (11 values)**

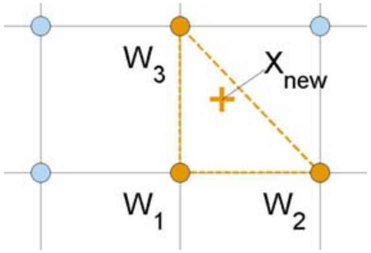


- **Interpolation (barycentric)**

Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**

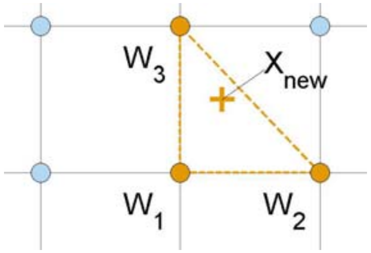
  Each new state in N-dim space represented by **N+1** weighted mesh nodes, each with weight W_k



Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**

  Each new state in N-dim space represented by **N+1** weighted mesh nodes, each with weight W_k



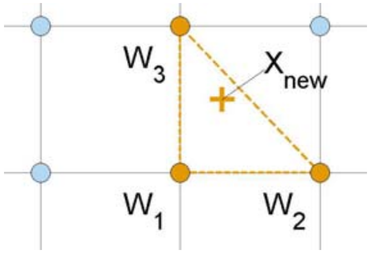
- **Define one-step cost; initialize C_{last}=$C_{onestep}$**

$$C_{onestep}(i) = \begin{cases} -1, & i \notin fallen \\ 0\ , & i \in fallen \end{cases}$$

Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**

  Each new state in N-dim space represented by **N+1** weighted mesh nodes, each with weight W_k



- **Define one-step cost; initialize $C_{last}=C_{onestep}$**

$$C_{onestep}(i) = \begin{cases} -1, & i \notin fallen \\ 0\ , & i \in fallen \end{cases}$$

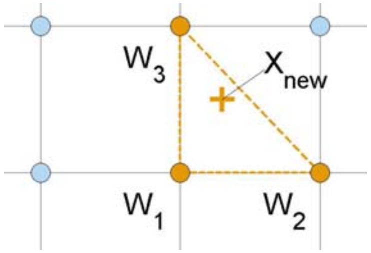
One-step cost of -1 maximizes steps taken before falling.

To maximize distance traveled, instead use: $C_{onestep}(i) = X_{m2}$

Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**

  Each new state in N-dim space represented by **N+1** weighted mesh nodes, each with weight W_k



- **Define one-step cost; initialize $C_{last}=C_{onestep}$**

$$C_{onestep}(i) = \begin{cases} -1, & i \notin fallen \\ 0\ , & i \in fallen \end{cases}$$

One-step cost of -1 maximizes steps taken before falling.

To maximize distance traveled, instead use: $C_{onestep}(i) = X_{m2}$

- **Iterate to minimize cost:**

$$C_{new}(i \mid a) = \sum_{k=1}^{5} W_k \left[\gamma \cdot C_{last}(k) + C_{onestep}(k)\right]$$

$$\gamma = 0.9$$

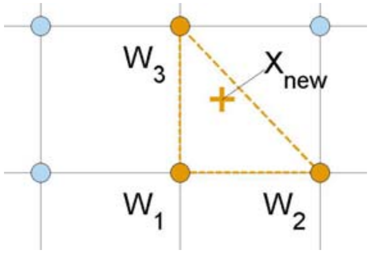
Iterative updates:

$$C_{last}(i) = C_{new}(i), \quad \forall i \qquad\qquad \pi(i) = \arg\min_a C_{new}(i \mid a)$$

Dynamic Programming (Value Iteration)

- **Pre-compute one-step dynamics**

  Each new state in N-dim space represented by **N+1** weighted mesh nodes, each with weight W_k



- **Define one-step cost; initialize $C_{last}=C_{onestep}$**

$$C_{onestep}(i) = \begin{cases} -1, & i \notin fallen \\ 0\ , & i \in fallen \end{cases}$$

One-step cost of -1 maximizes steps taken before falling.

To maximize distance traveled, instead use: $C_{onestep}(i) = X_{m2}$

- **Iterate to minimize cost:**

$$C_{new}(i \mid a) = \sum_{k=1}^{5} W_k \left[\gamma \cdot C_{last}(k) + C_{onestep}(k) \right]$$

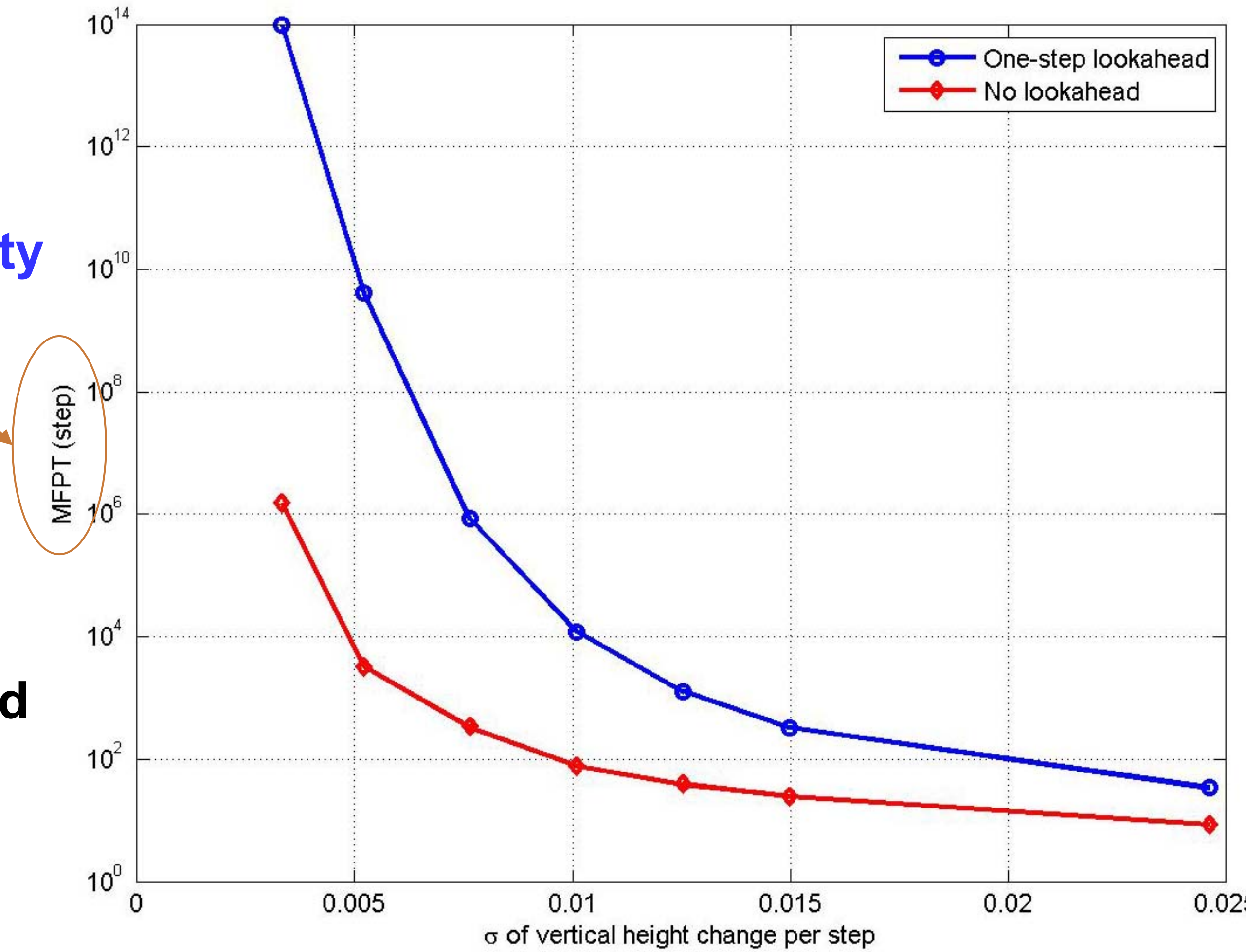
$$\gamma = 0.9$$

Iterative updates:

$$C_{last}(i) = C_{new}(i), \quad \forall i \qquad\qquad \pi(i) = \arg\min_a C_{new}(i \mid a)$$

Control on Stochastic Terrain

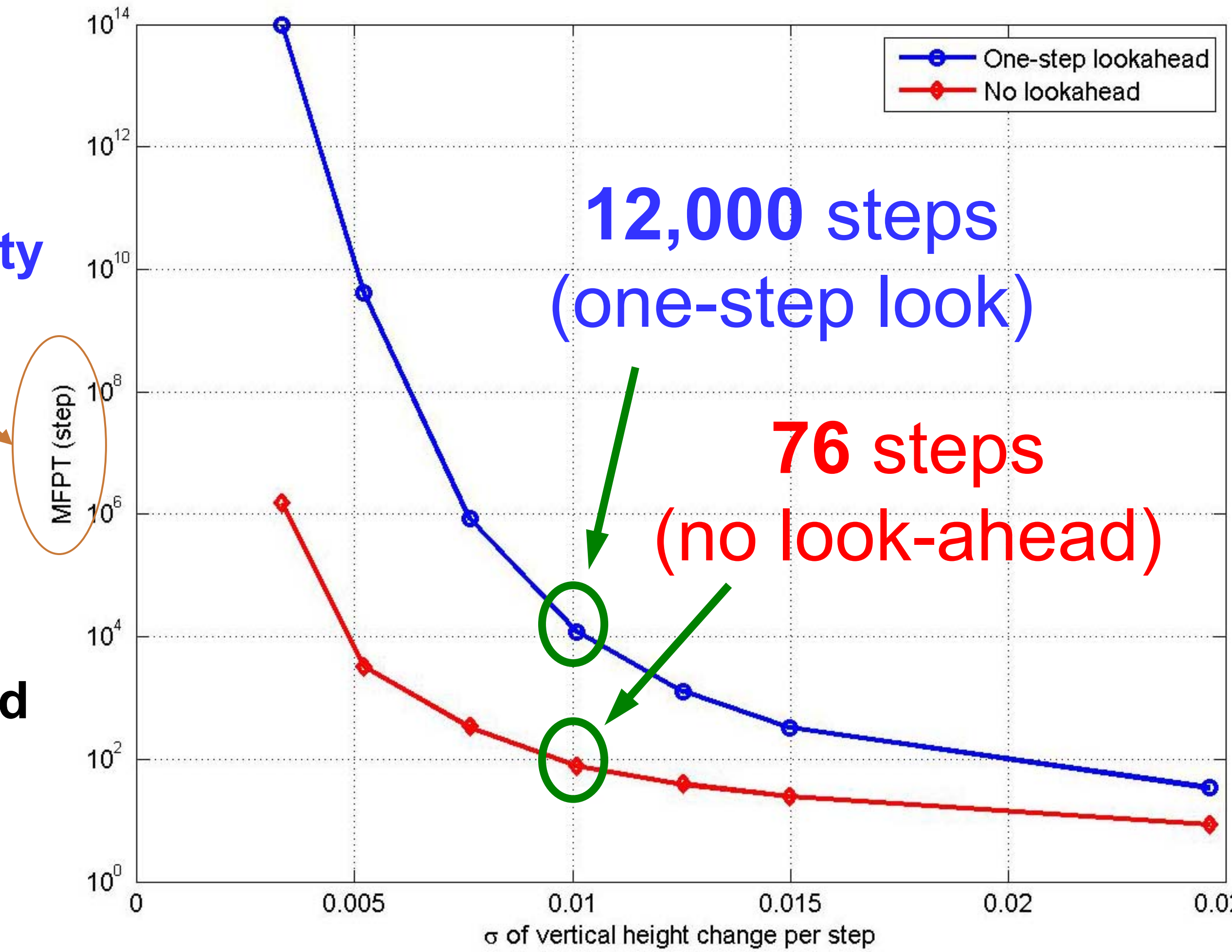
Mean first-passage time, MFPT, used to quantify stability



One-step look-ahead improves policy significantly

Control on Stochastic Terrain

Mean first-passage time, MFPT, used to quantify stability

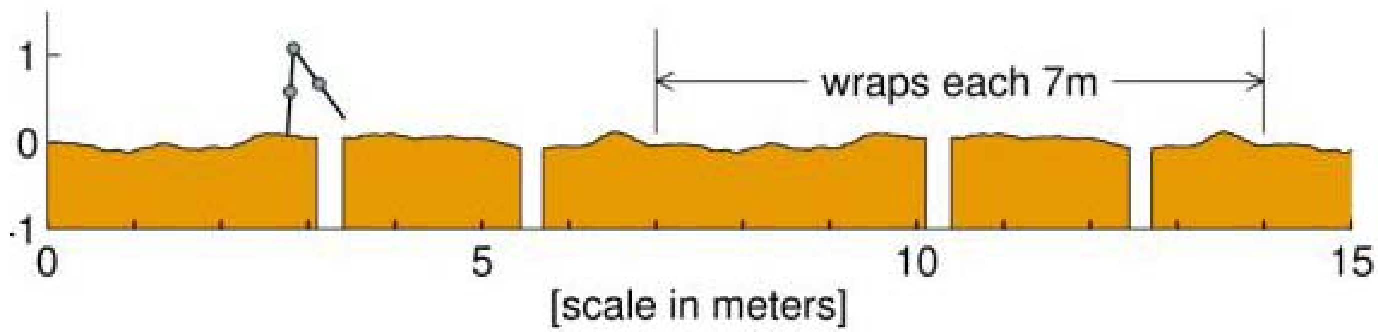


One-step look-ahead improves policy significantly

Control on Wrapping Terrain

- **For stochastic terrain:**
  - **N-step look-ahead requires 4+N total mesh dimensions**

- **Advantages of known, wrapping terrain:**
  - **Allows N-step look-ahead using only 4 mesh dimensions (4D)**
  - **N steps occur in iteration algorithm, not state representation**



Meshing: known, wrapping terrain

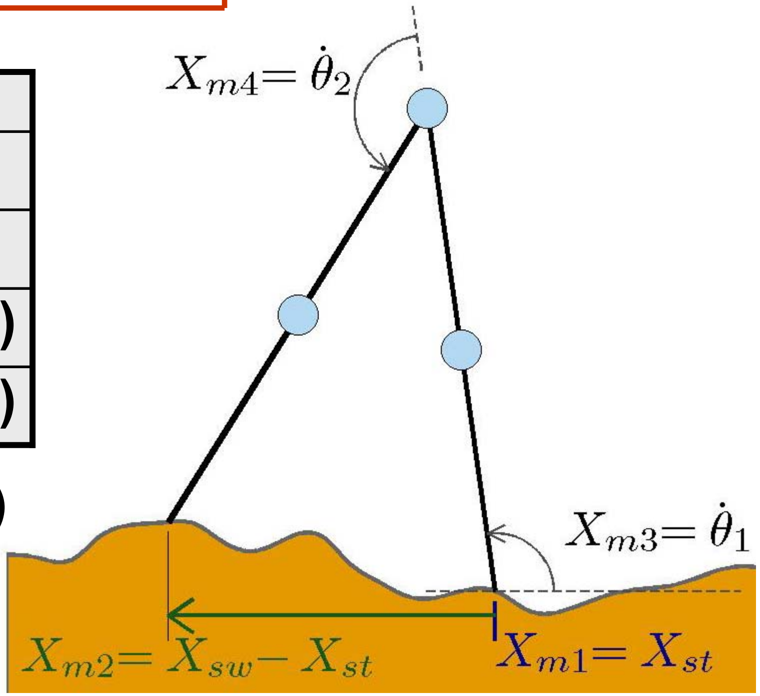
- **Post-collision meshing using 4 state variables**

$$X_{m1} = x_{st}$$
$$X_{m2} = \Delta x = x_{st} - x_{sw}$$
$$X_{m3} = \dot{\theta}_1$$
$$X_{m4} = \dot{\theta}_2$$

Including one extra "**fallen**" state, there are **411,601 mesh states**

state	# elem's	min	max	units
X_{m1}	140	0	7	(m)
X_{m2}	15	-0.85	-0.15	(m)
X_{m3}	14	-3.0	-0.4	(rad/s)
X_{m4}	14	-0.1	5.1	(rad/s)

- **Action**, α_{des} : 10 - 40 deg (13 values)



- **Interpolation (barycentric)**

Meshing: known, wrapping terrain

- **Post-collision meshing using 4 state variables**

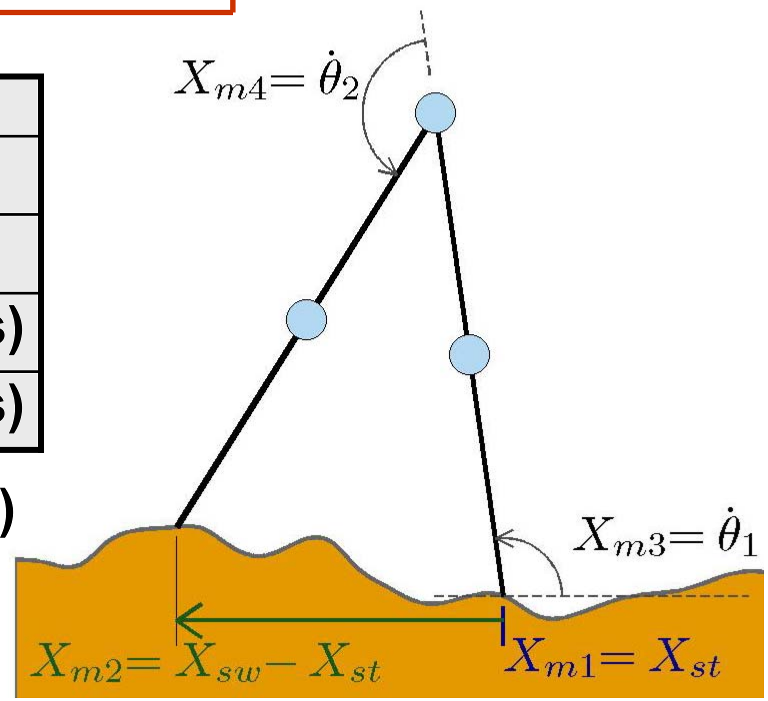
$$X_{m1} = x_{st}$$
$$X_{m2} = \Delta x = x_{st} - x_{sw}$$
$$X_{m3} = \dot{\theta}_1$$
$$X_{m4} = \dot{\theta}_2$$

only 1st state variable is different from stochastic modeling case

Including one extra "**fallen**" state, there are **411,601 mesh states**

state	# elem's	min	max	units
X_{m1}	140	0	7	(m)
X_{m2}	15	-0.85	-0.15	(m)
X_{m3}	14	-3.0	-0.4	(rad/s)
X_{m4}	14	-0.1	5.1	(rad/s)

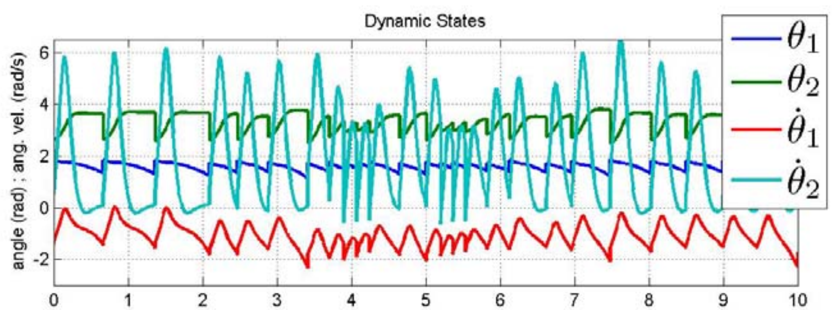
- **Action**, α_{des} : **10 - 40 deg (13 values)**

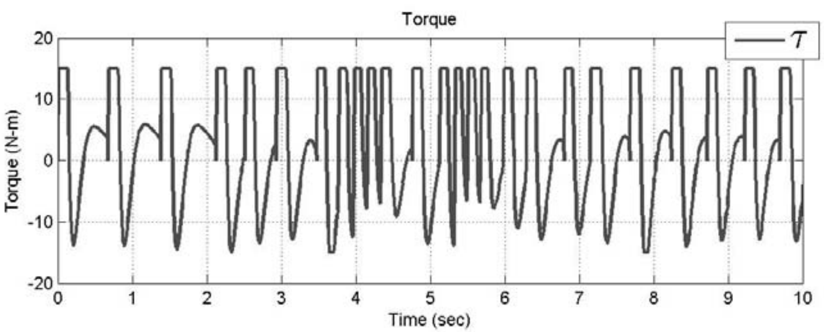


- **Interpolation (barycentric)**

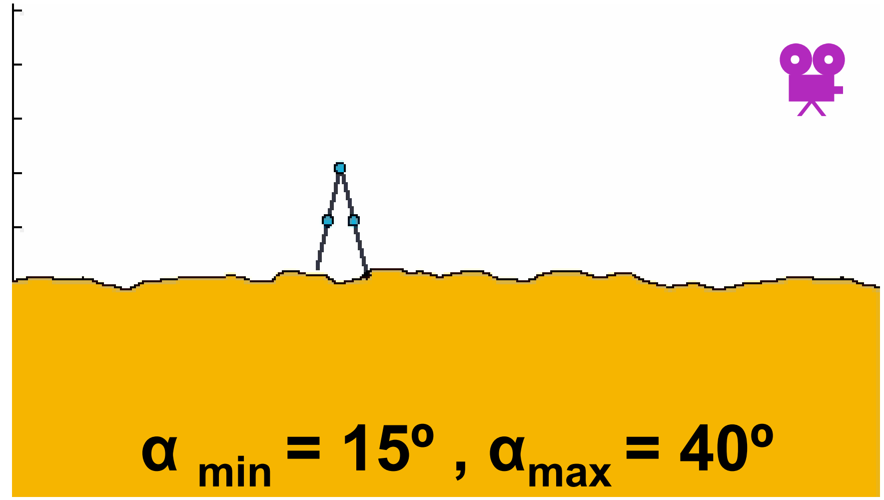
Results on Wrapping Terrain

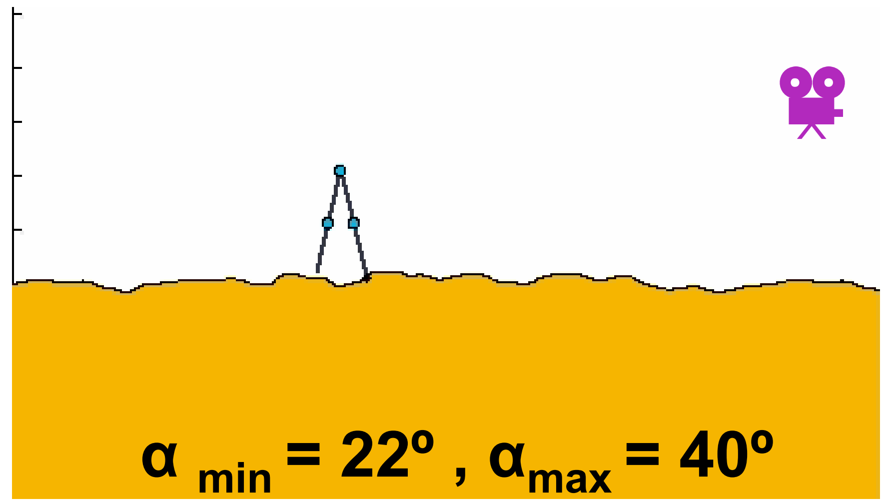
- **PD with impulsive toe-off**
  - **α is desired interleg angle**





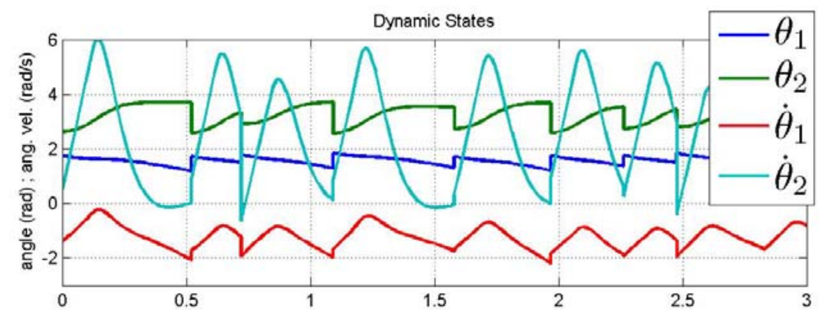
First **10** seconds of data

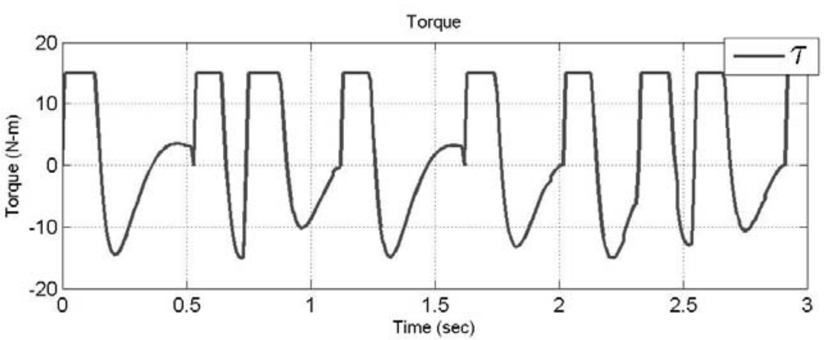




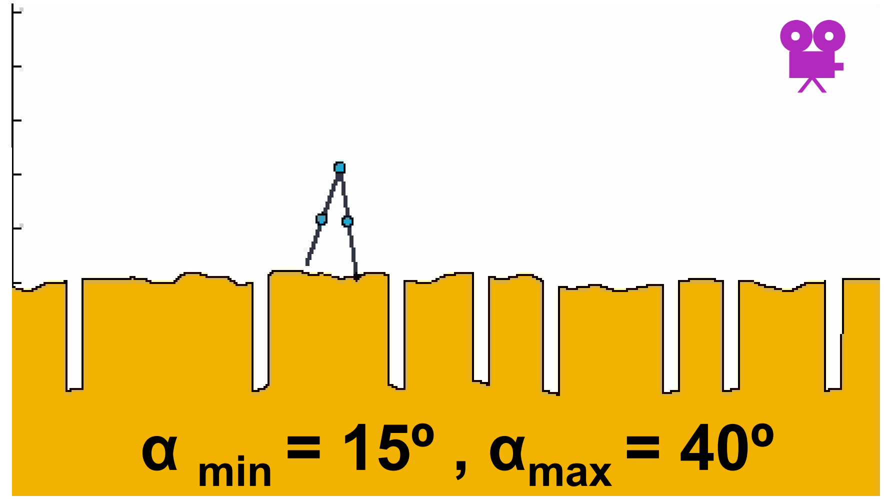
Results on Wrapping Terrain

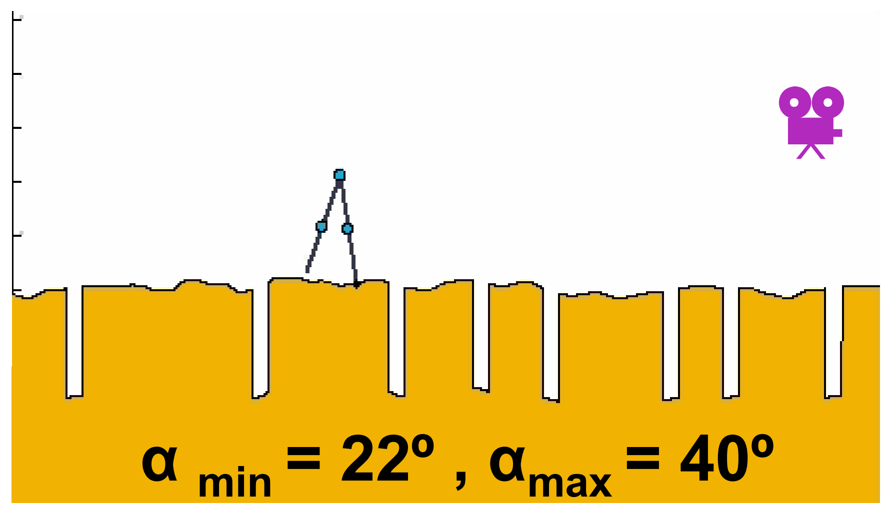
- **PD with impulsive toe-off**
  - **Gaps yield more pattern in footholds**





First **3** seconds of data

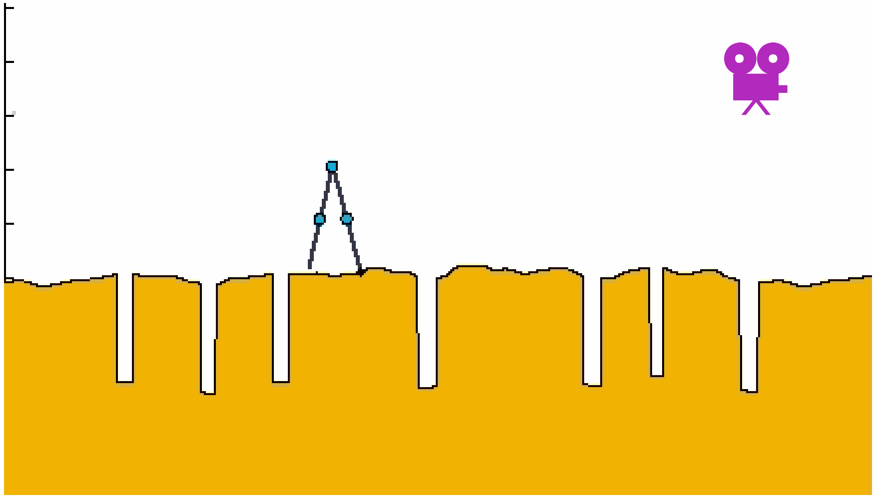


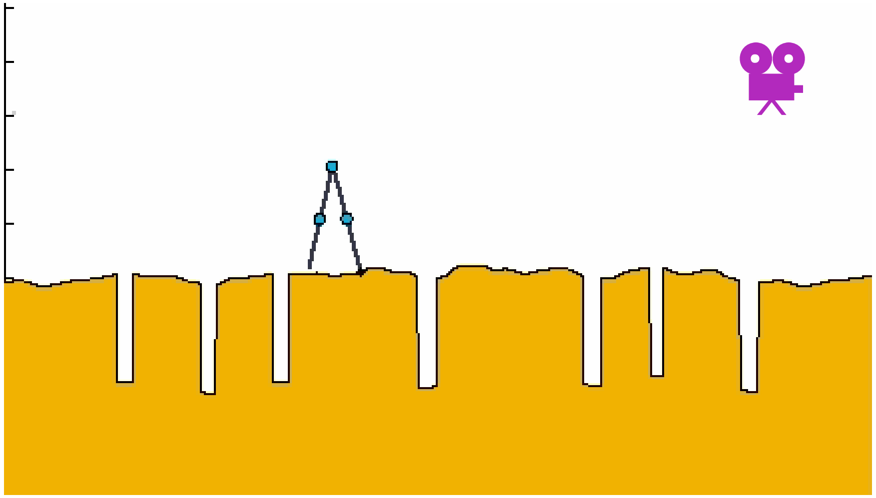


Discussion: One-step policy

- **Using heuristic cost functions on the <u>wrapping</u> mesh state also yields impressive results**
  - Implies lengthy value iteration computation and/or exact description of terrain are not essential
- **Although surprisingly good, one-step policy is inferior**
  - Performance sensitive to one-step heuristic used

Animations below use *only <u>slightly</u> different* one-step heuristics…





Future Work

- Use off-line policy from simulation as basis for **on-line policy learning** on **real robot**
  - **Direct-drive** hip torque
  - **Retracting toe**
  - Motor **encoder**
  - **Boom-mounted**
    - Repeating terrain
  - **Motion capture:**
    - Leg markers
    - Terrain markers

- Maximize **expected number of steps taken**

Summary

- Compass gait model with **hip torque** and **toe impulse** can negotiate **qualitatively rough terrain**

Summary

- Compass gait model with **hip torque** and **toe impulse** can negotiate **qualitatively rough terrain**

- Apply analytical tools toward creating **metastable locomotion**

Summary

- Compass gait model with **hip torque** and **toe impulse** can negotiate **qualitatively rough terrain**
- Apply analytical tools toward creating **metastable locomotion**
- **One-step look-ahead** greatly improves performance

Summary

- Compass gait model with **hip torque** and **toe impulse** can negotiate **qualitatively rough terrain**

- Apply analytical tools toward creating **metastable locomotion**

- **One-step look-ahead** greatly improves performance

- *What is possible if **better low-level control** is used?!?*

Summary

- Compass gait model with **hip torque** and **toe impulse** can negotiate **qualitatively rough terrain**
- Apply analytical tools toward creating **metastable locomotion**
- **One-step look-ahead** greatly improves performance
- *What is possible if **better low-level control** is used?!?*

- Same approach already shown to work on **known, wrapping terrain**:
  Byl and Tedrake, ICRA 2008 ➡ link to ICRA 2008 paper
- **Metastable walking** described further in upcoming work:
  Byl and Tedrake, RSS 2008 ➡ link to RSS 2008 paper

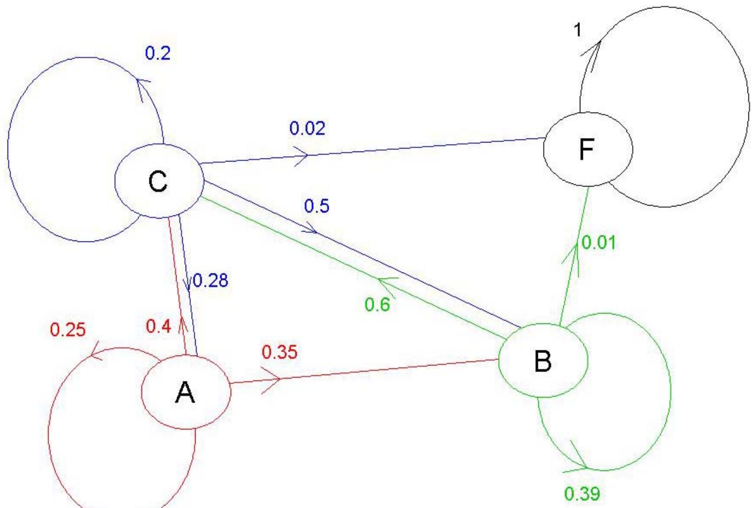
Questions?

Additional slides

- Details on **eigenanalysis** of discrete system
- **More results** on known, wrapping terrain
- Important details on **interpolation method**
- Fragility of **impulse-only** strategy
- **Dynamic** motion planning for a **stiff robot**

Eigenanalysis

- **Discretized system is a Markov chain**
  - **Analyze corresponding transition matrix**

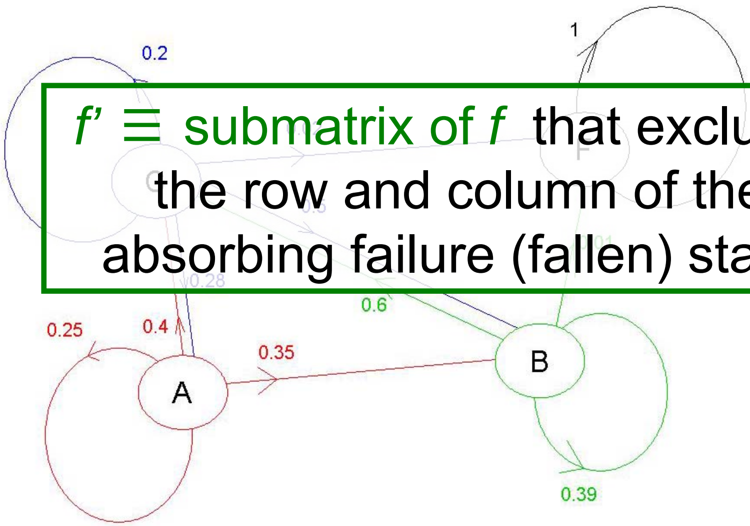


$$\boldsymbol{f} = \begin{bmatrix} 0.25 & 0.35 & 0.4 & 0 \\ 0 & 0.39 & 0.6 & 0.01 \\ 0.28 & 0.5 & 0.2 & 0.02 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$m_i = \sum_j f_{ij} m_j + 1 \quad \longrightarrow \quad \left(\mathbf{I} - \mathbf{f}'\right)\mathbf{m} = \mathbf{1} \quad \longrightarrow \quad \mathbf{m} = \left(\mathbf{I} - \mathbf{f}'\right)^{-1}\mathbf{1}$$

Eigenanalysis

- **Discretized system is a Markov chain**
  - **Analyze corresponding transition matrix**



f' ≡ submatrix of *f* that excludes the row and column of the absorbing failure (fallen) state.

$$\boldsymbol{f} = \begin{bmatrix} 0.25 & 0.35 & 0.4 & 0 \\ 0 & 0.39 & 0.6 & 0.01 \\ 0.28 & 0.5 & 0.2 & 0.02 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{m} \equiv$ mean first-passage time (MFPT)

$$m_i = \sum_j f_{ij} m_j + 1$$

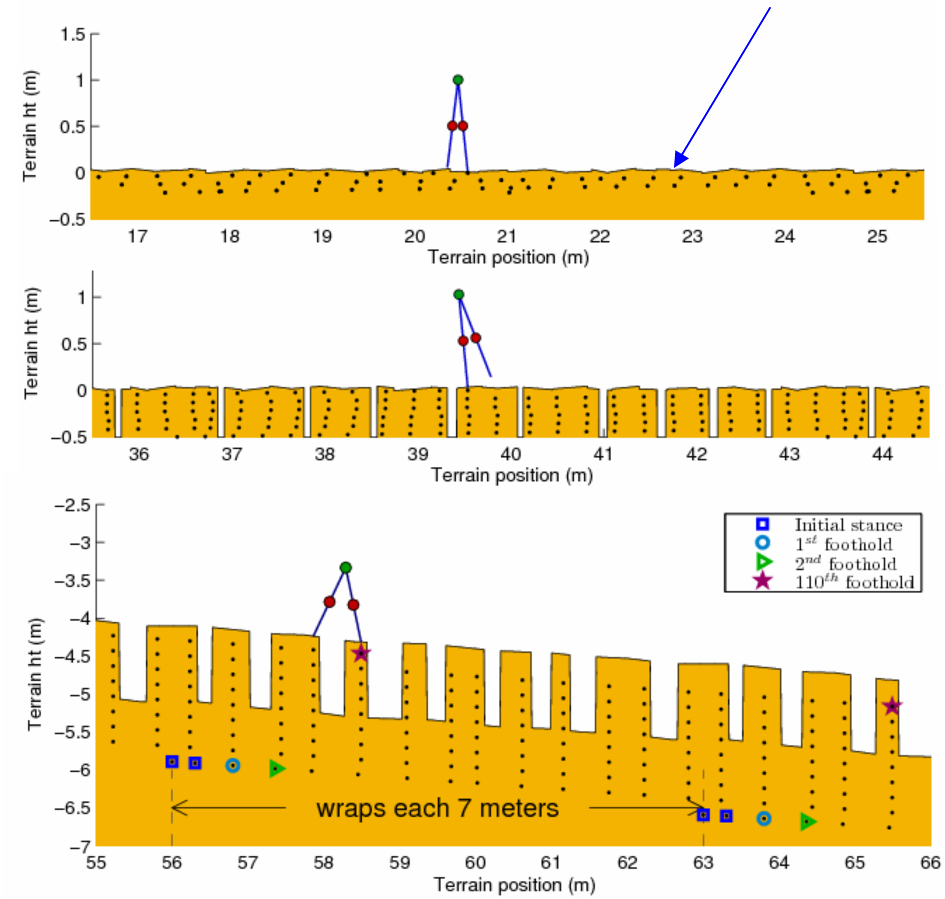
$$\left(\mathbf{I} - \mathbf{f}'\right)\mathbf{m} = \mathbf{1}$$

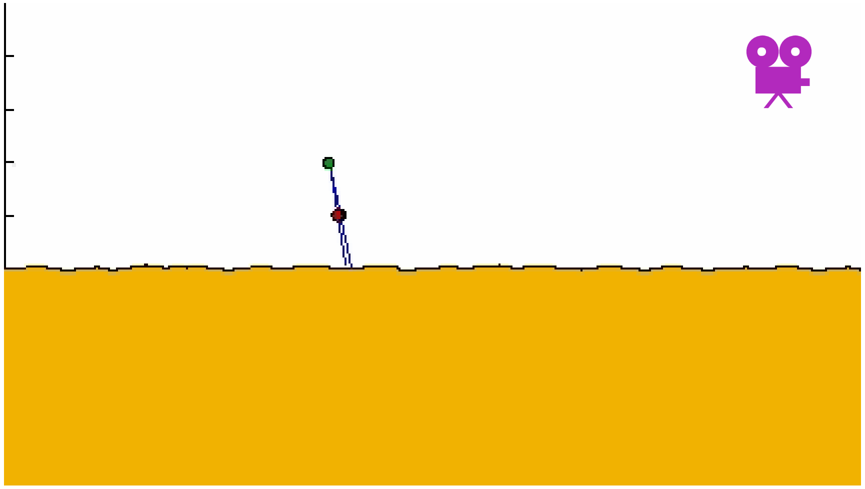
$$\mathbf{m} = \left(\mathbf{I} - \mathbf{f}'\right)^{-1}\mathbf{1}$$

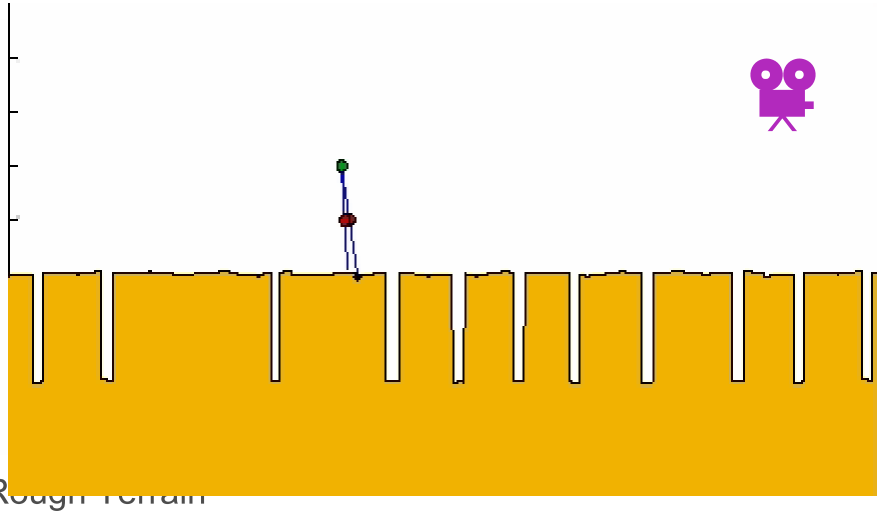
Results and Discussion

- Selecting only impulse magnitude (no PD) gives **fragile** results
- **PD-only** (used in examples below) works for **mild** or **downhill** terrain

Dots (wrapping) show previous footholds



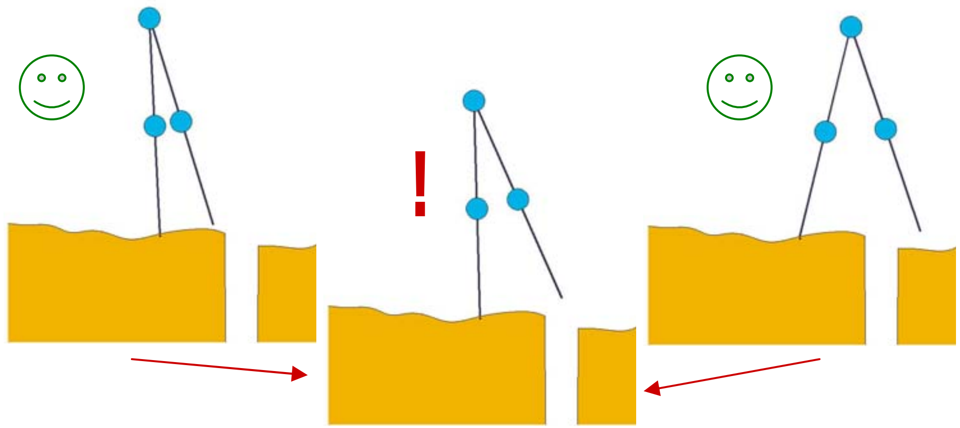




Discussion: Interpolation

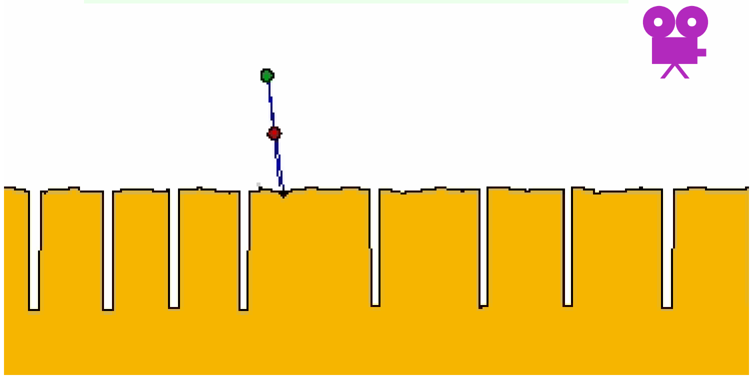
- **Method of interpolating optimal action is essential**
  - **Interpolating between actions oftens fails**
    - Small or large may be ok, while medium step fails:

Small step OK **Large step OK**



Interpolated step NOT OK

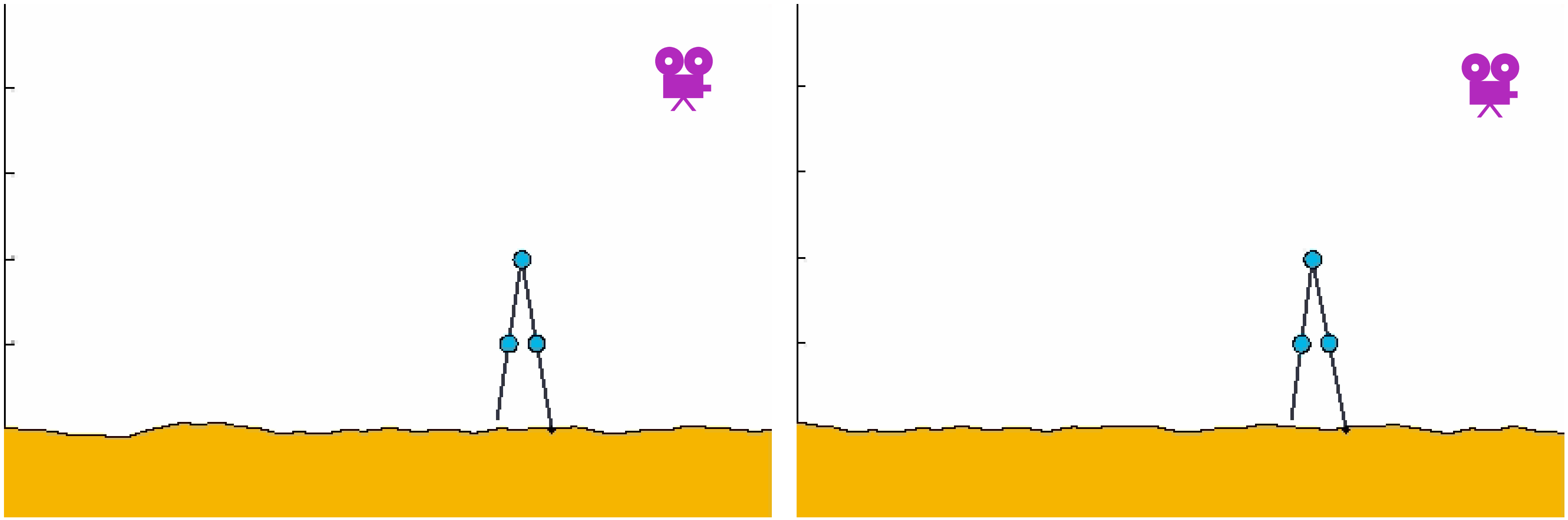
Watch for occasional steps into **no-go zones** in the animation below!



  - **Our solution:** simulate actual dynamics one step, then **select action resulting in new state with lowest cost**

Control on *Stochastic* Terrain

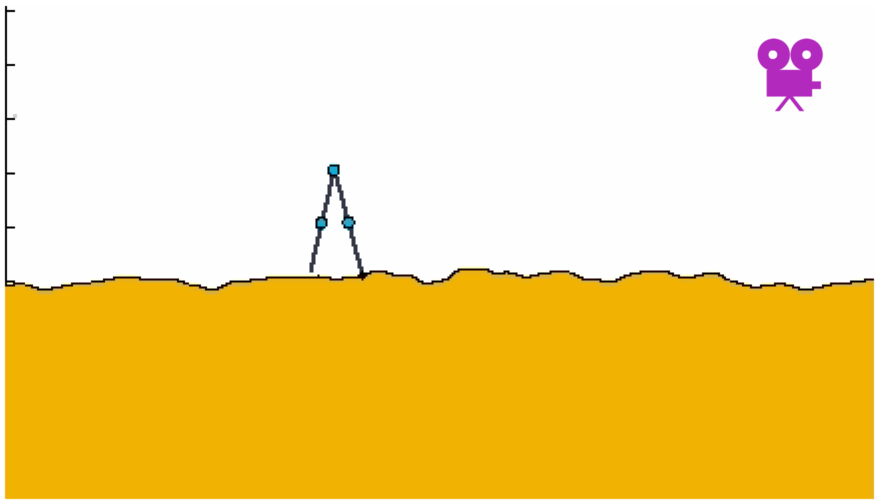
- ***One-step*** heuristic (below) on **random (no-wrap) terrain**



- Same ***optimization*** methodology can be applied using a **stochastic (e.g. Gaussian) description of terrain**

One-step on wrapping terrain

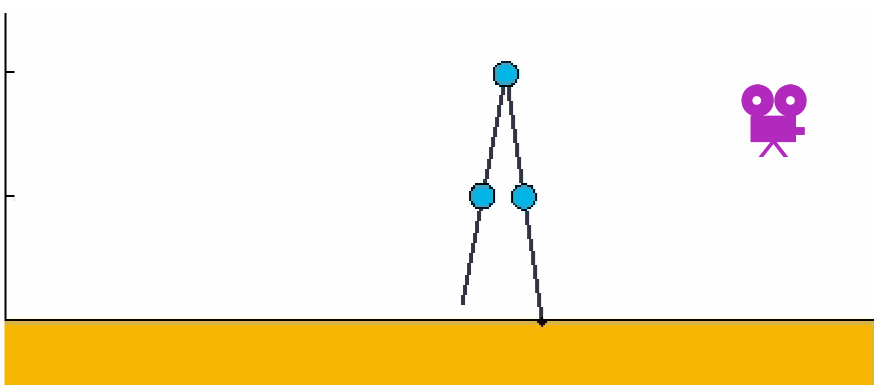
- **Results in continuous walking here**



Motivation

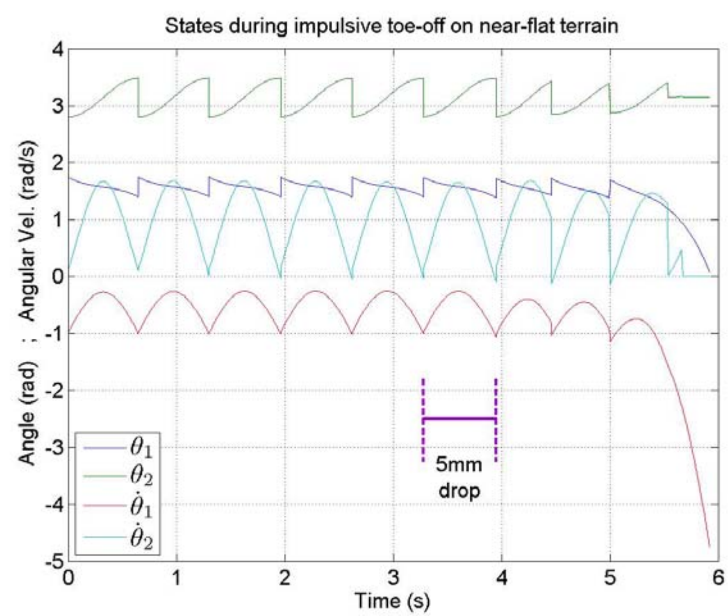
- **Passive-based walking is appealing for bipeds**
  - Captures fundamental, **pendular dynamics**
  - Seems likely to be **efficient**

- **Unfortunately, passive walkers are fragile!**
  - **Notoriously sensitive to initial conditions and perturbations**



Leg length = 1m

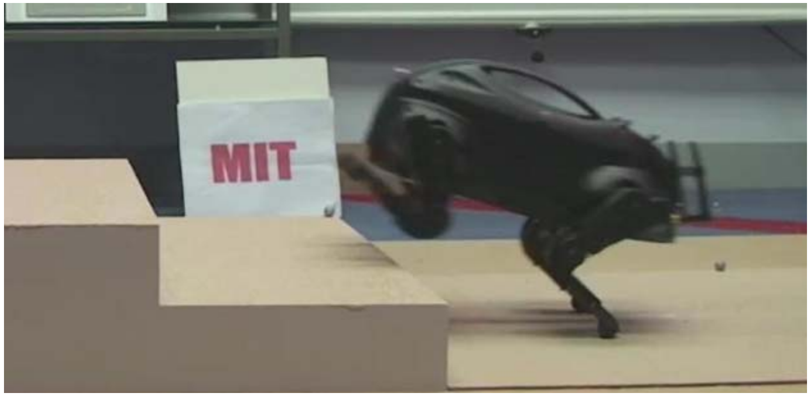
0.005m drop in .34m step, or about 1º



Underactuated stiff robots

- **Interested in applying same stochastic modeling to other, higher DOF robots**
  - **18 DOF (12 actuated, plus 6 DOF of body) LittleDog quadruped in dynamic, underactuated gaits and motions**
  - **Goal to learn policies which result in better stability**

Underactuated, double-support climbing motion →



- **See movies here:**

http://people.csail.mit.edu/katiebyl/ld/go_nogo_video/LittleDog_at_MIT_2008.mov

http://people.csail.mit.edu/katiebyl/ld/jersey_barrier/jersey_with_pacing.mov

people.csail.mit.edu/katiebyl/ld/newdog_terrainG/terrainG_newdog_withshove.mov