

# CS 450: Numerical Analysis<sup>1</sup>

## Numerical Optimization

University of Illinois at Urbana-Champaign

---

<sup>1</sup> *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:

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

where  $f \in \mathbb{R}^n \rightarrow \mathbb{R}$  is assumed to be differentiable.

- ▶ Without the constraints, i.e. with  $g = 0$  and  $h = 0$ , the problem is *unconstrained*.
- ▶ With constraints, the *constrained* optimization problem restricts the solution to elements of the *feasible region*:  $\{x : g(x) = 0 \text{ and } h(x) \leq 0\}$ .
- ▶ We consider linear, quadratic, and general nonlinear optimization problems:
  - ▶ If  $f$ ,  $g$ , and  $h$  are affine (linear and constant terms only) then we have *linear programming* problem.
  - ▶ If  $f$  is quadratic while  $g$  and  $h$  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 (strictly) convex if its (unique) local minimum is always a global minimum:
  - ▶ *Set  $S$  is convex if*

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

- ▶ *Function  $f$  is convex if*

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

- ▶ *A function may have a unique global minima but not be convex.*

## 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:

$$L(z) = \{\mathbf{x} : f(\mathbf{x}) = z\}$$

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

- ▶ If there exists a closed and bounded sublevel set in the domain of feasible points, then  $f$  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  $\mathbf{x}$  is an interior point in the feasible domain and is a local minima,

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

- ▶ If  $\frac{df}{dx_i}(\mathbf{x}) < 0$  an infinitesimal increment to  $x_i$  improves the solution,
  - ▶ if  $\frac{df}{dx_i}(\mathbf{x}) > 0$  an infinitesimal decrement to  $x_i$  improves the solution.
- ▶ **Critical points**  $\mathbf{x}$  satisfy  $\nabla f(\mathbf{x}) = \mathbf{0}$  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  $\mathbf{x}$ , for which  $\nabla f(\mathbf{x}) = \mathbf{0}$ , is a local minima, consider the *Hessian matrix*:

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

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

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

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

## Optimality on Feasible Region Border

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

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

- ▶  $\boldsymbol{\lambda}$  are referred to as the *Lagrange multipliers*.
- ▶ This condition implies that at  $\mathbf{x}^*$ , 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  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})$ , so they satisfy:

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

Seeking  $\boldsymbol{\lambda}^*$  to obtain a function  $k(\mathbf{x}) = \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*)$  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  $\epsilon$ :

$$\epsilon = f(x^* + h) - f(x^*) = \underbrace{f'(x^*)h}_0 + \frac{1}{2}f''(x^*)h^2 + O(h^3)$$

- ▶ *so the function value changes by  $\frac{1}{2}f''(x^*)h^2$ , which implies we need  $h = O(\sqrt{\epsilon})$ ,*
- ▶ *a perturbation to the function value in the  $k$ th significant digit, could result in the solution changing in the  $k/2$ th significant digit.*



# Golden Section Search

**Demo:** Golden Section Proportions

**Activity:** Golden Section Search

- ▶ Given bracket  $[a, b]$  with a unique local minimum ( $f$  is *unimodal* on the interval), *golden section search* considers 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 section 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  $\alpha_k$  if  $f(x_{k+1}) > f(x_k)$ .*

- ▶ 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.*

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

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

*such that  $f(\mathbf{x}_{k+1}) = \min_{\alpha_k} f(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k))$ , 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  $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{c}^T \mathbf{x}$  where  $\mathbf{A}$  is symmetric positive definite, consider the error  $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$ :
  - ▶ *We can quantify the error using the norm,  $\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\mathbf{x}^T \mathbf{A}\mathbf{x}}$ , as*

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

- ▶ *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  $\mathbf{A}$ , since*

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

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

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

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

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

*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 of  $\alpha_k$  and  $\beta_k$  at each iteration of an extrapolation method:

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

- ▶ *For SPD quadratic programming problems, conjugate gradient is an optimal 1st order method, converging in  $n$  iterations.*
- ▶ *It implicitly computes Lanczos iteration, searching along  $A$ -orthogonal directions at each step.*
- ▶ *Parallel tangents* implementation of the method proceeds as follows
  1. *Perform a step of steepest descent to generate  $\hat{\mathbf{x}}_k$  from  $\mathbf{x}_k$ .*
  2. *Generate  $\mathbf{x}_{k+1}$  by minimizing over the line passing through  $\mathbf{x}_{k-1}$  and  $\hat{\mathbf{x}}_k$ .*



# Nonlinear Conjugate Gradient

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

# Conjugate Gradient for Quadratic Optimization

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

# Krylov Optimization

- ▶ Conjugate Gradient finds the minimizer of  $f(x) = \frac{1}{2}x^T Ax - b^T x$  within the Krylov subspace of  $A$ :
  - ▶ *It constructs Krylov subspace  $\mathcal{K}_k(A, b) = \text{span}(b, Ab, \dots, A^{k-1}b)$ .*
  - ▶ *At the  $k$ th step conjugate gradient yields iterate*

$$x_k = \|b\|_2 Q_k T_k^{-1} e_1,$$

where  $Q_k$  are the Lanczos vectors associated with  $\mathcal{K}_k(A, b)$  and  $T_k = Q_k^T A Q_k$ .

- ▶ *This choice of  $x_k$  minimizes  $f(x)$  since*

$$\begin{aligned} \min_{x \in \mathcal{K}_k(A, b)} f(x) &= \min_{y \in \mathbb{R}^k} f(Q_k y) \\ &= \min_{y \in \mathbb{R}^k} y^T Q_k^T A Q_k y - b^T Q_k y \\ &= \min_{y \in \mathbb{R}^k} y^T T_k y - \|b\|_2 e_1^T y \end{aligned}$$

*is minimized by  $y = \|b\|_2^{-1} e_1$ .*

## Newton's Method

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

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

*The minima of this function can be determined by identifying critical points*

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

*thus to determine  $\mathbf{s}$  we solve the linear system,*

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

*Assuming invertibility of the Hessian, we can write the Newton's method iteration as*

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

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

## Quasi-Newton Methods

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

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

where  $\alpha_k$  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  $\mathbf{B}_k$  using  $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$  and  $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ :*

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

- ▶ *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  $f_{\mathbf{x}}(t)$  so that  $f_{\mathbf{x}}(t_i) \approx y_i$ :  
*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  $\mathbf{r}(\mathbf{x})$  so that  $r_i(\mathbf{x}) = y_i - f_{\mathbf{x}}(t_i)$  and minimize*

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

*Now the gradient is  $\nabla \phi(\mathbf{x}) = \mathbf{J}_{\mathbf{r}}^T(\mathbf{x}) \mathbf{r}(\mathbf{x})$  and the Hessian is*

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

# Gauss-Newton Method

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

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

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

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

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

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

*Recognizing the normal equations, we interpret the Gauss-Newton method as solving linear least squares problems  $\mathbf{J}_r(\mathbf{x}_k)\mathbf{s}_k \cong \mathbf{r}(\mathbf{x}_k)$ ,  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$ .*

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

# Constrained Optimization Problems

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

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

*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 constraints 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  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})$ ,

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

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

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

where

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

# 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
  - ▶ First, any minima  $x^*$  must be a feasible point, so  $g(x^*) = 0$  and  $h(x^*) \leq 0$ .
  - ▶ We say the  $i$ th inequality constraint is *active* at a minima  $x^*$  if  $h_i(x^*) = 0$ .
  - ▶ The collection of equality constraints and active inequality constraints  $q^*$ , satisfies  $q^*(x^*) = 0$ .
  - ▶ The negative gradient of the objective function at the minima must be in the row span of the Jacobian of this collection of constraints:

$$-\nabla f(x^*) = J_{q^*}^T(x^*)\lambda^* \quad \text{where } \lambda^* \text{ are Lagrange multipliers of constraints in } q^*.$$

- ▶ To use SQP for an inequality constrained optimization problem, consider at each iteration an *active set* of constraints:
  - ▶ Active set  $q_k$  contains all equality constraints and all inequality constraints that are exactly satisfied or violated at  $x_k$ .
  - ▶ Perform one step of Newton's method to minimize  $\mathcal{L}_k(x, \lambda) = f(x) + \lambda^T q_k(x)$  with respect to  $x$  and  $\lambda$ , 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  $\mathbf{g}(\mathbf{x}) = 0$ :

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

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

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

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

# Barrier Functions

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

- ▶ *Provided we start at a feasible point, modify objective function so it diverges to  $\infty$  when approaching border of feasible region.*
- ▶ *Inverse barrier function:*

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

- ▶ *Logarithmic barrier function:*

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

- ▶ *When using sufficiently small steps, we have  $\mathbf{x}_{\mu}^* \rightarrow \mathbf{x}^*$  as  $\mu \rightarrow 0$ .*
- ▶ *Barrier and penalty methods solve a sequence (for different values of  $\rho$  or  $\mu$ ) of unconstrained problems, requiring multiple executions of e.g., Newton's method*