Numerical Optimization for Economists

Todd Munson

Mathematics and Computer Science Division
Argonne National Laboratory

July 18–21, 2011

Part I

Numerical Optimization I: Static Models

Model Formulation

- Classify m people into two groups using v variables
  - $c \in \{0,1\}^m$ is the known classification
  - $d \in \Re^{m \times v}$ are the observations
  - $\beta \in \Re^{v+1}$ defines the separator
  - logit distribution function
- Maximum likelihood problem

$$\max_{\beta} \sum_{i=1}^{m} c_i \log(f(\beta, d_{i,\cdot})) + (1 - c_i)\log(1 - f(\beta, d_{i,\cdot}))$$

where

$$f(\beta, x) = \frac{\exp\left(\beta_0 + \sum_{j=1}^{v} \beta_j x_j\right)}{1 + \exp\left(\beta_0 + \sum_{j=1}^{v} \beta_j x_j\right)}$$

Solution Techniques

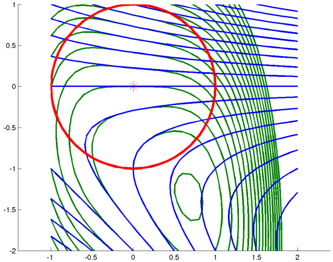
$$\min_x f(x)$$

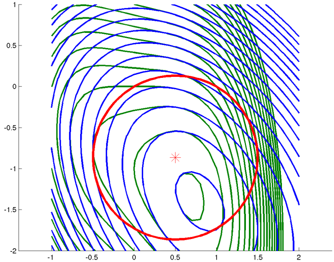
Main ingredients of solution approaches:

- Local method: given x_k (solution guess) compute a step s.
  - Gradient Descent
  - Quasi-Newton Approximation
  - Sequential Quadratic Programming
- Globalization strategy: converge from any starting point.
  - Trust region
  - Line search

Trust-Region Method

$$\begin{aligned} \min_{s} \quad & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T H(x_k) s \\ \text{subject to} \quad & \|s\| \leq \Delta_k \end{aligned}$$





Trust-Region Method

1. Initialize trust-region radius
   - Constant
   - Direction
   - Interpolation

Trust-Region Method

1. Initialize trust-region radius
   - Constant
   - Direction
   - Interpolation
2. Compute a new iterate
   1. Solve trust-region subproblem

$$\begin{array}{ll} \min_s & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T H(x_k) s \\ \text{subject to } \|s\| \leq \Delta_k & \end{array}$$

Trust-Region Method

1. Initialize trust-region radius
   - Constant
   - Direction
   - Interpolation
2. Compute a new iterate
   1. Solve trust-region subproblem

      $$\begin{array}{ll} \min_s & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T H(x_k) s \\ \text{subject to } \|s\| \leq \Delta_k & \end{array}$$

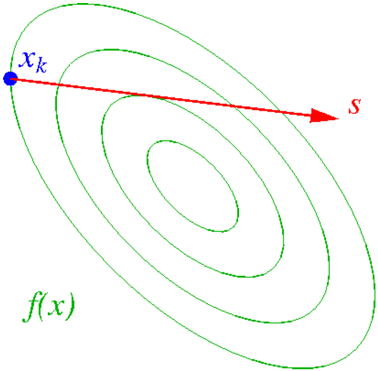
   2. Accept or reject iterate
   3. Update trust-region radius
      - Reduction
      - Interpolation
3. Check convergence

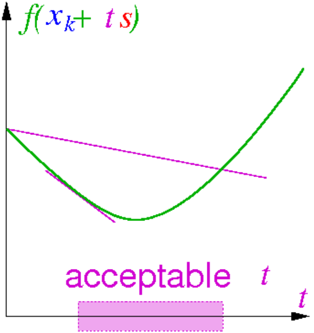
Solving the Subproblem

- Moré-Sorensen method
  - Computes global solution to subproblem
- Conjugate gradient method with trust region
  - Objective function decreases monotonically
  - Some choices need to be made
    - Preconditioner
    - Norm of direction and residual
    - Dealing with negative curvature

Line-Search Method

$$\min_s f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T (H(x_k) + \lambda_k I) s$$





Line-Search Method

1. Initialize perturbation to zero
2. Solve perturbed quadratic model

$$\min_s f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T (H(x_k) + \lambda_k I) s$$

Line-Search Method

1. Initialize perturbation to zero
2. Solve perturbed quadratic model

$$\min_s f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T (H(x_k) + \lambda_k I) s$$

3. Find new iterate
    1. Search along Newton direction
    2. Search along gradient-based direction

Line-Search Method

1. Initialize perturbation to zero
2. Solve perturbed quadratic model

$$\min_s f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T (H(x_k) + \lambda_k I) s$$

3. Find new iterate
    1. Search along Newton direction
    2. Search along gradient-based direction
4. Update perturbation
    - Decrease perturbation if the following hold
        - Iterative method succeeds
        - Search along Newton direction succeeds
    - Otherwise increase perturbation
5. Check convergence

Solving the Subproblem

- Conjugate gradient method

Solving the Subproblem

- Conjugate gradient method
- Conjugate gradient method with trust region
  - Initialize radius
    - Constant
    - Direction
    - Interpolation
  - Update radius
    - Reduction
    - Step length
    - Interpolation
  - Some choices need to be made
    - Preconditioner
    - Norm of direction and residual
    - Dealing with negative curvature

Performing the Line Search

- Backtracking Armijo Line search
  - Find t such that

$$f(x_k + ts) \leq f(x_k) + \sigma t \nabla f(x_k)^T s$$

  - Try $t = 1, \beta, \beta^2, \ldots$ for $0 < \beta < 1$
- More-Thuente Line search
  - Find t such that

$$\begin{aligned} f(x_k + ts) &\leq f(x_k) + \sigma t \nabla f(x_k)^T s \\ |\nabla f(x_k + ts)^T s| &\leq \delta |\nabla f(x_k)^T s| \end{aligned}$$

  - Construct cubic interpolant
  - Compute t to minimize interpolant
  - Refine interpolant

Updating the Perturbation

1. If increasing and $\Delta^k = 0$

$$\Delta^{k+1} = \mathrm{Proj}_{[\ell_0, u_0]}\left(\alpha_0 \|g(x^k)\|\right)$$

2. If increasing and $\Delta^k > 0$

$$\Delta^{k+1} = \mathrm{Proj}_{[\ell_i, u_i]}\left(\max\left(\alpha_i \|g(x^k)\|, \beta_i \Delta^k\right)\right)$$

3. If decreasing

$$\Delta^{k+1} = \min\left(\alpha_d \|g(x^k)\|, \beta_d \Delta^k\right)$$

4. If $\Delta^{k+1} < \ell_d$, then $\Delta^{k+1} = 0$

Trust-Region Line-Search Method

1. Initialize trust-region radius
   - Constant
   - Direction
   - Interpolation
2. Compute a new iterate
   1. Solve trust-region subproblem

      $$\begin{array}{ll} \min_s & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T H(x_k) s \\ \text{subject to } \|s\| \leq \Delta_k & \end{array}$$

   2. Search along direction
   3. Update trust-region radius
      - Reduction
      - Step length
      - Interpolation
3. Check convergence

Iterative Methods

- Conjugate gradient method
  - Stop if negative curvature encountered
  - Stop if residual norm is small

Iterative Methods

- Conjugate gradient method
  - Stop if negative curvature encountered
  - Stop if residual norm is small
- Conjugate gradient method with trust region
  - Nash
    - Follow direction to boundary if first iteration
    - Stop at base of direction otherwise
  - Steihaug-Toint
    - Follow direction to boundary
  - Generalized Lanczos
    - Compute tridiagonal approximation
    - Find global solution to approximate problem on boundary
    - Initialize perturbation with approximate minimum eigenvalue

Preconditioners

- No preconditioner
- Absolute value of Hessian diagonal
- Absolute value of perturbed Hessian diagonal
- Incomplete Cholesky factorization of Hessian
- Block Jacobi with Cholesky factorization of blocks
- Scaled BFGS approximation to Hessian matrix
  - None
  - Scalar
  - Diagonal of Broyden update
  - Rescaled diagonal of Broyden update
  - Absolute value of Hessian diagonal
  - Absolute value of perturbed Hessian diagonal

Norms

- Residual
  - Preconditioned – $\|r\|_{M^{-T}M^{-1}}$
  - Unpreconditioned – $\|r\|_2$
  - Natural – $\|r\|_{M^{-1}}$
- Direction
  - Preconditioned – $\|s\|_M \leq \Delta$
    - Monotonically increasing $\|s_{k+1}\|_M > \|s_k\|_M$.

Norms

- Residual
  - Preconditioned – $\|r\|_{M^{-T}M^{-1}}$
  - Unpreconditioned – $\|r\|_2$
  - Natural – $\|r\|_{M^{-1}}$
- Direction
  - Preconditioned – $\|s\|_M \leq \Delta$
    - Monotonically increasing $\|s_{k+1}\|_M > \|s_k\|_M$.
  - Unpreconditioned – $\|s\|_2 \leq \Delta$

Termination

- Typical convergence criteria
  - Absolute residual $\|\nabla f(x_k)\| < \tau_a$
  - Relative residual $\frac{\|\nabla f(x_k)\|}{\|\nabla f(x_k)\|} < \tau_r$
  - Unbounded objective $f(x_k) < \kappa$
  - Slow progress $|f(x_k) - f(x_{k-1})| < \epsilon$
  - Iteration limit
  - Time limit
- Solver status

Convergence Issues

- Quadratic convergence – best outcome
- Linear convergence
  - Far from a solution – $\|\nabla f(x_k)\|$ is large
  - Hessian is incorrect – disrupts quadratic convergence
  - Hessian is rank deficient – $\|\nabla f(x_k)\|$ is small
  - Limits of finite precision arithmetic
    1. $\|\nabla f(x_k)\|$ converges quadratically to small number
    2. $\|\nabla f(x_k)\|$ hovers around that number with no progress
- Domain violations such as $\frac{1}{x}$ when $x = 0$
  - Make implicit constraints explicit
- Nonglobal solution
  - Apply a multistart heuristic
  - Use global optimization solver

Some Available Software

- TRON – Newton method with trust-region
- LBFGS – Limited-memory quasi-Newton method with line search
- TAO – Toolkit for Advanced Optimization
  - NLS – Newton line-search method
  - NTR – Newton trust-region method
  - NTL – Newton line-search/trust-region method
  - LMVM – Limited-memory quasi-Newton method
  - CG – Nonlinear conjugate gradient methods

Model Formulation

- Economy with n agents and m commodities
  - $e \in \Re^{n \times m}$ are the endowments
  - $\alpha \in \Re^{n \times m}$ and $\beta \in \Re^{n \times m}$ are the utility parameters
  - $\lambda \in \Re^{n}$ are the social weights
- Social planning problem

$$\begin{array}{lll} \max\limits_{x \geq 0} & \displaystyle\sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{m} \frac{\alpha_{i,k}(1 + x_{i,k})^{1-\beta_{i,k}}}{1 - \beta_{i,k}} \right) & \\ \text{subject to} & \displaystyle\sum_{i=1}^{n} x_{i,k} \leq \sum_{i=1}^{n} e_{i,k} & \forall k = 1, \ldots, m \end{array}$$

Solving Constrained Optimization Problems

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{subject to} & c(x) \geq 0 \end{array}$$

Main ingredients of solution approaches:

- Local method: given x_k (solution guess) find a step s.
  - Sequential Quadratic Programming (SQP)
  - Sequential Linear/Quadratic Programming (SLQP)
  - Interior-Point Method (IPM)
- Globalization strategy: converge from any starting point.
  - Trust region
  - Line search
- Acceptance criteria: filter or penalty function.

Sequential Linear Programming

1. Initialize trust-region radius
2. Compute a new iterate

Sequential Linear Programming

1. Initialize trust-region radius
2. Compute a new iterate
    1. Solve linear program

$$
\begin{aligned}
\min_{s} \quad & f(x_k) + s^T \nabla f(x_k) \\
\text{subject to} \quad & c(x_k) + \nabla c(x_k)^T s \geq 0 \\
& \|s\| \leq \Delta_k
\end{aligned}
$$

Sequential Linear Programming

1. Initialize trust-region radius
2. Compute a new iterate
    1. Solve linear program

$$\begin{aligned} \min_{s} \quad & f(x_k) + s^T \nabla f(x_k) \\ \text{subject to } & c(x_k) + \nabla c(x_k)^T s \geq 0 \\ & \|s\| \leq \Delta_k \end{aligned}$$

    2. Accept or reject iterate
    3. Update trust-region radius
3. Check convergence

Sequential Quadratic Programming

1. Initialize trust-region radius
2. Compute a new iterate

Sequential Quadratic Programming

1. Initialize trust-region radius
2. Compute a new iterate
    1. Solve quadratic program

$$
\begin{array}{ll}
\min\limits_{s} & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T W(x_k) s \\
\text{subject to} & c(x_k) + \nabla c(x_k)^T s \geq 0 \\
 & \|s\| \leq \Delta_k
\end{array}
$$

Sequential Quadratic Programming

1. Initialize trust-region radius
2. Compute a new iterate
    1. Solve quadratic program

$$
\begin{array}{ll}
\min\limits_{s} & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T W(x_k) s \\
\text{subject to} & c(x_k) + \nabla c(x_k)^T s \geq 0 \\
 & \|s\| \leq \Delta_k
\end{array}
$$

    2. Accept or reject iterate
    3. Update trust-region radius
3. Check convergence

Sequential Linear Quadratic Programming

1. Initialize trust-region radius
2. Compute a new iterate

Sequential Linear Quadratic Programming

1. Initialize trust-region radius
2. Compute a new iterate
   1. Solve linear program to predict active set

$$
\begin{aligned}
\min_{d} \quad & f(x_k) + d^T \nabla f(x_k) \\
\text{subject to} \quad & c(x_k) + \nabla c(x_k)^T d \geq 0 \\
& \|d\| \leq \Delta_k
\end{aligned}
$$

Sequential Linear Quadratic Programming

1. Initialize trust-region radius
2. Compute a new iterate
   1. Solve linear program to predict active set

$$\begin{aligned} \min_{d} \quad & f(x_k) + d^T \nabla f(x_k) \\ \text{subject to} \quad & c(x_k) + \nabla c(x_k)^T d \geq 0 \\ & \|d\| \leq \Delta_k \end{aligned}$$

   2. Solve equality constrained quadratic program

$$\begin{aligned} \min_{s} \quad & f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T W(x_k) s \\ \text{subject to} \quad & c_{\mathcal{A}}(x_k) + \nabla c_{\mathcal{A}}(x_k)^T s = 0 \end{aligned}$$

   3. Accept or reject iterate
   4. Update trust-region radius
3. Check convergence

Acceptance Criteria

- Decrease objective function value: $f(x_k + s) \leq f(x_k)$
- Decrease constraint violation: $\|c_-(x_k + s)\| \leq \|c_-(x_k)\|$

Acceptance Criteria

- Decrease objective function value: $f(x_k + s) \leq f(x_k)$
- Decrease constraint violation: $\|c_-(x_k + s)\| \leq \|c_-(x_k)\|$
- Four possibilities
  1. step can decrease both $f(x)$ and $\|c_-(x)\|$ GOOD
  2. step can decrease $f(x)$ and increase $\|c_-(x)\|$???
  3. step can increase $f(x)$ and decrease $\|c_-(x)\|$???
  4. step can increase both $f(x)$ and $\|c_-(x)\|$ BAD

Acceptance Criteria

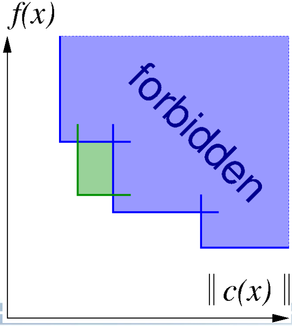
- Decrease objective function value: $f(x_k + s) \leq f(x_k)$
- Decrease constraint violation: $\|c_-(x_k + s)\| \leq \|c_-(x_k)\|$
- Four possibilities
  1. step can decrease both $f(x)$ and $\|c_-(x)\|$ GOOD
  2. step can decrease $f(x)$ and increase $\|c_-(x)\|$???
  3. step can increase $f(x)$ and decrease $\|c_-(x)\|$???
  4. step can increase both $f(x)$ and $\|c_-(x)\|$ BAD
- Filter uses concept from multi-objective optimization

(h_{k+1}, f_{k+1}) dominates (h_ℓ, f_ℓ) iff $h_{k+1} \leq h_\ell$ and $f_{k+1} \leq f_\ell$

Filter Framework

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

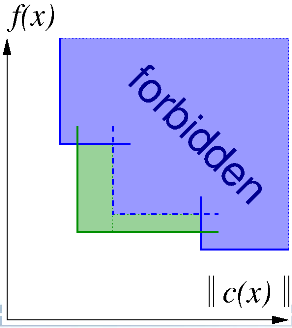
- new x_{k+1} is acceptable to filter $\mathcal{F}$ iff
  1. $h_{k+1} \leq h_\ell$ for all $\ell \in \mathcal{F}$ or
  2. $f_{k+1} \leq f_\ell$ for all $\ell \in \mathcal{F}$



Filter Framework

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

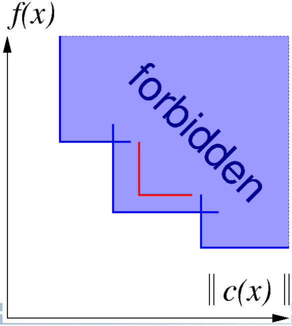
- new x_{k+1} is acceptable to filter $\mathcal{F}$ iff
  1. $h_{k+1} \leq h_\ell$ for all $\ell \in \mathcal{F}$ or
  2. $f_{k+1} \leq f_\ell$ for all $\ell \in \mathcal{F}$
- remove redundant filter entries



Filter Framework

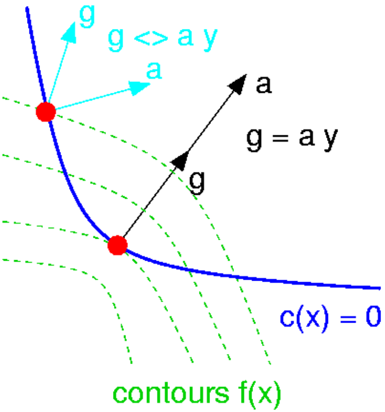
Filter $\mathcal{F}$: list of non-dominated pairs (h_ℓ, f_ℓ)

- new x_{k+1} is acceptable to filter $\mathcal{F}$ iff
  1. $h_{k+1} \leq h_\ell$ for all $\ell \in \mathcal{F}$ or
  2. $f_{k+1} \leq f_\ell$ for all $\ell \in \mathcal{F}$
- remove redundant filter entries
- new x_{k+1} is rejected if for some $\ell \in \mathcal{F}$
  1. $h_{k+1} > h_\ell$ and
  2. $f_{k+1} > f_\ell$



Convergence Criteria

- Feasible and no descent directions
  - Constraint qualification – LICQ, MFCQ
  - Linearized active constraints characterize directions
  - Objective gradient is a linear combination of constraint gradients



Optimality Conditions

- If x^* is a local minimizer and a constraint qualification holds, then there exist multipliers $\lambda^* \geq 0$ such that

$$\nabla f(x^*) - \nabla c_{\mathcal{A}}(x^*)^T \lambda^*_{\mathcal{A}} = 0$$

  - Lagrangian function $\mathcal{L}(x, \lambda) := f(x) - \lambda^T c(x)$
- Optimality conditions can be written as

$$\begin{aligned} &\nabla f(x) - \nabla c(x)^T \lambda = 0 \\ &0 \leq \lambda \perp c(x) \geq 0 \end{aligned}$$

  - Complementarity problem

Termination

- Feasible and complementary $\|\min(c(x_k), \lambda_k)\| \leq \tau_f$
- Optimal $\|\nabla_x \mathcal{L}(x_k, \lambda_k)\| \leq \tau_o$
- Other possible conditions
  - Slow progress
  - Iteration limit
  - Time limit
- Multipliers and reduced costs

```
display consumption.slack;      # Constraint violation
display consumption.dual;       # Lagrange multipliers

display x.rc;                   # Gradient of Lagrangian
```

Convergence Issues

- Quadratic convergence – best outcome
- Globally infeasible – linear constraints infeasible
- Locally infeasible – nonlinear constraints locally infeasible
- Unbounded objective – hard to detect
- Unbounded multipliers – constraint qualification not satisfied
- Linear convergence rate
  - Far from a solution – $\|\nabla f(x_k)\|$ is large
  - Hessian is incorrect – disrupts quadratic convergence
  - Hessian is rank deficient – $\|\nabla f(x_k)\|$ is small
  - Limits of finite precision arithmetic
- Domain violations such as $\frac{1}{x}$ when $x = 0$
  - Make implicit constraints explicit
- Nonglobal solutions
  - Apply a multistart heuristic
  - Use global optimization solver

Some Available Software

- 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
  - trust-region with ℓ_1 penalty
  - use with `knitro_options = "algorithm=3";`

Part II

Numerical Optimization II: Optimal Control

Model Formulation

- Maximize discounted utility
  - $u(\cdot)$ is the utility function
  - R is the retirement age
  - T is the terminal age
  - w is the wage
  - β is the discount factor
  - r is the interest rate
- Optimization problem

$$
\begin{aligned}
\max_{s,c} \quad & \sum_{t=0}^{T} \beta^t u(c_t) \\
\text{subject to} \quad & s_{t+1} = (1+r)s_t + w - c_t \quad t = 0, \ldots, R-1 \\
& s_{t+1} = (1+r)s_t - c_t \qquad\quad t = R, \ldots, T \\
& s_0 = s_{T+1} = 0
\end{aligned}
$$

Model: life1.mod

```
param R > 0, integer;                # Retirement age
param T > R, integer;                # Terminal age

param beta >= 0, < 1;                # Discount factor
param rate >= 0, < 1;                # Interest rate
param wage >= 0;                     # Wage rate
```

Model: life1.mod

```
param R > 0, integer;                   # Retirement age
param T > R, integer;                   # Terminal age

param beta >= 0, < 1;                   # Discount factor
param rate >= 0, < 1;                   # Interest rate
param wage >= 0;                        # Wage rate


var c{0..T};                            # Consumption
var s{0..T+1};                          # Savings
var u{t in 0..T} = -exp(-c[t]);         # Utility
```

Model: life1.mod

```
param R > 0, integer;                   # Retirement age
param T > R, integer;                   # Terminal age

param beta >= 0, < 1;                   # Discount factor
param rate >= 0, < 1;                   # Interest rate
param wage >= 0;                        # Wage rate


var c{0..T};                            # Consumption
var s{0..T+1};                          # Savings
var u{t in 0..T} = -exp(-c[t]);         # Utility


maximize utility:
    sum {t in 0..T} beta^t * u[t];

subject to
  working {t in 0..R-1}:
    s[t+1] = (1+rate)*s[t] + wage - c[t];

  retired {t in R..T}:
    s[t+1] = (1+rate)*s[t] - c[t];

  initial:
    s[0] = 0;

  terminal:
    s[T+1] = 0;
```

Data: life.dat

```
param R := 75;                          # Retirement age
param T := 100;                         # Terminal age

param beta := 0.9;                      # Discount factor
param rate := 0.2;                      # Interest rate
param wage := 1.0;                      # Wage rate
```

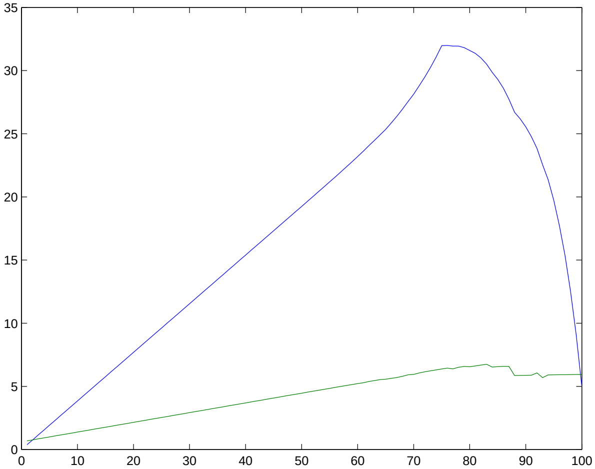
Commands: life1.cmd

```
# Load model and data
model life1.mod;
data life.dat;

# Specify solver and options
option solver mpec;

# Solve the instance
solve;

# Output results
printf {t in 0..T} "%2d %5.4e %5.4e\n", t, s[t], c[t] > out1.dat;
```

Output

```
ampl: include life1.cmd
AMPL interface to filter-MPEC: 20040408
: filter objective function   = -3.24322
       constraint violation = 1.01433e-11
Optimal solution found
14 iterations (0 for feasibility)
Evals: obj = 15, constr = 16, grad = 16, Hes = 15
ampl: quit;
```

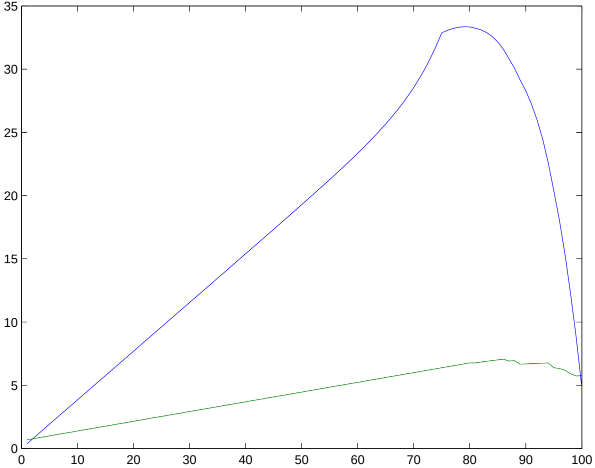
Plot of Output



Model: life2.mod

```
param R > 0, integer;                    # Retirement age
param T > R, integer;                    # Terminal age

param beta >= 0, < 1;                    # Discount factor
param rate >= 0, < 1;                    # Interest rate
param wage >= 0;                         # Wage rate

var cbar{0..T};                          # Scaled consumption
var c{t in 0..T} = cbar[t] / beta^t;     # Actual consumption
var s{0..T+1};                           # Savings
var u{t in 0..T} = -exp(-cbar[t] / beta^t);

maximize utility:
    sum {t in 0..T} beta^t * u[t];

subject to
  working {t in 0..R-1}:
    s[t+1] = (1+rate)*s[t] + wage - cbar[t] / beta^t;

  retired {t in R..T}:
    s[t+1] = (1+rate)*s[t] - cbar[t] / beta^t;

  initial:
    s[0] = 0;

  terminal:
    s[T+1] = 0;
```

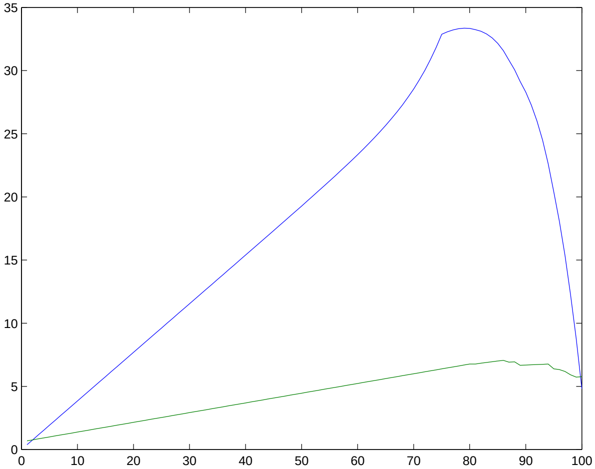
Plot of Output



Model: life3.mod

```
param R > 0, integer;                   # Retirement age
param T > R, integer;                   # Terminal age

param beta >= 0, < 1;                   # Discount factor
param rate >= 0, < 1;                   # Interest rate
param wage >= 0;                        # Wage rate

var cbar{0..T};                         # Scaled consumption
var c{t in 0..T} = cbar[t] / beta^t;    # Actual consumption
var s{0..T+1};                          # Savings
var u{t in 0..T} = -exp(-cbar[t] / beta^t);

maximize utility:
    sum {t in 0..T} beta^t * u[t];

subject to
  working {t in 0..R-1}:
    beta^t*s[t+1] = beta^t*(1+rate)*s[t] + beta^t*wage - cbar[t];

  retired {t in R..T}:
    beta^t*s[t+1] = beta^t*(1+rate)*s[t] - cbar[t];

  initial:
    s[0] = 0;

  terminal:
    s[T+1] = 0;
```

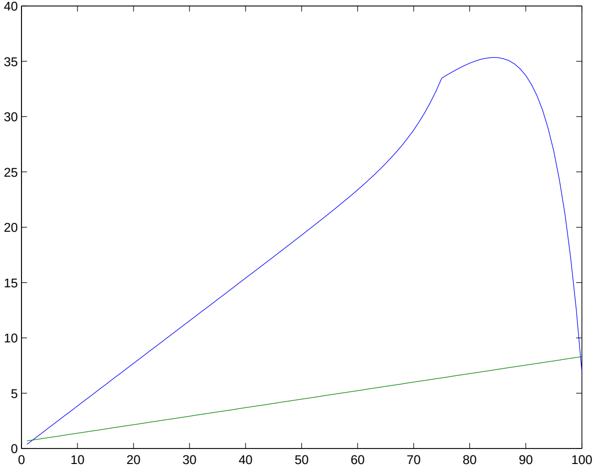
Plot of Output



Model: life4.mod

```
param R > 0, integer;                    # Retirement age
param T > R, integer;                    # Terminal age

param beta >= 0, < 1;                    # Discount factor
param rate >= 0, < 1;                    # Interest rate
param wage >= 0;                         # Wage rate

var cbar{0..T};                          # Scaled consumption
var c{t in 0..T} = cbar[t] / beta^t;     # Actual consumption
var sbar{0..T+1};                        # Scaled savings
var s{t in 0..T+1} = sbar[t] / beta^t;   # Actual savings
var u{t in 0..T} = -exp(-cbar[t] / beta^t);

maximize utility:
    sum {t in 0..T} beta^t * u[t];

subject to
  working {t in 0..R-1}:
    sbar[t+1]/beta = (1+rate)*sbar[t] + beta^t*wage - cbar[t];

  retired {t in R..T}:
    sbar[t+1]/beta = (1+rate)*sbar[t] - cbar[t];

  initial:
    sbar[0] = 0;

  terminal:
    sbar[T+1] = 0;
```

Plot of Output



Solving Constrained Optimization Problems

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \; & c(x) \geq 0 \end{aligned}$$

Main ingredients of solution approaches:

- Local method: given x_k (solution guess) find a step s.
  - Sequential Quadratic Programming (SQP)
  - Sequential Linear/Quadratic Programming (SLQP)
  - Interior-Point Method (IPM)
- Globalization strategy: converge from any starting point.
  - Trust region
  - Line search
- Acceptance criteria: filter or penalty function.

Interior-Point Method

- Reformulate optimization problem with slacks

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

- Construct perturbed optimality conditions

$$F_\tau(x, y, z) = \begin{bmatrix} \nabla f(x) - \nabla c(x)^T y - z \\ c(x) \\ Xz - \tau e \end{bmatrix}$$

- Central path $\{x(\tau), y(\tau), z(\tau) \mid \tau > 0\}$
- Apply Newton's method for sequence $\tau \searrow 0$

Interior-Point Method

1. Compute a new iterate
    1. Solve linear system of equations

$$\begin{bmatrix} W_k & -\nabla c(x_k)^T & -I \\ \nabla c(x_k) & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = -F_\mu(x_k, y_k, z_k)$$

    2. Accept or reject iterate
    3. Update parameters
2. Check convergence

Convergence Issues

- Quadratic convergence – best outcome
- Globally infeasible – linear constraints infeasible
- Locally infeasible – nonlinear constraints locally infeasible
- Dual infeasible – dual problem is locally infeasible
- Unbounded objective – hard to detect
- Unbounded multipliers – constraint qualification not satisfied
- Duality gap
- Domain violations such as $\frac{1}{x}$ when $x = 0$
    - Make implicit constraints explicit
- Nonglobal solutions
    - Apply a multistart heuristic
    - Use global optimization solver

Some Available Software

- IPOPT – open source in COIN-OR
  - line-search filter algorithm
- 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
  - Newton system: modified Cholesky factorization

Optimal Technology

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

- Maximize social welfare
- Constraints
  - Limit total greenhouse gas emissions
  - Low-carbon technology less costly as it becomes widespread
- Assumptions on emission rates, economic growth, and energy costs

Model Formulation

- Finite time: $t \in [0, T]$
- Instantaneous energy output: $q^o(t)$ and $q^n(t)$
- Cumulative energy output: $x^o(t)$ and $x^n(t)$

$$x^n(t) = \int_0^t q^n(\tau)d\tau$$

- Discounted greenhouse gases emissions

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

- Consumer surplus $S(Q(t), t)$ derived from utility
- Production costs
  - c_o per unit cost of old technology
  - $c_n(x^n(t))$ per unit cost of new technology (learning by doing)

Continuous-Time Model

$$\max_{\{q^o,q^n,x^n,z\}(t)} \int_0^T e^{-rt}\left[S(q^o(t)+q^n(t),t) - c_o q^o(t) - c_n(x^n(t))q^n(t)\right] dt$$

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

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

$$z(T) \leq z_T$$

$$q^o(t) \geq 0, \quad q^n(t) \geq 0.$$

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

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

Optimal Technology Penetration

Discretization:

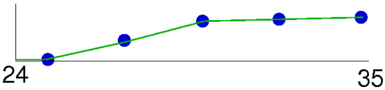
- $t \in [0, T]$ replaced by $N + 1$ equally spaced points $t_i = ih$
- $h := T/N$ time integration step-length
- approximate $q_i^n \simeq q^n(t_i)$ etc.

Replace differential equation

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

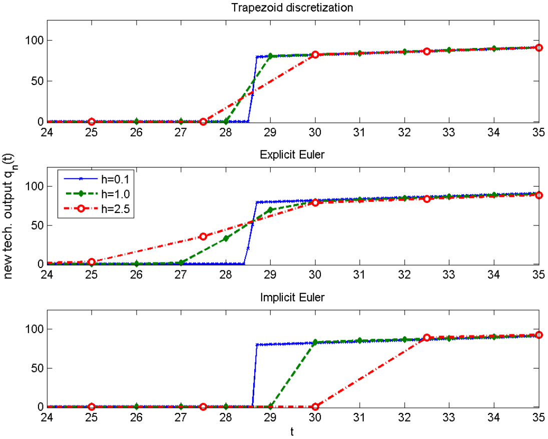
by

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



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

Solution with Varying h



Output for different discretization schemes and step-sizes

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 $\bar{K}_0^j$:

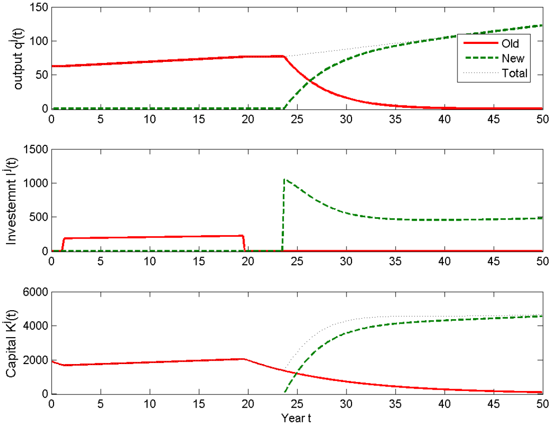
Notation:

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

Optimal Technology Penetration

$$
\begin{aligned}
\underset{\{q^j, K^j, I^j, x, z\}(t)}{\text{maximize}} \quad & \left\{ \int_0^T e^{-rt} \left[\tilde{S}(Q(t), t) - C(t) - K(t) \right] dt + e^{-rT} K(T) \right\} \\
\text{subject to} \quad & \dot{x}(t) = q^n(t), \quad x(0) = x_0 = 0 \\
& \dot{K}^j(t) = -\delta K^j(t) + I^j(t), \quad K^j(0) = \bar{K}_0^j, \quad j \in \{o, n\} \\
& \dot{z}(t) = e^{-at}[b_o q^o(t) + b_n q^n(t)], \quad z(0) = z_0 = 0 \\
& z(T) \le z_T \\
& q^j(t) \ge 0, \; j \in \{o, n\} \\
& I^j(t) \ge 0, \; j \in \{o, n\}
\end{aligned}
$$

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

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

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

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

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

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

Discretize with 2nd order RK

$$\text{minimize } \frac{h}{2}\sum_{k=0}^{K-1} u^2_{k+1/2} + 2y^2_{k+1/2}$$

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

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

Pitfalls of Discretizations [Hager, 2000]

Optimal Control Problem

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

subject to

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

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

Discretize with 2nd order RK

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

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

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

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!

Tips to Solve Continuous-Time Problems

- Use discretize-then-optimize with different schemes
- Refine discretization: $h = 1$ discretization is nonsense
- Check implied discretization of adjoints

Tips to Solve Continuous-Time Problems

- Use discretize-then-optimize with different schemes
- Refine discretization: $h = 1$ discretization is nonsense
- Check implied discretization of adjoints

Alternative: Optimize-Then-Discretize

- Consistent adjoint/dual discretization
- Discretized gradients can be wrong!
- Harder for inequality constraints

Ordered Sets

```
param V, integer;                              # Number of vertices
param E, integer;                              # Number of elements

set VERTICES := {1..V};                        # Vertex indices
set ELEMENTS := {1..E};                        # Element indices
set COORDS   := {1..3} ordered;                # Spatial coordinates

param T{ELEMENTS, 1..4} in VERTICES;           # Tetrahedral elements

var x{VERTICES, COORDS};                       # Position of vertices

var norm{e in ELEMENTS} = sum{i in COORDS, j in 1..4}
  (x[T[e,j], i] - x[T[e,1], i])^2;

var area{e in ELEMENTS} = sum{i in COORDS}
  (x[T[e,2], i] - x[T[e,1], i]) *
    ((x[T[e,3], nextw(i)] - x[T[e,1], nextw(i)]) *
     (x[T[e,4], prevw(i)] - x[T[e,1], prevw(i)]) -
     (x[T[e,3], prevw(i)] - x[T[e,1], prevw(i)]) *
     (x[T[e,4], nextw(i)] - x[T[e,1], nextw(i)]));

minimize f: sum {e in ELEMENTS} norm[e] / max(area[e], 0) ^ (2 / 3);
```

Circular Sets

```
param V, integer;                              # Number of vertices
param E, integer;                              # Number of elements

set VERTICES := {1..V};                        # Vertex indices
set ELEMENTS := {1..E};                        # Element indices
set COORDS   := {1..3} circular;               # Spatial coordinates

param T{ELEMENTS, 1..4} in VERTICES;           # Tetrahedral elements

var x{VERTICES, COORDS};                       # Position of vertices

var norm{e in ELEMENTS} = sum{i in COORDS, j in 1..4}
  (x[T[e,j], i] - x[T[e,1], i])^2;

var area{e in ELEMENTS} = sum{i in COORDS}
  (x[T[e,2], i] - x[T[e,1], i]) *
    ((x[T[e,3], next(i)] - x[T[e,1], next(i)]) *
     (x[T[e,4], prev(i)] - x[T[e,1], prev(i)]) -
     (x[T[e,3], prev(i)] - x[T[e,1], prev(i)]) *
     (x[T[e,4], next(i)] - x[T[e,1], next(i)]));

minimize f: sum {e in ELEMENTS} norm[e] / max(area[e], 0) ^ (2 / 3);
```

Part III

Numerical Optimization III: Complementarity Constraints

Nash Games

- Non-cooperative game played by n individuals
  - Each player selects a strategy to optimize their objective
  - Strategies for the other players are fixed
- Equilibrium reached when no improvement is possible

Nash Games

- Non-cooperative game played by n individuals
  - Each player selects a strategy to optimize their objective
  - Strategies for the other players are fixed
- Equilibrium reached when no improvement is possible
- Characterization of two player equilibrium (x^*, y^*)

$$
\begin{aligned}
x^* &\in \begin{cases} \underset{x \geq 0}{\arg\min} & f_1(x, y^*) \\ \text{subject to} & c_1(x) \leq 0 \end{cases} \\
y^* &\in \begin{cases} \underset{y \geq 0}{\arg\min} & f_2(x^*, y) \\ \text{subject to} & c_2(y) \leq 0 \end{cases}
\end{aligned}
$$

Nash Games

- Non-cooperative game played by n individuals
  - Each player selects a strategy to optimize their objective
  - Strategies for the other players are fixed
- Equilibrium reached when no improvement is possible
- Characterization of two player equilibrium (x^*, y^*)

$$x^* \in \begin{cases} \arg\min\limits_{x \geq 0} & f_1(x, y^*) \\ \text{subject to } c_1(x) \leq 0 \end{cases}$$
$$y^* \in \begin{cases} \arg\min\limits_{y \geq 0} & f_2(x^*, y) \\ \text{subject to } c_2(y) \leq 0 \end{cases}$$

- Many applications in economics
  - Bimatrix games
  - Cournot duopoly models
  - General equilibrium models
  - Arrow-Debreau models

Complementarity Formulation

- Assume each optimization problem is convex
  - $f_1(\cdot, y)$ is convex for each y
  - $f_2(x, \cdot)$ is convex for each x
  - $c_1(\cdot)$ and $c_2(\cdot)$ satisfy constraint qualification
- Then the first-order conditions are necessary and sufficient

$$\begin{array}{ll} \min\limits_{x \geq 0} & f_1(x, y^*) \\ \text{subject to} & c_1(x) \leq 0 \end{array} \Leftrightarrow \begin{array}{l} 0 \leq x \perp \nabla_x f_1(x, y^*) + \lambda_1^T \nabla_x c_1(x) \geq 0 \\ 0 \leq \lambda_1 \perp -c_1(x) \geq 0 \end{array}$$

Complementarity Formulation

- Assume each optimization problem is convex
  - $f_1(\cdot, y)$ is convex for each y
  - $f_2(x, \cdot)$ is convex for each x
  - $c_1(\cdot)$ and $c_2(\cdot)$ satisfy constraint qualification
- Then the first-order conditions are necessary and sufficient

$$\begin{array}{ll} \min\limits_{y \geq 0} & f_2(x^*, y) \\ \text{subject to } c_2(y) \leq 0 & \end{array} \Leftrightarrow \begin{array}{l} 0 \leq y \;\perp \nabla_y f_2(x^*, y) + \lambda_2^T \nabla_y c_2(y) \geq 0 \\ 0 \leq \lambda_2 \perp -c_2(y) \geq 0 \end{array}$$

Complementarity Formulation

- Assume each optimization problem is convex
  - $f_1(\cdot, y)$ is convex for each y
  - $f_2(x, \cdot)$ is convex for each x
  - $c_1(\cdot)$ and $c_2(\cdot)$ satisfy constraint qualification
- Then the first-order conditions are necessary and sufficient

$$
\begin{array}{l}
0 \leq x \quad \perp \nabla_x f_1(x, y) + \lambda_1^T \nabla_x c_1(x) \geq 0 \\
0 \leq y \quad \perp \nabla_y f_2(x, y) + \lambda_2^T \nabla_y c_2(y) \geq 0 \\
0 \leq \lambda_1 \perp -c_1(y) \geq 0 \\
0 \leq \lambda_2 \perp -c_2(y) \geq 0
\end{array}
$$

- Nonlinear complementarity problem
  - Square system – number of variables and constraints the same
  - Each solution is an equilibrium for the Nash game

Model Formulation

- Economy with n agents and m commodities
  - $e \in \Re^{n \times m}$ are the endowments
  - $\alpha \in \Re^{n \times m}$ and $\beta \in \Re^{n \times m}$ are the utility parameters
  - $p \in \Re^{m}$ are the commodity prices
- Agent i maximizes utility with budget constraint

$$\begin{array}{ll} \max\limits_{x_{i,*} \geq 0} & \displaystyle\sum_{k=1}^{m} \frac{\alpha_{i,k}(1 + x_{i,k})^{1-\beta_{i,k}}}{1 - \beta_{i,k}} \\ \text{subject to} & \displaystyle\sum_{k=1}^{m} p_k \left(x_{i,k} - e_{i,k}\right) \leq 0 \end{array}$$

- Market k sets price for the commodity

$$0 \leq p_k \perp \sum_{i=1}^{n} \left(e_{i,k} - x_{i,k}\right) \geq 0$$

Model: cge.mod

```
set AGENTS;                                    # Agents
set COMMODITIES;                               # Commodities

param e {AGENTS, COMMODITIES} >= 0, default 1; # Endowment

param alpha {AGENTS, COMMODITIES} > 0;         # Utility parameters
param beta {AGENTS, COMMODITIES} > 0;

var x {AGENTS, COMMODITIES};                   # Consumption (no bounds!)
var l {AGENTS};                                # Multipliers (no bounds!)
var p {COMMODITIES};                           # Prices (no bounds!)

var du {i in AGENTS, k in COMMODITIES} =       # Marginal prices
  alpha[i,k] / (1 + x[i,k])^beta[i,k];

subject to
  optimality {i in AGENTS, k in COMMODITIES}:
    0 <= x[i,k] complements -du[i,k] + p[k] * l[i] >= 0;

  budget {i in AGENTS}:
    0 <= l[i]   complements sum {k in COMMODITIES} p[k]*(e[i,k] - x[i,k]) >= 0;

  market {k in COMMODITIES}:
    0 <= p[k]   complements sum {i in AGENTS} (e[i,k] - x[i,k]) >= 0;
```

Data: cge.dat

```
set AGENTS := Jorge, Sven, Todd;
set COMMODITIES := Books, Cars, Food, Pens;

param alpha : Books  Cars  Food  Pens :=
     Jorge      1      1     1     1
     Sven       1      2     3     4
     Todd       2      1     1     5;

param beta (tr): Jorge  Sven  Todd :=
     Books         1.5    2     0.6
     Cars          1.6    3     0.7
     Food          1.7    2     2.0
     Pens          1.8    2     2.5;
```

Commands: cge.cmd

```
# Load model and data
model cge.mod;
data cge.dat;

# Specify solver and options
option presolve 0;
option solver "pathampl";

# Solve the instance
solve;

# Output results
printf {i in AGENTS, k in COMMODITIES} "%5s %5s: % 5.4e\n", i, k, x[i,k] > cge.out;
printf "\n" > cge.out;
printf {k in COMMODITIES} "%5s: % 5.4e\n", k, p[k] > cge.out;
```

Results: cge.out

```
Jorge Books:  8.9825e-01
Jorge  Cars:  1.4651e+00
Jorge  Food:  1.2021e+00
Jorge  Pens:  6.8392e-01
 Sven Books:  2.5392e-01
 Sven  Cars:  7.2054e-01
 Sven  Food:  1.6271e+00
 Sven  Pens:  1.4787e+00
 Todd Books:  1.8478e+00
 Todd  Cars:  8.1431e-01
 Todd  Food:  1.7081e-01
 Todd  Pens:  8.3738e-01

Books:  1.0825e+01
 Cars:  6.6835e+00
 Food:  7.3983e+00
 Pens:  1.1081e+01
```

Commands: cgenum.cmd

```
# Load model and data
model cge.mod;
data cge.dat;

# Specify solver and options
option presolve 0;
option solver "pathampl";

# Solve the instance
drop market['Books'];
fix p['Books'] := 1;
solve;

# Output results
printf {i in AGENTS, k in COMMODITIES} "%5s %5s: % 5.4e\n", i, k, x[i,k] > cgenum.out;
printf "\n" > cgenum.out;
printf {k in COMMODITIES} "%5s: % 5.4e\n", k, p[k] > cgenum.out;
```

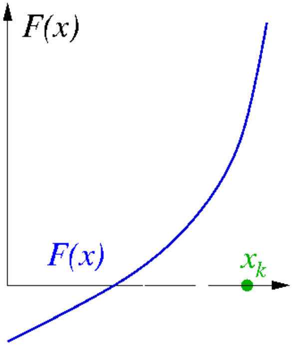
Results: cgenum.out

```
Jorge Books:  8.9825e-01
Jorge  Cars:  1.4651e+00
Jorge  Food:  1.2021e+00
Jorge  Pens:  6.8392e-01
 Sven Books:  2.5392e-01
 Sven  Cars:  7.2054e-01
 Sven  Food:  1.6271e+00
 Sven  Pens:  1.4787e+00
 Todd Books:  1.8478e+00
 Todd  Cars:  8.1431e-01
 Todd  Food:  1.7081e-01
 Todd  Pens:  8.3738e-01

Books:  1.0000e+00
 Cars:  6.1742e-01
 Food:  6.8345e-01
 Pens:  1.0237e+00
```

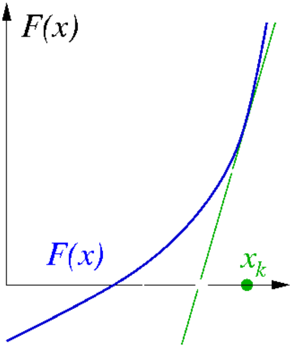
Pitfalls

- Nonsquare systems
  - Side variables
  - Side constraints
- Orientation of equations
  - Skew symmetry preferred
  - Proximal point perturbation
- AMPL presolve
  - `option presolve 0;`

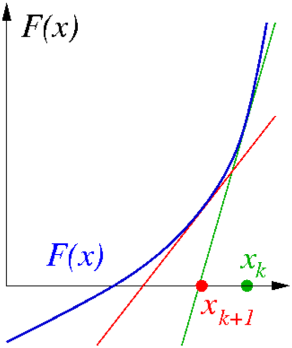
Newton Method for Nonlinear Equations



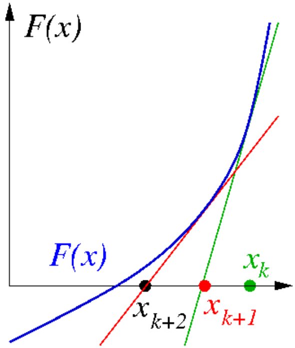
Newton Method for Nonlinear Equations



Newton Method for Nonlinear Equations



Newton Method for Nonlinear Equations



Methods for Complementarity Problems

- Sequential linearization methods (PATH)
  1. Solve the linear complementarity problem

$$0 \leq x \quad \perp \quad F(x_k) + \nabla F(x_k)(x - x_k) \geq 0$$

  2. Perform a line search along merit function
  3. Repeat until convergence

Methods for Complementarity Problems

- Sequential linearization methods (PATH)
  1. Solve the linear complementarity problem

     $$0 \leq x \quad \perp \quad F(x_k) + \nabla F(x_k)(x - x_k) \geq 0$$

  2. Perform a line search along merit function
  3. Repeat until convergence
- Semismooth reformulation methods (SEMI)
  - Solve linear system of equations to obtain direction
  - Globalize with a trust region or line search
  - Less robust in general
- Interior-point methods

Semismooth Reformulation

- Define Fischer-Burmeister function

$$\phi(a, b) := a + b - \sqrt{a^2 + b^2}$$

  - $\phi(a, b) = 0$ iff $a \geq 0,\ b \geq 0$, and $ab = 0$
- Define the system

$$[\Phi(x)]_i = \phi(x_i, F_i(x))$$

  - x^* solves complementarity problem iff $\Phi(x^*) = 0$
- Nonsmooth system of equations

Semismooth Algorithm

1. Calculate $H^k \in \partial_B \Phi(x^k)$ and solve the following system for d^k:

$$H^k d^k = -\Phi(x^k)$$

If this system either has no solution, or

$$\nabla \Psi(x^k)^T d^k \leq -p_1 \|d^k\|^{p_2}$$

is not satisfied, let $d^k = -\nabla \Psi(x^k)$.

Semismooth Algorithm

1. Calculate $H^k \in \partial_B \Phi(x^k)$ and solve the following system for d^k:

   $$H^k d^k = -\Phi(x^k)$$

   If this system either has no solution, or

   $$\nabla \Psi(x^k)^T d^k \leq -p_1 \|d^k\|^{p_2}$$

   is not satisfied, let $d^k = -\nabla \Psi(x^k)$.

2. Compute smallest nonnegative integer i^k such that

   $$\Psi(x^k + \beta^{i^k} d^k) \leq \Psi(x^k) + \sigma \beta^{i^k} \nabla \Psi(x^k) d^k$$

3. Set $x^{k+1} = x^k + \beta^{i^k} d^k$, $k = k+1$, and go to 1.

Convergence Issues

- Quadratic convergence – best outcome
- Linear convergence
  - Far from a solution – $r(x_k)$ is large
  - Jacobian is incorrect – disrupts quadratic convergence
  - Jacobian is rank deficient – $\|\nabla r(x_k)\|$ is small
  - Converge to local minimizer – guarantees rank deficiency
  - Limits of finite precision arithmetic
    1. $r(x_k)$ converges quadratically to small number
    2. $r(x_k)$ hovers around that number with no progress
- Domain violations such as $\frac{1}{x}$ when $x = 0$

Some Available Software

- PATH – sequential linearization method
- MILES – sequential linearization method
- SEMI – semismooth linesearch method
- TAO – Toolkit for Advanced Optimization
  - SSLS – full-space semismooth linesearch methods
  - ASLS – active-set semismooth linesearch methods
  - RSCS – reduced-space method

Definition

- Leader-follower game
  - Dominant player (leader) selects a strategy y^*
  - Then followers respond by playing a Nash game

$$x_i^* \in \begin{cases} \displaystyle\operatorname*{arg\,min}_{x_i \geq 0} & f_i(x, y) \\ \text{subject to } c_i(x_i) \leq 0 \end{cases}$$

- Leader solves optimization problem with equilibrium constraints

$$\begin{array}{ll} \displaystyle\min_{y \geq 0, x, \lambda} & g(x, y) \\ \text{subject to } h(y) \leq 0 & \\ & 0 \leq x_i \quad \perp \quad \nabla_{x_i} f_i(x, y) + \lambda_i^T \nabla_{x_i} c_i(x_i) \geq 0 \\ & 0 \leq \lambda_i \quad \perp \quad -c_i(x_i) \geq 0 \end{array}$$

- Many applications in economics
  - Optimal taxation
  - Tolling problems

Model Formulation

- Economy with n agents and m commodities
  - $e \in \Re^{n \times m}$ are the endowments
  - $\alpha \in \Re^{n \times m}$ and $\beta \in \Re^{n \times m}$ are the utility parameters
  - $p \in \Re^{m}$ are the commodity prices
- Agent i maximizes utility with budget constraint

$$\begin{array}{ll} \max\limits_{x_{i,*} \geq 0} & \displaystyle\sum_{k=1}^{m} \frac{\alpha_{i,k}(1 + x_{i,k})^{1-\beta_{i,k}}}{1 - \beta_{i,k}} \\ \text{subject to} & \displaystyle\sum_{k=1}^{m} p_k \left(x_{i,k} - e_{i,k}\right) \leq 0 \end{array}$$

- Market k sets price for the commodity

$$0 \leq p_k \perp \sum_{i=1}^{n} (e_{i,k} - x_{i,k}) \geq 0$$

Model: cgempec.mod

```
set LEADER;                                      # Leader
set FOLLOWERS;                                   # Followers
set AGENTS := LEADER union FOLLOWERS;            # All the agents
check: (card(LEADER) == 1 && card(LEADER inter FOLLOWERS) == 0);

set COMMODITIES;                                 # Commodities

param e {AGENTS, COMMODITIES} >= 0, default 1; # Endowment

param alpha {AGENTS, COMMODITIES} > 0;           # Utility parameters
param beta {AGENTS, COMMODITIES} > 0;

var x {AGENTS, COMMODITIES};                     # Consumption (no bounds!)
var l {FOLLOWERS};                               # Multipliers (no bounds!)
var p {COMMODITIES};                             # Prices (no bounds!)

var u {i in AGENTS} =                            # Utility
  sum {k in COMMODITIES} alpha[i,k] * (1 + x[i,k])^(1 - beta[i,k]) / (1 - beta[i,k]);
var du {i in AGENTS, k in COMMODITIES} =         # Marginal prices
  alpha[i,k] / (1 + x[i,k])^beta[i,k];
```

Model: cgempec.mod

```
maximize
  objective: sum {i in LEADER} u[i];

subject to
  leader_budget {i in LEADER}:
    sum {k in COMMODITIES} p[k]*(e[i,k] - x[i,k]) >= 0;

  optimality {i in FOLLOWERS, k in COMMODITIES}:
    0 <= x[i,k] complements -du[i,k] + p[k] * l[i] >= 0;

  budget {i in FOLLOWERS}:
    0 <= l[i]   complements sum {k in COMMODITIES} p[k]*(e[i,k] - x[i,k]) >= 0;

  market {k in COMMODITIES}:
    0 <= p[k]   complements sum {i in AGENTS} (e[i,k] - x[i,k]) >= 0;
```

Data: cgempec.dat

```
set LEADER := Jorge;
set FOLLOWERS := Sven, Todd;
set COMMODITIES := Books, Cars, Food, Pens;

param alpha : Books  Cars  Food  Pens :=
     Jorge      1     1     1     1
     Sven       1     2     3     4
     Todd       2     1     1     5;

param beta (tr): Jorge  Sven  Todd :=
     Books        1.5    2     0.6
     Cars         1.6    3     0.7
     Food         1.7    2     2.0
     Pens         1.8    2     2.5;
```

Commands: cgempec.cmd

```
# Load model and data
model cgempec.mod;
data cgempec.dat;

# Specify solver and options
option presolve 0;
option solver "loqo";

# Solve the instance
drop market['Books'];
fix p['Books'] := 1;
solve;

# Output results
printf {i in AGENTS, k in COMMODITIES} "%5s %5s: % 5.4e\n", i, k, x[i,k] > cgempec.out;
printf "\n" > cgempec.out;
printf {k in COMMODITIES} "%5s: % 5.4e\n", k, p[k] > cgempec.out;
```

Output: cgempec.out

```
     Stackleberg                         Nash Game
Jorge Books:  9.2452e-01          Jorge Books:  8.9825e-01
Jorge  Cars:  1.3666e+00          Jorge  Cars:  1.4651e+00
Jorge  Food:  1.1508e+00          Jorge  Food:  1.2021e+00
Jorge  Pens:  7.7259e-01          Jorge  Pens:  6.8392e-01
 Sven Books:  2.5499e-01           Sven Books:  2.5392e-01
 Sven  Cars:  7.4173e-01           Sven  Cars:  7.2054e-01
 Sven  Food:  1.6657e+00           Sven  Food:  1.6271e+00
 Sven  Pens:  1.4265e+00           Sven  Pens:  1.4787e+00
 Todd Books:  1.8205e+00           Todd Books:  1.8478e+00
 Todd  Cars:  8.9169e-01           Todd  Cars:  8.1431e-01
 Todd  Food:  1.8355e-01           Todd  Food:  1.7081e-01
 Todd  Pens:  8.0093e-01           Todd  Pens:  8.3738e-01

Books:  1.0000e+00                Books:  1.0000e+00
 Cars:  5.9617e-01                 Cars:  6.1742e-01
 Food:  6.6496e-01                 Food:  6.8345e-01
 Pens:  1.0700e+00                 Pens:  1.0237e+00
```

Nonlinear Programming Formulation

$$
\begin{aligned}
\min_{x,y,\lambda,s,t\geq 0} \quad & g(x,y) \\
\text{subject to } & h(y) \leq 0 \\
& s_i = \nabla_{x_i} f_i(x,y) + \lambda_i^T \nabla_{x_i} c_i(x_i) \\
& t_i = -c_i(x_i) \\
& \sum_i \left(s_i^T x_i + \lambda_i t_i \right) \leq 0
\end{aligned}
$$

- Constraint qualification fails
  - Lagrange multiplier set unbounded
  - Constraint gradients linearly dependent
  - Central path does not exist
- Able to prove convergence results for some methods
- Reformulation very successful and versatile in practice

Penalization Approach

$$
\begin{aligned}
\min_{x,y,\lambda,s,t\geq 0} \quad & g(x,y) + \pi \sum_i \left(s_i^T x_i + \lambda_i t_i \right) \\
\text{subject to } & h(y) \leq 0 \\
& s_i = \nabla_{x_i} f_i(x,y) + \lambda_i^T \nabla_{x_i} c_i(x_i) \\
& t_i = -c_i(x_i)
\end{aligned}
$$

- Optimization problem satisfies constraint qualification
- Need to increase π

Relaxation Approach

$$
\begin{aligned}
\min_{x,y,\lambda,s,t\geq 0} \quad & g(x,y) \\
\text{subject to} \quad & h(y) \leq 0 \\
& s_i = \nabla_{x_i} f_i(x,y) + \lambda_i^T \nabla_{x_i} c_i(x_i) \\
& t_i = -c_i(x_i) \\
& \sum_i \left(s_i^T x_i + \lambda_i t_i \right) \leq \tau
\end{aligned}
$$

- Need to decrease τ

Limitations

- Multipliers may not exist
- Solvers can have a hard time computing solutions
  - Try different algorithms
  - Compute feasible starting point
- Stationary points may have descent directions
  - Checking for descent is an exponential problem
  - Strong stationary points found in certain cases
- Many stationary points – global optimization

Limitations

- Multipliers may not exist
- Solvers can have a hard time computing solutions
  - Try different algorithms
  - Compute feasible starting point
- Stationary points may have descent directions
  - Checking for descent is an exponential problem
  - Strong stationary points found in certain cases
- Many stationary points – global optimization
- Formulation of follower problem
  - Multiple solutions to Nash game
  - Nonconvex objective or constraints
  - Existence of multipliers

Model Formulation

- Firm $f \in \mathcal{F}$ chooses output x_f to maximize profit
  - u is the utility function

$$u = \left(1 + \sum_{f \in \mathcal{F}} x_f^\alpha\right)^{\frac{\eta}{\alpha}}$$

  - α and η are parameters
  - c_f is the unit cost for each firm
- In particular, for each firm $f \in \mathcal{F}$

$$x_f^* \in \arg\max_{x_f \geq 0} \left(\frac{\partial u}{\partial x_f} - c_f\right) x_f$$

- First-order optimality conditions

$$0 \leq x_f \perp c_f - \frac{\partial u}{\partial x_f} - x_f \frac{\partial^2 u}{\partial x_f^2} \geq 0$$

Model: oligopoly.mod

```
set FIRMS;                                       # Firms in problem

param c {FIRMS};                                 # Unit cost
param alpha > 0;                                 # Constants
param eta > 0;

var x {FIRMS} default 0.1;                       # Output (no bounds!)

var s = 1 + sum {f in FIRMS} x[f]^alpha;         # Summation term
var u = s^(eta/alpha);                           # Utility
var du {f in FIRMS} =                            # Marginal price
  eta * s^(eta/alpha - 1) * x[f]^(alpha - 1);
var dudu {f in FIRMS} =                          # Derivative
  eta * (eta - alpha) * s^(eta/alpha - 2) * x[f]^(2 * alpha - 2) +
  eta * (alpha - 1   ) * s^(eta/alpha - 1) * x[f]^(    alpha - 2);

compl {f in FIRMS}:
  0 <= x[f] complements c[f] - du[f] - x[f] * dudu[f] >= 0;
```

Data: oligopoly.dat

```
param: FIRMS :  c :=
         1     0.07
         2     0.08
         3     0.09;

param alpha := 0.999;
param eta := 0.2;
```

Commands: oligopoly.cmd

```
# Load model and data
model oligopoly.mod;
data oligopoly.dat;

# Specify solver and options
option presolve 0;
option solver "pathampl";

# Solve complementarity problem
solve;

# Output the results
printf {f in FIRMS} "Output for firm %2d: % 5.4e\n", f, x[f] > oligcomp.out;
```

Results: oligopoly.out

```
Output for firm  1:  8.3735e-01
Output for firm  2:  5.0720e-01
Output for firm  3:  1.7921e-01
```

Model Formulation

- Players select strategies to minimize loss
  - $p \in \Re^n$ is the probability player 1 chooses each strategy
  - $q \in \Re^m$ is the probability player 2 chooses each strategy
  - $A \in \Re^{n \times m}$ is the loss matrix for player 1
  - $B \in \Re^{n \times m}$ is the loss matrix for player 2
- Optimization problem for player 1

$$\begin{array}{ll} \min\limits_{0 \le p \le 1} & p^T A q \\ \text{subject to} & e^T p = 1 \end{array}$$

- Optimization problem for player 2

$$\begin{array}{ll} \min\limits_{0 \le q \le 1} & p^T B q \\ \text{subject to} & e^T q = 1 \end{array}$$

Model Formulation

- Players select strategies to minimize loss
  - $p \in \Re^n$ is the probability player 1 chooses each strategy
  - $q \in \Re^m$ is the probability player 2 chooses each strategy
  - $A \in \Re^{n \times m}$ is the loss matrix for player 1
  - $B \in \Re^{n \times m}$ is the loss matrix for player 2
- Complementarity problem

$$
\begin{array}{ll}
0 \leq p \leq 1 & \perp Aq - \lambda_1 \\
0 \leq q \leq 1 & \perp B^T p - \lambda_2 \\
\lambda_1 \text{ free} & \perp e^T p = 1 \\
\lambda_2 \text{ free} & \perp e^T q = 1
\end{array}
$$

Model: bimatrix1.mod

```
param n > 0, integer;          # Strategies for player 1
param m > 0, integer;          # Strategies for player 2

param A{1..n, 1..m};           # Loss matrix for player 1
param B{1..n, 1..m};           # Loss matrix for player 2

var p{1..n};                   # Probability player 1 selects strategy i
var q{1..m};                   # Probability player 2 selects strategy j
var lambda1;                   # Multiplier for constraint
var lambda2;                   # Multiplier for constraint

subject to
  opt1 {i in 1..n}:            # Optimality conditions for player 1
    0 <= p[i] <= 1 complements sum{j in 1..m} A[i,j] * q[j] - lambda1;

  opt2 {j in 1..m}:            # Optimality conditions for player 2
    0 <= q[j] <= 1 complements sum{i in 1..n} B[i,j] * p[i] - lambda2;

  con1:
    lambda1 complements sum{i in 1..n} p[i] = 1;

  con2:
    lambda2 complements sum{j in 1..m} q[j] = 1;
```

Model: bimatrix2.mod

```
param n > 0, integer;           # Strategies for player 1
param m > 0, integer;           # Strategies for player 2

param A{1..n, 1..m};            # Loss matrix for player 1
param B{1..n, 1..m};            # Loss matrix for player 2

var p{1..n};                    # Probability player 1 selects strategy i
var q{1..m};                    # Probability player 2 selects strategy j
var lambda1;                    # Multiplier for constraint
var lambda2;                    # Multiplier for constraint

subject to
  opt1 {i in 1..n}:             # Optimality conditions for player 1
    0 <= p[i] complements sum{j in 1..m} A[i,j] * q[j] - lambda1 >= 0;

  opt2 {j in 1..m}:             # Optimality conditions for player 2
    0 <= q[j] complements sum{i in 1..n} B[i,j] * p[i] - lambda2 >= 0;

  con1:
    0 <= lambda1 complements sum{i in 1..n} p[i] >= 1;

  con2:
    0 <= lambda2 complements sum{j in 1..m} q[j] >= 1;
```

Model: bimatrix3.mod

```
param n > 0, integer;            # Strategies for player 1
param m > 0, integer;            # Strategies for player 2

param A{1..n, 1..m};             # Loss matrix for player 1
param B{1..n, 1..m};             # Loss matrix for player 2

var p{1..n};                     # Probability player 1 selects strategy i
var q{1..m};                     # Probability player 2 selects strategy j

subject to
  opt1 {i in 1..n}:              # Optimality conditions for player 1
    0 <= p[i] complements sum{j in 1..m} A[i,j] * q[j] >= 1;

  opt2 {j in 1..m}:              # Optimality conditions for player 2
    0 <= q[j] complements sum{i in 1..n} B[i,j] * p[i] >= 1;
```

Part IV

Numerical Optimization IV: Extensions

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

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:

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

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

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/P