CS 450: Numerical Anlaysis[1]

Numerical Optimization

University of Illinois at Urbana-Champaign

[1] *These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book "Scientific Computing: An Introductory Survey" by Michael T. Heath (slides).*

Numerical Optimization

- Our focus will be on *continuous* rather than *combinatorial* optimization:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0} \quad \text{and} \quad \boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$$

- We consider linear, quadratic, and general nonlinear optimization problems:

Local Minima and Convexity

- Without knowledge of the analytical form of the function, numerical optimization methods at best achieve convergence to a *local* rather than *global* minimum:

- A set is *convex* if it includes all points on any line, while a function is (strictly) convex if its (unique) local minimum is always a global minimum:

Existence of Local Minima

- *Level sets* are all points for which f has a given value, *sublevel sets* are all points for which the value of f is less than a given value:

- If there exists a closed and bounded sublevel set in the domain of feasible points, then f has has a global minimum in that set:

Optimality Conditions

- If $\boldsymbol{x}$ is an interior point in the feasible domain and is a local minima,

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{df}{dx_1}(\boldsymbol{x}) & \cdots & \frac{df}{dx_n}(\boldsymbol{x}) \end{bmatrix}^T = \mathbf{0} :$$

- *Critical points* $\boldsymbol{x}$ satisfy $\nabla f(\boldsymbol{x}) = \mathbf{0}$ and can be minima, maxima, or saddle points:

Hessian Matrix

- To ascertain whether a critical point $\boldsymbol{x}$, for which $\nabla f(\boldsymbol{x}) = \mathbf{0}$, is a local minima, consider the *Hessian matrix*:
- If $\boldsymbol{x}^*$ is a minima of f, then $\boldsymbol{H}_f(\boldsymbol{x}^*)$ is positive semi-definite:

Optimality on Feasible Region Border

- Given an equality constraint $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$, it is no longer necessarily the case that $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$. Instead, it may be that directions in which the gradient decreases lead to points outside the feasible region:

$$\exists \boldsymbol{\lambda} \in \mathbb{R}^n, \quad -\nabla f(\boldsymbol{x}^*) = \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda}$$

- Such *constrained minima* are critical points of the Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$, so they satisfy:

$$\nabla \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}^*) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}^*) \end{bmatrix} = \mathbf{0}$$

Sensitivity and Conditioning

- The condition number of solving a nonlinear equations is $1/f'(x^*)$, however for a minimizer x^*, we have $f'(x^*) = 0$, so conditioning of optimization is inherently bad:

- To analyze worst case error, consider how far we have to move from a root $\boldsymbol{x}^*$ to perturb the function value by ϵ:

Golden Section Search

- Given bracket $[a, b]$ with a unique local minimum (f is *unimodal* on the interval), *golden section search* considers consider points $f(x_1), f(x_2)$, $a < x_1 < x_2 < b$ and discards subinterval $[a, x_1]$ or $[x_2, b]$:

- Since one point remains in the interval, golden section search selects x_1 and x_2 so one of them can be effectively reused in the next iteration:

Newton's Method for Optimization

- At each iteration, approximate function by quadratic and find minimum of quadratic function:

- The new approximate guess will be given by $x_{k+1} - x_k = -f'(x_k)/f''(x_k)$:

Successive Parabolic Interpolation

- Interpolate f with a quadratic function at each step and find its minima:

- The convergence rate of the resulting method is roughly 1.324

Safeguarded 1D Optimization

- Safeguarding can be done by bracketing via golden section search:

- Backtracking and step-size control:

General Multidimensional Optimization

- Direct search methods by simplex (*Nelder-Mead*):

- Steepest descent: find the minimizer in the direction of the negative gradient:

Convergence of Steepest Descent

- Steepest descent converges linearly with a constant that can be arbitrarily close to 1:

- Given quadratic optimization problem $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$ where $\boldsymbol{A}$ is symmetric positive definite, the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$ satisfies

Gradient Methods with Extrapolation

- We can improve the constant in the linear rate of convergence of steepest descent by leveraging *extrapolation methods*, which consider two previous iterates (maintain *momentum* in the direction $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$):
- The *heavy ball method*, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

Conjugate Gradient Method

- The *conjugate gradient method* is capable of making the optimal choice of α_k and β_k at each iteration of an extrapolation method:

- *Parallel tangents* implementation of the method proceeds as follows

Nonlinear Conjugate Gradient

- Various formulations of conjugate gradient are possible for nonlinear objective functions, which differ in how they compute β below
- Fletcher-Reeves is among the most common, computes the following at each iteration
  1. Perform 1D minimization for α in $f(\boldsymbol{x}_k + \alpha \boldsymbol{s}_k)$
  2. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\alpha} \boldsymbol{s}_k$
  3. Compute gradient $\boldsymbol{g}_{k+1} = \nabla f(\boldsymbol{x}_{k+1})$
  4. Compute $\beta = \boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1} / (\boldsymbol{g}_k^T \boldsymbol{g}_{k+1})$
  5. $\boldsymbol{s}_{k+1} = -\boldsymbol{g}_{k+1} + \beta \boldsymbol{s}_k$

Conjugate Gradient for Quadratic Optimization

- Conjugate gradient is an optimal iterative method for quadratic optimization, $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$
- For such problems, it can be expressed in an efficient and succinct form, computing at each iteration
  1. $\alpha = \boldsymbol{r}_k^T\boldsymbol{r}_k/\boldsymbol{s}_k^T\boldsymbol{A}\boldsymbol{s}_k$
  2. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\boldsymbol{s}_k$
  3. Compute gradient $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\boldsymbol{A}\boldsymbol{s}_k$
  4. Compute $\beta = \boldsymbol{r}_{k+1}^T\boldsymbol{r}_{k+1}/(\boldsymbol{r}_k^T\boldsymbol{r}_{k+1})$
  5. $\boldsymbol{s}_{k+1} = \boldsymbol{r}_{k+1} + \beta\boldsymbol{s}_k$
- Note that for quadratic optimization, the negative gradient $-\boldsymbol{g}$ corresponds to the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$

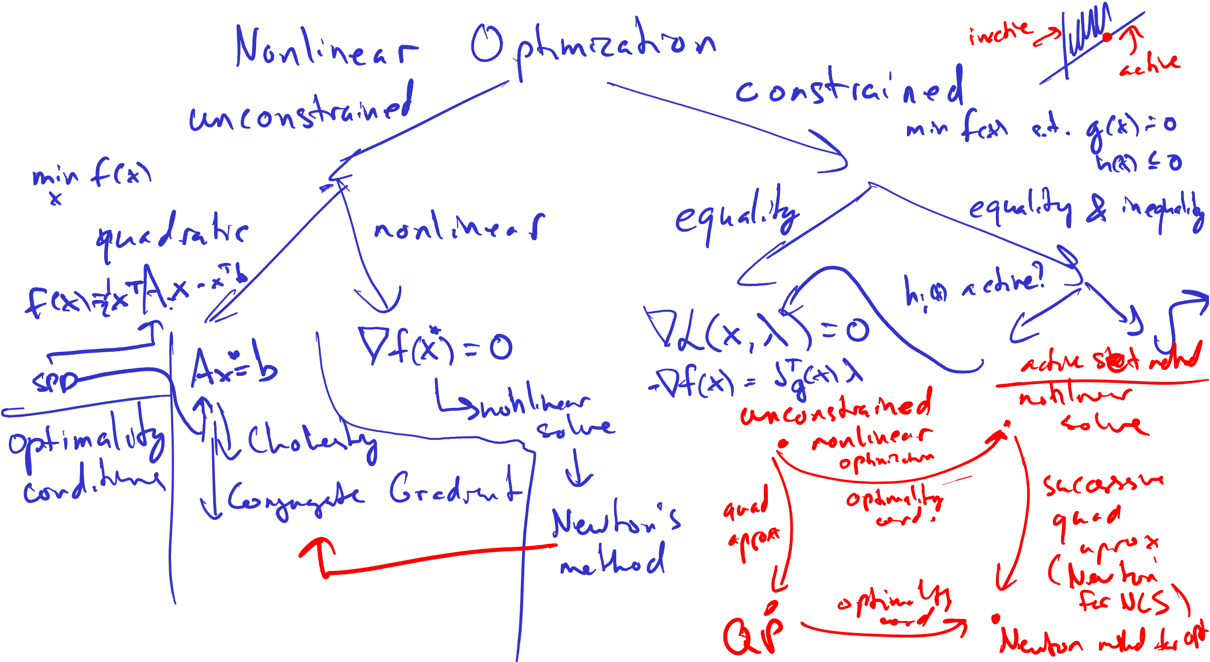
Krylov Optimization

- Conjugate Gradient finds the minimizer of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ within the Krylov subspace of $\boldsymbol{A}$:

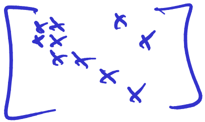
Newton's Method

- Newton's method in n dimensions is given by finding minima of n-dimensional quadratic approximation:

Nonlinear Optimization



Conjugate Gradient (CG)

- Nonlinear optimization method
  - variants
- Primarily used for QPs, $f(x) = \frac{1}{2}x^T A x - x^T b$
  - require matrix-vector products with A
    - good if A is sparse $A =$ 
  - linear conv rate, with constant $c \le \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$
  - short recurrence, store only 2 vectors
    - ~ Lanczos (Krylov for symmetric) $Q_k^T A Q_k = H_k$
  - each search direction is A-conjugate w.r.t. previous direction

$x^T A y = 0$

$x^T y = 0$

- CG will converge to exact solution after k steps where k is # unique eigenvalues in A

$k \leq \dim(A)$

Quasi-Newton Methods

- *Quasi-Newton* methods compute approximations to the Hessian at each step:

- The *BFGS* method is a secant update method, similar to Broyden's method:

Nonlinear Least Squares

- An important special case of multidimensional optimization is *nonlinear least squares*, the problem of fitting a nonlinear function $f_{\boldsymbol{x}}(t)$ so that $f_{\boldsymbol{x}}(t_i) \approx y_i$:

$f(t) = x_1 t^2 + x_2 t + x_3 + x_4 \log(t)$ $\quad (t_1, y_1), \dots, (t_m, y_m)$

- We can cast nonlinear least squares as an optimization problem and solve it by Newton's method:

$r_i(x) = y_i - f_x(t_i)$ | objective fun.

$e(x) = \frac{1}{2}\|r(x)\|_2^2 = \frac{1}{2} r(x)^T r(x)$

$\nabla e(x) = J_r^T(x)\, r(x)$

$H_e(x) = J_r^T(x) J_r(x) + \sum_{i=1}^{m} r_i(x) H_{r_i}(x)$

$\approx 0 \Rightarrow \approx 0$

Gauss-Newton Method

- The Hessian for nonlinear least squares problems has the form:

$$H_G(x) = J_r^T(x) J_r(x) + \sum_{i=1}^{m} r_i(x) H_{r_i}(x)$$

- The *Gauss-Newton* method is Newton iteration with an approximate Hessian:

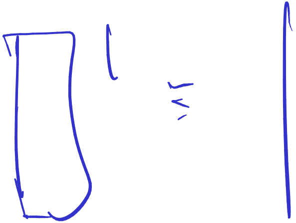
$$\hat{H}_G(x) = J_r^T(x) J_r(x) \approx H_G(x)$$

$$s_k = x_{k+1} - x_k = \hat{H}_G(x_k)^{-1} \nabla G(x_k) = \left(J_r^T(x_k) J_r(x_k)\right)^{-1} J_r^T(x_k) r(x_k)$$

$$J_r(x_k) s_k \cong r(x_k)$$

- The Levenberg-Marquardt method incorporates Tykhonov regularization into the linear least squares problems within the Gauss-Newton method.

$$J_r(x_k)\, s_k \overset{\sim}{=} r(x_k)$$



Tykhonov Regularization

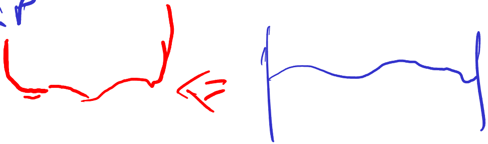
$$\begin{bmatrix} \lambda I \\ J_r(x_k) \end{bmatrix} s_k \overset{\sim}{=} \begin{bmatrix} 0 \\ r(x_k) \end{bmatrix}$$

Constrained Optimization Problems

- We now return to the general case of *constrained* optimization problems:

$$\min_x f(x) \quad \text{s.t.} \quad g(x) = 0, \quad h(x) \leq 0$$

f quad → QP



- Generally, we will seek to reduce constrained optimization problems to a series of unconstrained optimization problems:
  - *sequential quadratic programming*: (SQP) → sequence of QPs (equality constrained; unconstrained)
  - *penalty-based methods*: (interior point methods) approximate constraints by penalties to objective function
  - *active set methods*: select 'active' constraints from inequality constraints

Sequential Quadratic Programming

- *Sequential quadratic programming* (SQP) corresponds to using Newton's method to solve the equality constrained optimality conditions, by finding critical points of the Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$,

$$\nabla \mathcal{L}(x, \lambda) = \begin{bmatrix} \nabla f(x) + J_g^T(x)\lambda \\ g(x) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- At each iteration, SQP computes $\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{\lambda}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{s}_k \\ \boldsymbol{\delta}_k \end{bmatrix}$ by solving

$$H_{\mathcal{L}}(x_k, \lambda_k) \begin{bmatrix} s_k \\ \delta_k \end{bmatrix} = -\nabla \mathcal{L}(x_k, \lambda_k)$$

$$H_{\mathcal{L}}(x_k, \lambda_k) = \begin{bmatrix} B(x_k, \lambda_k) & J_g^T(x_k) \\ J_g(x_k) & 0 \end{bmatrix}$$

saddle point matrix

with $B(x, \lambda) = H_f(x) + \sum_{i=1}^{n} \lambda_i H_{g_i}(x)$

Inequality Constrained Optimality Conditions

- The *Karush-Kuhn-Tucker (KKT)* conditions hold for local minima of a problem with equality and inequality constraints, the key conditions are
  - x^* must be in feasible region $g(x^*) = 0$, $h(x^*) \le 0$
  - ith ineq. constraint is active if $h_i(x^*) = 0$
  - the collection of equality and active ineq. constraints $q^*(x^*) = 0$, $q^* = \begin{bmatrix} g(x^*) \\ h_{active}(x^*) \end{bmatrix}$
  - $-\nabla f(x^*) = J_{q^*}^T(x^*) \lambda^*$
- To use SQP for an inequality constrained optimization problem, consider at each iteration an *active set* of constraints:
  - q_k to be active $h_i(x)_k$ constraints at iteration k and $g(x)$
    - equality, ineq. violated or saturated at x_k
  - perform Newton's method to minimize $\mathcal{L}_k(x, \lambda) = f(x) + \lambda^T q_k(x)$

Penalty Functions

- Alternatively, we can reduce constrained optimization problems to unconstrained ones by modifying the objective function. *Penalty* functions are effective for equality constraints $\boldsymbol{g}(\boldsymbol{x}) = 0$:

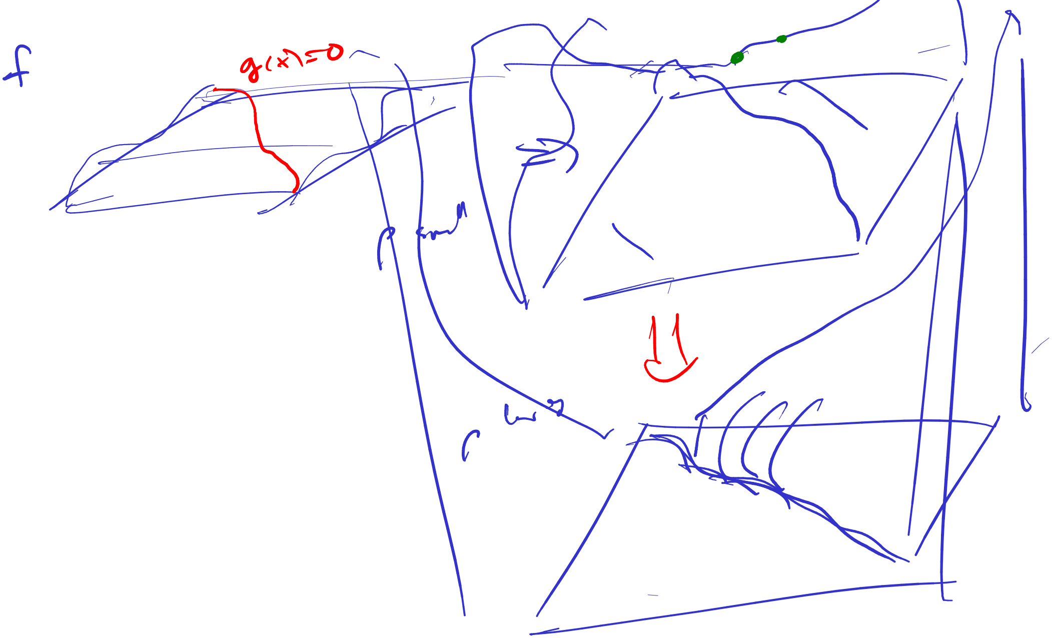
$$\varphi_\rho(x) = f(x) + \frac{1}{2}\rho\, g^T(x)\, g(x)$$

x^*_ρ is min of $\varphi_\rho(x)$, $\lim_{\rho \to \infty} x^*_\rho = x^*$

Hessian of φ_ρ is ill-conditioned for large ρ

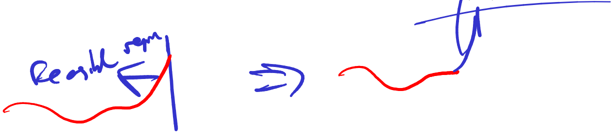
- The augmented Lagrangian function provides a more numerically robust approach:

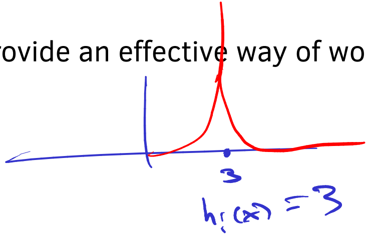
$$\mathcal{L}_\rho(x, \lambda) = f(x) + \lambda^T g(x) + \frac{1}{2}\rho\, g(x)^T g(x)$$


f
g(x)=0

Barrier Functions

- *Barrier functions* (*interior point methods*) provide an effective way of working with inequality constraints $\boldsymbol{h}(\boldsymbol{x}) \le \boldsymbol{0}$:





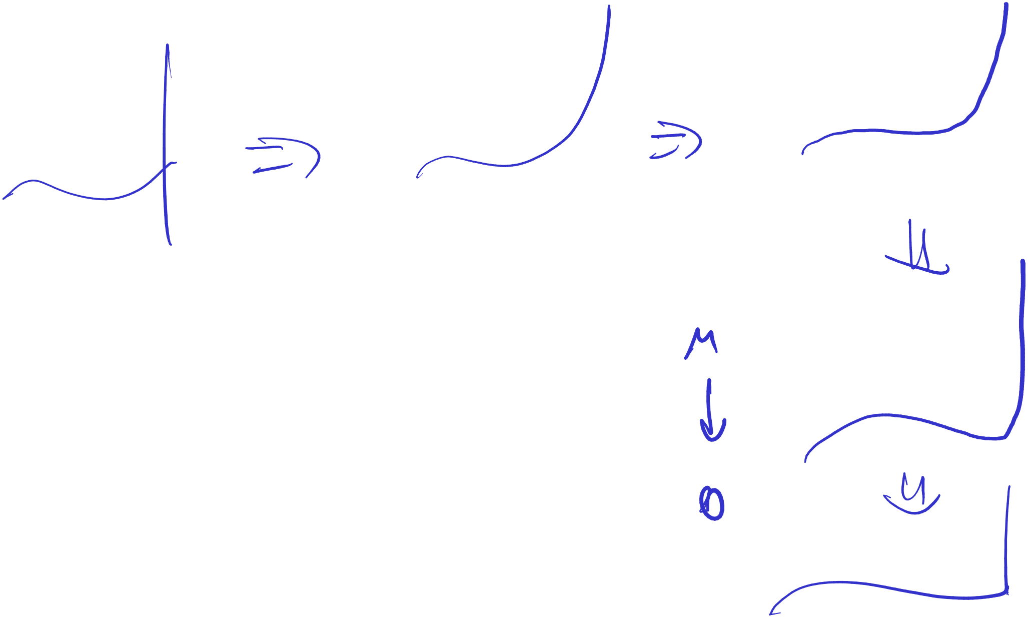
- inverse barrier function

$$\varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{m} \frac{1}{h_i(x)}$$

- logarithmic barrier function

$$\varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{m} \log(-h_i(x))$$

$$x^*_\mu \to x^* \quad \text{as } \mu \to 0$$


M
0