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:

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

  where $f \in \mathbb{R}^n \to \mathbb{R}$ is assumed to be differentiable.
  - *Without the constraints, i.e. with $\boldsymbol{g} = \boldsymbol{0}$ and $\boldsymbol{h} = \boldsymbol{0}$, the problem is unconstrained.*
  - *With constraints, the constrained optimization problem restricts the solution to elements of the feasible region: $\{\boldsymbol{x} : \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \text{ and } \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}\}$.*
- We consider linear, quadratic, and general nonlinear optimization problems:
  - *If f, $\boldsymbol{g}$, and $\boldsymbol{h}$ are affine (linear and constant terms only) then we have linear programming problem.*
  - *If f is quadratic while $\boldsymbol{g}$ and $\boldsymbol{h}$ are linear, then we have a quadratic programming problem, for which specialized methods exist.*
  - *Generally, we have a nonlinear programming problem.*

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:

  If the input domain is infinite or the global minimum is in an infinitesimally narrow trough, it may be impossible to find the global minimum in finite time.

- A set is *convex* if it includes all points on any line, while a function is convex if it is greater or equal to points on any of its tangent lines:
  - *Set S is convex if*

    $$\forall \boldsymbol{x}, \boldsymbol{y} \in S, \quad \alpha \in [0,1], \quad \alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y} \in S.$$

  - *Function f is convex if*

    $$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha)f(\boldsymbol{y}).$$

  - *A twice-differentiable convex function always has nonnegative second derivative, hence a local minima of a convex function is also a global minima.*

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:

$$L(z) = \{\boldsymbol{x} : f(\boldsymbol{x}) = z\}$$
$$S(z) = \{\boldsymbol{x} : f(\boldsymbol{x}) \leq z\}$$

- 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:

  Need a value z such that $S(z)$ has finite size, is contiguous, and includes its own boundary.

Optimality Conditions

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

$$\nabla f(\boldsymbol{x}) = \left[\frac{df}{dx_1}(\boldsymbol{x}) \quad \cdots \frac{df}{dx_n}(\boldsymbol{x})\right]^T = \mathbf{0} :$$

  - *If* $\frac{df}{dx_i}(\boldsymbol{x}) < 0$ *an infinitesimal increment to* x_i *improves the solution,*
  - *if* $\frac{df}{dx_i}(\boldsymbol{x}) > 0$ *an infinitesimal decrement to* x_i *improves the solution.*

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

  For scalar function f*, can distinguish the three by considering sign of* $f''(x)$*.*

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*:

$$\boldsymbol{H}_f(\boldsymbol{x}) = \boldsymbol{J}_{\nabla f}(\boldsymbol{x}) = \begin{bmatrix} \frac{d^2 f}{dx_1^2}(\boldsymbol{x}) & \cdots & \frac{d^2 f}{dx_1 dx_n}(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \frac{d^2 f}{dx_n dx_1}(\boldsymbol{x}) & \cdots & \frac{d^2 f}{dx_n^2}(\boldsymbol{x}) \end{bmatrix}$$

  The Hessian matrix is always symmetric if f is twice differentiable.

- If $\boldsymbol{x}^*$ is a minima of f, then $\boldsymbol{H}_f(\boldsymbol{x}^*)$ is positive semi-definite:

  If $\boldsymbol{H}_f(\boldsymbol{x}^*)$ is not positive semi-definite, there exists normalized vector $\boldsymbol{s}$ such that $\boldsymbol{s}^T\boldsymbol{H}_f(\boldsymbol{x}^*)\boldsymbol{s} < 0$, which means that for a sufficiently small α, $\hat{\boldsymbol{x}} = \boldsymbol{x}^* + \alpha\boldsymbol{s}$ will have be a better solution, $f(\hat{\boldsymbol{x}}) < f(\boldsymbol{x}^*)$, since the gradient is zero at $\boldsymbol{x}^*$ and decreases for an infinitesimal perturbation of $\boldsymbol{x}^*$ in the direction $\boldsymbol{s}$.

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:

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

  - *$\boldsymbol{\lambda}$ are referred to as the Lagrange multipliers.*
  - *This necessary condition implies that at $\boldsymbol{x}^*$, the direction in which f decreases is in the span of directions moving along which would exit the feasible region.*

- 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:

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

  Seeking $\boldsymbol{\lambda}^*$ to obtain a function $k(\boldsymbol{x}) = \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}^*)$ with maximum global minimum is the dual optimization problem.

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:

  Consider perturbation of function values for a function that changes slowly near the minimum.

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

$$\epsilon = f(x^* + h) - f(x^*) = \underbrace{f'(x^*)h}_{0} + \frac{1}{2}f''(x^*)h^2 + O(h^3)$$

  - *so if the function value changes by a infinitesimal perturbation ϵ, we have the error to the solution h, satisfies $h = O(\sqrt{\epsilon/f''(x^*)})$*
  - *a perturbation to the function value in the kth significant digit, could result in the solution changing in the $k/2$th significant digit.*

Demo: *Golden Section Proportions*

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 $[x_2, b]$:
  - *If a function is strictly convex and bounded on $[a, b]$, it is unimodal on that interval, but a unimodal function may be non-convex.*
  - *Because the function is unimodal, if we have $f(x_1) < f(x_2)$ then the unique local minima f in $[a, b]$ has to be in the interval $[a, x_2]$.*
  - *So, if $f(x_1) < f(x_2)$ can restrict search to $[a, x_2]$ and otherwise to $[x_1, b]$.*
- 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:
  - *For example, when $f(x_1) > f(x_2)$, x_2 is inside $[x_1, b]$ and we would like x_2 to serve as the x_1 for the next iteration.*
  - *To ensure this, and minimize resulting interval length, we pick $x_2 = a + (b - a)(\sqrt{5} - 1)/2$ and $x_1 = b - (b - a)(\sqrt{5} - 1)/2$.*
  - *Consequently, the convergence of golden secetion search is linear with constant $(\sqrt{5} - 1)/2$ per function evaluation.*

Newton's Method for Optimization

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

  Pick quadratic function $\hat{f}$ as first three terms of Taylor expansion of f about x_k, matching value and first two derivatives of f at x_k.

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

$$f(x) \approx \hat{f}(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2$$

  since the function is quadratic, we can find its unique critical point to find its minima,

$$\hat{f}'(x_{k+1}) = f'(x_k) + f''(x_k)(x_{k+1} - x_k) = 0.$$

Successive Parabolic Interpolation

- Interpolate f with a quadratic function at each step and find its minima:
  Given three points, there is a unique quadratic function interpolating them.
- The convergence rate of the resulting method is roughly 1.324
  By comparison, the convergence of golden section search is linear with a constant of 0.618*, while Newton's method converges quadratically.*

Safeguarded 1D Optimization

- Safeguarding can be done by bracketing via golden section search:
  Combination of Newton and golden section search
  - *achieves quadratic convergence locally,*
  - *is guaranteed convergence provided unimodality of function.*
- Backtracking and step-size control:
  - *Can take smaller step* $x_{k+1} = x_k - \alpha_k f'(x_k)/f''(x_k)$ *for some* $\alpha_k < 1$.
  - *Can backtrack and choose smaller* α_k *if* $f(x_{k+1}) > f(x_k)$.

Demo: *Nelder-Mead Method*

General Multidimensional Optimization

- Direct search methods by simplex (*Nelder-Mead*):
  - *form a $n+1$-point polytope in n-dimensional space and adjust worst point (highest function value) by moving it along a line passing through the centroid of the remaining points,*
  - *relies on function evaluations only, but can converge to nonstationary points even for convex 2D functions.*
- Steepest descent: find the minimizer in the direction of the negative gradient:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

such that $f(\boldsymbol{x}_{k+1}) = \min_{\alpha_k} f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k))$, i.e. perform a line search (solve 1D optimization problem) 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:
  - *Convergence is slow locally, in the worst case, and generally depends on the Hessian near the minima.*
  - *If the gradient is changing quickly, it serves as good approximation only within a small local neighborhood, so the line search may result in arbitrarily small steps.*
- 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, consider the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$:
  - *We can quantify the error using the norm, $||\boldsymbol{x}||_{\boldsymbol{A}} = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$, as*

$$\lim_{k\to\infty} \frac{||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}}}{||\boldsymbol{e}_k||_{\boldsymbol{A}}} = \frac{\sigma_{\mathit{max}}(\boldsymbol{A}) - \sigma_{\mathit{min}}(\boldsymbol{A})}{\sigma_{\mathit{max}}(\boldsymbol{A}) + \sigma_{\mathit{min}}(\boldsymbol{A})}$$

  - *When sufficiently close to a local minima, general nonlinear optimization problems are described by such an SPD quadratic problem.*
  - *Convergence rate depends on the conditioning of $\boldsymbol{A}$, since*

$$\frac{\sigma_{\mathit{max}}(\boldsymbol{A}) - \sigma_{\mathit{min}}(\boldsymbol{A})}{\sigma_{\mathit{max}}(\boldsymbol{A}) + \sigma_{\mathit{min}}(\boldsymbol{A})} = \frac{\kappa(\boldsymbol{A}) - 1}{\kappa(\boldsymbol{A}) + 1}.$$

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 $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$):

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

- The *heavy ball method*, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

  For a quadratic program defined by $\boldsymbol{A}$, these exist α and β, such that the convergence rate of the heavy ball method is

$$\lim_{k\to\infty} \frac{||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}}}{||\boldsymbol{e}_k||_{\boldsymbol{A}}} = \frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1}$$

  Nesterov's gradient optimization method is another instance of an extrapolation method that provides further improved optimality guarantees.

Conjugate Gradient Method

- The *conjugate gradient method* is capable of making the optimal choice (for quadratic programs) of α_k and β_k at each iteration:

$$(\alpha_k, \beta_k) = \underset{\alpha_k, \beta_k}{\operatorname{argmin}} \left[f\Big(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \Big) \right]$$

  - *For SPD quadratic programming problems, conjugate gradient is an optimal 1st order method, converging in $n-1$ iterations.*
  - *It implicitly computes Lanczos iteration, searching along $\boldsymbol{A}$-orthogonal directions at each step.*

- *Parallel tangents* implementation of the method in a general nonlinear setting proceeds as follows

  1. *Perform a step of steepest descent to generate $\hat{\boldsymbol{x}}_k$ from $\boldsymbol{x}_k$.*
  2. *Generate $\boldsymbol{x}_{k+1}$ by minimizing over the line passing through $\boldsymbol{x}_{k-1}$ and $\hat{\boldsymbol{x}}_k$.*

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. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{s}_k$
  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 $\boldsymbol{A}$:
  - *It constructs Krylov subspace $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}) =$ span$(\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{A}^{r-1}\boldsymbol{b})$.*
  - *At the kth step conjugate gradient yields iterate*

    $$\boldsymbol{x}_k = ||\boldsymbol{b}||_2\boldsymbol{Q}_k\boldsymbol{T}_k^{-1}\boldsymbol{e}_1,$$

    where $\boldsymbol{Q}_k$ are the Lanczos vectors associated with $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ and $\boldsymbol{T}_k = \boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k$.
  - *This choice of $\boldsymbol{x}_k$ minimizes $f(\boldsymbol{x})$ since*

    $$\begin{aligned}\min_{\boldsymbol{x}\in\mathcal{K}_k(\boldsymbol{A},\boldsymbol{c})} f(\boldsymbol{x}) &= \min_{\boldsymbol{y}\in\mathbb{R}^k} f(\boldsymbol{Q}_k\boldsymbol{y}) \\ &= \min_{\boldsymbol{y}\in\mathbb{R}^k} \boldsymbol{y}^T\boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} - \boldsymbol{b}^T\boldsymbol{Q}_k\boldsymbol{y} \\ &= \min_{\boldsymbol{y}\in\mathbb{R}^k} \boldsymbol{y}^T\boldsymbol{T}_k\boldsymbol{y} - ||\boldsymbol{b}||_2\boldsymbol{e}_1^T\boldsymbol{y}\end{aligned}$$

    is minimized by $\boldsymbol{y} = ||\boldsymbol{b}||_2\boldsymbol{T}_k^{-1}\boldsymbol{e}_1$.
  - *Since $\boldsymbol{T}_k$ differs from $\boldsymbol{T}_{k-1}$ only in addition of a single row and column, by Sherman-Morrison-Woodbury, efficient updates exist to solve for each $\boldsymbol{y}$.*

Newton's Method

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

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx \hat{f}(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \boldsymbol{s}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{H}_f(\boldsymbol{x}_k)\boldsymbol{s}.$$

The existence of second derivatives of f at $\boldsymbol{x}_k$ ($\boldsymbol{H}_f(\boldsymbol{x}_k)$) is needed.
The minima of this function can be determined by identifying critical points

$$\boldsymbol{0} = \nabla \hat{f}(\boldsymbol{s}) = \nabla f(\boldsymbol{x}_k) + \boldsymbol{H}_f(\boldsymbol{x}_k)\boldsymbol{s},$$

thus to determine s we solve the linear system,

$$\boldsymbol{H}_f(\boldsymbol{x}_k)\boldsymbol{s} = -\nabla f(\boldsymbol{x}_k).$$

Assuming invertibility of the Hessian, Newton's method iteration is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \underbrace{\boldsymbol{H}_f(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k)}_{\boldsymbol{s}_k}.$$

Quadratic convergence follows by equivalence to Newton's method for solving nonlinear system of optimality equations $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$.

Quasi-Newton Methods

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

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{B}_k^{-1} \nabla f(\boldsymbol{x}_k)$$

  where α_k is a line search parameter. Quasi-Newton methods can be more robust than Newton's method, as the Newton's method step can lead to a direction in which the objective function is strictly increasing.

- The *BFGS* method is a secant update method, similar to Broyden's method:
  - *At each iteration, perform a rank-2 update to $\boldsymbol{B}_k$ using $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y_k} = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$:*

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{B}_k}{\boldsymbol{s}_k^T \boldsymbol{B}_k \boldsymbol{s}_k}$$

  - *Can update inverse with $O(n^2)$ work, but its more stable and efficient to update a symmetric indefinite factorization.*
  - *The BFGS method also preserves symmetry of the Hessian approximation.*

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 $f_{\boldsymbol{x}}(t_i) \approx y_i$:
  For example, consider fitting $f_{[x_1,x_2]}(t) = x_1 \sin(x_2 t)$ *so that*
  $$\begin{bmatrix} f_{[x_1,x_2]}(1.5) \\ f_{[x_1,x_2]}(1.9) \\ f_{[x_1,x_2]}(3.2) \end{bmatrix} \approx \begin{bmatrix} -1.2 \\ 4.5 \\ 7.3 \end{bmatrix}.$$
- We can cast nonlinear least squares as an optimization problem and solve it by Newton's method:
  Define residual vector function $\boldsymbol{r}(\boldsymbol{x})$ *so that* $r_i(\boldsymbol{x}) = y_i - f_{\boldsymbol{x}}(t_i)$ *and minimize*
  $$\phi(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{r}(\boldsymbol{x})||_2^2 = \frac{1}{2}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}).$$
  Now the gradient is $\nabla\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})$ *and the Hessian is*
  $$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_{r_i}(\boldsymbol{x}).$$

Gauss-Newton Method

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

$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_{r_i}(\boldsymbol{x}).$$

  The second term is small when the residual function $\boldsymbol{r}(\boldsymbol{x})$ is small, so approximate

$$\boldsymbol{H}_\phi(\boldsymbol{x}) \approx \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}).$$

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

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}_k)^{-1}\nabla\phi(\boldsymbol{x}_k) = \boldsymbol{x}_k - (\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k))^{-1}\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k).$$

  - *recognizing the normal equations, we interpret the Gauss-Newton method as solving linear least squares problems $\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k)\boldsymbol{s}_k \cong \boldsymbol{r}(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$.*
  - *Gauss-Newton can also be derived by taking a linear approximation of f at $\boldsymbol{x}_k$.*
  - *Tykhonov regularization is often incorporated, yielding Levenberg-Marquardt.*

Constrained Optimization Problems

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

  $$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \textit{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \quad \textit{and} \quad \boldsymbol{h}(\boldsymbol{x}) \le \boldsymbol{0}$$

  When f is quadratic, while h, g is linear, this is a quadratic optimization problem.

- Generally, we will seek to reduce constrained optimization problems to a series of unconstrained optimization problems:
  - *sequential quadratic programming*: solve an unconstrained quadratic optimization problem at each iteration,
  - *penalty-based methods*: solve a series of more complicated (more ill-conditioned) unconstrained optimization problems,
  - *active set methods*: define sequence of optimization problems with inequality constrains ignored or treated as equality 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 $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$,

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

- 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

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \begin{bmatrix} \boldsymbol{s}_{k} \\ \boldsymbol{\delta}_{k} \end{bmatrix} = -\nabla \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$$

where

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) = \begin{bmatrix} \boldsymbol{B}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) & \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}_k) \\ \boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}_k) & \boldsymbol{0} \end{bmatrix} \quad \textit{with} \quad \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{H}_f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \boldsymbol{H}_{g_i}(\boldsymbol{x})$$

Inequality Constrained Optimality Conditions

- The *Karush-Kuhn-Tucker (KKT)* conditions are necessary coniditions for local minima of a problem with equality and inequality constraints, they include
  - *First, any minima $\boldsymbol{x}^*$ must be a feasible point, so $\boldsymbol{g}(\boldsymbol{x}^*) = \mathbf{0}$ and $\boldsymbol{h}(\boldsymbol{x}^*) \leq \mathbf{0}$.*
  - *We say the ith inequality constraint is active at a minima $\boldsymbol{x}^*$ if $h_i(\boldsymbol{x}^*) = 0$.*
  - *The collection of equality constraints and active inequality constraints $\boldsymbol{q}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}) & \boldsymbol{h}(\boldsymbol{x}) \end{bmatrix}^T$, satisfies $\boldsymbol{q}(\boldsymbol{x}^*) = \mathbf{0}$.*
  - *The negative gradient of the objective function at the minima must be in the row span of the Jacobian of this collection of constraints:*

$$-\nabla f(\boldsymbol{x}^*) = \boldsymbol{J}_{\boldsymbol{q}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda}^* \quad \text{where } \boldsymbol{\lambda}^* = \begin{bmatrix} \boldsymbol{\lambda}_1 & \boldsymbol{\lambda}_2 \end{bmatrix}^T \text{ and } \boldsymbol{\lambda}_2 \leq 0.$$

- To use SQP for an inequality constrained optimization problem, consider at each iteration an *active set* of constraints:
  - *Active set $\boldsymbol{q}_k$ contains all equality constraints and all inequality constraints that are exactly satisfied or violated at $\boldsymbol{x}_k$.*
  - *Active set method: perform one step of Newton's method to minimize $\mathcal{L}_k(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{q}_k(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, then update active set.*

Penalty Functions

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

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

  is a simple merit function, and its solutions $\boldsymbol{x}_\rho^*$ satisfy $\lim_{\rho\to\infty} \boldsymbol{x}_\rho^* = \boldsymbol{x}^*$. However, the Hessian of ϕ_ρ becomes increasingly ill-conditioned for large ρ, leading to slow convergence.

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

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

Barrier Functions

- *Barrier functions* (*interior point methods*) provide an effective way of working with inequality constraints $\boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$:
  - *Provided we start at a feasible point, modify objective function so it diverges to ∞ when approaching border of feasible region.*
  - *Inverse barrier function:*

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

  - *Logarithmic barrier function:*

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

  - *When using sufficiently small steps, we have $\boldsymbol{x}^*_\mu \to \boldsymbol{x}^*$ as $\mu \to 0$.*
  - *Barrier and penality methods solve a sequence of unconstrained problems (for changing ρ or μ), requiring multiple executions of e.g., Newton's method.*
  - *Primal-dual interior point methods can also be derived from the KKT conditions.*