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 \( f \in \mathbb{R}^n \to \mathbb{R} \) 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**
    \[
    \forall x, y \in S, \quad \alpha \in [0, 1], \quad \alpha x + (1 - \alpha)y \in S.
    \]

  - **Function $f$ is convex if**
    \[
    f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y).
    \]

- 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:

\[
L(z) = \{ x : f(x) = z \}
\]
\[
S(z) = \{ x : f(x) \leq z \}
\]

- 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,

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

- 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 $x$, for which $\nabla f(x) = 0$, is a local minima, consider the Hessian matrix:

$$
H_f(x) = J_{\nabla f}(x) =
\begin{bmatrix}
\frac{d^2 f}{dx_1^2}(x) & \cdots & \frac{d^2 f}{dx_1 dx_n}(x) \\
\vdots & \ddots & \vdots \\
\frac{d^2 f}{dx_n dx_1}(x) & \cdots & \frac{d^2 f}{dx_n^2}(x)
\end{bmatrix}
$$

The Hessian matrix is always symmetric.

If $x^*$ is a minima of $f$, then $H_f(x^*)$ is positive semi-definite:

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

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

\[
\exists \lambda \in \mathbb{R}^n, \quad -\nabla f(x^*) = J_g^T(x^*)\lambda
\]

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

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

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

$$\epsilon = f(x^* + h) - f(x^*) = f'(x^*)h + 0 + O(h^2)$$

▶ 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

- Given bracket \([a, b]\) with a unique 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 \( f(x_{k+1}) > f(x_k) \).
General Multidimensional Optimization

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

*Form a \(n\)-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:

\[
x_{k+1} = x_k - \alpha_k \nabla f(x_k)
\]

such that \(f(x_{k+1}) = \min_{\alpha_k} f(x_k - \alpha_k \nabla f(x_k))\), 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 \( f(x) = \frac{1}{2} x^T A x + c^T x \) where \( A \) is symmetric positive definite, the error \( e_k = x_k - x^* \) satisfies
  \[
  \|e_{k+1}\|_A = e_{k+1}^T A e_{k+1} = \frac{\sigma_{\text{max}}(A) - \sigma_{\text{min}}(A)}{\sigma_{\text{max}}(A) + \sigma_{\text{min}}(A)} \|e_k\|_A
  \]

- 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 \( A \), since
  \[
  \frac{\sigma_{\text{max}}(A) - \sigma_{\text{min}}(A)}{\sigma_{\text{max}}(A) + \sigma_{\text{min}}(A)} = \frac{\kappa(A) - 1}{\kappa(A) + 1}.
  \]
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 $x_k - x_{k-1}$):

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta_k (x_k - x_{k-1})$$

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

$$\|e_{k+1}\|_A = \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \|e_k\|_A$$

*Nesterov’s gradient optimization method is another instance of an extrapolation method that provides further improved optimality guarantees.*
The conjugate gradient method is capable of making the optimal choice of $\alpha_k$ and $\beta_k$ at each iteration of an extrapolation method:

$$(\alpha_k, \beta_k) = \arg\min_{\alpha_k, \beta_k} \left[ f\left( x_k - \alpha_k \nabla f(x_k) + \beta_k (x_k - x_{k-1}) \right) \right]$$

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 $\hat{x}_k$ from $x_k$.
2. Generate $x_{k+1}$ by minimizing over the line passing through $x_{k-1}$ and $\hat{x}_k$. 

Demo: Conjugate Gradient Method
Krylov Optimization

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

- **It constructs Krylov subspace** \( K_k(A, c) = \text{span}(c, Ac, \ldots, A^{r-1}c) \).
- **At the \( k \)th step conjugate gradient yields iterate**

\[
x_k = -\|c\|_2 Q_k T_k^{-1} e_1,
\]

where \( Q_k \) are the Lanczos vectors associated with \( K_k(A, c) \) and \( T_k = Q_k^T A Q_k \).

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

\[
\min_{x \in K_k(A, c)} f(x) = \min_{y \in \mathbb{R}^k} f(Q_k y) \\
= \min_{y \in \mathbb{R}^k} y^T Q_k^T A Q_k y + c^T Q_k y \\
= \min_{y \in \mathbb{R}^k} y^T T_k y + \|c\|_2 e_1^T y
\]

is minimized by \( y = -\|c\|_2 T_k^{-1} e_1 \).
Newton’s Method

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

$$f(x_k + s) \approx \hat{f}(s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T H_f(x_k) s.$$ 

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

$$0 = \nabla \hat{f}(s) = \nabla f(x_k) + H_f(x_k)s,$$

thus to determine $s$ we solve the linear system,

$$H_f(x_k)s = -\nabla f(x_k).$$

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

$$x_{k+1} = x_k - H_f(x_k)^{-1} \nabla f(x_k),$$

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

Demo: Newton’s Method in $n$ dimensions
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 it’s 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 $f_x(t)$ so that $f_x(t_i) \approx y_i$:

*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 $r(x)$ so that $r_i(x) = y_i - f_x(t_i)$ and minimize*

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

*Now the gradient is $\nabla \phi(x) = J_r^T(x) r(x)$ and the Hessian is*

$$
H_{\phi}(x) = J_r^T(x) J_r(x) + \sum_{i=1}^{m} r_i(x) H_{r_i}(x).
$$
The Hessian for nonlinear least squares problems has the form:

\[
H_\phi(x) = J_r^T(x)J_r(x) + \sum_{i=1}^{m} r_i(x)H_{r_i}(x).
\]

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

\[
H_\phi(x) \approx \hat{H}_\phi(x) = J_r^T(x)J_r(x).
\]

The \textit{Gauss-Newton} method is Newton iteration with an approximate Hessian:

\[
x_{k+1} = x_k - \hat{H}_\phi(x_k)^{-1}\nabla f(x_k) = x_k - (J_r^T(x_k)J_r(x_k))^{-1}J_r^T(x_k)r(x_k).
\]

Recognizing the normal equations, we interpret the Gauss-Newton method as solving linear least squares problems

\[
J_r(x_k)s_k \approx r(x_k), \quad x_{k+1} = x_k + s_k.
\]

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

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

- At each iteration, SQP computes $[x_{k+1} \quad \lambda_{k+1}] = [x_k \quad \lambda_k] + [s_k \quad \delta_k]$ by solving

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

where

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

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

- *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:*

\[
-\nabla f(x^*) = J^T_{q^*}(x^*)\lambda^* \quad \text{where } \lambda^* \text{ are Lagrange multipliers of constraints in } q^*. 
\]

- 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 \( L_k(x, \lambda) = f(x) + \lambda^T q_k(x) \) 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 $g(x) = 0$:

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

  is a simple merit function, and its solutions $x_\rho^*$ satisfy $\lim_{\rho \to \infty} x_\rho^* = x^*$. 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:

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

- **Barrier functions** (*interior point methods*) provide an effective way of working with inequality constraints $h(x) \leq 0$:
  - Provided we start at a feasible point, modify objective function so it diverges to $\infty$ when approaching border of feasible region.
  - **Inverse barrier function**:
    \[ \phi_{\mu}(x) = f(x) - \mu \sum_{i=1}^{m} \frac{1}{h_i(x)}. \]
  - **Logarithmic barrier function**:
    \[ \phi_{\mu}(x) = f(x) - \mu \sum_{i=1}^{m} \log(-h_i(x)). \]
  - When using sufficiently small steps, we have $x^*_\mu \to x^*$ as $\mu \to 0$. 