Numerical Optimization

Todd Munson Stefan Wild
Sven Leyffer

Mathematics and Computer Science Division
Argonne National Laboratory

January 25 to February 1, 2017

Overview (i)

Overview (ii)

Overview (iii)

Part V: Optimization Without Derivatives

Part VI: Stochastic Approximation Methods

Overview (iv)

Part VII: Advanced Topics

Continuous-Time Optimal Control
- Technology Penetration Model
- Discretize then Optimize

Part I

Principles of Numerical Computing

Types of Computing

- Symbolic computing (e.g. mathematica)
  - Represent and manipulate mathematical expressions
  - Apply rules of algebra to simplify expressions
  - Irrational numbers are represented exactly

Types of Computing

- Symbolic computing (e.g. mathematica)
  - Represent and manipulate mathematical expressions
  - Apply rules of algebra to simplify expressions
  - Irrational numbers are represented exactly
- Numerical computing (e.g. matlab)
  - Finite precision arithmetic
  - Apply basic algebra operations
  - Irrational numbers are approximated
  - Numerical analysis of algorithms required

Finite Precision Arithmetic

- Representation of numbers (9.123×10^{-4})
  - Significand and its sign (9123)
  - Exponent and its sign (-4)
  - Special numbers (Inf/NaN)

Finite Precision Arithmetic

- Representation of numbers (9.123×10^{-4})
  - Significand and its sign (9123)
  - Exponent and its sign (-4)
  - Special numbers (Inf/NaN)
  - Finite collection
    - Single precision: 2^{32}, $\epsilon = 5.96 \times 10^{-8}$
    - Double precision: 2^{64}, $\epsilon = 1.11 \times 10^{-16}$

Finite Precision Arithmetic

- Representation of numbers (9.123×10^{-4})
  - Significand and its sign (9123)
  - Exponent and its sign (-4)
  - Special numbers (Inf/NaN)
  - Finite collection
    - Single precision: 2^{32}, $\epsilon = 5.96 \times 10^{-8}$
    - Double precision: 2^{64}, $\epsilon = 1.11 \times 10^{-16}$
- Addition example ($9.123 \times 10^{-4} + 9.123 \times 10^{-5}$)
  - Shift for common exponent ($9.123 \times 10^{-4} + 0.9123 \times 10^{-4}$)
  - Add (10.0353×10^{-4})
  - Round result (10.04×10^{-4})
  - Normalize (1.004×10^{-3})
  - Accumulate roundoff error (-4.700^{-7})
    - Importance of ϵ: $1.0 + \epsilon = 1.0$
    - Adding many numbers: $x^{k+1} = x^k + \epsilon$, $x^0 = 1.0$

Finite Precision Arithmetic

- Representation of numbers (9.123×10^{-4})
  - Significand and its sign (9123)
  - Exponent and its sign (-4)
  - Special numbers (Inf/NaN)
  - Finite collection
    - Single precision: 2^{32}, $\epsilon = 5.96 \times 10^{-8}$
    - Double precision: 2^{64}, $\epsilon = 1.11 \times 10^{-16}$
- Addition example ($9.123 \times 10^{-4} + 9.123 \times 10^{-5}$)
  - Shift for common exponent ($9.123 \times 10^{-4} + 0.9123 \times 10^{-4}$)
  - Add (10.0353×10^{-4})
  - Round result (10.04×10^{-4})
  - Normalize (1.004×10^{-3})
  - Accumulate roundoff error (-4.700^{-7})
    - Importance of ϵ: $1.0 + \epsilon = 1.0$
    - Adding many numbers: $x^{k+1} = x^k + \epsilon$, $x^0 = 1.0$
- Subtraction results in loss of precision
- Multiplication and division similar

Dealing with Roundoff Errors

- Structure computation to minimize roundoff error
  - Adding positive numbers from least magnitude to largest
  - Requires keeping sorted list of numbers
  - Minimum is very difficult in general
  - Reduce the error when possible
- Perform computations tolerant to roundoff error
  - Analyze algorithms
  - Prove statements about the error
  - Field of numerical analysis
- Numerical linear algebra widely studied

Notation

- Scalar: $a \in \Re$
- Vector: $b \in \Re^n$, b_i is the ith element
- Matrix: $A \in \Re^{m \times n}$
  - $A_{i,\cdot}$ is the ith row
  - $A_{\cdot,j}$ is the jth column
  - $A_{i,j}$ is the element in row i and column j

Notation

- Scalar: $a \in \Re$
- Vector: $b \in \Re^n$, b_i is the ith element
- Matrix: $A \in \Re^{m \times n}$
  - $A_{i,\cdot}$ is the ith row
  - $A_{\cdot,j}$ is the jth column
  - $A_{i,j}$ is the element in row i and column j
- Types of matrices
  - Square: $A \in \Re^{n \times n}$
  - Symmetric: $A = A^T$
  - Skew symmetric: $A = -A^T$

Notation

- Scalar: $a \in \Re$
- Vector: $b \in \Re^n$, b_i is the ith element
- Matrix: $A \in \Re^{m \times n}$
  - $A_{i,\cdot}$ is the ith row
  - $A_{\cdot,j}$ is the jth column
  - $A_{i,j}$ is the element in row i and column j
- Types of matrices
  - Square: $A \in \Re^{n \times n}$
  - Symmetric: $A = A^T$
  - Skew symmetric: $A = -A^T$
- Storage of matrices
  - Dense matrices store mn entries
  - Sparse matrices store
    - Nonzero structure: (i, j) such that $A_{i,j} \neq 0$
    - Nonzero entries: $A_{i,j} \neq 0$
  - Sparse linear algebra used for performance

Linear Algebra

- Matrix-vector products

$$y = Ax + b$$

- Matrix-matrix products

$$C = AB$$

- Linear systems of equations

$$Ax = b$$

- Condition number estimation

Direct Methods for Medium Problems

$$Ax = b$$

- Use backslash $x = A\backslash b$ rather than inverse $x = inv(A)b$
- LL' Cholesky factorization (sparse, symmetric, positive definite)
- LDL' factorization (sparse, symmetric, indefinite)
- LU factorization (sparse, generic, full rank)
- QR factorization (dense, generic)
- USV' singular value decomposition (dense, generic)

Iterative Methods for Large Problems

$$Ax = b$$

- Conjugate gradient method (sparse, symmetric, positive definite)
- Generalized minimum residual (sparse, generic)
- Quasi-minimum residual (sparse, generic)
- Transpose-free quasi-minimum residual (sparse, generic)

Functions

- Objective function

$$f : \Re^n \to \Re$$

  - Linear: $f(x) = a^T x$ for vector $a \in \Re^n$
  - Quadratic: $f(x) = \frac{1}{2} x^T Q x + a^T x$ for symmetric matrix $Q \in \Re^{n \times n}$
  - Nonlinear:
    - Continuous versus discontinuous
    - Smooth versus nonsmooth

Functions

- Objective function

$$f : \Re^n \to \Re$$

  - Linear: $f(x) = a^T x$ for vector $a \in \Re^n$
  - Quadratic: $f(x) = \frac{1}{2} x^T Q x + a^T x$ for symmetric matrix $Q \in \Re^{n \times n}$
  - Nonlinear:
    - Continuous versus discontinuous
    - Smooth versus nonsmooth
- Constraints

$$c : \Re^n \to \Re^m$$

  - Linear: $c(x) = Ax + b$ for matrix $A \in \Re^{m \times n}$ and vector $b \in \Re^m$
  - Quadratic: $c_i(x) = \frac{1}{2} x^T Q_i x + a_i^T x + b_i$ for symmetric $Q_i \in \Re^{n \times n}$
  - Nonlinear
    - Continuous versus discontinuous
    - Smooth versus nonsmooth

Functions

- Objective function

$$f : \Re^n \rightarrow \Re$$

  - Linear: $f(x) = a^T x$ for vector $a \in \Re^n$
  - Quadratic: $f(x) = \frac{1}{2}x^T Qx + a^T x$ for symmetric matrix $Q \in \Re^{n \times n}$
  - Nonlinear:
    - Continuous versus discontinuous
    - Smooth versus nonsmooth
- Constraints

$$c : \Re^n \rightarrow \Re^m$$

  - Linear: $c(x) = Ax + b$ for matrix $A \in \Re^{m \times n}$ and vector $b \in \Re^m$
  - Quadratic: $c_i(x) = \frac{1}{2}x^T Q_i x + a_i^T x + b_i$ for symmetric $Q_i \in \Re^{n \times n}$
  - Nonlinear
    - Continuous versus discontinuous
    - Smooth versus nonsmooth
- Code functions using intrinsic operations
  - Basic algebra operations
  - Transcendental functions
  - Reduce roundoff errors

Derivatives

- Gradient of f (vector in $\Re^n$)

$$[\nabla f(x)]_j = \frac{\partial f}{\partial x_j}$$

- Hessian of f (symmetric matrix in $\Re^{n\times n}$)

$$[\nabla^2 f(x)]_{j,k} = \frac{\partial^2 f}{\partial x_j \partial x_k}$$

- Jacobian of c (matrix in $\Re^{m\times n}$)

$$[\nabla c(x)]_{i,j} = \frac{\partial c_i}{\partial x_j}$$

- Second derivatives of c (tensor in $\Re^{m\times n\times n}$, $\nabla^2 c_i(x)$ symmetric)

$$[\nabla c(x)]_{i,j,k} = \frac{\partial^2 c_i}{\partial x_j \partial x_k}$$

Computation of Derivatives

- Numerical derivatives via finite differences

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}$$

  where e_i is the ith column of the identity matrix
  - Can requires $n + 1$ function evaluations
  - Quality depends on function and stepsize
  - Subtraction results in loss of precision
  - Ask Stefan Wild about optimal stepsize

Computation of Derivatives

- Numerical derivatives via finite differences

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}$$

  where e_i is the ith column of the identity matrix
  - Can requires $n + 1$ function evaluations
  - Quality depends on function and stepsize
  - Subtraction results in loss of precision
  - Ask Stefan Wild about optimal stepsize
- Analyze the computational graph of the function
  - Use calculus rules for the intrinsic functions
  - Apply the chain rule to assemble partials
    - Forward mode – easy to implement, but can be slow
    - Reverse mode – hard to implement, but is fast
  - Precision approximately equal to function precision
  - Provable bounds on cost that does not grow in n

Computation of Derivatives

- Numerical derivatives via finite differences

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}$$

  where e_i is the ith column of the identity matrix
  - Can requires $n + 1$ function evaluations
  - Quality depends on function and stepsize
  - Subtraction results in loss of precision
  - Ask Stefan Wild about optimal stepsize
- Analyze the computational graph of the function
  - Use calculus rules for the intrinsic functions
  - Apply the chain rule to assemble partials
    - Forward mode – easy to implement, but can be slow
    - Reverse mode – hard to implement, but is fast
  - Precision approximately equal to function precision
  - Provable bounds on cost that does not grow in n
- Check analytic derivatives against numerical derivatives

Computation of Derivatives

- Numerical derivatives via finite differences

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}$$

  where e_i is the ith column of the identity matrix
  - Can requires $n + 1$ function evaluations
  - Quality depends on function and stepsize
  - Subtraction results in loss of precision
  - Ask Stefan Wild about optimal stepsize
- Analyze the computational graph of the function
  - Use calculus rules for the intrinsic functions
  - Apply the chain rule to assemble partials
    - Forward mode – easy to implement, but can be slow
    - Reverse mode – hard to implement, but is fast
  - Precision approximately equal to function precision
  - Provable bounds on cost that does not grow in n
- Check analytic derivatives against numerical derivatives
- *Assume derivatives available for now*

Lack of Derivatives

- Functions are nondifferentiable

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand
  - Modeling language provides derivatives
  - Assisted differentiation methods are available

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand
  - Modeling language provides derivatives
  - Assisted differentiation methods are available
- Too much memory consumed

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand
  - Modeling language provides derivatives
  - Assisted differentiation methods are available
- Too much memory consumed
  - Exploit sparse matrices
  - Matrix-free methods use matrix-vector products

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand
  - Modeling language provides derivatives
  - Assisted differentiation methods are available
- Too much memory consumed
  - Exploit sparse matrices
  - Matrix-free methods use matrix-vector products
- Too much time to compute them

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand
  - Modeling language provides derivatives
  - Assisted differentiation methods are available
- Too much memory consumed
  - Exploit sparse matrices
  - Matrix-free methods use matrix-vector products
- Too much time to compute them
  - Smart finite differences
  - Use reverse mode of differentiation

Lack of Derivatives

- Functions are nondifferentiable
  - Transformations from nonsmooth to smooth
  - Approximation methods for nonsmooth functions
- Too hard to construct by hand
  - Modeling language provides derivatives
  - Assisted differentiation methods are available
- Too much memory consumed
  - Exploit sparse matrices
  - Matrix-free methods use matrix-vector products
- Too much time to compute them
  - Smart finite differences
  - Use reverse mode of differentiation
- Derivative-free methods are a gateway or last resort

Solving Systems of Equations

Locally Convergent Newton Method

- Most good algorithms are variants of Newton's method
- Compute solutions $F(x) = 0$ where $F : \Re^n \rightarrow \Re^n$
  - Form Taylor series approximation around x^k

$$F(x) \approx \nabla F(x^k)(x - x^k) + F(x^k)$$

  - Solve for x and iterate

$$x^{k+1} = x^k - \nabla F(x^k) \backslash F(x^k)$$

Solving Systems of Equations

Locally Convergent Newton Method

- Most good algorithms are variants of Newton's method
- Compute solutions $F(x) = 0$ where $F : \Re^n \to \Re^n$
  - Form Taylor series approximation around x^k

$$F(x) \approx \nabla F(x^k)(x - x^k) + F(x^k)$$

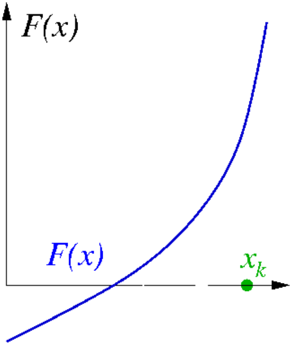
  - Solve for x and iterate

$$x^{k+1} = x^k - \nabla F(x^k) \backslash F(x^k)$$

- Several possible outcomes
  - Convergence: $\lim_{k \to \infty} x^k = x^*$
  - Iterate divergence: $\lim_{k \to \infty} \|x^k\| \to \infty$
  - Sequence cycles:
    - Multiple convergent subsequences (limit points)
    - Limit points are not solutions

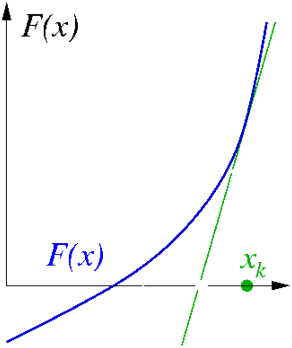
Solving Systems of Equations

Illustration of Newton's Method



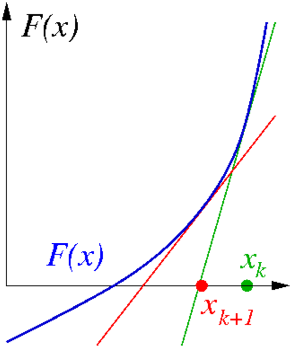
Solving Systems of Equations

Illustration of Newton's Method



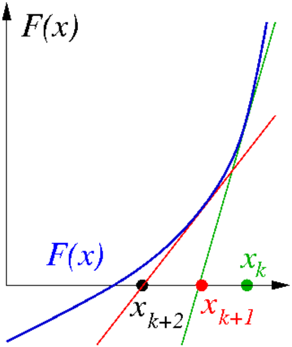
Solving Systems of Equations

Illustration of Newton's Method



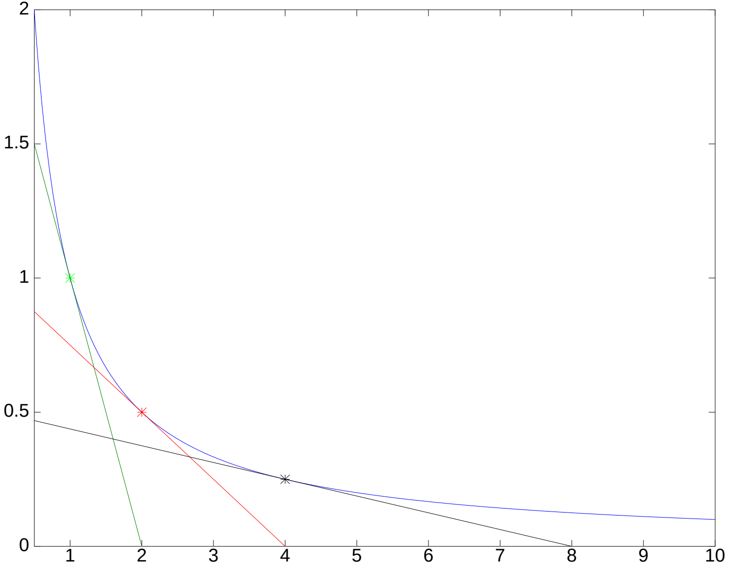
Solving Systems of Equations

Illustration of Newton's Method



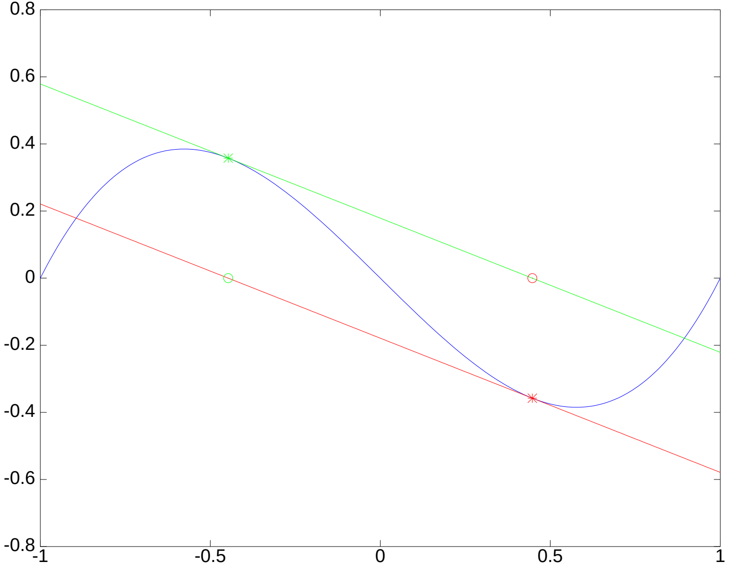
Solving Systems of Equations

Illustration of Iterate Divergence



Solving Systems of Equations

Illustration of Cycling



Solving Systems of Equations

Globally Convergent Newton Method

- Use Newton method to compute a step

$$s^k = -\nabla F(x^k) \backslash F(x^k)$$

- Check direction for descent using merit function $\Psi(x) = \|F(x)\|_2^2$

$$\nabla \Psi(x^k)' s^k < 0$$

- Minimize merit function along step

$$t_k \in \arg \min_{t \in (0,1]} \Psi(x^k + t s^k)$$

- Update iterate

$$x^{k+1} = x^k + t_k s^k$$

  and repeat until convergence
- Several possible outcomes
  - Good convergence: $\lim_{k \to \infty} x^k \to x^*$ with $F(x^*) = 0$
  - Bad convergence: $\lim_{k \to \infty} x^k \to x^*$ with $\Psi(x^*) > 0$ and $\nabla \Psi(x^*) = 0$
  - Iterate divergence: $\lim_{k \to \infty} \|x^k\| \to \infty$

Solving Systems of Equations

Implementation Details

- Numerical linear algebra
  - Direct methods compute a (sparse) factorization

    $$\nabla F(x^k) = LU$$

    where L is lower triangular and U is upper triangular
  - Iterative methods compute an approximation
    - Krylov subspace method such as generalize minimum residual
    - Large part of the computation is matrix-vector products
    - Accelerate convergence using a preconditioner
  - Recourse when $\nabla F(x^k)$ is not invertible

Solving Systems of Equations

Implementation Details

- Numerical linear algebra
  - Direct methods compute a (sparse) factorization

    $$\nabla F(x^k) = LU$$

    where L is lower triangular and U is upper triangular
  - Iterative methods compute an approximation
    - Krylov subspace method such as generalize minimum residual
    - Large part of the computation is matrix-vector products
    - Accelerate convergence using a preconditioner
  - Recourse when $\nabla F(x^k)$ is not invertible
- Minimizing merit function along the step
  - Golden section and Fibonacci search
  - Armijo backtracking line search: find smallest integer i with

    $$\Psi(x^k + \beta^i s^k) \leq \Psi(x^k) + \sigma \beta^i \nabla \Psi(x^k)' s^k$$
  - Moré-Thuente: cubic interpolation and refinement

Solving Systems of Equations

Convergence Results

- Global convergence under suitable conditions
- Local fast convergence under suitable conditions
  - If x^k is near a solution, method converges to a solution x^*
  - The distance to the solution decreases quickly; ideally,

$$\|x^{k+1} - x^*\| \leq c\|x^k - x^*\|^2$$

Solving Systems of Equations

Newton Method with Proximal Perturbation

- Recourse when $\nabla F(x^k)$ is not invertible or bad direction
- Solve linear system of equations

$$(\nabla F(x^k) + \lambda_k I)s^k = -F(x^k)$$

- Check direction for descent using merit function $\Psi(x) = \|F(x)\|_2^2$

$$\nabla \Psi(x^k)' s^k < -p_1 \|s^k\|^{p_2}$$

  otherwise use steepest descent direction $s^k = -\nabla \Psi(x^k)$
- Minimize merit function along step

$$t_k \in \arg \min_{t \in (0,1]} \Psi(x^k + t s^k)$$

- Update perturbation
- Update iterate

$$x^{k+1} = x^k + t_k s^k$$

  and repeat until convergence

Nonsquare Nonlinear Systems of Equations

- Given $F : \Re^n \to \Re^m$, compute x such that
$$F(x) = 0$$
- System is underdetermined if $m < n$
  - More variables than constraints
  - Solution typically not unique
  - Need to select one solution
$$\min_x \; \|x\|_2 \text{ subject to } F(x) = 0$$
- System is overdetermined if $m > n$
  - More constraints than variables
  - Solution typically does not exist
  - Need to select approximate solution
$$\min_x \; \|F(x)\|_2$$
- System is square if $m = n$
  - Jacobian has full rank then solution is unique
  - If Jacobian is rank deficient then
    - Underdetermined when compatible
    - Overdetermined when incompatible

Part II

Foundations of Optimization

Foundations of Optimization

- Generic nonlinear optimization problem

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

  - f represents the objective function
  - c represents the constraints
- Least squares problems
- Further classification
  - Types of functions
  - Types of constraints
- Model transformations

Ordinary Least Squares

- Minimize two norm: $\|y\|_2^2 = \sum_i y_i^2$

$$\min_x \ \|Ax - b\|_2^2$$

  - Data representation: $A \in \Re^{m \times n}$
  - Vector of observations: $b \in \Re^n$
    - m is the number of measurements
    - n is the number of variables
    - Both can be very large

Ordinary Least Squares

Basic Algorithm

- Assuming A has full column rank

  $$x = (A^T A) \backslash (A^T b)$$

  where $A^T A \in \Re^{n \times n}$ is symmetric
- Standard errors use $\mathrm{diag}((A^T A)^{-1})$
- Two cases
  - Number of variables is small
  - Number of variables is large

Ordinary Least Squares

Small Number of Variables

- Form the matrix $B = A^T A$

$$B_{i,j} = \sum_k A_{k,i} A_{k,j}$$

  - Requires $O(mn^2)$ operations
  - Control roundoff errors for large m
  - Dense linear algebra
- Solve $x = B \backslash (A^T b)$
  - Requires $O(n^3 + mn)$ operations
  - Dense linear algebra
- Compute standard errors using $\mathrm{diag}(B^{-1})$
  - Find diagonal entries of $B \backslash I$
  - Compute $inv(B)$
  - Dense linear algebra

Generalized Least Squares

Basic Algorithm

- Minimize matrix norm: $\|y\|_M^2 = y^T M y$, M positive definite

$$\min_x \|Ax - b\|_M^2$$

  - Data representation: $A \in \Re^{m \times n}$
  - Matrix representation: $M \in \Re^{m \times m}$
  - Vector of observations: $b \in \Re^n$
    - m is the number of measurements
    - n is the number of variables
- Assuming A has full column rank

$$x = (A^T M A)^{-1} A^T M b$$

  where $A^T M A \in \Re^{n \times n}$ is symmetric
- Typical weight is Σ^{-1}
  - Large dense matrix
  - Must have a special representation
- Forming $A^T \Sigma^{-1} A$ is a bad idea
- Utilize matrix-free iterative methods

Convex Functions

- $f : \Re^n \to \Re$ is convex if
$$\forall\, x, y \;:\; f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$
for all $t \in [0,1]$
- Differentiable $f : \Re^n \to \Re$ is convex if
$$\forall\, x, y \;:\; f(y) \geq f(x) + \nabla f(x)^T (y - x)$$
- Twice differentiable $f : \Re^n \to \Re$ is convex if
$$\forall x, y \;:\; y^T \nabla^2 f(x) y \geq 0$$
- A matrix $Q \in \Re^{n \times n}$ is positive semidefinite if
$$\forall y \;:\; y^T Q y \geq 0$$
  - Many equivalent characterizations
  - For symmetric Q, eigenvalues are nonnegative
- A set $X \subseteq \Re^n$ is convex if
$$\forall\, x \in X,\; y \in X \;:\; tx + (1-t)y \in X$$
for all $t \in [0,1]$

Illustration of Convexity

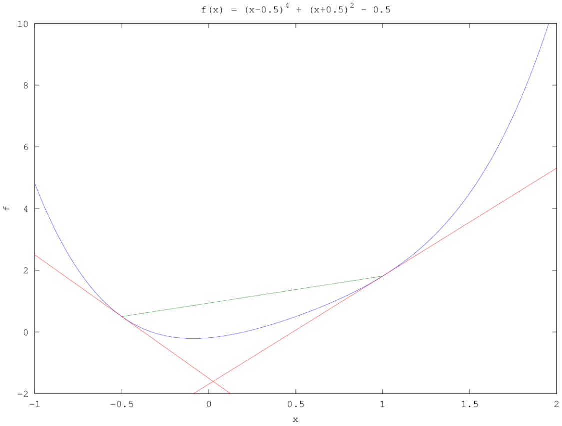


Illustration of Nonconvexity

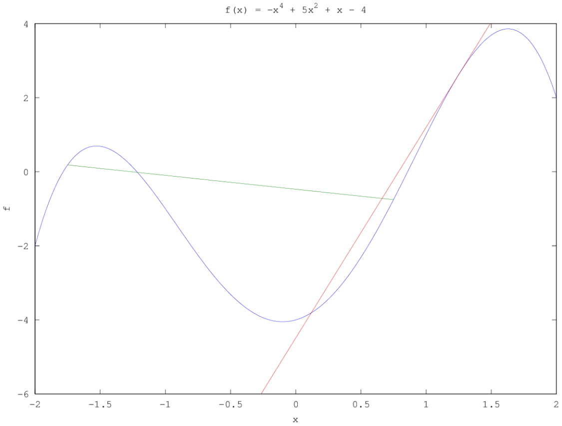
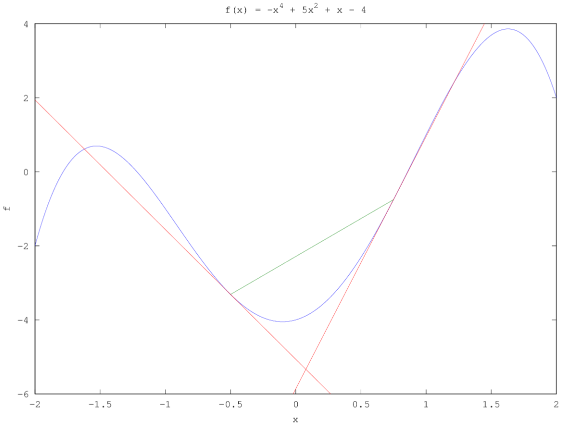


Illustration of Local Convexity



Convexity

Proving and Disproving

- Generally a hard problem

Convexity

Proving and Disproving

- Generally a hard problem
- Proving convexity analyzes functional form
  - Representation as an abstract syntax tree
  - Apply convexity rules
  - E.g. If $g(x)$ and $h(x)$ are convex, then $g(x) + h(x)$ is convex
  - Analyzer can be found for AMPL models
  - Cannot disprove convexity

Convexity

Proving and Disproving

- Generally a hard problem
- Proving convexity analyzes functional form
  - Representation as an abstract syntax tree
  - Apply convexity rules
  - E.g. If $g(x)$ and $h(x)$ are convex, then $g(x) + h(x)$ is convex
  - Analyzer can be found for AMPL models
  - Cannot disprove convexity
- Disproving convexity uses the definitions
  - Choose finite points x^k and y^k
  - Choose finite steps $\{t_\ell\} \subset (0, 1)$
  - Not convex if there exists k and ℓ such that

$$f(t_\ell x^k + (1 - t_\ell)y^k) > t_\ell f(x^k) + (1 - t_\ell)f(y^k)$$

  - Cannot prove convexity

Taxonomy

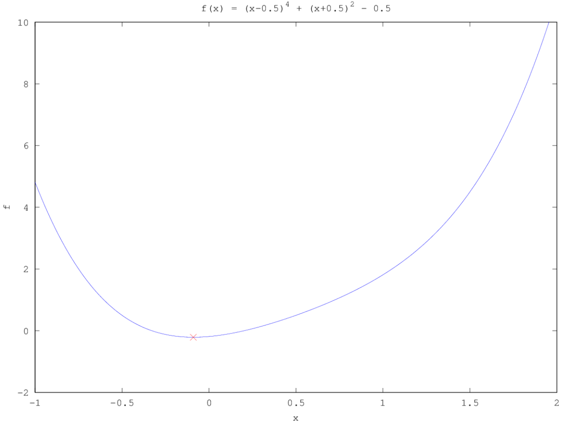
Unconstrained Optimization

$$\min_x \; f(x)$$

- Linear: $f(x) = c^T x$ (unbounded solution)
- Quadratic: $f(x) = \frac{1}{2} x^T Q x + c^T x$
  - Convex if Q is positive semidefinite
  - Concave if $-Q$ is positive semidefinite (unbounded solution)
  - Otherwise it is neither convex nor concave
  - Conjugate gradient iterative methods
- Nonlinear function
  - Convex under suitable conditions
  - Concave if $-f(x)$ is convex (unbounded solution)
  - Otherwise it is neither convex nor concave
- Implications of convexity
  - Convex implies local solution is a global solution
  - Strictly convex implies at most one solution
  - Strongly convex implies one solution
  - Convex implies convex solution set
- Provable global solutions for nonconvex problems is difficult
- Algorithms discussed in next session

Convex Unconstrained Optimization

Critical Points



- Global minimizer: $\nabla f(x) = 4(x-0.5)^3 + 2(x+0.5) = 0$

Taxonomy

Nonconvex Unconstrained Optimization

$$\min_x \; f(x)$$

- Global solution x^* if

$$f(x^*) \le f(x) \quad \forall\, x$$

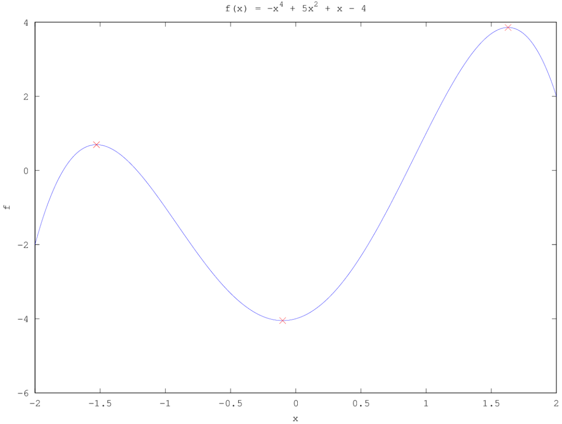
  - Global solutions need not be unique
  - All global solutions have the same objective value
- Local solution x^* if

$$f(x^*) \le f(x) \quad \forall\, x \in \mathcal{N}(x^*)$$

  where $\mathcal{N}(x^*)$ is a neighborhood of x^*
  - Local solutions need not be unique
  - Local solutions need not have the same objective value

Nonconvex Unconstrained Optimization

Critical Points



- Stationary: $\nabla f(x) = -4x^3 + 10x + 1 = 0$
- Local maximizer: $\nabla^2 f(x) = -12x^2 + 10 < 0$
- Local minimizer: $\nabla^2 f(x) = -12x^2 + 10 > 0$

Taxonomy

Bound Constrained Optimization

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{subject to} & \ell \leq x \leq u \end{array}$$

- Linear: $f(x) = c^T x$
- Quadratic: $f(x) = \frac{1}{2} x^T Q x + c^T x$
  - Convex if Q is positive semidefinite (special methods)
  - Concave if $-Q$ is positive semidefinite
  - Otherwise it is neither convex nor concave
  - Conjugate gradient iterative methods
- Nonlinear function
  - Convex under suitable conditions
  - Concave if $-f(x)$ is convex
  - Otherwise it is neither convex nor concave
- Implications of convexity
  - Convex implies local solution is a global solution
  - Strictly convex implies at most one solution
  - Strongly convex implies one solution
  - Convex implies convex solution set
- Provable global solutions for nonconvex problems is difficult

Taxonomy

Linear Program

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Ax + b \le 0 \\ & x \ge 0 \end{aligned}$$

- Convex optimization problem
- Specialized solution methods
  - Simplex method and variants (restarts well on related problems)
  - Interior-point methods (may not restart well)
  - Available software
    - CLP – open source from COIN-OR
    - CPLEX and GUROBI – commercial

Taxonomy

Linear Program

- Primal linear program

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Ax + b \leq 0 \\ & x \geq 0 \end{aligned}$$

- Dual linear program

$$\begin{aligned} \max_\mu \quad & b^T \mu \\ \text{subject to} \quad & A^T \mu + c \geq 0 \\ & \mu \geq 0 \end{aligned}$$

- Strong duality theory applies
  - Both feasible and optimal solutions are equal
  - One is infeasible and the other is unbounded
  - Both are infeasible
- Representation depends variables and constraints
  - Primal has n variables and m constraints
  - Dual has m variables and n constraints

Taxonomy

Quadratic Program

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{2}x^T Q x + c^T x \\ \text{subject to} & Ax + b \leq 0 \\ & x \geq 0 \end{array}$$

- Q is positive semidefinite implies
  - Convex optimization problem
  - Local solutions are global solutions
  - Specialized solution methods
    - Active-set methods (restarts well for related problems)
    - Interior-point methods (may not restart well)
  - Available software
    - OOQP – source available
    - CPLEX and GUROBI – commercial
- Otherwise problem is nonconvex
  - Attempt to compute local solution
  - May only find a stationary point
  - Provable global solutions for nonconvex problems is difficult

Taxonomy

Nonlinear Program

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

- Convex functions f and c_i
  - Convex optimization problem
  - Feasible set $X = \{x \mid c(x) \leq 0\}$ is convex
  - Local solutions are global solutions
- Solutions methods
  - Attempt to compute local solution
  - Approximation methods
  - Interior-point methods
  - Provable global solutions for nonconvex problems is difficult
  - Algorithms discussed in next session

Taxonomy

Nonlinear Program

$$\min_{x \in X} \; f(x)$$

with $X = \{x \mid c(x) \leq 0\}$

- Global solution x^*:

  $$f(x^*) \leq f(x) \; \forall \, x \in X$$

  - Global solutions need not be unique
  - All global solutions have the same objective value
- Local solution x^*:

  $$f(x^*) \leq f(x) \; \forall \, x \in \mathcal{N}(x^*) \cap X$$

  - Local solutions need not be unique
  - Local solutions need not have the same objective value

Taxonomy

Other Types of Nonlinear Programs (Maybe Discussed)

- Dynamic programming – see other lectures
- Optimal control problem

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

- Bilevel optimization problem

$$
\begin{aligned}
\min_{x,y} \quad & f(x,y) \\
\text{subject to} \quad & y \in \arg \left\{ \begin{aligned} \min_{\bar{y}} \quad & g(x,\bar{y}) \\ \text{subject to} \quad & c(x,\bar{y}) \le 0 \end{aligned} \right.
\end{aligned}
$$

- Multi-objective optimization (Pareto surfaces)

Taxonomy

Other Types of Nonlinear Programs (Not Discussed)

- Semidefinite programs
- Second-order cone constraints
- Two-stage stochastic programs

$$\begin{array}{lll} \underset{x,y}{\min} & f(x) + \sum_{s \in S} p_s g_s(x, y_s) & \\ \text{subject to} & c_s(x, y_s) \leq 0 & \forall\, s \in S \end{array}$$

- Semi-infinite optimization problem

$$\begin{array}{lll} \underset{x}{\min} & f(x) & \\ \text{subject to} & c(x, u) \leq 0 & \forall\, u \in \mathcal{U} \end{array}$$

- Discrete optimization and categorical variables

Modeling

Transformations of Nonsmooth Problems

- One norm: $\|x\|_1 = \sum_i |x_i|$

$$\min_x \; f(x) + \|x\|_1$$

- Smooth reformulation

$$\begin{array}{ll} \displaystyle\min_{x,y} & f(x) + e^T y \\ \text{subject to} & -y \le x \le y \end{array}$$

  where $y \in \Re^n$ is a vector
- Add n variables and $2n$ constraints
- Minimization is important
- Not applicable to

$$\max_x \; f(x) + \|x\|_1$$

- Infinity norm: $\|x\|_\infty = \max_i |x_i|$

$$\min_x \; f(x) + \|x\|_\infty$$

- Smooth reformulation

$$\begin{array}{ll} \min\limits_{x,y} & f(x) + y \\ \text{subject to} & -y \le x \le y \end{array}$$

  where $y \in \Re$ is a scalar
- Add one variable and $2n$ constraints
- Minimization is important
- Not applicable to

$$\max_x \; f(x) + \|x\|_1$$

Modeling

Transformations of Nonsmooth Problems

- Pointwise maximum

$$\min_x \; f(x) + \max_i \{c_i(x)\}$$

- Smooth reformulation

$$\begin{array}{ll} \displaystyle\min_{x,y} & f(x) + y \\ \text{subject to} & c_i(x) \leq y \quad \forall\, i \end{array}$$

  where $y \in \Re$ is a scalar
- Add one variable and several constraints
- Minimization is important
- Not applicable to

$$\max_x \; f(x) + \max_i \{c_i(x)\}$$

Modeling

Transformations of Constraints

- One norm constraint: $\|x\|_1 = \sum_i |x_i|$

$$\begin{array}{ll} \displaystyle\min_x & f(x) \\ \text{subject to} & \|x\|_1 \le \Delta \end{array}$$

- Smooth reformulation

$$\begin{array}{ll} \displaystyle\min_{x,y} & f(x) \\ \text{subject to} & -y \le x \le y \\ & e^T y \le \Delta \end{array}$$

where $y \in \Re^n$ is a vector

- Add n variables and $2n$ constraints
- Not applicable to

$$\begin{array}{ll} \displaystyle\min_x & f(x) \\ \text{subject to} & \|x\|_1 \ge \Delta \end{array}$$

Modeling

Transformations of Constraints

- Infinity norm constraint: $\|x\|_\infty = \max_i |x_i|$

$$\begin{array}{ll} \min\limits_x & f(x) \\ \text{subject to} & \|x\|_\infty \le \Delta \end{array}$$

- Smooth reformulation

$$\begin{array}{ll} \min\limits_x & f(x) \\ \text{subject to} & -\Delta \le x \le \Delta \end{array}$$

- Not applicable to

$$\begin{array}{ll} \min\limits_x & f(x) \\ \text{subject to} & \|x\|_\infty \ge \Delta \end{array}$$

Modeling

Smooth Approximations of Nonsmooth Functions

- Some examples
  - $|x| \approx \sqrt{x^2 + \epsilon^2} - \epsilon$
  - $\sqrt{x^2 + y^2} \approx \sqrt{x^2 + y^2 + \epsilon^2} - \epsilon$
  - $(tx^p + (1-t)y^p)^{\frac{1}{p}} \approx (t(x+\epsilon)^p + (1-t)(y+\epsilon)^p)^{\frac{1}{p}} - \epsilon$
- Derivatives can become bad as $\epsilon \to 0$

Bad Transformations

Do Not Make Them!

- Square transformation

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

  is equivalent to

$$\min_{y} \quad f(y^2)$$

- Exponential transformation

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

  is equivalent to

$$\min_{y} \quad f(a^y)$$

  for $a > 0$

Student Discussion

Part III

Continuous Optimization Algorithms

Overview of Continuous Optimization

- One-dimensional unconstrained optimization
  - Characterization of critical points
  - Basic algorithms
- Multi-dimensional unconstrained optimization
  - Critical points and their types
  - Computation of local minimizers
- Multi-dimensional constrained optimization
  - Critical points and Lagrange multipliers
  - Second-order sufficiency conditions
  - Globally-convergent algorithms
- Complementarity constraints
  - Stationarity concepts
  - Constraint qualifications
  - Numerical methods

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

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

where

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

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

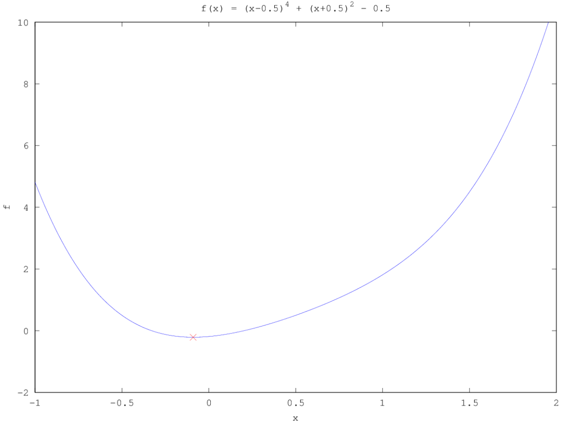
$$\min_{\beta} \quad - \left(\sum_{i=1}^{m} c_i \log(f(\beta, d_{i,\cdot})) + (1 - c_i) \log(1 - f(\beta, d_{i,\cdot})) \right)$$

where

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

Convex Unconstrained Optimization

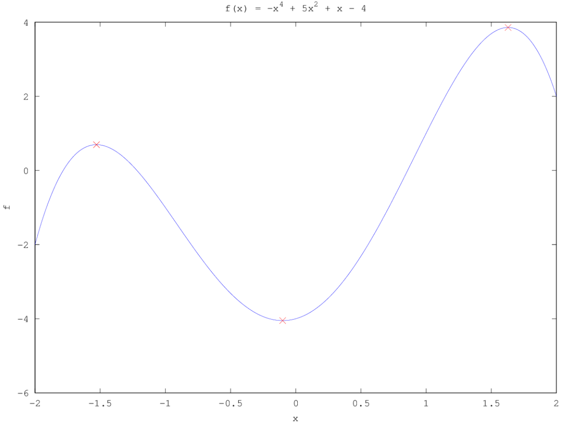
Critical Points



- Global minimizer: $\nabla f(x) = 4(x-0.5)^3 + 2(x+0.5) = 0$

Nonconvex Unconstrained Optimization

Critical Points



- Stationary: $\nabla f(x) = -4x^3 + 10x + 1 = 0$
- Local maximizer: $\nabla^2 f(x) = -12x^2 + 10 < 0$
- Local minimizer: $\nabla^2 f(x) = -12x^2 + 10 > 0$

Locally Convergent Newton Method for Optimization

- Most good algorithms are variants of Newton's method
- Attempt to compute local minimizers for $f : \Re^n \to \Re$
- Settle for stationary points $\nabla f(x) = 0$
  - Form Taylor series approximation around x^k

$$f(x) \approx f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k)$$

  - Solve quadratic optimization problem for s^k

$$\min_s \; f(x^k) + \nabla f(x^k)^T s + \frac{1}{2} s^T \nabla^2 f(x^k) s$$

    - Convex case solutions satisfy

$$\nabla^2 f(x^k) s^k = -\nabla f(x^k)$$

    - Nonconvex case can be unbounded
  - Update iterate

$$x^{k+1} = x^k + s^k$$

  and repeat until convergence

Locally Convergent Newton Method for Optimization

Some Possible Outcomes

- Convergence: $\lim_{k\to\infty} x^k = x^*$
- Divergence: $\lim_{k\to\infty} \|x^k\| \to \infty$
- Unbounded below: $\lim_{k\to\infty} f(x^k) \to -\infty$
- Sequence cycles
  - Multiple convergent subsequences (limit points)
  - Limit points are not solutions

Illustration of Convex Problem

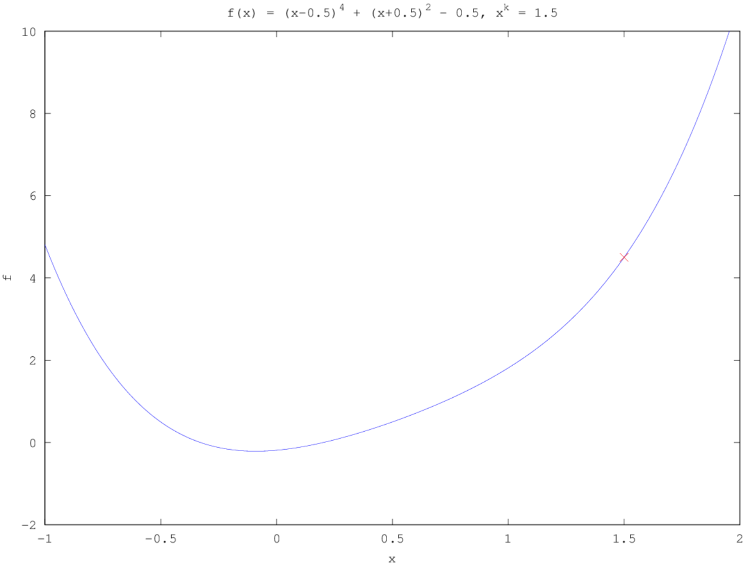


Illustration of Convex Problem

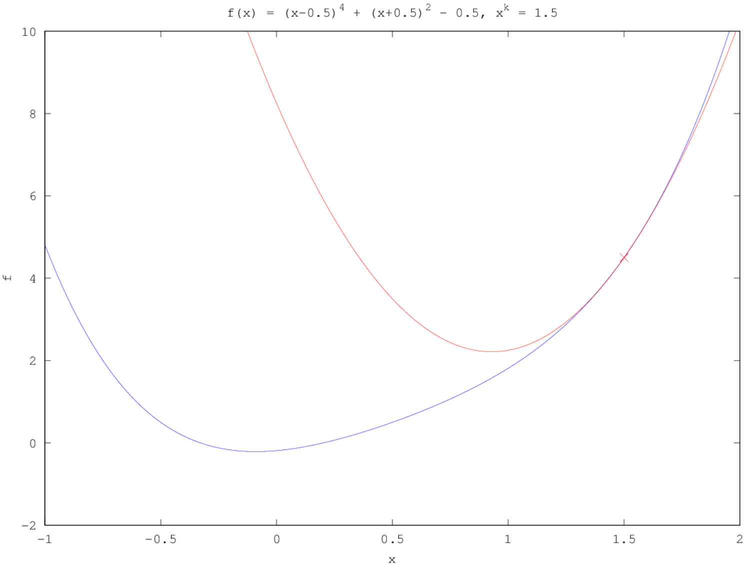


Illustration of Convex Problem

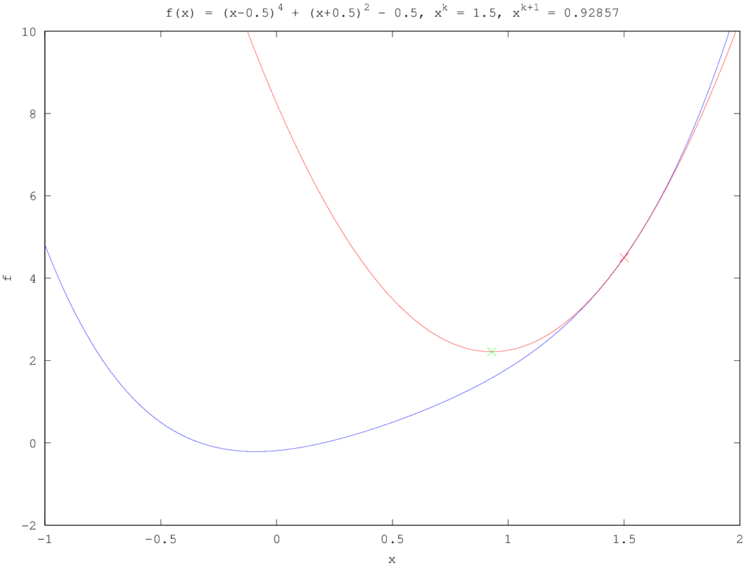


Illustration of Convex Problem

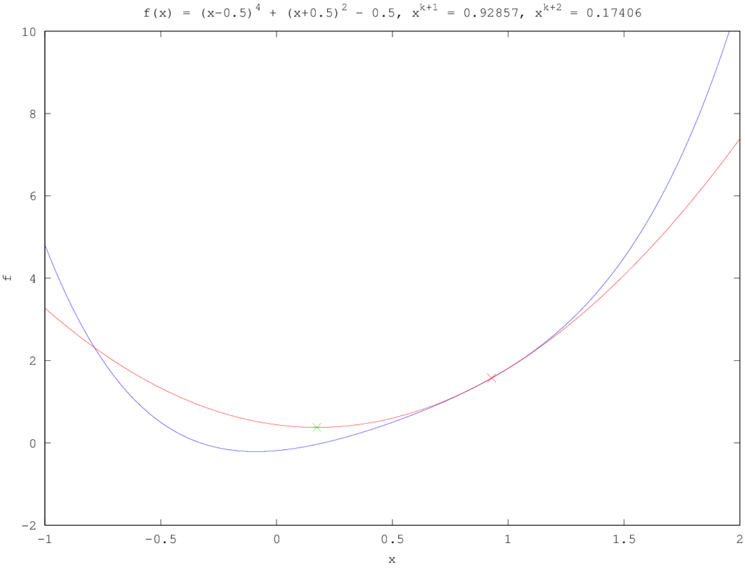


Illustration of Convex Problem

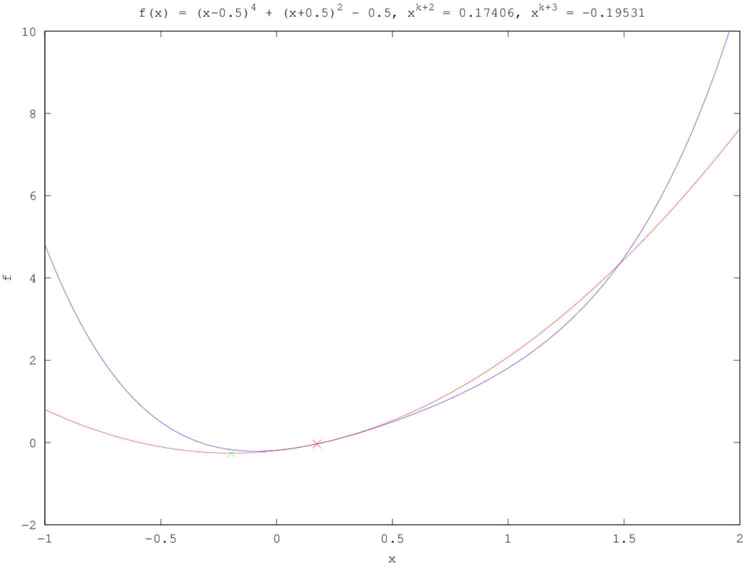


Illustration of Convex Problem

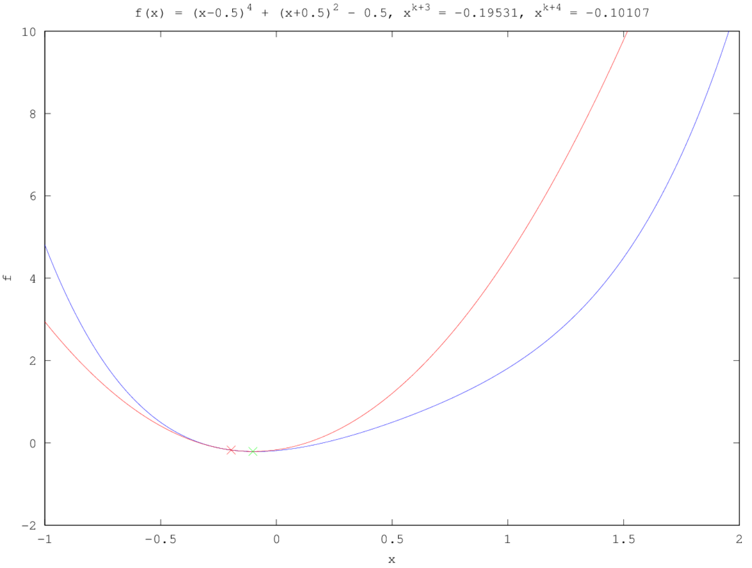


Illustration on Nonconvex Problem

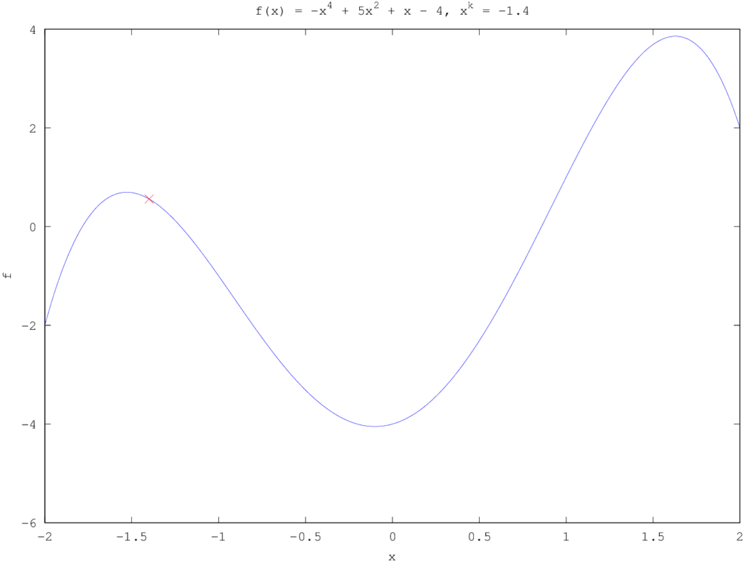
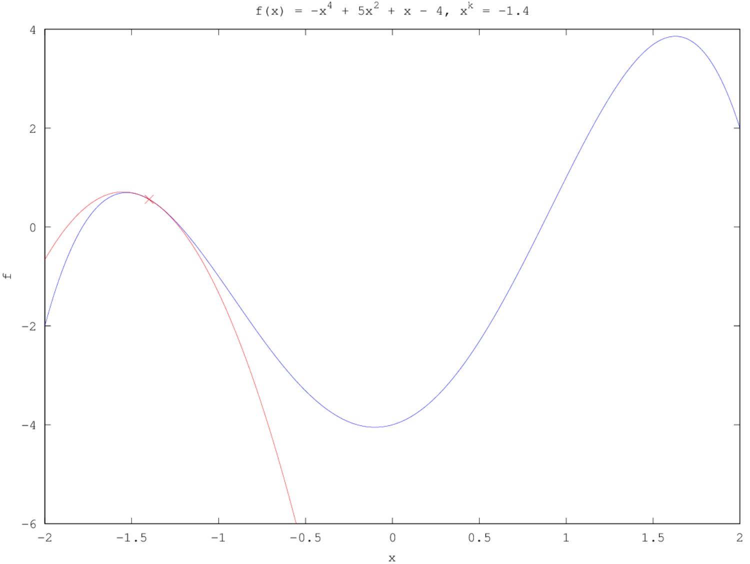


Illustration on Nonconvex Problem



Basic Theory

$$\min_x f(x)$$

- Convex functions – local minimizers are global minimizers
- Nonconvex functions
  - Stationarity: $\nabla f(x) = 0$
  - Local minimizer: $\nabla^2 f(x)$ is positive definite (min eig positive)
  - Local maximizer: $\nabla^2 f(x)$ is negative definite (max eig negative)

Solving Unconstrained Optimization Problems

$$\min_x f(x)$$

Main ingredients of solution approaches:

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

Trust-Region Method

$$\begin{array}{ll} \min\limits_{s} & f(x^k) + s^T \nabla f(x^k) + \frac{1}{2} s^T H(x^k) s \\ \text{subject to} & \|s\|_2 \le \Delta_k \end{array}$$

where $H(x^k)$ approximates $\nabla^2 f(x^k)$

Illustration on Nonconvex Problem

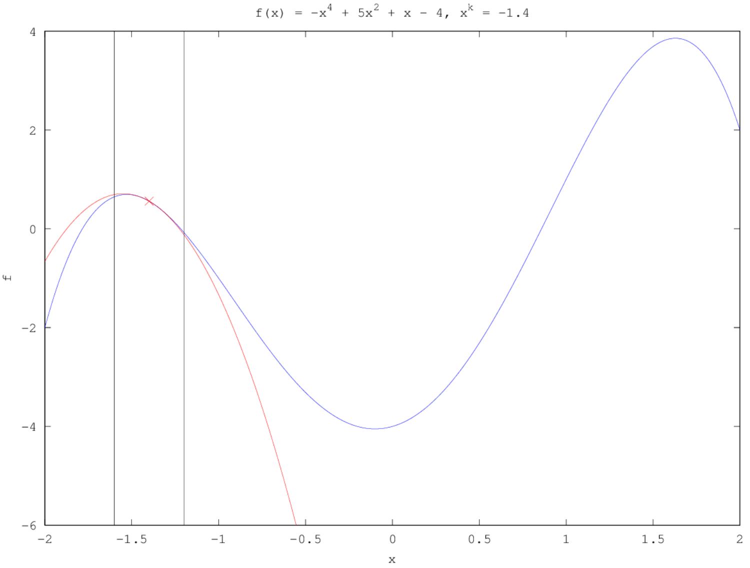
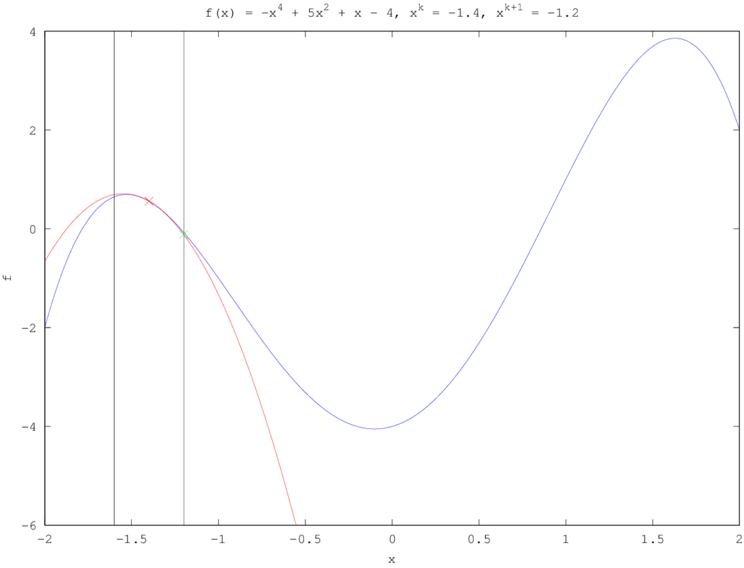
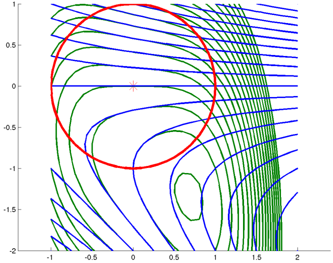
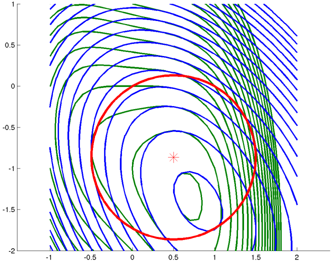


Illustration on Nonconvex Problem



Two-Dimensional Example





Trust-Region Method

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

$$\begin{array}{ll} \min_s & f(x^k) + s^T \nabla f(x^k) + \frac{1}{2} s^T H(x^k) s \\ \text{subject to} & \|s\|_2 \leq \Delta_k \end{array}$$

Trust-Region Method

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

$$\begin{array}{ll} \min_s & f(x^k) + s^T \nabla f(x^k) + \frac{1}{2} s^T H(x^k) s \\ \text{subject to} & \|s\|_2 \leq \Delta_k \end{array}$$

    2.2 Accept or reject iterate
    2.3 Update trust-region radius
        - Increase if actual reduction more than predicted
        - Decrease if actual reduction less than predicted
3. Check convergence

Illustration on Nonconvex Problem

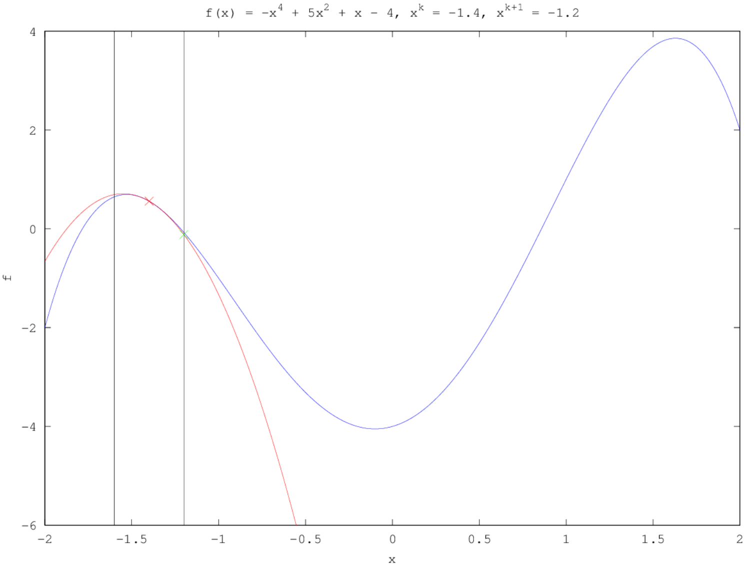


Illustration on Nonconvex Problem

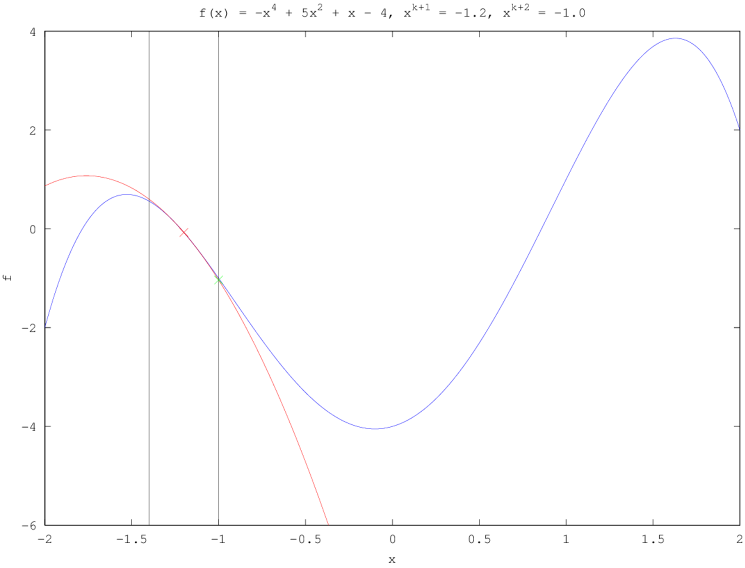


Illustration on Nonconvex Problem

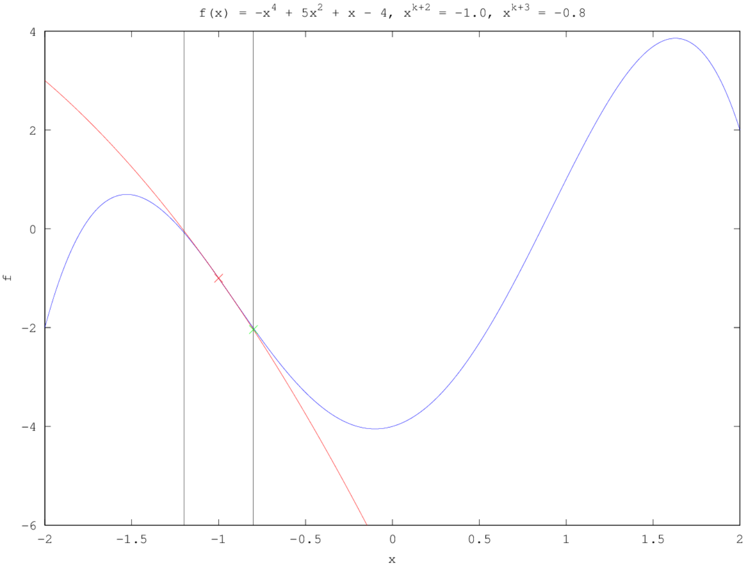


Illustration on Nonconvex Problem

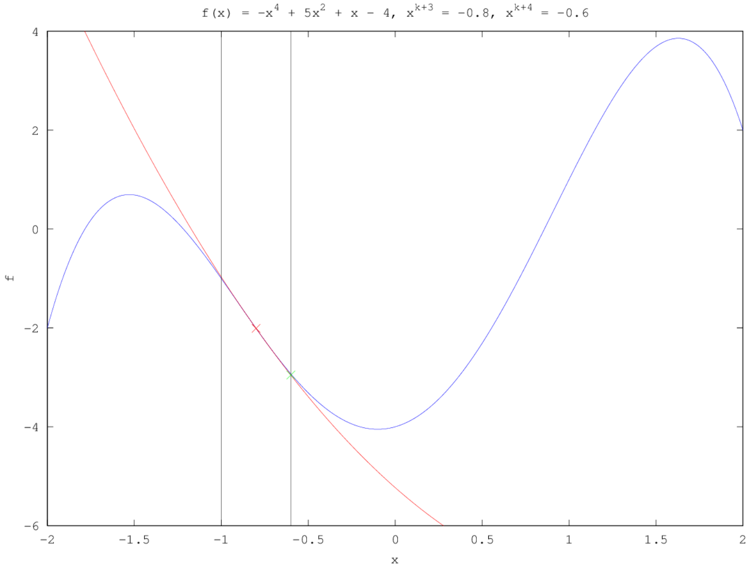


Illustration on Nonconvex Problem

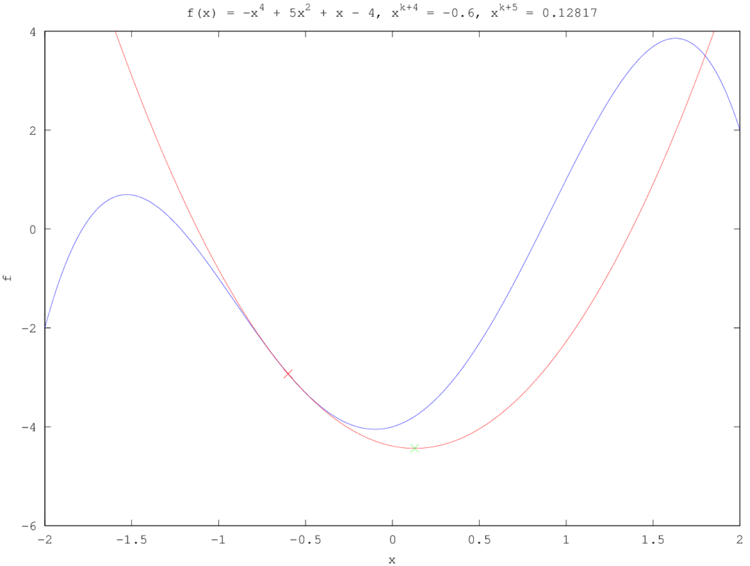
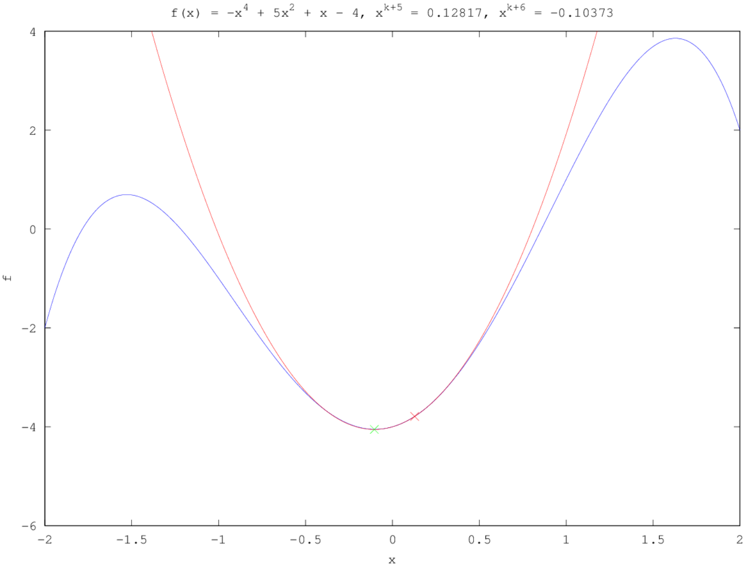


Illustration on Nonconvex Problem



Solving a Convex Quadratic Program

- Assume the quadratic program is strictly convex

$$\min_{s} \quad \frac{1}{2} s^T H s + c^T s$$

  - H is symmetric and positive definite
  - H^{-1} exists
- Stationary points are necessary and sufficient

$$Hs = -c$$

  - Cholesky factorization
    - Compute (sparse) lower triangular matrix with $H = LL^T$
    - Solve $s = L^{-T}(L^{-1}c)$ exploiting lower triangular property
  - Conjugate gradient method
    - Iteratively compute a set of H conjugate directions
    - Analytically minimize quadratic along the directions
    - Objective function decreases monotonically
    - Guaranteed convergence in n steps

Solving a Nonconvex Quadratic Program

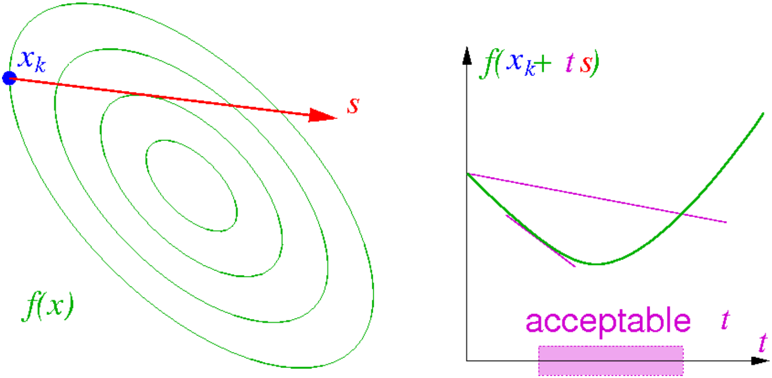
- No assumptions on the quadratic program

$$\begin{array}{ll} \min_s & \frac{1}{2} s^T H s + c^T s \\ \text{subject to} & \|s\|_2 \leq \Delta_k \end{array}$$

  - Trust region bounds objective function
  - No unbounded solutions
- Can detect inertia with a LDL^T factorization and use direct method
- Global solutions computed with Moré-Sorensen method
  - Requires repeated factorization of a matrix
  - Can be expensive to calculate
  - Little benefit
- Conjugate gradient method with a trust region
  - Iteratively compute a set of H conjugate directions
  - Analytically minimize quadratic along the directions
  - Stop when trust region boundary is encountered
  - Objective function decreases monotonically

Line-Search Method

$$\min_{s} \quad f(x^k) + s^T \nabla f(x^k) + \frac{1}{2} s^T (H(x^k) + \lambda_k I) s$$



Line-Search Method

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

$$\min_{s} \quad f(x^k) + s^T \nabla f(x^k) + \frac{1}{2} s^T (H(x^k) + \lambda_k I) s$$

Line-Search Method

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

$$\min_{s} \quad f(x^k) + s^T \nabla f(x^k) + \frac{1}{2} s^T (H(x^k) + \lambda_k I) s$$

3. Find new iterate
   3.1 Search along Newton direction
   3.2 Search along gradient-based direction

Line-Search Method

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

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

3. Find new iterate
   3.1 Search along Newton direction
   3.2 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

- Use LDL^T to determine inertia and update perturbation
- Apply conjugate gradient method and stop on unbounded directions

Solving the Subproblem

- Use LDL^T to determine inertia and update perturbation
- Apply conjugate gradient method and stop on unbounded directions
- Conjugate gradient method with trust region
  - Initialize radius
  - Update radius

Performing the Line Search

- Backtracking Armijo line search
  - Find t to satisfy sufficient decrease condition

$$f(x^k + ts) \le f(x^k) + \sigma t \nabla f(x^k)^T s$$

  - Try $t = 1, \beta, \beta^2, \ldots$ for $0 < \beta < 1$
- More-Thuente line search
  - Find t to satisfy strong Wolfe conditions

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

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

Updating the Perturbation

1. If increasing and $\lambda_k = 0$
$$\lambda_{k+1} = \mathrm{Proj}_{[\ell_0, u_0]} \left(\alpha_0 \|\nabla f(x^k)\| \right)$$
2. If increasing and $\lambda_k > 0$
$$\lambda_{k+1} = \mathrm{Proj}_{[\ell_i, u_i]} \left(\max \left(\alpha_i \|\nabla f(x^k)\|, \beta_i \lambda_k \right) \right)$$
3. If decreasing
$$\lambda_{k+1} = \min \left(\alpha_d \|\nabla f(x^k)\|, \beta_d \lambda_k \right)$$
4. If $\lambda_{k+1} < \ell_d$, then $\lambda_{k+1} = 0$

Trust-Region Line-Search Method

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

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

   2.2 Search along direction
   2.3 Update trust-region radius
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 to Improve Performance

- Modify system of equations solved

$$MHs = -Mc$$

  - M is symmetric positive definite
  - M^{-1} can be easily applied to vector
  - MH is well conditioned or has clustered eigenvalues
  - Corresponds to changing to an elliptic trust region

$$\begin{array}{ll} \min_s & \frac{1}{2}s^T Hs + c^T s \\ \text{subject to} & \|s\|_M \leq \Delta_k \end{array}$$

  where $\|x\|_M = \sqrt{x^T Mx}$ and M defines the ellipse
- Preconditioners are problem specific
- Many possibly preconditioners
  - No preconditioner – $M = I$
  - Diagonal of Hessian – $M = |\text{diag}(H(x^k))|$
  - Diagonal of perturbed Hessian – $M = |\text{diag}(H(x^k) + \lambda_k I)|$
  - Quasi-newton approximation to Hessian matrix
  - Incomplete Cholesky factorization of Hessian
  - Block Jacobi with Cholesky factorization of blocks

Termination

- Typical convergence criteria
  - Absolute residual $\|\nabla f(x^k)\| < \tau_a$
  - Relative residual $\frac{\|\nabla f(x^k)\|}{\|\nabla f(x_0)\|} < \tau_r$
  - Unbounded objective $f(x^k) < \kappa$
  - Slow progress $|f(x^k) - f(x_{k-1})| < \epsilon$
  - Iteration limit
  - Time limit
- Check the 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

Social Planning Model

- 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

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

Life-Cycle Saving Model

- 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

$$
\begin{array}{lll}
\max\limits_{s,c} & \sum\limits_{t=0}^{T} \beta^t u(c_t) & \\
\text{subject to} & s_{t+1} = (1+r)s_t + w - c_t & t = 0, \dots, R-1 \\
 & s_{t+1} = (1+r)s_t - c_t & t = R, \dots, T \\
 & s_0 = s_{T+1} = 0 &
\end{array}
$$

Theory Revisited

- Strict descent direction s

$$\nabla f(x)^T s < 0$$

- Stationarity conditions (first-order conditions)
  - No feasible, strict descent directions
  - For all feasible directions s

$$\nabla f(x)^T s \geq 0$$

  - Unconstrained case, $s \in \Re^n$ and

$$\nabla f(x) = 0$$

  - Constrained cases
    - Characterize feasible directions
    - Requires constraint qualification

Basic Theory

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

- Feasible and no strict descent directions
  - Constraint qualification – LICQ, MFCQ
  - Linearized active constraints characterize directions
  - Objective gradient is a linear combination of constraint gradients
    - Under a constraint qualification, $s = 0$ solves the linear program

      $$\begin{array}{ll} \min\limits_{s} & \nabla f(x^*)^T s + f(x^*) \\ \text{subject to} & \nabla c(x^*) s + c(x^*) \leq 0 \end{array}$$

      Note: x^* feasible, implies the linear program is feasible
    - Dual linear program

      $$\begin{array}{ll} \max\limits_{\lambda} & c(x^*)^T \lambda \\ \text{subject to} & \nabla f(x^*) + \nabla c(x^*)^T \lambda = 0 \\ & \lambda \geq 0 \end{array}$$

    - Linear programming theory
- Lagrangian: $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$

Convergence Criteria

Satisfies Constraint Qualification (Slater)

$$\begin{array}{ll} \min\limits_{x} & x \\ \text{subject to} & x^2 - 1 \leq 0 \end{array}$$

has solution $x = -1$

- Primal linear program

$$\begin{array}{ll} \min\limits_{s} & s \\ \text{subject to} & -2s \leq 0 \end{array}$$

- Optimal solution is $s = 0$
- Dual program produces $\lambda = \frac{1}{2}$

Convergence Criteria

Lacks Constraint Qualification (Slater/LICQ)

$$\begin{array}{ll} \min\limits_{x} & x \\ \text{subject to} & x^2 \le 0 \end{array}$$

has solution $x = 0$

- Primal linear program

$$\begin{array}{ll} \min\limits_{s} & s \\ \text{subject to} & 0 \le 0 \end{array}$$

- Optimal solution is $s = -\infty$
- Dual program is infeasible

Convergence Criteria

Lacks Constraint Qualification (Slater/LICQ)

$$
\begin{array}{ll}
\min\limits_{x} & x^2 \\
\text{subject to} & x^2 \leq 0
\end{array}
$$

has solution $x = 0$

- Primal linear program

$$
\begin{array}{ll}
\min\limits_{s} & 0 \\
\text{subject to} & 0 \leq 0
\end{array}
$$

- Optimal solution is $s \in (-\infty, \infty)$
- Dual program produces $\lambda \in [0, \infty)$
- Constraint qualification is sufficient but not required

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

$$\begin{array}{l} \nabla f(x) + \nabla c(x)^T \lambda = 0 \\ 0 \leq \lambda \quad \perp \quad -c(x) \geq 0 \end{array}$$

  - Complementarity problem

Solving Constrained Optimization Problems

Main ingredients of solution approaches:

- Local method: given x^k (solution guess) find a step s^k
  - 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
   2.1 Solve linear program

$$
\begin{array}{ll}
\min\limits_{s} & f(x^k) + \nabla f(x^k)^T s \\
\text{subject to} & c(x^k) + \nabla c(x^k) s \leq 0 \\
 & \|s\| \leq \Delta_k
\end{array}
$$

Sequential Linear Programming

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

$$\begin{array}{ll} \min\limits_{s} & f(x^k) + \nabla f(x^k)^T s \\ \text{subject to} & c(x^k) + \nabla c(x^k) s \leq 0 \\ & \|s\| \leq \Delta_k \end{array}$$

   2.2 Accept or reject iterate
   2.3 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
   2.1 Solve quadratic program

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

where $W(x^k, \lambda^k)$ approximates $\nabla^2_{x,x}\mathcal{L}(x^k, \lambda^k)$

Sequential Quadratic Programming

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

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

   where $W(x^k, \lambda^k)$ approximates $\nabla^2_{x,x} \mathcal{L}(x^k, \lambda^k)$

   2.2 Accept or reject iterate

   2.3 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
   2.1 Solve linear program to predict active set

$$
\begin{array}{ll}
\min\limits_{d} & f(x^k) + \nabla f(x^k)^T d \\
\text{subject to} & c(x^k) + \nabla c(x^k) d \leq 0 \\
& \|d\| \leq \Delta_k
\end{array}
$$

   where $\mathcal{A}_k = \{i \mid c(x^k) + \nabla c_i(x^k)^T d^k = 0\}$

Sequential Linear Quadratic Programming

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

$$\begin{array}{ll} \min\limits_{d} & f(x^k) + \nabla f(x^k)^T d \\ \text{subject to} & c(x^k) + \nabla c(x^k) d \leq 0 \\ & \|d\| \leq \Delta_k \end{array}$$

   where $\mathcal{A}_k = \{i \mid c(x^k) + \nabla c_i(x^k)^T d^k = 0\}$

   2.2 Solve equality constrained quadratic program

$$\begin{array}{ll} \min\limits_{s} & f(x^k) + \nabla f(x^k)^T s + \frac{1}{2} s^T W(x^k, \lambda^k) s \\ \text{subject to} & c_{\mathcal{A}_k}(x^k) + \nabla c_{\mathcal{A}_k}(x^k) s = 0 \end{array}$$

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

Acceptance Criteria

- Constraint violation: $h(x) = \| \max(c(x), 0) \|$
- Decrease objective function value: $f(x^k + s^k) \leq f(x^k)$
- Decrease constraint violation: $h(x^k + s^k) \leq h(x^k)$

Acceptance Criteria

- Constraint violation: $h(x) = \| \max(c(x), 0) \|$
- Decrease objective function value: $f(x^k + s^k) \leq f(x^k)$
- Decrease constraint violation: $h(x^k + s^k) \leq h(x^k)$
- Four possibilities
  1. step can decrease both $f(x)$ and $h(x)$ GOOD
  2. step can decrease $f(x)$ and increase $h(x)$???
  3. step can increase $f(x)$ and decrease $h(x)$???
  4. step can increase both $f(x)$ and $h(x)$ BAD

Acceptance Criteria

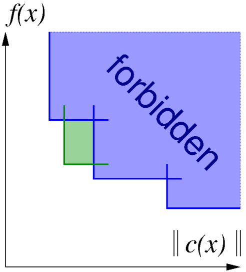
- Constraint violation: $h(x) = \| \max(c(x), 0) \|$
- Decrease objective function value: $f(x^k + s^k) \leq f(x^k)$
- Decrease constraint violation: $h(x^k + s^k) \leq h(x^k)$
- Four possibilities
  1. step can decrease both $f(x)$ and $h(x)$ GOOD
  2. step can decrease $f(x)$ and increase $h(x)$???
  3. step can increase $f(x)$ and decrease $h(x)$???
  4. step can increase both $f(x)$ and $h(x)$ 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)

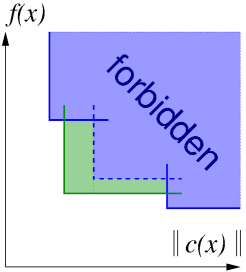
- new $x^{k+1} = x^k + s^k$ is acceptable to filter $\mathcal{F}$ iff for all $\ell \in \mathcal{F}$
  1. $h_{k+1} \le h_\ell$ or
  2. $f_{k+1} \le f_\ell$



Filter Framework

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

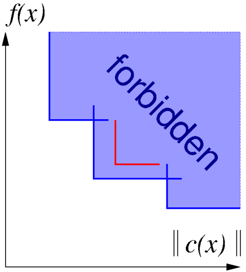
- new $x^{k+1} = x^k + s^k$ is acceptable to filter $\mathcal{F}$ iff for all $\ell \in \mathcal{F}$
  1. $h_{k+1} \leq h_\ell$ or
  2. $f_{k+1} \leq f_\ell$
- remove redundant filter entries



Filter Framework

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

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



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

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
  - Hessian is incorrect – disrupts quadratic convergence
  - Hessian is rank deficient
  - 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

- 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";`

Interior-Point Method

- Original optimization problem

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

Interior-Point Method

- Original optimization problem

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

- Reformulate by adding slacks

$$\begin{aligned} \min_{x,u} \quad & f(x) \\ \text{subject to} \quad & c(x) + u = 0 \\ & x \ge 0, \; u \ge 0 \end{aligned}$$

Interior-Point Method

- Equality constrained optimization problem

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

- Construct perturbed optimality conditions

$$
F_\tau(x, \lambda, \nu) = \begin{bmatrix} \nabla f(x) + \nabla c(x)^T \lambda - \mu \\ -c(x) \\ x \odot \mu - \tau e \end{bmatrix}
$$

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

Interior-Point Method

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

$$\begin{bmatrix} W^k & \nabla c(x^k)^T & -I \\ -\nabla c(x^k) & 0 & 0 \\ \operatorname{diag}(\mu^k) & 0 & \operatorname{diag}(x^k) \end{bmatrix} \begin{pmatrix} s_x \\ s_\lambda \\ s_\mu \end{pmatrix} = -F_\tau(x^k, \lambda^k, \mu^k)$$

   1.2 Accept or reject iterate
   1.3 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

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

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

Part IV

Equilibrium Problems

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^*)

$$
\begin{aligned}
x^* &\in \left\{ \begin{array}{ll} \operatorname*{arg\,min}_{x \geq 0} & f_1(x, y^*) \\ \text{subject to} & c_1(x) \leq 0 \end{array} \right. \\
y^* &\in \left\{ \begin{array}{ll} \operatorname*{arg\,min}_{y \geq 0} & f_2(x^*, y) \\ \text{subject to} & c_2(y) \leq 0 \end{array} \right.
\end{aligned}
$$

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^*)

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

- 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 $c_2(\cdot)$ are convex and satisfy constraint qualification
- Then first-order conditions are necessary and sufficient

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

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 $c_2(\cdot)$ are convex and satisfy constraint qualification
- Then first-order conditions are necessary and sufficient

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

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 $c_2(\cdot)$ are convex and satisfy constraint qualification
- Then first-order conditions are necessary and sufficient

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

- Nonlinear complementarity problem

$$0 \leq x \quad \perp \quad F(x) \geq 0$$

  - 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

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

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
  - Jacobian does not exist on set of measure zero
  - Degeneracy: $x_i = F_i(x) = 0$
- Merit function $\Psi(x) = \|\Phi(x)\|_2^2$ is differentiable!

Semismooth Algorithm

- Calculate $H^k \in \partial_B \Phi(x^k)$ and solve

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

- If this system has no solution or

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

then let $s^k = -\nabla \Psi(x^k)$

Semismooth Algorithm

- Calculate $H^k \in \partial_B \Phi(x^k)$ and solve
$$H^k s^k = -\Phi(x^k)$$
- If this system has no solution or
$$\nabla \Psi(x^k)^T s^k > -p_1 \|s^k\|^{p_2}$$
then let $s^k = -\nabla \Psi(x^k)$
- Compute smallest nonnegative integer i_k such that
$$\Psi(x^k + \beta^{i_k} s^k) \leq \Psi(x^k) + \sigma \beta^{i_k} \nabla \Psi(x^k) s^k$$
- Update the iterate
$$x^{k+1} = x^k + \beta^{i_k} s^k$$
and repeat until convergence

Convergence Issues

- Quadratic convergence – best outcome
- Linear convergence
  - Far from a solution – $\Psi(x^k)$ is large
  - Jacobian is incorrect – disrupts quadratic convergence
  - Jacobian is rank deficient – $\|\nabla \Psi(x^k)\|$ is small
  - Converge to local minimizer – guarantees rank deficiency
  - Limits of finite precision arithmetic
    1. $\Psi(x^k)$ converges quadratically to small number
    2. $\Psi(x^k)$ 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

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

- Leader solves optimization problem with equilibrium constraints

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

- 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

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

Nonlinear Programming Formulation

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

- 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

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

- Optimization problem satisfies constraint qualification
- Need to increase π

Relaxation Approach

$$
\begin{array}{ll}
\min\limits_{x,y,\lambda,s,t\geq 0} & g(x,y) \\
\text{subject to} & h(y) \leq 0 \\
& s_i = \nabla_{x_i} f_i(x,y) + \lambda_i^T \nabla_{x_i} c_i(x_i) \\
& t_i = -c_i(x_i) \\
& \sum\limits_i \left(s_i^T x_i + \lambda_i t_i \right) \leq \tau
\end{array}
$$

- 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

Part V

Optimization Without Derivatives

Part VI

Stochastic Approximation Methods

Stochastic Methods for Two Types of Problems

A. Stochastic optimization
   - Modeling and algorithms for optimization under uncertainty
   - Stochasticity from problem and/or algorithm

B. Deterministic optimization
   - Objectives and constraints deterministic
   - Methods are "randomized"

Stochastic Methods for Two Types of Problems

A. Stochastic optimization
   - Modeling and algorithms for optimization under uncertainty
   - Stochasticity from problem and/or algorithm

B. Deterministic optimization
   - Objectives and constraints deterministic
   - Methods are "randomized"

→ Methods and analysis are related

A.
Stochastic Optimization Problems and Methods

Stochastic Optimization

General problem

$$\min \{ f(x) = \mathbb{E}_\xi \left[F(x,\xi) \right] : \; x \in X \} \tag{1}$$

- $x \in \mathbb{R}^n$ decision variables
- ξ vector of random variables
  - independent of x
  - $P(\xi)$ distribution function for ξ
  - ξ has support Ξ
- $F(x,\cdot)$ functional form of uncertainty for decision x
- $X \subseteq \mathbb{R}^n$ set defined by deterministic constraints
  - Also: stochastic/probabilistic constraints (not addressed here)

Approach of Sampling Methods for $f(x) = \mathbb{E}_\xi \left[F(x,\xi)\right]$

- Let $\xi^1, \xi^2, \cdots, \xi^N \sim P$
- For $x \in X$, define:

$$f_N(x) = \frac{1}{N}\sum_{i=1}^{N} F(x,\xi^i)$$

  - f_N is a random variable (really, a stochastic process) (depends on $(\xi^1, \xi^2, \cdots, \xi^N)$)
  - Motivated by $\mathbb{E}_\xi \left[f_N(x)\right] = f(x)$

Bias of Sampling Methods

- Let $f^* = f(x^*)$ for $x^* \in X^* \subseteq X$

Bias of Sampling Methods

- Let $f^* = f(x^*)$ for $x^* \in X^* \subseteq X$
- For any $N \geq 1$:
$$\mathbb{E}_\xi [f_N^*] \leq f^* = \mathbb{E}_\xi [F(x^*, \xi)]$$
because
$$\mathbb{E}_\xi [f_1^*] = \mathbb{E}_\xi [\min \{F(x, \xi) : x \in X\}] \leq \min \{\mathbb{E}_\xi [F(x, \xi)] : x \in X\} = f^*$$

Bias of Sampling Methods

- Let $f^* = f(x^*)$ for $x^* \in X^* \subseteq X$
- For any $N \geq 1$:
$$\mathbb{E}_\xi [f_N^*] \leq f^* = \mathbb{E}_\xi [F(x^*, \xi)]$$
because
$$\mathbb{E}_\xi [f_1^*] = \mathbb{E}_\xi [\min \{F(x, \xi) : x \in X\}] \leq \min \{\mathbb{E}_\xi [F(x, \xi)] : x \in X\} = f^*$$

- Sampling problems result in optimal values below f^*
- f_N^* is biased estimator of f^*

Sample Average Approximation

- Draw realizations $\hat{\xi}^1, \hat{\xi}^2, \cdots, \hat{\xi}^N \sim P$ of $\left(\xi^1, \xi^2, \cdots, \xi^N\right)$
- Replace (1) with

$$\min \left\{ \frac{1}{N} \sum_{i=1}^{N} F(x, \hat{\xi}^i) : \ x \in X \right\} \tag{2}$$

  - $\hat{f}_N(x) = \frac{1}{N} \sum_{i=1}^{N} F(x, \hat{\xi}^i)$ deterministic
  - Follows mean of the N sample paths defined by the (fixed) $\hat{\xi}^i$

SAA Algorithm

Input N, (maybe $x^0 \in X$)

1. Generate $\hat{\xi}^1, \hat{\xi}^2, \cdots, \hat{\xi}^N \sim P$
2. Solve the deterministic problem

$$\min \left\{ \frac{1}{N} \sum_{i=1}^{N} F(x, \hat{\xi}^i) : \; x \in X \right\}$$

Output x_N^* (or X_N^*).

Convergence with N

- A sufficient condition:
  - For any $\epsilon > 0$ there exists N_ϵ so that

$$\left|\hat{f}_N(x) - f(x)\right| < \epsilon \qquad \forall\, N \geq N_\epsilon \quad \forall\, x \in X$$

  with probability 1 (*wp1*).
- Then $\hat{f}_N^* \to f^*$ *wp1*.
- (With additional assumptions on f and $X^* \subset X$):

$$\mathrm{dist}(x_N^*, X^*) \to 0$$

- ($+$ uniqueness, $X^* = x^*$):

$$x_N^* \to x^*$$

Stochastic Approximation Method

Basically just:

Input x^0

1. $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$, $\qquad k = 0, 1, \ldots$

- α_k a step size
- s^k a random direction

Stochastic Approximation Method

Basically just:

Input x^0

1. $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$, $\quad k = 0, 1, \ldots$

- α_k a step size
- s^k a random direction

Generally assume:

α_k: $\sum_{k=0}^{\infty} \alpha_k = \infty$, $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$

s^k: $\mathbb{E}\left[\nabla f(x^k)^T s^k\right] > 0$

s^k is an ascent direction (in expectation) at x^k

Stochastic Approximation Method

Basically just:

Input x^0

1. $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$, $\quad k = 0, 1, \ldots$

- α_k a step size
- s^k a random direction

Generally assume:

α_k: $\sum_{k=0}^{\infty} \alpha_k = \infty$, $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$

s^k: $\mathbb{E}\left[\nabla f(x^k)^T s^k\right] > 0$

s^k is an ascent direction (in expectation) at x^k

- "Exact" Stochastic Gradient Descent: $s^k = \nabla f(x^k)$

Classic SA Algorithms

- "Original" method is Robbins-Monro (1951)
- Without derivatives: Kiefer-Wolfowitz (1952)
  replaces gradient with finite-difference approximation, e.g.,

$$1.\quad x^{k+1} \leftarrow x^k - \alpha_k s^k, \qquad\qquad k = 0, 1, \ldots$$

  - where
$$s^k = \frac{F(x^k + h_k I_n; \hat{\xi}^k) - F(x^k - h_k I_n; \hat{\xi}^{k+1/2})}{2h_k}$$

Classic SA Algorithms

- "Original" method is Robbins-Monro (1951)
- Without derivatives: Kiefer-Wolfowitz (1952)
  replaces gradient with finite-difference approximation, e.g.,

1. $x^{k+1} \leftarrow x^k - \alpha_k s^k, \qquad k = 0, 1, \ldots$

  - where
$$s^k = \frac{F(x^k + h_k I_n; \hat{\xi}^k) - F(x^k - h_k I_n; \hat{\xi}^{k+1/2})}{2h_k}$$
  - Requires $2n$ evaluations every iteration
  - Can appeal to variance reduction techniques (e.g., common RNs)
  - Convergence $x^k \to x^*$ if f strongly convex (near x^*), usual conditions on α_k, $h_k \to 0$, $\sum_k \frac{\alpha_k^2}{h_k^2} < \infty$
  - K-W recommend: $\alpha_k = \frac{1}{k}$, $h_k = \frac{1}{k^{1/3}}$

Classic SA Algorithms

- "Original" method is Robbins-Monro (1951)
- Without derivatives: Kiefer-Wolfowitz (1952) replaces gradient with finite-difference approximation, e.g.,

$$1.\quad x^{k+1} \leftarrow x^k - \alpha_k s^k, \qquad k = 0, 1, \ldots$$

  - where
$$s^k = \frac{F(x^k + h_k I_n; \hat{\xi}^k) - F(x^k - h_k I_n; \hat{\xi}^{k+1/2})}{2h_k}$$
  - Requires $2n$ evaluations every iteration
  - Can appeal to variance reduction techniques (e.g., common RNs)
  - Convergence $x^k \to x^*$ if f strongly convex (near x^*), usual conditions on α_k, $h_k \to 0$, $\sum_k \frac{\alpha_k^2}{h_k^2} < \infty$
  - K-W recommend: $\alpha_k = \frac{1}{k}$, $h_k = \frac{1}{k^{1/3}}$
- Extensions such as SPSA (Spall) reduce number of evaluations (see randomized methods slides. . .)

Derivative-Based Stochastic Gradient Descent

Input x^0; Repeat:

1. Draw realization $\hat{\xi}^k \sim P$ of ξ^k
2. Compute $s^k = \nabla_x F(x^k; \hat{\xi}^k)$
3. Update $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$

Derivative-Based Stochastic Gradient Descent

Input x^0; Repeat:

1. Draw realization $\hat{\xi}^k \sim P$ of ξ^k
2. Compute $s^k = \nabla_x F(x^k; \hat{\xi}^k)$
3. Update $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$

- $\nabla_x F(x^k; \hat{\xi}^k)$ is an unbiased estimator for $\nabla f(x^k)$

Derivative-Based Stochastic Gradient Descent

Input x^0; Repeat:

1. Draw realization $\hat{\xi}^k \sim P$ of ξ^k
2. Compute $s^k = \nabla_x F(x^k; \hat{\xi}^k)$
3. Update $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$

- $\nabla_x F(x^k; \hat{\xi}^k)$ is an unbiased estimator for $\nabla f(x^k)$
- Can incorporate curvature if desired

  e.g., $B^k s^k$ an unbiased estimator for $\left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k)$
- Can work with subgradients
- Can even output $x^N = \frac{1}{N} \sum_{k=1}^{N} x^k$

Modern Stochastic Gradient Descent Codes

Stochastic gradient descent seems inherently sequential

- Better in special cases, e.g.,

$$f(x) = \sum_{e \in \mathcal{E}} f_e(x_e), \qquad e \subset \{1, \cdots, n\}$$

$|\mathcal{E}|$ and n large

Modern Stochastic Gradient Descent Codes

Stochastic gradient descent seems inherently sequential

- Better in special cases, e.g.,

$$f(x) = \sum_{e \in \mathcal{E}} f_e(x_e), \qquad e \subset \{1, \cdots, n\}$$

  $|\mathcal{E}|$ and n large
- HOGWILD! (Niu, Recht, Ré, Wright)
  - parallel, asynchronous implementation
  - `http://i.stanford.edu/hazy/victor/Hogwild/`
  - Basic idea: Each processor samples an e uniformly from $\mathcal{E}$ and updates the coordinates x_e, ... ties broken arbitrarily

B. Randomized Algorithms for Deterministic Problems

Randomized Algorithms for Deterministic Problems

$$\min \{f(x) : x \in X \subseteq \mathbb{R}^n\}$$

- f deterministic
- Random variables are now generated by the method, *not from the problem*
- Often assume properties of f

  e.g., ∇f is L'-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L' \|x - y\| \qquad \forall x, y \in X$$

  e.g., f is strongly convex (with parameter τ):

$$f(x) \geq f(y) + (x - y)^T \nabla f(y) + \frac{\tau}{2}\|x - y\|^2 \qquad \forall x, y \in X$$

Basic Algorithms

Matyas (e.g., 1965):

- Input x^0; repeat:
  1. Generate Gaussian u^k (centered about 0)
  2. Evaluate $f(x^k + u^k)$
  3. $x^{k+1} = \begin{cases} x^k + u^k & \text{if } f(x^k + u^k) < f(x^k) \\ x^k & \text{otherwise.} \end{cases}$

Basic Algorithms

Matyas (e.g., 1965):

- Input x^0; repeat:
  1. Generate Gaussian u^k (centered about 0)
  2. Evaluate $f(x^k + u^k)$
  3. $x^{k+1} = \begin{cases} x^k + u^k & \text{if } f(x^k + u^k) < f(x^k) \\ x^k & \text{otherwise.} \end{cases}$

Poljak (e.g., 1987)

- Input x^0, $\{h_k, \mu_k\}_k$; repeat:
  1. Generate a random $u^k \in R^n$
  2. $x^{k+1} = x^k - h_k \dfrac{f(x^k + \mu_k u^k) - f(x^k)}{\mu_k} u^k$
     - $h_k > 0$ is the step size
     - $\mu_k > 0$ is called the smoothing parameter

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \le M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \dots, n$$

Input x^0; Repeat:

1. Choose $i_k = \arg\max_{i=1,\dots,n} |\nabla_i f(x^k)|$
2. Update $x^{k+1} = x^k - \frac{1}{M}\nabla_{i_k} f(x^k) e_{i_k}$

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Input x^0; Repeat:

1. Choose $i_k = \arg\max_{i=1,\ldots,n} |\nabla_i f(x^k)|$
2. Update $x^{k+1} = x^k - \frac{1}{M}\nabla_{i_k} f(x^k) e_{i_k}$

- Generates $f(x^k) - f^* \leq \frac{2nMR^2}{k+4}$, where $R \geq \|x^0 - x^*\|$

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Input x^0; Repeat:

1. Choose $i_k = \arg\max_{i=1,\ldots,n} |\nabla_i f(x^k)|$
2. Update $x^{k+1} = x^k - \frac{1}{M} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $f(x^k) - f^* \leq \frac{2nMR^2}{k+4}$, where $R \geq \|x^0 - x^*\|$
- Good: only updates x_{i_k}
- Bad: requires entire gradient $\nabla f(x^k)$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \dots, n$$

Input x^0; Repeat:

1. Choose i_k uniformly at random from $\{1, \cdots, n\}$
2. Update $x^{k+1} = x^k - \frac{1}{L_i} \nabla_{i_k} f(x^k) e_{i_k}$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \le L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Input x^0; Repeat:

1. Choose i_k uniformly at random from $\{1, \cdots, n\}$
2. Update $x^{k+1} = x^k - \frac{1}{L_i} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $\mathbb{E}\left[f(x^k)\right] - f^* \le \frac{2nR_1^2}{k+4}$, where $R_1 = \max\{\|x - x^*\|_1 : f(x) \le f(x_0)\}$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \le L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \dots, n$$

Input x^0; Repeat:

1. Choose i_k uniformly at random from $\{1, \cdots, n\}$
2. Update $x^{k+1} = x^k - \frac{1}{L_i} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $\mathbb{E}\left[f(x^k)\right] - f^* \le \frac{2nR_1^2}{k+4}$, where $R_1 = \max\{\|x - x^*\|_1 : f(x) \le f(x_0)\}$
- Good: only updates x_{i_k}
- Better: requires only component i_k of gradient $\nabla f(x^k)$
- Can also:
  - generate i_k proportional to coordinate Lipschitz parameters $\{L_i\}_i$
  - perform block-coordinate (and other subspace) operations

Gaussian Smoothing

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be deterministic
- $u \in \mathbb{R}^n$ from a Gaussian distribution, $\mathbb{E}_u[u] = 0$
  - Here: Covariance matrix I_n, general C OK
- For scalar $\mu > 0$, Gaussian-smoothed version of f:

$$f_\mu(x) = \mathbb{E}_u[f(x + \mu u)]$$

Gaussian Smoothing

- Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be deterministic
- $u \in \mathbb{R}^n$ from a Gaussian distribution, $\mathbb{E}_u[u] = 0$
  - Here: Covariance matrix I_n, general C OK
- For scalar $\mu > 0$, Gaussian-smoothed version of f:

$$f_\mu(x) = \mathbb{E}_u[f(x + \mu u)]$$

  - If f is convex, then $f_\mu(x) \geq f(x)$
  - If f is convex and ∇f is L'-Lipschitz, then

$$|f_\mu(x) - f(x)| \leq \frac{\mu^2}{2} L' n$$

Gaussian Smoothing and Directional Derivatives

$$f_\mu(x) = \mathbb{E}_u \left[f(x + \mu u) \right]$$

- Derivative of f in the direction u: $f'(x; u) = \lim_{h \downarrow 0} \frac{f(x+hu) - f(x)}{h}$

Gaussian Smoothing and Directional Derivatives

$$f_\mu(x) = \mathbb{E}_u \left[f(x + \mu u) \right]$$

- Derivative of f in the direction u: $f'(x; u) = \lim_{h \downarrow 0} \frac{f(x+hu)-f(x)}{h}$
- $g_0(x) = f'_u(x) u$
  - If f is convex, then $\mathbb{E}_u \left[g_0(x) \right]$ is a subgradient of f
  - If f is differentiable at x, then

$$\mathbb{E}_u \left[\|g_0(x)\|^2 \right] \leq (n+4) \|\nabla f(x)\|^2$$

Gaussian Smoothing and Directional Derivatives

$$f_\mu(x) = \mathbb{E}_u\left[f(x+\mu u)\right]$$

- Derivative of f in the direction u: $f'(x;u) = \lim_{h\downarrow 0} \frac{f(x+hu)-f(x)}{h}$
- $g_0(x) = f'_u(x)u$
  - If f is convex, then $\mathbb{E}_u\left[g_0(x)\right]$ is a subgradient of f
  - If f is differentiable at x, then

$$\mathbb{E}_u\left[\|g_0(x)\|^2\right] \le (n+4)\|\nabla f(x)\|^2$$

- $g_\mu(x) = \frac{f(x+\mu u)-f(x)}{\mu} u$
  - If f is differentiable at x, then $\mathbb{E}_u\left[g_\mu(x)\right] = \nabla f_\mu(x)$
  - If f is differentiable at x and ∇f is L'-Lipschitz, then

$$\mathbb{E}_u\left[\|g_\mu(x)\|^2\right] \le 2(n+4)\|\nabla f(x)\|^2 + \frac{\mu^2}{2} L'^2 (n+6)^3$$

Random Gradient Method

Input $x^0 \in X$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_0(x^k) = f'_{u^k}(x^k)u^k$
2. $x^{k+1} = \mathcal{P}_X\left\{x^k - h_k g_0(x^k)\right\}$

Random Gradient Method

Input $x^0 \in X$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_0(x^k) = f'_{u^k}(x^k)u^k$
2. $x^{k+1} = \mathcal{P}_X \left\{x^k - h_k g_0(x^k)\right\}$

- Key result (Nesterov) for convex (but possibly nonsmooth) f:
  For fixed $h_k = \frac{R}{\sqrt{n+4}\sqrt{N+1}L}$ and any $\epsilon > 0$,
  $$\mathbb{E}_u\left[f(\hat{x}^N)\right] - f^* \leq \epsilon, \qquad \text{where} \quad \hat{x}^N = \arg\min_{i=1,\ldots,N} f(x^i)$$
  in $\mathcal{O}\left(\frac{n}{\epsilon^2}\right)$ iterations
- Also works for convex stochastic optimization and convex smooth f (with improved bounds and rates)

Random Gradient-Free Method

Input $x^0 \in X$, $\mu > 0$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(x^k) = \frac{f(x^k+u^k)-f(x^k)}{\mu} u^k$
2. $x^{k+1} = \mathcal{P}_X \left\{x^k - h_k g_\mu(x^k)\right\}$

Random Gradient-Free Method

Input $x^0 \in X$, $\mu > 0$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(x^k) = \frac{f(x^k+u^k)-f(x^k)}{\mu} u^k$
2. $x^{k+1} = \mathcal{P}_X\left\{x^k - h_k g_\mu(x^k)\right\}$

- Key result (Nesterov) for convex (but possibly nonsmooth) f:
  For fixed $h_k = \frac{R}{(n+4)\sqrt{N+1}L}$, $\mu = \frac{\epsilon}{2L\sqrt{n}}$, and any $\epsilon > 0$,

  $$\mathbb{E}_u\left[f(\hat{x}^N)\right] - f^* \le \epsilon, \qquad \text{where} \quad \hat{x}^N = \arg\min_{i=1,\ldots,N} f(x^i)$$

  in $\mathcal{O}\left(\frac{n^2}{\epsilon^2}\right)$ iterations
- Also works for convex stochastic optimization and convex smooth f (with improved bounds and rates)

Accelerated Random Gradient-Free Method

f strongly convex (with convexity parameter τ)

Input $v^0 = x^0$, $\mu > 0$, $\gamma_0 \geq \tau$, $\{h_k\}_k$; repeat:

1. Obtain $\alpha_k > 0$ satisfying $16(n+1)^2 L' \alpha_k^2 = (1-\alpha_k)\gamma_k + \tau\alpha_k$
2. Set $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \tau\alpha_k$, $\lambda_k = \frac{\alpha_k \tau}{\gamma_{k+1}}$, $\beta_k = \frac{\alpha_k \gamma_k}{\gamma_k + \alpha_k \tau}$
3. Set $y^k = (1-\beta_k)x^k + \beta_k v^k$
4. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(y^k) = \frac{f(y^k+u^k)-f(y^k)}{\mu} u^k$
5. Update

$$
\begin{aligned}
x^{k+1} &= \quad y^k - \frac{1}{4(n+4)L'} g_\mu(y^k) \\
v^{k+1} &= \quad (1-\lambda_k)v^k + \lambda_k y^k - \frac{1}{16(n+1)^2 L' \alpha_k} g_\mu(y^k)
\end{aligned}
$$

Accelerated Random Gradient-Free Method

f strongly convex (with convexity parameter τ)

Input $v^0 = x^0$, $\mu > 0$, $\gamma_0 \geq \tau$, $\{h_k\}_k$; repeat:

1. Obtain $\alpha_k > 0$ satisfying $16(n+1)^2 L' \alpha_k^2 = (1-\alpha_k)\gamma_k + \tau\alpha_k$
2. Set $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \tau\alpha_k$, $\lambda_k = \frac{\alpha_k \tau}{\gamma_{k+1}}$, $\beta_k = \frac{\alpha_k \gamma_k}{\gamma_k + \alpha_k \tau}$
3. Set $y^k = (1-\beta_k)x^k + \beta_k v^k$
4. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(y^k) = \frac{f(y^k+u^k)-f(y^k)}{\mu} u^k$
5. Update

$$\begin{aligned} x^{k+1} &= \quad y^k - \tfrac{1}{4(n+4)L'} g_\mu(y^k) \\ v^{k+1} &= \quad (1-\lambda_k)v^k + \lambda_k y^k - \tfrac{1}{16(n+1)^2 L' \alpha_k} g_\mu(y^k) \end{aligned}$$

- Key result (Nesterov): for $\tau = 0$ functions $\exists \mu > 0$ so that

$$\mathbb{E}_u \left[f(\hat{x}^N) \right] - f^* \leq \epsilon, \qquad \text{where} \quad \hat{x}^N = \arg \min_{i=1,\ldots,N} f(x^i)$$

in $\mathcal{O}\left(\frac{n}{\epsilon^{1/2}}\right)$ iterations

Applying SA-Like Ideas to Special Cases

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

m huge

Applying SA-Like Ideas to Special Cases

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

m huge

Ex.- *Nonlinear Least Squares* Warning: likely nonconvex!
$F_i(x) = \|\phi(x;\theta^i) - d^i\|^2$
Evaluating $\phi(\cdot,\cdot)$ requires solving a large PDE

Ex.- *Sample Average Approximation*
$F_i(x) = R(x;\hat{\xi}^i)$
$\hat{\xi}^i \in \Omega$ a scenario/RV realization
(and R depends nontrivially on $\hat{\xi}^i$)

Applying SA-Like Ideas to Special Cases

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

m huge

Ex.- *Nonlinear Least Squares* Warning: likely nonconvex!
$F_i(x) = \|\phi(x;\theta^i) - d^i\|^2$
Evaluating $\phi(\cdot,\cdot)$ requires solving a large PDE

Ex.- *Sample Average Approximation*
$F_i(x) = R(x;\hat{\xi}^i)$
$\hat{\xi}^i \in \Omega$ a scenario/RV realization
(and R depends nontrivially on $\hat{\xi}^i$)

The good:

- $\nabla f(x) = \sum_{i=1}^{m} \nabla F_i(x)$

The bad:

- m still huge

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$
- Under minimal assumptions:

$$\mathbb{E}\left[\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} F_i(x)\right] = f(x) \qquad \text{and} \qquad \mathbb{E}\left[\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla F_i(x)\right] = \nabla f(x)$$

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$
- Under minimal assumptions:

$$\mathbb{E}\left[\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} F_i(x)\right] = f(x) \quad \text{and} \quad \mathbb{E}\left[\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla F_i(x)\right] = \nabla f(x)$$

- Use $-\nabla f_{\mathcal{S}} = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla F_i(x)$ as direction s^k

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$
- Under minimal assumptions:

$$\mathbb{E}\left[\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} F_i(x)\right] = f(x) \qquad \text{and} \qquad \mathbb{E}\left[\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla F_i(x)\right] = \nabla f(x)$$

- Use $-\nabla f_{\mathcal{S}} = -\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla F_i(x)$ as direction s^k
- How to choose $\mathcal{S}$?

$$\mathbb{E}\left[\|\nabla f_{\mathcal{S}_n} - \nabla f\|^2\right] = \left(1 - \frac{|\mathcal{S}|}{m}\right) \mathbb{E}\left[\|\nabla f_{\mathcal{S}_r} - \nabla f\|^2\right]$$

$\Rightarrow$ sampling *without replacement* ($\mathcal{S}_n$) gives lower variance than does sampling *with replacement* ($\mathcal{S}_r$)

Summary

- Methods for stochastic optimization and randomized methods for deterministic optimization closely related

+ Incredibly simple to code basic implementation
+ Well-studied complexity bounds, especially for convex cases; can show that asymptotic rates are optimal
+ Even useful when gradient/subgradient unavailable

Summary

- Methods for stochastic optimization and randomized methods for deterministic optimization closely related

+ Incredibly simple to code basic implementation
+ Well-studied complexity bounds, especially for convex cases; can show that asymptotic rates are optimal
+ Even useful when gradient/subgradient unavailable

- Bounds and parameters depend on characteristics of function (e.g., Lipschitz parameters, level set diameters, strong convexity)
- (Some) Practitioners remain nervous about performance deviations from the mean (active research area)

Part VII

Advanced Topics

Optimal Technology Penetration

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

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

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

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

$$z(T) \leq z_T$$

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

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:

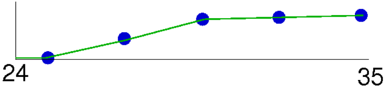
- $t \in [0, T]$ replaced by $N+1$ equally spaced points $t_i = ih$
- $h := T/N$ time integration step-length
- approximate $q_i^n \simeq q^n(t_i)$ etc.

Replace differential equation

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

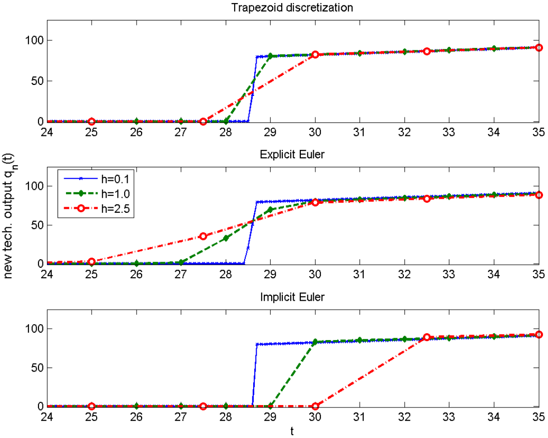
by

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



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 $\bar{K}_0^j$:

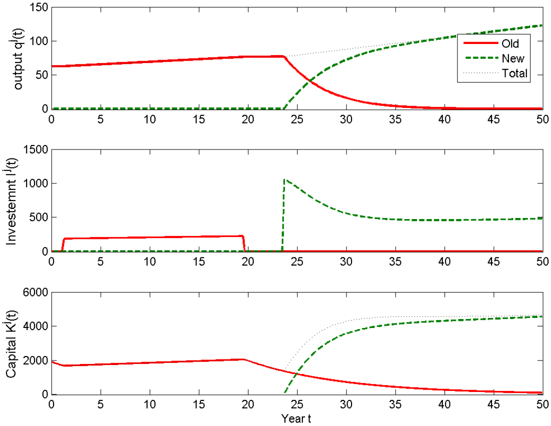
Notation:

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

Optimal Technology Penetration

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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!

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