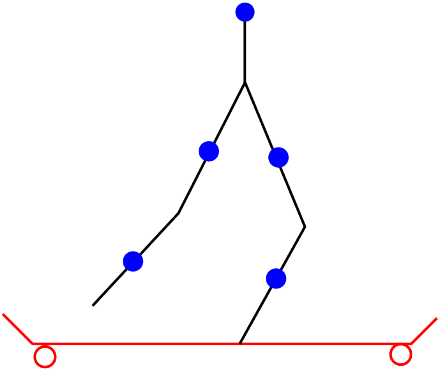
Modeling & Control of a Longboard-Riding Robot

Matt Keeter
mkeeter@mit.edu

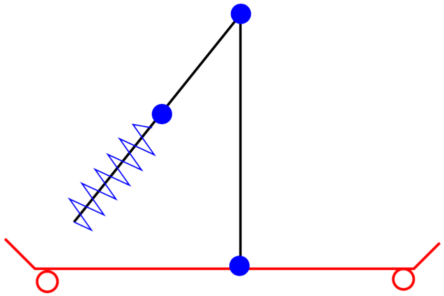
6.832 Final Project
Spring 2012

Inspiration

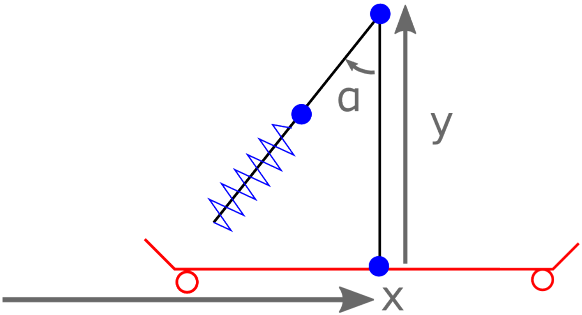
System Model



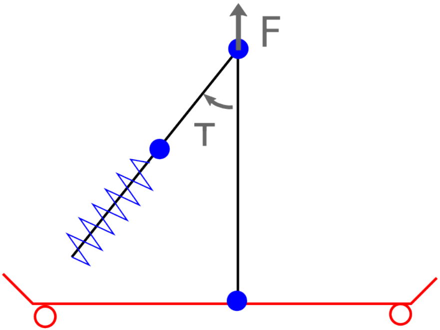
Simplified Model



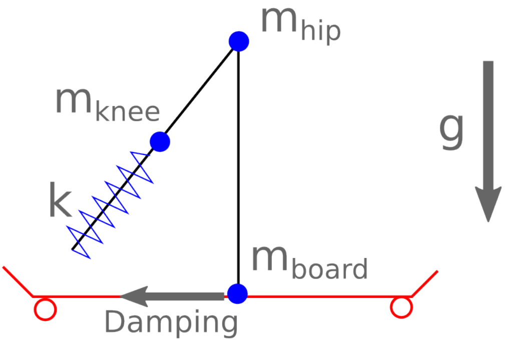
State Variables



Control Inputs



System Parameters



System Summary

- $\mathbf{q} = [x, y, \alpha]'$

System Summary

- $\mathbf{q} = [x, y, \alpha]'$
- $\mathbf{u} = [F, \tau]'$

System Summary

- $\mathbf{q} = [x, y, \alpha]'$
- $\mathbf{u} = [F, \tau]'$
- Gliding and pushing modes
  - $y_{\text{toe}} \leq 0 \rightarrow$ pushing
  - $y_{\text{toe}} > 0 \rightarrow$ gliding

- Wrote tools to automatically solve dynamics

Dynamics

- Wrote tools to automatically solve dynamics
- Written in Python using Sage

Dynamics

- Wrote tools to automatically solve dynamics
- Written in Python using Sage
- Solves for:
    - Second derivatives $\ddot{\mathbf{q}}$
    - Co-located PFL equations

Dynamics

- Wrote tools to automatically solve dynamics
- Written in Python using Sage
- Solves for:
  - Second derivatives $\ddot{\mathbf{q}}$
  - Co-located PFL equations
- Exports as MATLAB scripts

Feedback Linearization

- Solve dynamics equations for $\ddot{x}$, F, τ

Feedback Linearization

- Solve dynamics equations for $\ddot{x}$, F, τ
- Solutions are in terms of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{y}$, $\ddot{\alpha}$ (as well as constants)

Feedback Linearization

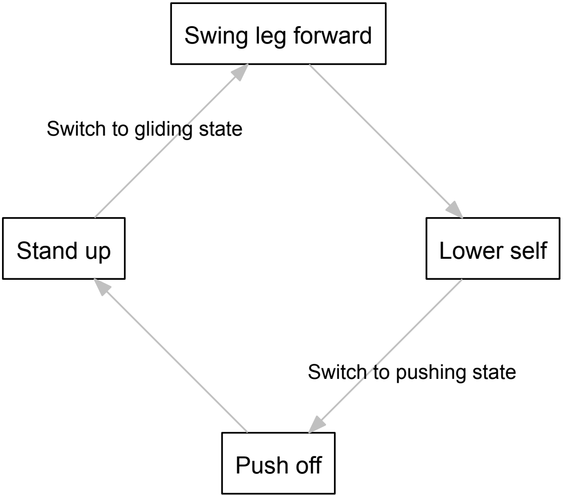
- Solve dynamics equations for $\ddot{x}$, F, τ
- Solutions are in terms of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{y}$, $\ddot{\alpha}$ (as well as constants)
- Plug in $\ddot{y}$, $\ddot{\alpha}$ for feedback linearization
  - $\ddot{y} = \ddot{y}_{\text{desired}}$
  - $\ddot{\alpha} = \ddot{\alpha}_{\text{desired}}$

Feedback Linearization

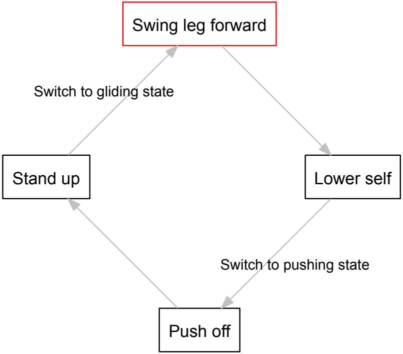
- Solve dynamics equations for $\ddot{x}$, F, τ
- Solutions are in terms of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{y}$, $\ddot{\alpha}$ (as well as constants)
- Plug in $\ddot{y}$, $\ddot{\alpha}$ for feedback linearization
  - $\ddot{y} = \ddot{y}_{\text{desired}}$
  - $\ddot{\alpha} = \ddot{\alpha}_{\text{desired}}$
- Double integrator control

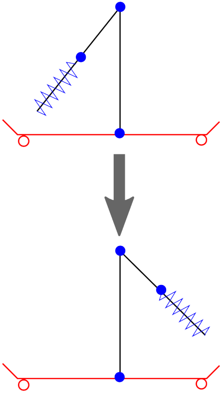
Controller Strategy

High-level strategy breaks motion into stages

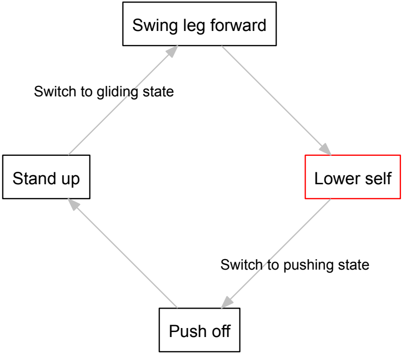


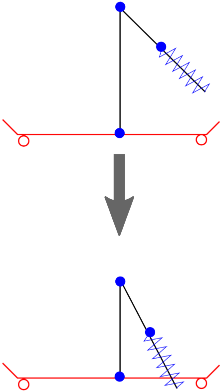
Controller Strategy



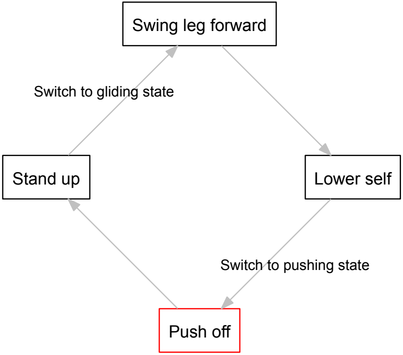


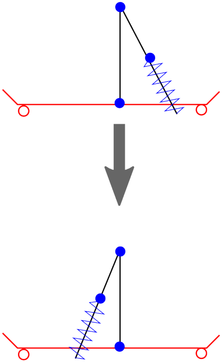
Controller Strategy



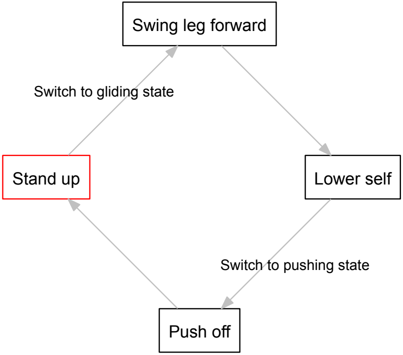


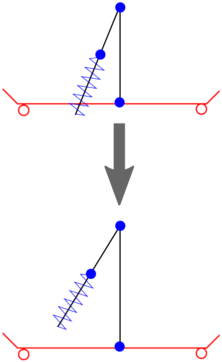
Controller Strategy



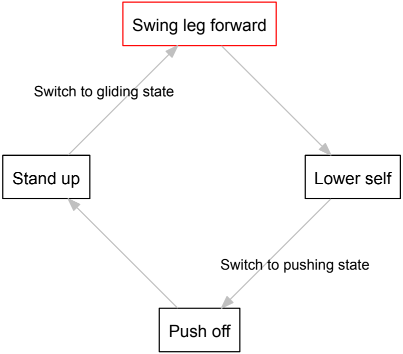


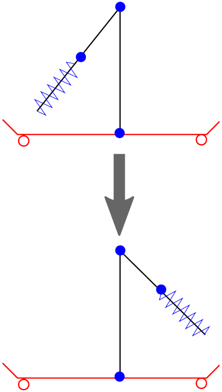
Controller Strategy





Controller Strategy





Demo

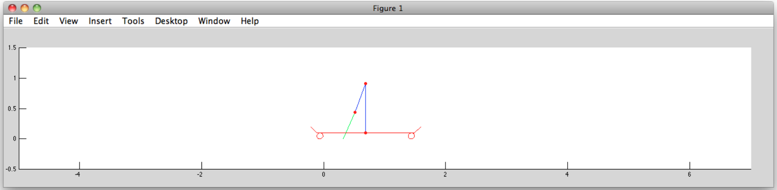


Figure 1
File Edit View Insert Tools Desktop Window Help
1.5
1
0.5
0
-0.5
-4
-2
0
2
4
6

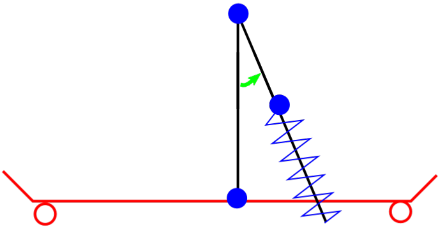
Controller Parameters

Controller is parameterized by three terms:

α_{hit}	Angle of collision
α_{stand}	Angle ending push
$\ddot{\alpha}_{\text{swing}}$	Swinging acceleration

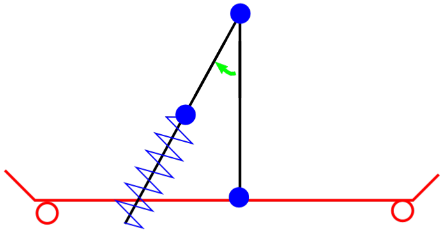
Controller Parameters

α_{hit}	Angle of collision
α_{stand}	Angle ending push
$\ddot{\alpha}_{\text{swing}}$	Swinging acceleration



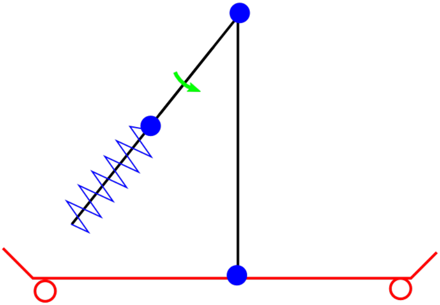
Controller Parameters

α_{hit}	Angle of collision
α_{stand}	Angle ending push
$\ddot{\alpha}_{\text{swing}}$	Swinging acceleration



Controller Parameters

α_{hit}	Angle of collision
α_{stand}	Angle ending push
$\ddot{\alpha}_{\text{swing}}$	Swinging acceleration

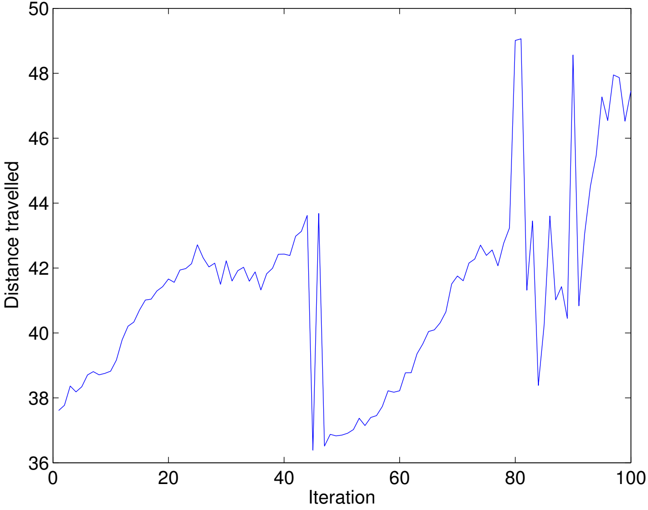


Stochastic Gradient Descent

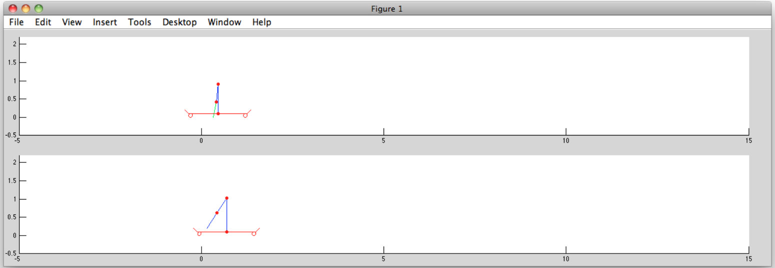
Optimized for distance travelled in fixed time

Stochastic Gradient Descent

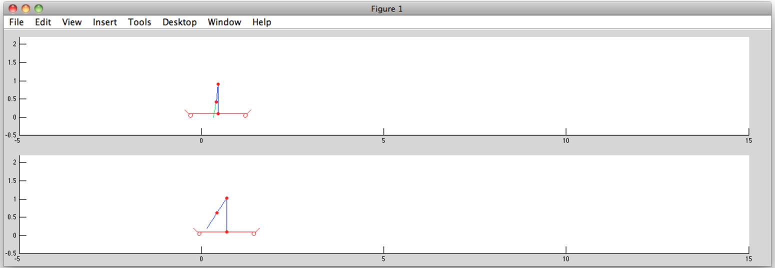
Optimized for distance travelled in fixed time



Optimized Demo



Optimized Demo



20% improvement!

Summary

- Developed simplified system model
- Wrote dynamics-solving tools
- Designed high-level controller behavior
- Used gradient descent to optimize parameters

Questions?