Optimization Software Survey

Sven Leyffer and Todd Munson
Mathematics and Computer Division
Argonne National Laboratory
{leyffer,tmunson}@mcs.anl.gov

Institute for Computational Economics
University of Chicago
July 29, 2008

Overview

1. Optimization Methods
   - Active-Set Methods: SQP/SLQP
   - Interior Point Methods
   - Global Convergence
2. Optimization Software
   - Available Solvers
   - Failures & Exception Handling
   - Local Solutions
3. Beyond Nonlinear Optimization
   - Optimization with Integer Variables
   - Global Optimization & Optimization Without Derivatives
   - Control and Optimization

Generic Nonlinear Optimization Problem

Nonlinear Programming (NLP) problem

$$\left\{\begin{array}{lll} \underset{x}{\text{minimize}} & f(x) & \text{objective} \\ \text{subject to} & c(x) = 0 & \text{constraints} \\ & x \geq 0 & \text{variables} \end{array}\right.$$

- $f : R^n \rightarrow R$, $c : R^n \rightarrow R^m$ smooth (typically $\mathcal{C}^2$)
- $x \in R^n$ finite dimensional (may be large)
- more general $l \leq c(x) \leq u$ possible

Solving Nonlinear Optimization Problems

$$(P) \quad \underset{x}{\text{minimize}} \; f(x) \quad \text{subject to } c(x) = 0, \quad x \geq 0$$

Main ingredients of iterative solution approaches:

1. Local Method: Given x_k (solution guess) find a step s.
   - Local problem should be easier to solve than (P).
   - Ensure fast (quadratic) local convergence.
   - Connection to global convergence ...
2. Forcing Strategy: Global convergence from remote starting points.
3. Forcing Mechanism: Truncate step s to force progress:
   - Trust-region to restrict s of local problem.
   - Back-tracking line-search along step s.

... look at each ingredient in turn.

Optimality Conditions for NLP

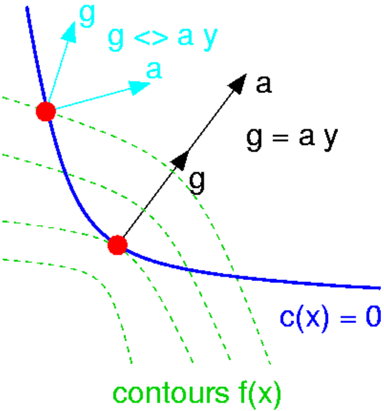
Constraint qualification (CQ)
Linearizations of $c(x) = 0$ characterize all feasible perturbations
$\Rightarrow$ rules out cusps etc.

x^* local minimizer & CQ holds $\Rightarrow \exists$ multipliers y^*, z^*:

$$\begin{aligned} \nabla f(x^*) - \nabla c(x^*)^T y^* - z^* &= 0 \\ c(x^*) &= 0 \\ X^* z^* &= 0 \\ x^* \geq 0,\ z^* &\geq 0 \end{aligned}$$

where $X^* = \text{diag}(x^*)$, thus $X^* z^* = 0 \Leftrightarrow x_i^* z_i^* = 0$
Lagrangian: $\mathcal{L}(x, y, z) := f(x) - y^T c(x) - z^T x$

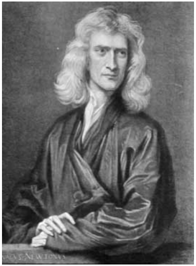
Optimality Conditions for NLP



Objective gradient is linear combination of constraint gradients

$$g(x) = A(x)y, \qquad \text{where } g(x) := \nabla f(x),\ A(x) := \nabla c(x)^T$$

Newton's Method for Nonlinear Equations



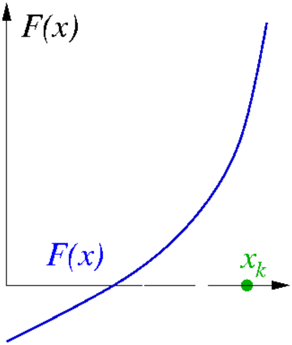
Solve $F(x) = 0$:
Get approx. x_{k+1} of solution of $F(x) = 0$
by solving linear model about x_k:

$$F(x_k) + \nabla F(x_k)^T (x - x_k) = 0$$

for $k = 0, 1, \ldots$

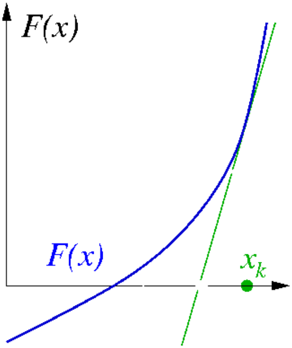
<u>Theorem</u>: If $F \in \mathcal{C}^2$, and $\nabla F(x^*)$ nonsingular,
then Newton converges quadratically near x^*.

Newton's Method for Nonlinear Equations



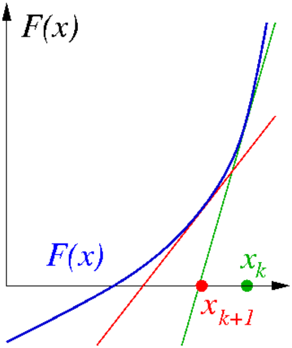
Next: two classes of methods based on Newton ...

Newton's Method for Nonlinear Equations



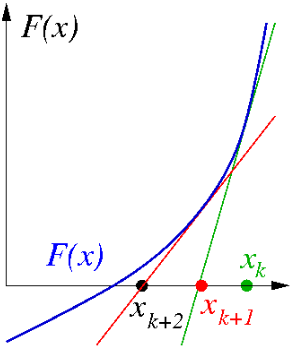
Next: two classes of methods based on Newton ...

Newton's Method for Nonlinear Equations



Next: two classes of methods based on Newton ...

Newton's Method for Nonlinear Equations



Next: two classes of methods based on Newton ...

Active-Set Methods

Sequential Quadratic Programming (SQP)

Consider equality constrained NLP

$$\underset{x}{\text{minimize}}\ f(x) \quad \text{subject to } c(x) = 0$$

Optimality conditions:

$$\begin{aligned} \nabla f(x) - \nabla c(x)^T y &= 0 \quad \text{and} \\ c(x) &= 0 \end{aligned}$$

... system of nonlinear equations: $F(w) = 0$ for $w = (x, y)$.

... solve using Newton's method

Sequential Quadratic Programming (SQP)

Nonlinear system of equations (KKT conditions)

$$\nabla f(x) - \nabla c(x)^T y = 0 \quad \text{and} \quad c(x) = 0$$

Apply Newton's method from $w_k = (x_k, y_k)$ ... $H_k = \nabla^2 \mathcal{L}(x_k, y_k)$

$$\begin{bmatrix} H_k & -A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} s_x \\ s_y \end{pmatrix} = - \begin{pmatrix} \nabla_x \mathcal{L}(x_k, y_k) \\ c_k \end{pmatrix}$$

... set $(x_{k+1}, y_{k+1}) = (x_k + s_x, y_k + s_y)$ ... $A^k = \nabla c(x_k)^T$
... solve for $y_{k+1} = y_k + s_y$ directly instead:

$$\begin{bmatrix} H_k & -A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} s \\ y_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

... set $(x_{k+1}, y_{k+1}) = (x_k + s, y_{k+1})$

Sequential Quadratic Programming (SQP)

Newton's Method for KKT conditions leads to:

$$\begin{bmatrix} H_k & -A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} s \\ y_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f_k \\ c_k \end{pmatrix}$$

... are optimality conditions of QP

$$\begin{cases} \underset{s}{\text{minimize}} & \nabla f_k^T s + \frac{1}{2} s^T H_k s \\ \text{subject to} & c_k + A_k^T s = 0 \end{cases}$$

... hence Sequential Quadratic Programming

Parenthesis: Saddle Point Problems

Given H symmetric $n \times n$, and A $m \times n$ matrices.

Let $K = \begin{bmatrix} H & -A \\ A^T & 0 \end{bmatrix}$

When is K nonsingular (i.e. invertible)?

Lemma If A has full rank, and if

$$Au = 0, u \neq 0 \Rightarrow u^T H u > 0$$

then K is nonsingular.

i.e. partial positive definiteness of H covers null-space of A

Sequential Quadratic Programming (SQP)

SQP for inequality constrained NLP:

$$\underset{x}{\text{minimize}} \; f(x) \quad \text{subject to } c(x) = 0 \quad \& \quad x \geq 0$$

REPEAT

1. Solve QP for (s, y_{k+1}, z_{k+1})

$$\begin{cases} \underset{s}{\text{minimize}} & \nabla f_k^T s + \frac{1}{2} s^T H_k s \\ \text{subject to} & c_k + A_k^T s = 0 \\ & x_k + s \geq 0 \end{cases}$$

2. Set $x_{k+1} = x_k + s$

... QP solve computationally expensive

Sequential Quadratic Programming

$$\text{NLP:} \quad \underset{x}{\text{minimize}} \;\; f(x) \qquad \text{subject to } c(x) = 0, \; x \geq 0$$

Sequential Quadratic Programming (SQP)

$$\begin{array}{ll} \underset{s}{\text{minimize}} & g_k^T s + \frac{1}{2} s^T W_k s \\ \text{subject to} & c_k + A_k^T s = 0 \\ & x_k + s \geq 0 \end{array}$$

where $g_k = \nabla f(x_k)$, $A_k = \nabla c(x_k)^T$, $W_k = \nabla^2 \mathcal{L}(x_k, y_k)$

set $x_{k+1} \leftarrow x_k + s$, update trust-region etc.

- unsuitable for large problems: QP pivoting $\Rightarrow$ basis factors

Sequential Linear Programming

NLP: $\underset{x}{\text{minimize}}\; f(x) \qquad \text{subject to } c(x) = 0,\; x \geq 0$

Sequential Linear Programming (SLP)

$$\begin{array}{ll} \underset{s}{\text{minimize}} & g_k^T s + \frac{1}{2} s^T W_k s \\ \text{subject to} & c_k + A_k^T s = 0 \\ & x_k + s \geq 0 \qquad \|s\|_\infty \leq \Delta_k \end{array}$$

where $g_k = \nabla f(x_k)$, $A_k = \nabla c(x_k)^T$, $W_k = \nabla^2 \mathcal{L}(x_k, y_k)$

set $x_{k+1} \leftarrow x_k + s$, update trust-region etc.

- unsuitable for large problems: QP pivoting $\Rightarrow$ basis factors
- solve LPs with million unknowns on PC
  trust-region $\|s\|_\infty \leq \Delta_k$ to avoid unbounded LP

Sequential Linear Programming

while (not optimal) **begin**

1. Compute displacement s_{LP} by solving LP subproblem

3. **if** step s acceptable **then**

   $x_{k+1} = x_k + s$ & increase TR $\Delta = 2 * \Delta$

   else $x_{k+1} = x_k$ & decrease TR $\Delta = \Delta/2$

end

- SLP ⇒ slow local convergence ... steepest descent

Sequential Linear Programming with EQP

while (not optimal) **begin**

1. Compute displacement s_{LP} by solving LP subproblem
2. Identify active constraints: $\mathcal{A} = \{i : c_i + a_i^T s_{LP} = 0\}$

$$(\text{EQP}) \begin{bmatrix} W & -A_{:,\mathcal{A}} \\ A_{:,\mathcal{A}}^T & \end{bmatrix} \begin{pmatrix} s \\ y_{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} -g \\ -c_{\mathcal{A}} \end{pmatrix}$$

   ... solve equality QP for step s

3. **if** step s acceptable **then**

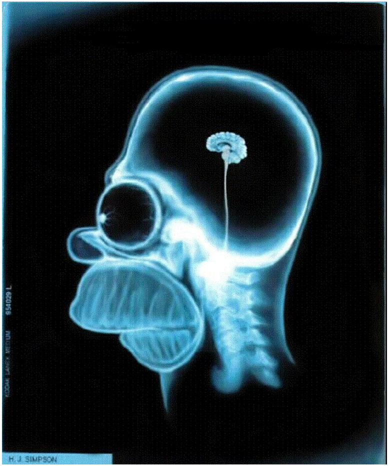
   $x_{k+1} = x_k + s$ & increase TR $\Delta = 2 * \Delta$

   else $x_{k+1} = x_k$ & decrease TR $\Delta = \Delta/2$

end

- SLP $\Rightarrow$ slow local convergence ... steepest descent
- EQP $\Rightarrow$ fast local convergence ... $\simeq$ Newton on $A_{:,\mathcal{A}}$
- use with `knitro_options = "algorithm=3";` ... or ASTROS

Modern Interior-Point Methods (IPM)



Modern Interior-Point Methods (IPM)

General NLP

$$\underset{x}{\text{minimize}} \; f(x) \quad \text{subject to } c(x) = 0 \quad \& \quad x \geq 0$$

Perturbed $\mu > 0$ optimality conditions ($x, z > 0$)

$$F_\mu(x, y, z) = \left\{ \begin{array}{r} \nabla f(x) - \nabla c(x)^T y - z \\ c(x) \\ Xz - \mu e \end{array} \right\} = 0$$

- Primal-dual formulation, where $X = \text{diag}(x)$
- Central path $\{x(\mu), y(\mu), z(\mu) \; : \; \mu > 0\}$
- Apply Newton's method for sequence $\mu \searrow 0$

Modern Interior-Point Methods (IPM)

Newton's method applied to primal-dual system ...

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k & -A_k & -I \\ A_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = -F_\mu(x_k, y_k, z_k)$$

where $A_k = \nabla c(x_k)^T$, X_k diagonal matrix of x_k.

Polynomial run-time guarantee for convex problems

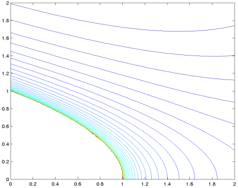
Classical Interior-Point Methods (IPM)

$$\underset{x}{\text{minimize}} \ f(x) \quad \text{subject to } c(x) = 0 \quad \& \quad x \geq 0$$

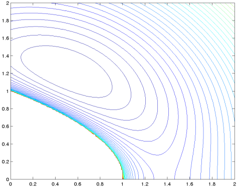
Related to classical barrier methods [Fiacco & McCormick]

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) - \mu \sum \log(x_i) \\ \text{subject to} & c(x) = 0 \end{cases}$$

$\mu = 10$



$\mu = 1$



$$\text{minimize } x_1^2 + x_2^2 - \mu \log\left(x_1 + x_2^2 - 1\right)$$

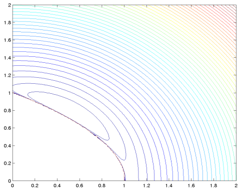
Classical Interior-Point Methods (IPM)

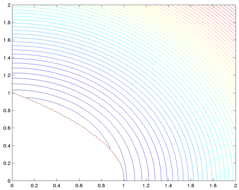
$$\underset{x}{\text{minimize}}\ f(x) \quad \text{subject to } c(x) = 0 \quad \& \quad x \geq 0$$

Related to classical barrier methods [Fiacco & McCormick]

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) - \mu \sum \log(x_i) \\ \text{subject to} & c(x) = 0 \end{cases}$$

$\mu = 0.1$ $\mu = 0.001$





$$\text{minimize}\, x_1^2 + x_2^2 - \mu \log\left(x_1 + x_2^2 - 1\right)$$

Classical Interior-Point Methods (IPM)

$$\underset{x}{\text{minimize}} \; f(x) \quad \text{subject to } c(x) = 0 \quad \& \quad x \geq 0$$

Relationship to barrier methods

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) - \mu \sum \log(x_i) \\ \text{subject to} & c(x) = 0 \end{cases}$$

First order conditions

$$\begin{aligned} \nabla f(x) - \mu X^{-1} e - A(x) y &= 0 \\ c(x) &= 0 \end{aligned}$$

... apply Newton's method ...

Classical Interior-Point Methods (IPM)

Newton's method for barrier problem from x_k ...

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k + \mu X_k^{-2} & -A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \ldots$$

Introduce $Z(x_k) := \mu X_k^{-1}$ ... or ... $Z(x_k) X_k = \mu e$

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k + Z(x_k) X_k^{-1} & -A_k \\ A_k & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \ldots$$

... compare to primal-dual system ...

Classical Interior-Point Methods (IPM)

Recall: Newton's method applied to primal-dual system ...

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k & -A_k & -I \\ A_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = -F_\mu(x_k, y_k, z_k)$$

Eliminate $\Delta z = -X^{-1} Z \Delta x - Ze - \mu X^{-1} e$

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k + Z_k X_k^{-1} & -A_k \\ A_k & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \ldots$$

Interior-Point Methods (IPM)

Primal-dual system ...

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k + Z_k X_k^{-1} & -A_k \\ A_k & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \ldots$$

... compare to barrier system ...

$$\begin{bmatrix} \nabla^2 \mathcal{L}_k + Z(x_k) X_k^{-1} & -A_k \\ A_k & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \ldots$$

- Z_k is free, not $Z(x_k) = \mu X_k^{-1}$ (primal multiplier)
- avoid difficulties with barrier ill-conditioning

Solving Nonlinear Optimization Problems

$$(P) \quad \underset{x}{\text{minimize}} \ f(x) \quad \text{subject to } c(x) \geq 0$$

Main ingredients of iterative solution approaches:

1. Local Method: Given x_k (solution guess) find a step s.
   - Sequential Quadratic Programming (SQP)
   - Sequential Linear/Quadratic Programming (SLQP)
   - Interior-Point Methods
2. Forcing Strategy: Global convergence from remote starting points.
3. Forcing Mechanism: Truncate step s to force progress:
   - Trust-region to restrict s of local problem ... used in this talk.
   - Back-tracking line-search along step s.

Enforcing Convergence

When's a New Point Better?

Easy for unconstrained minimize $f(x)$ (quadratic model $q_k(s)$):

$$x_{k+1} = x_k + s \text{ better, iff } f(x_{k+1}) \le f(x_k) - 10^{-4} q_k(s)$$

... actual reduction matches portion of reduction predicted by model.

Unclear for constrained problem: $c(x) = 0$

- step s can reduce both $f(x)$ and $\|c(x)\|$ GOOD
- step s increases $f(x)$ and decreases $\|c(x)\|$???
- step s decreases $f(x)$ and increases $\|c(x)\|$???
- step s can increase both $f(x)$ and $\|c(x)\|$ BAD

Penalty Functions (i)

Augmented Lagrangian Methods

$$\underset{x}{\text{minimize}} \; L(x, y_k, \rho_k) \; = \; f(x) \; - \; y_k^T c(x) + \frac{1}{2}\rho_k \|c(x)\|^2$$

As $y_k \to y_*$:
- $x_k \to x_*$ for $\rho_k > \bar{\rho}$
- No ill-conditioning, improves convergence rate

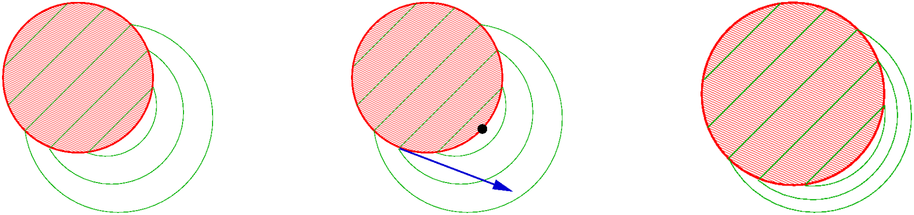
- update ρ_k based on reduction in $\|c(x)\|^2$
- approx. minimize $L(x, y_k, \rho_k)$
- first-order multiplier update: $y_{k+1} = y_k - \rho_k c(x_k)$
  $\Rightarrow$ dual iteration

Penalty Functions (ii)

Exact Penalty Function: minimize$_x$ $\Phi(x, \pi) = f(x) + \pi \|c(x)\|$

- combine constraints and objective
- equivalence of optimality $\Rightarrow$ exact for $\pi > \|y^*\|_D$

  ... now apply unconstrained techniques
- Φ nonsmooth, but equivalent to smooth problem (exercise)



Filter Methods for NLP

Penalty function can be inefficient

- Penalty parameter not known a priori
- Large penalty parameter $\Rightarrow$ slow convergence

Two competing aims in optimization:

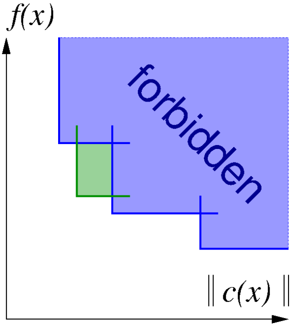
1. Minimize $f(x)$
2. Minimize $h(x) := \|c(x)\|$ ... more important

$\Rightarrow$ concept from multi-objective optimization:
(h_{k+1}, f_{k+1}) dominates (h_l, f_l) iff $h_{k+1} \le h_l$ & $f_{k+1} \le f_l$

Filter Methods for NLP

Filter $\mathcal{F}$: list of non-dominated pairs (h_l, f_l)

- new x_{k+1} acceptable to filter $\mathcal{F}$, iff
  1. $h_{k+1} \le h_l \; \forall l \in \mathcal{F}$, or
  2. $f_{k+1} \le f_l \; \forall l \in \mathcal{F}$

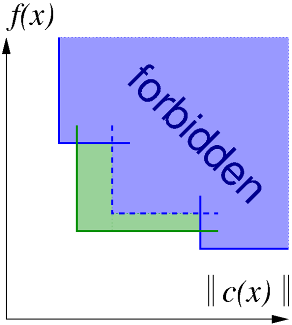


$\Rightarrow$ often accept new x_{k+1}, even if penalty function increases

Filter Methods for NLP

Filter $\mathcal{F}$: list of non-dominated pairs (h_l, f_l)

- new x_{k+1} acceptable to filter $\mathcal{F}$, iff
  1. $h_{k+1} \leq h_l \; \forall l \in \mathcal{F}$, or
  2. $f_{k+1} \leq f_l \; \forall l \in \mathcal{F}$
- remove redundant entries

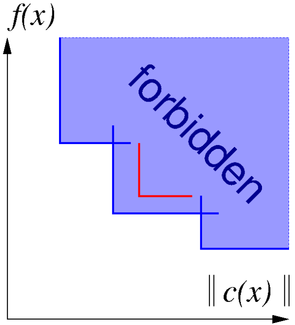


$\Rightarrow$ often accept new x_{k+1}, even if penalty function increases

Filter Methods for NLP

Filter $\mathcal{F}$: list of non-dominated pairs (h_l, f_l)

- new x_{k+1} acceptable to filter $\mathcal{F}$, iff
  1. $h_{k+1} \leq h_l \; \forall l \in \mathcal{F}$, or
  2. $f_{k+1} \leq f_l \; \forall l \in \mathcal{F}$
- remove redundant entries
- reject new x_{k+1},
  if $h_{k+1} > h_l$ & $f_{k+1} > f_l$
  ... reduce trust region radius $\Delta = \Delta/2$



$\Rightarrow$ often accept new x_{k+1}, even if penalty function increases

Solving Nonlinear Optimization Problems

$$(P) \quad \underset{x}{\text{minimize}} \; f(x) \quad \text{subject to } c(x) \geq 0$$

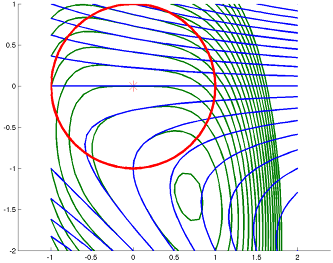
Main ingredients of iterative solution approaches:

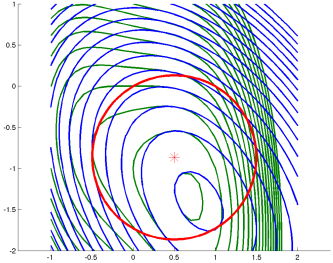
1. Local Method: Given x_k (solution guess) find a step s.
   - Sequential Quadratic Programming (SQP)
   - Sequential Linear/Quadratic Programming (SLQP)
   - Interior-Point Methods
2. Forcing Strategy: Augmented Lagrangian, penalty, filter.
3. Forcing Mechanism: Truncate step s to force progress:
   - Trust-region to restrict s of local problem ... used in this talk.
   - Back-tracking line-search along step s.

Trust-Region Methods

Globalize SQP/IPM using trust region, $\Delta^k > 0$:
Consider unconstrained $f(x)$ minimization by trust-region

$$\underset{s}{\text{minimize}} \; q_k(s) := f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T H(x_k) s \text{ subject to } \|s\| \le \Delta^k$$





Trust-Region Framework for Nonlinear Optimization

$$\underset{x}{\text{minimize}} \;\; f(x) \quad \text{subject to } c(x) = 0, \quad x \geq 0$$

E.g. SQP: given x_0 starting point, set $k = 0$

repeat

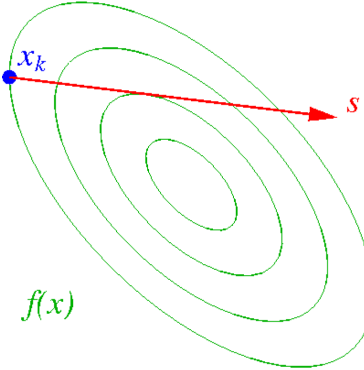
1. solve trust-region problem around x_k for step s:

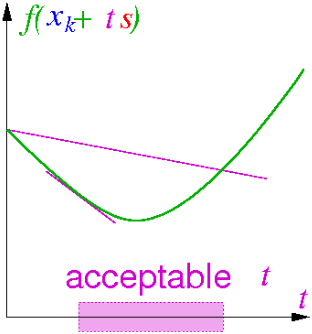
$$\min_s q_k(s) \text{ s.t. } c_k + A_k^T s = 0, \; x_k + s \geq 0, \; \|s\| \leq \Delta^k$$

2. **if** $x_k + s$ improves on x_k **then**
   accept step: $x_{k+1} = x_k + s$
   else reject step: $x_{k+1} = x_k$
3. $k = k + 1$ & house-keeping

until convergence

Line-Search Methods





Line-Search Methods

SQP/IPM compute s descend direction or penalty function: $s^T \nabla \Phi < 0$

Backtracking-Armijo line search

Given $\alpha^0 = 1$, $\beta = 0.1$, set $l = 0$

REPEAT

1. $\alpha^{l+1} = \alpha^l / 2$ & evaluate $\Phi(x + \alpha^{l+1} s)$
2. $l = l + 1$

UNTIL $\Phi(x + \alpha^l s) \leq f(x) + \alpha^l \beta s^T \nabla \Phi$

Converges to stationary point, or unbounded, or zero descend

Overview

Sequential Quadratic Programming

- ASTROS Active-Set Trust-Region Optimization Solvers
- filterSQP
    - trust-region SQP; robust QP solver
    - filter to promote global convergence
- SNOPT
    - line-search SQP; null-space CG option
    - ℓ_1 exact penalty function
- SLIQUE (part of KNITRO)
    - SLP-EQP ("SQP" for larger problems)
    - trust-region with ℓ_1 penalty
    - use with knitro_options = "algorithm=3";

Other Methods: CONOPT generalized reduced gradient method

Interior Point Methods

- IPOPT (free: part of COIN-OR)
  - line-search filter algorithm
  - 2nd order convergence analysis for filter
- KNITRO
  - trust-region Newton to solve barrier problem
  - ℓ_1 penalty barrier function
  - Newton system: direct solves or null-space CG
- LOQO
  - line-search method
  - Cholesky factorization; no convergence analysis

Other solvers: MOSEK (unsuitable or nonconvex problem)

Augmented Lagrangian Methods

- LANCELOT
  - minimize augmented Lagrangian subject to bounds
  - trust-region to force convergence
  - iterative (CG) solves
- MINOS
  - minimize augmented Lagrangian subject to linear constraints
  - line-search; recent convergence analysis
  - direct factorization of linear constraints
- PENNON
  - suitable for semi-definite optimization
  - alternative penalty terms

COIN-OR

http://www.coin-or.org

- COmputational INfrastructure for Operations Research
- A library of (interoperable) software tools for optimization
- A development platform for open source projects in the OR community
- Possibly Relevant Modules:
    - OSI: Open Solver Interface
    - CGL: Cut Generation Library
    - CLP: Coin Linear Programming Toolkit
    - CBC: Coin Branch and Cut
    - IPOPT: Interior Point OPTimizer for NLP
    - NLPAPI: NonLinear Programming API

Other: `SOPLEX` ... (MI)LP solver almost as good as CPLEX

Active-Set vs. Interior-Point

Active-Set usually more robust (identify degeneracy)

- LP/QP solve become bottleneck for large problems
  combinatorial pivoting & dense linear algebra
- robust LP/QP find linearly independent set of constraints
  $\Rightarrow$ ensures LICQ for subset of constraints
- good warm-start properties ... solving related problems

Interior-Point often faster (in terms of CPU time)

- solve single linear system per iteration
  $\Rightarrow$ much faster than LP/QP solve
- poor warm-start properties ... initial point $x, z > \mu$
- carry all constraints around at all times
  $\Rightarrow$ affected by degeneracy ... cond(KKT) $= \mathcal{O}(\mu^{-1})$

... but there are practical differences too, see `hs044.mod`

Automatic Differentiation

How do I get the derivatives $\nabla c(x)$, $\nabla^2 c(x)$ etc?

- hand-coded derivatives are error prone
- finite differences $\frac{\partial c_i(x)}{\partial x_j} \simeq \frac{c_i(x+\delta e_j)-c_i(x)}{\delta}$ can be dangerous
  where $e_j = (0,\ldots,0,1,0,\ldots,0)$ is j^{th} unit vector

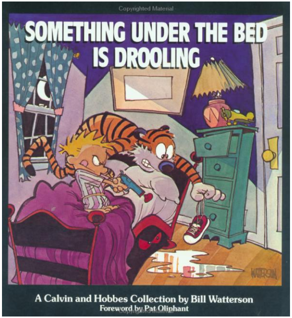
Automatic Differentiation

- chain rule techniques to differentiate program
- recursive application $\Rightarrow$ "exact" derivatives
- suitable for huge problems, see `www.autodiff.org`

... already done for you in AMPL/GAMS etc.

Something Under the Bed is Drooling

1. exception handling
   - floating point (IEEE) exceptions
   - unbounded problems
2. local solutions
   - (locally) inconsistent problems
   - suboptimal solutions



... identify problem & suggest remedies

Floating Point (IEEE) Exceptions

Bad example: minimize barrier function, `barrier.mod`

```
param mu default 1;
var x{1..2} >= -10, <= 10;
var s;
minimize barrier: x[1]^2 + x[2]^2 - mu*log(s);
subject to
   cons: s = x[1] + x[2]^2 - 1;
```

... results in error message like

`Cannot evaluate objective at start`

... change initialization of `s`:

`var s := 1;` ... difficult, if IEEE during solve ...

Unbounded Objective

Penalized MPEC (wait till tomorrow) $\pi = 1$:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & x_1^2 + x_2^2 - 4x_1x_2 \quad + \pi x_1 x_2 \\ \text{subject to} & x_1, x_2 \geq 0 \end{array}$$

... unbounded below for all $\pi < 2$

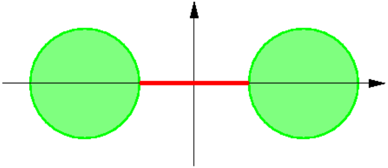
```
param pi >= 0, default 1;   # ... penalty parameter
var x{1..2} >= 0;
minimize MPECpen: x[1]^2 + x[2]^2 - 4*x[1]*x[2] + pi*x[1]*x[2];
```

... what happens to `L1penalty.mod`?

Locally Inconsistent Problems

NLP may have no feasible point

```
var x{1..2} >= -1;
minimize objf: -1000*x[2];
subject to
   con1: (x[1]+2)^2 + x[2]^2 <= 1;
   con2: (x[1]-2)^2 + x[2]^2 <= 1;
```



feasible set: intersection of circles

Locally Inconsistent Problems

LOQO

```
     |         Primal             |           Dual
Iter |  Obj Value     Infeas      |   Obj Value     Infeas
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  1   -1.000000e+03   4.2e+00    -6.000000e+00   1.0e-00
[...]
500    2.312535e-04   7.9e-01     1.715213e+12   1.5e-01
LOQO 6.06: iteration limit
```

... fails to converge ... not useful for user

dual unbounded $\rightarrow \infty \Rightarrow$ primal infeasible

Locally Inconsistent Problems

FILTER

```
iter  |    rho    |   ||d||     |    f / hJ  |   ||c||/hJt
------+----------+------------+------------+------------
   0:0  10.0000    0.00000     -1000.0000    16.000000
   1:1  10.0000    2.00000     -1000.0000    8.0000000
[...]
   8:2  2.00000   0.320001E-02  7.9999693   0.10240052E-04
   9:2  2.00000   0.512000E-05  8.0000000   0.26214586E-10
filterSQP: Nonlinear constraints locally infeasible
```

... fast convergence to minimum infeasibility

... identify "blocking" constraints ... modify model/data

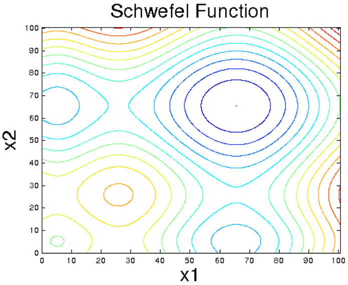
Locally Inconsistent Problems

Remedies for locally infeasible problems:

1. check your model: print constraints & residuals, e.g.
   ```
   solve;
   display _conname, _con.lb, _con.body, _con.ub;
   display _varname, _var.lb, _var, _var.ub;
   ```
   ... look at violated and active constraints
2. try different nonlinear solvers (easy with AMPL)
3. build-up model from few constraints at a time
4. try different starting points ... global optimization

Suboptimal Solution & Multi-start

Problems can have many local minimizers




```
param pi := 3.1416;
param n integer, >= 0, default 2;
set N := 1..n;
var x{N} >= 0, <= 32*pi, := 1;
minimize objf:
- sum{i in N} x[i]*sin(sqrt(x[i]));
```

default start point converges to local minimizer

Suboptimal Solution & Multi-start

```
param nD := 5;       # discretization
set    D := 1..nD;
param  hD := 32*pi/(nD-1);
param optval{D,D};
model schwefel.mod;  # load model

for {i in D}{
   let x[1] := (i-1)*hD;
   for {j in D}{
      let x[2] := (j-1)*hD;
      solve;
      let optval[i,j] := objf;
   }; # end for
}; # end for
```

Suboptimal Solution & Multi-start

```
display optval;
optval [*,*]
:     1         2          3          4          5    :=
1     0         24.003    -36.29     -50.927     56.909
2    24.003    -7.8906    -67.580    -67.580    -67.580
3   -36.29     -67.5803   -127.27    -127.27    -127.27
4   -50.927    -67.5803   -127.27    -127.27    -127.27
5    56.909    -67.5803   -127.27    -127.27    -127.27
;
```

... there exist better multi-start procedures

Overview

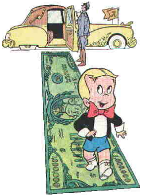
Optimization with Integer Variables

Mixed-Integer Nonlinear Program (MINLP)

- modeling discrete choices $\Rightarrow 0 - 1$ variables
- modeling integer decisions $\Rightarrow$ integer variables
  e.g. number of different stocks in portfolio (8-10)
  not number of beers sold at Goose Island (millions)

MINLP solvers:

- branch (separate $z_i = 0$ and $z_i = 1$) and cut
- solve millions of NLP relaxations: `MINLPBB`, `SBB`
- outer approximation: iterate MILP and NLP solvers
  `BONMIN (COIN-OR)` & FilMINT on NEOS

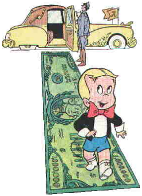


Portfolio Management

- N: Universe of asset to purchase
- x_i: Amount of asset i to hold
- B: Budget

$$\text{minimize } \; u(x) \quad \text{subject to } \sum_{i \in N} x_i = B, \quad x \geq 0$$

Portfolio Management

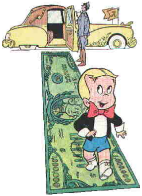


- N: Universe of asset to purchase
- x_i: Amount of asset i to hold
- B: Budget

$$\text{minimize } u(x) \quad \text{subject to } \sum_{i \in N} x_i = B, \quad x \geq 0$$

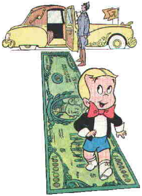
- Markowitz: $u(x) \stackrel{\text{def}}{=} -\alpha^T x + \lambda x^T Q x$
  - α: maximize expected returns
  - Q: variance-covariance matrix of expected returns
  - λ: minimize risk; aversion parameter

More Realistic Models



- $b \in \mathbb{R}^{|N|}$ of "benchmark" holdings
- Benchmark Tracking: $u(x) \stackrel{\text{def}}{=} (x-b)^T Q(x-b)$
  - Constraint on $\mathbb{E}$[Return]: $\alpha^T x \geq r$

More Realistic Models



- $b \in \mathbb{R}^{|N|}$ of "benchmark" holdings
- Benchmark Tracking: $u(x) \stackrel{\text{def}}{=} (x-b)^T Q(x-b)$
  - Constraint on $\mathbb{E}$[Return]: $\alpha^T x \geq r$
- Limit Names: $|i \in N \ : \ x_i > 0| \leq K$
  - Use binary indicator variables to model the implication $x_i > 0 \Rightarrow y_i = 1$
  - Implication modeled with variable upper bounds:

$$x_i \leq B y_i \qquad \forall i \in N$$

  - $\sum_{i \in N} y_i \leq K$

Global Optimization

I need to find the GLOBAL minimum!

- use any NLP solver (often work well!)
- use the multi-start trick from previous slides
- global optimization based on branch-and-reduce: `BARON`
    - constructs global underestimators
    - refines region by branching
    - tightens bounds by solving LPs
    - solve problems with 100s of variables
- "voodoo" solvers: genetic algorithm & simulated annealing
  no convergence theory ... usually worse than deterministic

Derivative-Free Optimization

My model does not have derivatives!

- Change your model ... good models have derivatives!
- pattern-search methods for $\min f(x)$
  - evaluate $f(x)$ at stencil $x_k + \Delta M$
  - move to new best point
  - extend to NLP; some convergence theory h
  - matlab: `NOMADm.m`; parallel `APPSPACK`
- solvers based on building interpolating quadratic models
  - DFO project on `www.coin-or.org`
  - Mike Powell's `NEWUOA` quadratic model
- "voodoo" solvers: genetic algorithm & simulated annealing
  no convergence theory ... usually worse than deterministic

Optimal Technology Penetration

Avoid global warming without ruining the economy!

Optimal Technology Penetration

Goal: Optimize energy production schedule and transition between old and new reduced-carbon technology to meet carbon targets

- Maximize social welfare
- Constraints:
  - GHG target at end of time
  - Reduced-carbon technology subject to learning effects
    ... reduced unit cost as new technology becomes widespread
- Assumptions on GHG emission rates, economic growth, energy costs

$\Rightarrow$ Optimal control problem

... model as finite-dimensional optimization problem...

Optimal Technology Penetration

Time: $t \in [0, T]$: function $x(t)$, derivative $\dot{x}(t) = \frac{dx(t)}{dt}$

Energy Output: old & new technology energy output: $q^o(t)$ and $q^n(t)$; total energy output: $Q(t) = q^o(t) + q^n(t)$.

Demand and Consumer Surplus: $\tilde{S}(Q, t)$: integral of demand derived from CES utility

Production Costs: c_o unit cost of old technology new technology from learning by doing: $x(t) = \int_0^t q^n(\tau) d\tau$

Greenhouse Gases Emissions: discount at environmental time preference rate:

$$\int_0^T e^{-at} \big(b_o q^o(t) + b_n q^n(t) \big) dt \le z_T$$

Optimal Technology Penetration

$$
\begin{aligned}
\underset{\{q^o,q^n,x,z\}(t)}{\text{maximize}} \quad & \int_0^T e^{-rt}\left[\tilde{S}(q^o(t)+q^n(t),t) - c_o q^o(t) - c_n(x(t))q^n(t)\right] dt \\
\text{subject to} \quad & \dot{x}(t) = q^n(t), \quad x(0) = x_0 = 0 \\
& \dot{z}(t) = e^{-at}\big(b_o q^o(t) + b_n q^n(t)\big), \quad z(0) = z_0 = 0 \\
& z(T) \le z_T \\
& q^o(t) \ge 0, \quad q^n(t) \ge 0.
\end{aligned}
$$

Optimal Technology Penetration

Discretization:

- $t \in [0, T]$ replaced by $N + 1$ equally spaced points $t_i = ih$
- $h := T/N$ time integration step-length
- approximate $q_i^n \simeq q^n(t_i)$ etc.

Replace differential equation

$$\dot{x}(t) = q^n(t)$$

by

$$x_{i+1} = x_i + hq_i^n$$

... use $h = 1$ (or even $h = 3$) years

Optimal Technology Penetration

Discretization:

- $t \in [0, T]$ replaced by $N + 1$ equally spaced points $t_i = ih$
- $h := T/N$ time integration step-length
- approximate $q_i^n \simeq q^n(t_i)$ etc.

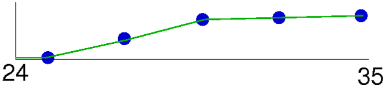
Replace differential equation

$$\dot{x}(t) = q^n(t)$$

by

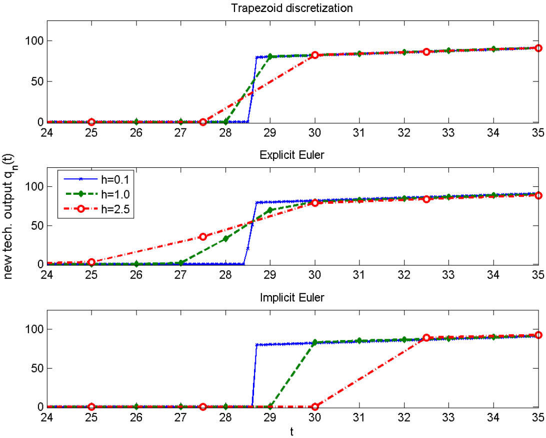
$$x_{i+1} = x_i + hq_i^n$$

... use $h = 1$ (or even $h = 3$) years



Output of new technology between $t = 24$ and $t = 35$

Optimal Technology Penetration with Varying h



Output of new technology for different discretization schemes and step-sizes ⇒ sharp transition (does not make sense economically)

Optimal Technology Penetration

Add adjustment cost to model building of capacity:

Capital and Investment:

- $K^j(t)$ amount of capital in technology j at t.
- $I^j(t)$ investment to increase $K^j(t)$.
- initial capital level as $\bar{K}_0^j$:

Notation:

- $Q(t) = q^o(t) + q^n(t)$
- $C(t) = C^o(q^o(t), K^o(t)) + C^n(q^n(t), K^n(t))$
- $I(t) = I^o(t) + I^n(t)$
- $K(t) = K^o(t) + K^n(t)$

Optimal Technology Penetration

$$\underset{\{q^j,K^j,I^j,x,z\}(t)}{\text{maximize}} \quad \left\{ \int_0^T e^{-rt} \left[\tilde{S}(Q(t),t) - C(t) - K(t) \right] dt + e^{-rT} K(T) \right\}$$

subject to

$$\dot{x}(t) = q^n(t), \quad x(0) = x_0 = 0$$

$$\dot{K}^j(t) = -\delta K^j(t) + I^j(t), \quad K^j(0) = \bar{K}_0^j, \quad j \in \{o,n\}$$

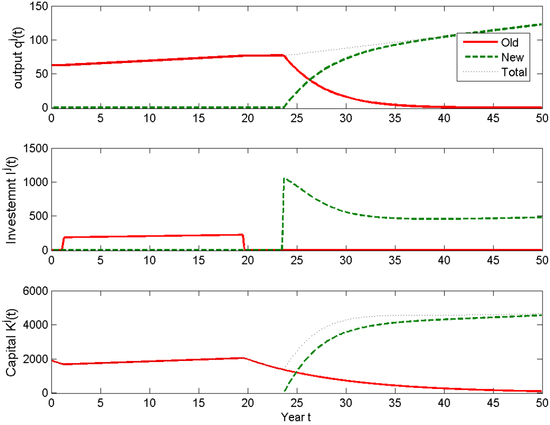
$$\dot{z}(t) = e^{-at}[b_o q^o(t) + b_n q^n(t)], \quad z(0) = z_0 = 0$$

$$z(T) \le z_T$$

$$q^j(t) \ge 0, \; j \in \{o,n\}$$

$$I^j(t) \ge 0, \; j \in \{o,n\}$$

Optimal Technology Penetration



Optimal output, investment, and capital for 50% CO2 reduction.

Pitfalls of Discretizations [Hager, 2000]

Optimal Control Problem

$$\text{minimize } \frac{1}{2}\int_0^1 u^2(t) + 2y^2(t)dt$$

subject to

$$\begin{array}{rcl} \dot{y}(t) & = & \frac{1}{2}y(t) + u(t),\ t \in [0,1], \\ y(0) & = & 1. \end{array}$$

$$\Rightarrow y^*(t) = \frac{2e^{3t} + e^3}{e^{3t/2}(2 + e^3)},$$
$$u^*(t) = \frac{2(e^{3t} - e^3)}{e^{3t/2}(2 + e^3)}.$$

Pitfalls of Discretizations [Hager, 2000]

Optimal Control Problem

$$\text{minimize } \tfrac{1}{2}\int_0^1 u^2(t) + 2y^2(t)dt$$

subject to

$$\begin{aligned} \dot{y}(t) &= \tfrac{1}{2}y(t) + u(t),\ t \in [0,1], \\ y(0) &= 1. \end{aligned}$$

$$\begin{aligned} \Rightarrow y^*(t) &= \frac{2e^{3t} + e^3}{e^{3t/2}(2+e^3)}, \\ u^*(t) &= \frac{2(e^{3t} - e^3)}{e^{3t/2}(2+e^3)}. \end{aligned}$$

Discretize with 2nd order RK

$$\text{minimize } \frac{h}{2}\sum_{k=0}^{K-1} u_{k+1/2}^2 + 2y_{k+1/2}^2$$

subject to $(k = 0, \ldots, K)$:

$$\begin{aligned} y_{k+1/2} &= y_k + \frac{h}{2}(\tfrac{1}{2}y_k + u_k), \\ y_{k+1} &= y_k + h(\tfrac{1}{2}y_{k+1/2} + u_{k+1/2}) \end{aligned}$$

Pitfalls of Discretizations [Hager, 2000]

Optimal Control Problem

$$\text{minimize } \tfrac{1}{2}\int_0^1 u^2(t) + 2y^2(t)dt$$

subject to

$$\begin{aligned} \dot{y}(t) &= \tfrac{1}{2}y(t) + u(t),\ t \in [0,1], \\ y(0) &= 1. \end{aligned}$$

$$\Rightarrow y^*(t) = \frac{2e^{3t} + e^3}{e^{3t/2}(2+e^3)},$$
$$u^*(t) = \frac{2(e^{3t} - e^3)}{e^{3t/2}(2+e^3)}.$$

Discretize with 2nd order RK

$$\text{minimize } \frac{h}{2}\sum_{k=0}^{K-1} u_{k+1/2}^2 + 2y_{k+1/2}^2$$

subject to $(k = 0, \ldots, K)$:

$$\begin{aligned} y_{k+1/2} &= y_k + \frac{h}{2}(\tfrac{1}{2}y_k + u_k), \\ y_{k+1} &= y_k + h(\tfrac{1}{2}y_{k+1/2} + u_{k+1/2}), \end{aligned}$$

Discrete solution $(k = 0, \ldots, K)$:

$$y_k = 1, \quad y_{k+1/2} = 0,$$
$$u_k = -\frac{4+h}{2h}, \quad u_{k+1/2} = 0,$$

DOES NOT CONVERGE!

Discretize-Then-Optimize

Discretization state equation implies discretization of adjoint

... may have different convergence properties.

Example problem (independent of solution of discretized problem!)

$$\dot{y}(t) = \tfrac{1}{2} y(t) + u(t), \qquad y(0) = 1,$$

$$y_{k+1/2} = y_k + \frac{\Delta t}{2}(\tfrac{1}{2} y_k + u_k), \qquad y_{k+1} = y_k + \Delta t(\tfrac{1}{2} y_{k+1/2} + u_{k+1/2}),$$

$$\dot{\lambda}(t) = -\tfrac{1}{2}\lambda(t) + 2y(t), \qquad \lambda(1) = 0,$$

$$\lambda_{k+1/2} = \Delta t(\tfrac{1}{2}\lambda_{k+1} - 2y_{k+1/2}), \qquad \lambda_k = \lambda_{k+1} + (1 + \Delta t/4)\lambda_{k+1/2},$$

$$u(t) - \lambda(t) = 0.$$

$$-\lambda_{k+1/2} = 0, \qquad u_{k+1/2} - \lambda_{k+1} = 0.$$

Tips to Solve Continuous-Time Problems

Alternative: Optimize-Then-Discretize

- consistent adjoint/dual discretization
- discretized gradients can be wrong!
- OK for equality constraints; harder for inequality constraints

Tips for handling continuous-time models

1. use discretize-then-optimize (easier)
2. refine discretization: $h = 1$ year discretization is nonsense
3. use different discretization schemes ... refine answers
4. check implied discretization of adjoints

... always be wary of fixed step-lengths

Optimization Conclusions

Optimization is General Modeling Paradigm

- linear, nonlinear, equations, inequalities
- integer variables, equilibrium, control

AMPL (GAMS) Modeling and Programming Languages

- express optimization problems
- use automatic differentiation
- easy access to state-of-the-art solvers

Optimization Software

- open-source: COIN-OR, IPOPT, SOPLEX, & ASTROS (soon)
- current solver limitations on laptop:
    - 1,000,000 variables/constraints for LPs
    - 100,000 variables/constraints for NLPs/NCPs
    - 100 variables/constraints for global optimization
    - 500,000,000 variable LP on BlueGene/L