

Optimal Control using Iterative Dynamic Programming

Daniel M. Webb, W. Fred Ramirez advising

Department of Chemical and Biological Engineering
University of Colorado, Boulder

May 16, 2007

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation
- Real-time optimal control

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation
- Real-time optimal control

Some basic modeling steps:

- Formulate

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation
- Real-time optimal control

Some basic modeling steps:

- Formulate
- Train (identify model parameters)

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation
- Real-time optimal control

Some basic modeling steps:

- Formulate
- Train (identify model parameters)
- Offline optimization

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation
- Real-time optimal control

Some basic modeling steps:

- Formulate
- Train (identify model parameters)
- Offline optimization
- Online optimization (real-time optimal control)

Problem Introduction

ODE models:

- $\dot{\mathbf{X}} = f(\mathbf{X}, v(\mathbf{X}, \omega))$
- $X(t = 0) = X_0$

Model uses (especially batch systems):

- Optimize yield
- Minimize unwanted products
- Reduce run-to-run variation
- Real-time optimal control

Some basic modeling steps:

- Formulate
- Train (identify model parameters)
- **Offline optimization**
- Online optimization (real-time optimal control)

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

	Accumulation	Generation	Input	Dilution
$\dot{V} =$				
$\dot{X} =$				
$\dot{S} =$				
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$				
$\dot{S} =$				
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,

	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$				
$\dot{S} =$				
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

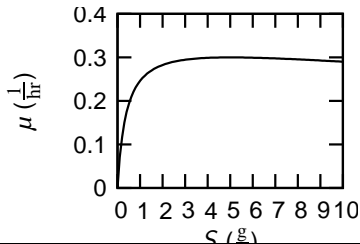
- cell growth,

	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$				
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,

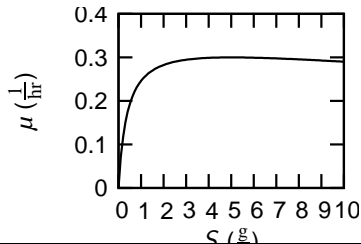


	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$				
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,

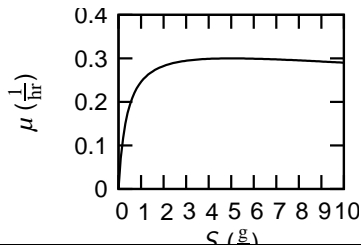


	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$				
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,



	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,
- foreign protein production,

	<i>Accumulation</i>	<i>Generation</i>	<i>Input</i>	<i>Dilution</i>
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$				
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

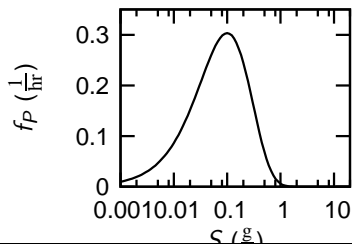
- cell growth,
- substrate consumption,
- foreign protein production,

	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,
- foreign protein production,



	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,
- foreign protein production,
- and foreign protein secretion.

	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$				

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

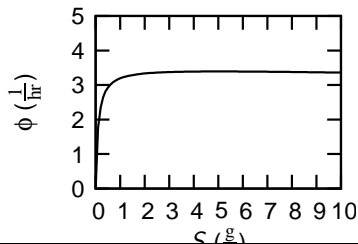
- cell growth,
- substrate consumption,
- foreign protein production,
- and foreign protein secretion.

	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$		$\phi(P_T - P_M)$		$-\frac{q}{V} P_M$

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,
- foreign protein production,
- and foreign protein secretion.



	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$		$\phi(P_T - P_M)$		$-\frac{q}{V} P_M$

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,
- foreign protein production,
- and foreign protein secretion.

The optimal control problem:

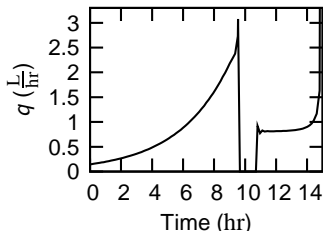
$$\begin{aligned} & \text{MAX}(\Phi) \\ \Phi &= P_M(t_f) \cdot V(t_f) \end{aligned}$$

	<i>Accumulation</i>	<i>Generation</i>	<i>Input</i>	<i>Dilution</i>
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$		$\phi(P_T - P_M)$		$-\frac{q}{V} P_M$

Park-Ramirez Bioreactor Model

For genetically modified yeast in a fed-batch reactor, predict:

- cell growth,
- substrate consumption,
- foreign protein production,
- and foreign protein secretion.



	Accumulation	Generation	Input	Dilution
$\dot{V} =$			q	
$\dot{X} =$		μX		$-\frac{q}{V} X$
$\dot{S} =$		$-Y\mu X$	$+\frac{q}{V} S_f$	$-\frac{q}{V} S$
$\dot{P}_T =$		$f_P X$		$-\frac{q}{V} P_T$
$\dot{P}_M =$		$\phi(P_T - P_M)$		$-\frac{q}{V} P_M$

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.
 - Seems to be much less common than control vector parametrization.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.
 - Seems to be much less common than control vector parametrization.
- 3 Necessary conditions and Pontryagin's Maximum Principle

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.
 - Seems to be much less common than control vector parametrization.
- 3 Necessary conditions and Pontryagin's Maximum Principle
 - Can provide much more insight into the problem solution.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.
 - Seems to be much less common than control vector parametrization.
- 3 Necessary conditions and Pontryagin's Maximum Principle
 - Can provide much more insight into the problem solution.
 - Requires more skill/education than parametrization methods.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.
 - Seems to be much less common than control vector parametrization.
- 3 Necessary conditions and Pontryagin's Maximum Principle
 - Can provide much more insight into the problem solution.
 - Requires more skill/education than parametrization methods.
 - Difficult to apply to singular control problems.

Optimal Control Solution Methods

The four primary ways currently used to solve optimal control problems:

- 1 Control vector parametrization (CVP)
 - Break the control $u(t)$ into piecewise vectors.
 - Each piecewise vector is a function approximation $v(\omega)$ (ie. constant, linear, polynomial).
 - Find the sensitivities $\frac{\partial \Phi}{\partial \omega}$.
 - Solve for ω using a NLP solver.
- 2 Collocation
 - Similar to control vector parametrization except discretizing states too.
 - Seems to be much less common than control vector parametrization.
- 3 Necessary conditions and Pontryagin's Maximum Principle
 - Can provide much more insight into the problem solution.
 - Requires more skill/education than parametrization methods.
 - Difficult to apply to singular control problems.
- 4 Iterative dynamic programming (IDP)

Dynamic Programming

What is dynamic programming?

A way to solve discrete multi-stage decision problem by:

Dynamic Programming

What is dynamic programming?

A way to solve discrete multi-stage decision problem by:

- Breaking the problem into smaller subproblems.

Dynamic Programming

What is dynamic programming?

A way to solve discrete multi-stage decision problem by:

- Breaking the problem into smaller subproblems.
- Remembering the best solutions to the subproblems.

Dynamic Programming

What is dynamic programming?

A way to solve discrete multi-stage decision problem by:

- Breaking the problem into smaller subproblems.
- Remembering the best solutions to the subproblems.
- Combining the solutions to the subproblems to get the overall solution.

Iterative Dynamic Programming (IDP)

What is iterative dynamic programming (IDP)?

A type of dynamic programming to solve the optimal control problem by:

Iterative Dynamic Programming (IDP)

What is iterative dynamic programming (IDP)?

A type of dynamic programming to solve the optimal control problem by:

- Breaking the control $u(t)$ into k stages.

Iterative Dynamic Programming (IDP)

What is iterative dynamic programming (IDP)?

A type of dynamic programming to solve the optimal control problem by:

- Breaking the control $u(t)$ into k stages.
- Guess several profiles for the stagewise constant controls u_k .

Iterative Dynamic Programming (IDP)

What is iterative dynamic programming (IDP)?

A type of dynamic programming to solve the optimal control problem by:

- Breaking the control $u(t)$ into k stages.
- Guess several profiles for the stagewise constant controls u_k .
- Use the guessed controls u_k to calculate a guessed continuous state profiles $X(t)$, $S(t)$, etc.

Iterative Dynamic Programming (IDP)

What is iterative dynamic programming (IDP)?

A type of dynamic programming to solve the optimal control problem by:

- Breaking the control $u(t)$ into k stages.
- Guess several profiles for the stagewise constant controls u_k .
- Use the guessed controls u_k to calculate a guessed continuous state profiles $X(t)$, $S(t)$, etc.
- Starting at the final time and working backward, find out which of the guessed controls was the best and remember it for the next iteration.

Iterative Dynamic Programming (IDP)

What is iterative dynamic programming (IDP)?

A type of dynamic programming to solve the optimal control problem by:

- Breaking the control $u(t)$ into k stages.
- Guess several profiles for the stagewise constant controls u_k .
- Use the guessed controls u_k to calculate a guessed continuous state profiles $X(t)$, $S(t)$, etc.
- Starting at the final time and working backward, find out which of the guessed controls was the best and remember it for the next iteration.

Animation 1: Basic IDP algorithm

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.
- IDP sometimes has noisy controls.

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.
- IDP sometimes has noisy controls.

What's good about IDP though?

- Stochastic method: if sufficient samples are taken, likely to find global optimum.

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.
- IDP sometimes has noisy controls.

What's good about IDP though?

- Stochastic method: if sufficient samples are taken, likely to find global optimum.
- No sensitivity derivatives needed.

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.
- IDP sometimes has noisy controls.

What's good about IDP though?

- Stochastic method: if sufficient samples are taken, likely to find global optimum.
- No sensitivity derivatives needed.
- Obtains an approximate solution very quickly.

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.
- IDP sometimes has noisy controls.

What's good about IDP though?

- Stochastic method: if sufficient samples are taken, likely to find global optimum.
- No sensitivity derivatives needed.
- Obtains an approximate solution very quickly.
- Very simple algorithm

CVP vs. IDP

CVP seems to be more popular than IDP. Why?

- Fast! At least 4x faster than IDP, often 20x faster.
- Easy (Matlab optimization toolbox, etc).
- Gradient method: fast in general, very fast near the solution.
- IDP sometimes has noisy controls.

What's good about IDP though?

- Stochastic method: if sufficient samples are taken, likely to find global optimum.
- No sensitivity derivatives needed.
- Obtains an approximate solution very quickly.
- Very simple algorithm (although maybe not after I'm done with it).

Speeding up IDP

IDP is slow, how can we speed it up?

Speeding up IDP

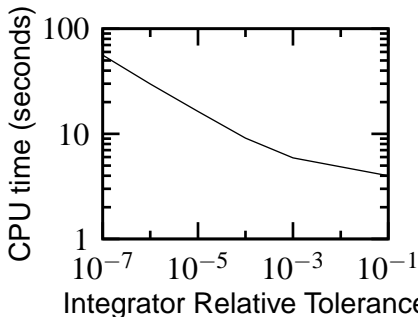
IDP is slow, how can we speed it up?

- Reduce integrator relative tolerance.

Speeding up IDP

IDP is slow, how can we speed it up?

- Reduce integrator relative tolerance.



Speeding up IDP

IDP is slow, how can we speed it up?

- Reduce integrator relative tolerance.
- Use my new adaptive region size update methods.

Speeding up IDP

IDP is slow, how can we speed it up?

- Reduce integrator relative tolerance.
- Use my new adaptive region size update methods.



First-order Control Filter

IDP often leads to noisy controls:

- Animation 3: Basic IDP with 50 stages

First-order Control Filter

IDP often leads to noisy controls:

- Animation 3: Basic IDP with 50 stages
- Try a first-order control filter after every iteration.

First-order Control Filter

IDP often leads to noisy controls:

- Animation 3: Basic IDP with 50 stages
- Try a first-order control filter after every iteration.
- Animation 4: First-order control filter

Simulated Annealing Control Filter

First-order control filter can over-filter.

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

1: **if** no test controls improve Φ **then**

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
- 2: **for** each test control **do**

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
- 2: **for** each test control **do**
- 3: **if** test control leads to a smoother control profile **then**

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
- 2: **for** each test control **do**
- 3: **if** test control leads to a smoother control profile **then**
- 4: $\gamma_s = \exp \left[\frac{-\Delta\Phi}{T \cdot M_\Phi} \right] - R(0, 1)$
- 5: **end if**

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
- 2: **for** each test control **do**
- 3: **if** test control leads to a smoother control profile **then**
- 4: $\gamma_s = \exp \left[\frac{-\Delta\Phi}{T \cdot M_\Phi} \right] - R(0, 1)$
- 5: **end if**
- 6: **end for**
- 7: Choose the test control with the largest γ_s
- 8: **end if**

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
 - 2: **for** each test control **do**
 - 3: **if** test control leads to a smoother control profile **then**
 - 4: $\gamma_s = \exp \left[\frac{-\Delta\Phi}{T \cdot M_\Phi} \right] - R(0, 1)$
 - 5: **end if**
 - 6: **end for**
 - 7: Choose the test control with the largest γ_s
 - 8: **end if**
- Temperature cools with control region size and number of iterations.

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
 - 2: **for** each test control **do**
 - 3: **if** test control leads to a smoother control profile **then**
 - 4: $\gamma_s = \exp \left[\frac{-\Delta\Phi}{T \cdot M_\Phi} \right] - R(0, 1)$
 - 5: **end if**
 - 6: **end for**
 - 7: Choose the test control with the largest γ_s
 - 8: **end if**
- Temperature cools with control region size and number of iterations.
 - Animation 5: Simulated annealing filter

Simulated Annealing Control Filter

First-order control filter can over-filter.

How about filtering more where Φ is hurt less?

- 1: **if** no test controls improve Φ **then**
- 2: **for** each test control **do**
- 3: **if** test control leads to a smoother control profile **then**
- 4: $\gamma_s = \exp \left[\frac{-\Delta\Phi}{T \cdot M_\Phi} \right] - R(0, 1)$
- 5: **end if**
- 6: **end for**
- 7: Choose the test control with the largest γ_s
- 8: **end if**

- Temperature cools with control region size and number of iterations.
- Animation 5: Simulated annealing filter
- Less likely to find local minima for this problem.

Control Damping

How about punishing control activity directly?

Control Damping

How about punishing control activity directly?

- $\Phi^* = \Phi - \Phi_d$

Control Damping

How about punishing control activity directly?

- $\Phi^* = \Phi - \Phi_d$
- $\Phi_d =$ discrete second derivative of controls.

Control Damping

How about punishing control activity directly?

- $\Phi^* = \Phi - \Phi_d$
- $\Phi_d =$ discrete second derivative of controls.
- Animation 6: Damping

Control Damping

How about punishing control activity directly?

- $\Phi^* = \Phi - \Phi_d$
- $\Phi_d =$ discrete second derivative of controls.
- Animation 6: Damping
- Changes the problem!

Control Damping

How about punishing control activity directly?

- $\Phi^* = \Phi - \Phi_d$
- $\Phi_d =$ discrete second derivative of controls.
- Animation 6: Damping
- Changes the problem!
- Was often harmful to solution if large enough to filter well.
- Good in small doses in combination with other methods.

Pivot Point Test Controls

Regular test controls work backwards one stage at a time.

Pivot Point Test Controls

Regular test controls work backwards one stage at a time.

Why not change two stage controls at a time?

Pivot Point Test Controls

Regular test controls work backwards one stage at a time.

Why not change two stage controls at a time?

- Animation 7: Pivot points (show every test control)

Pivot Point Test Controls

Regular test controls work backwards one stage at a time.

Why not change two stage controls at a time?

- Animation 7: Pivot points (show every test control)
- Animation 8: Pivot points (show every stage)

Pivot Point Test Controls

Regular test controls work backwards one stage at a time.

Why not change two stage controls at a time?

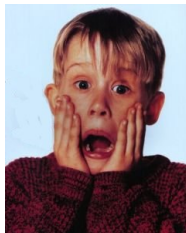
- Animation 7: Pivot points (show every test control)
- Animation 8: Pivot points (show every stage)
- Very fast convergence (to local minima).

Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.

Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.



Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.

Solve using two-step process:

Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.

Solve using two-step process:

- 1 Solve most of the way using basic IDP.

Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.

Solve using two-step process:

- 1 Solve most of the way using basic IDP.
- 2 Solve some more with smoothed IDP.

Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.

Solve using two-step process:

- 1 Solve most of the way using basic IDP.
- 2 Solve some more with smoothed IDP.

The two-step results?

Big Problem with Smoothing

All the smoothing techniques caused local minima to be found.

Solve using two-step process:

- 1 Solve most of the way using basic IDP.
- 2 Solve some more with smoothed IDP.

The two-step results?



Stagewise Linear Continuous Controls

Why not try a stagewise linear discretization of controls?

Stagewise Linear Continuous Controls

Why not try a stagewise linear discretization of controls?

- Animation 9: Continuous linear controls (show every stage)

Stagewise Linear Continuous Controls

Why not try a stagewise linear discretization of controls?

- Animation 9: Continuous linear controls (show every stage)
- Useful; seems to obtain equivalent Φ but smoother.

Conclusions

- Several smoothing methods were developed for IDP which greatly increase convergence speed. If used with the two-step procedure, you get the best of both worlds:

Conclusions

- Several smoothing methods were developed for IDP which greatly increase convergence speed. If used with the two-step procedure, you get the best of both worlds:
 - Resistance to local minima.

Conclusions

- Several smoothing methods were developed for IDP which greatly increase convergence speed. If used with the two-step procedure, you get the best of both worlds:
 - Resistance to local minima.
 - Fast convergence near the solution.

Acknowledgments

- Dr. W. Fred Ramirez

Acknowledgments

- Dr. W. Fred Ramirez
- National Science Foundation (funding)

Acknowledgments

- Dr. W. Fred Ramirez
- National Science Foundation (funding)
- The Free software community for the software tools I used.

Acknowledgments

- Dr. W. Fred Ramirez
- National Science Foundation (funding)
- The Free software community for the software tools I used.
- My wife Janet and daughter Ani for being patient with me.

Acknowledgments

- Dr. W. Fred Ramirez
- National Science Foundation (funding)
- The Free software community for the software tools I used.
- My wife Janet and daughter Ani for being patient with me.

