10.34 Numerical Methods Applied to Chemical Engineering Fall 2015

Quiz #1 Review

Study guide based on notes developed by J.A. Paulson, modified by K. Severson

Linear Algebra

We've covered three major topics in linear algebra: characterizing matrices, linear equations, and eigenvalue problems. We'll start with some preliminaries for notation in linear algebra and then review each of these topics.

Vectors

1. A vector is an ordered set of numbers e.g. $\mathbf{x} = [x_1, \dots x_N]$
2. Transpose operator

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{x}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}.$$

3. Inner product given by $\mathbf{x}^T\mathbf{y} = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{N} x_i y_i$. Only works for same size vectors and the result is a scalar.
4. Outer product (dyadic) given by $\mathbf{x}\mathbf{y}^T = \mathbf{x} \otimes \mathbf{y}$. Can have any size vectors and the result is a matrix.

Matrices

1. A matrix has ordered sets of numbers, e.g. $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}$. A matrix represents a transformation. It can also be thought of as a map between vector spaces.
2. Given $\mathbf{C} = \mathbf{A}^T$, the elements of $\mathbf{C}$ are $C_{ij} = A_{ji}$.
3. Trace is denoted Tr $\mathbf{A} = \sum_{i=1}^{N} A_{ii}$ (sum of diagonals). Only valid for square matrices!
4. Matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$. For $\mathbf{A} \in \mathbb{R}^{N \times M}$, this scales as $O(NM)$. Column-view,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1M} \\ A_{21} & A_{22} & \cdots & A_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NM} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix} = x_1 \begin{bmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{N1} \end{bmatrix} + x_2 \begin{bmatrix} A_{12} \\ A_{22} \\ \vdots \\ A_{N2} \end{bmatrix} + \cdots + x_M \begin{bmatrix} A_{1M} \\ A_{2M} \\ \vdots \\ A_{NM} \end{bmatrix}$$

5. Matrix-matrix product $\mathbf{C} = \mathbf{AB}$. For $\mathbf{A} \in \mathbb{R}^{N \times M}$ and $\mathbf{B} \in \mathbb{R}^{M \times P}$, this scales as $O(NMP)$. Remember properties in Lecture 1 slide 26.

6. Matrix inverse $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. Exists if $\mathbf{A}$ is non-singular.

7. Determinant of a matrix $\det(\mathbf{A})$ can be defined in terms of its minors,

$$\det(\mathbf{A}) = \sum_{i=1}^{N} (-1)^{i+j} \underbrace{A_{ij}}_{\text{cofactor}} \underbrace{M_{ij}(\mathbf{A})}_{\text{minor}}$$

where the minor $M_{ij}(\mathbf{A})$ is simply the matrix $\mathbf{A}$ with the i^{th} row and j^{th} column removed. When $\mathbf{A} \in \mathbb{R}^{N \times N}$, then $M_{ij}(\mathbf{A}) \in \mathbb{R}^{N-1 \times N-1}$. Only valid for square matrices! Remember determinant properties e.g., $\det(\mathbf{A}^T) = \det(\mathbf{A})$ and $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$.

Characterizing vectors and matrices

1. Vector norms

   (a) A vector norm maps a vector to a scalar and has three properties:
   $\|\mathbf{x}\| \geq 0$, $\|\mathbf{x}\| = 0$ iff $\mathbf{x} = 0$
   $\|\mathbf{x}+\mathbf{x}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ Note that this property is referred to as the triangle inequality.
   $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ where $\alpha \in \mathbb{C}$

   (b) The most commonly used vector norms in our class are p-norms: $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p}$ where $p \geq 1$.

2. Matrix norms

   (a) A matrix norm also maps a matrix to a scalar.
   (b) In class we discussed the induced norm
   $\|\mathbf{A}\|_p = \max_{\mathbf{x} \neq 0} \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p}$
   (c) The induced norm can be interpreted as "how much can can $\mathbf{A}$ stretch $\mathbf{x}$"
   (d) We looked at several specific induced norms:
   $\|\mathbf{A}\|_2 = \max \sigma_i$ where σ_i are the singular values of $\mathbf{A}$
   $\|\mathbf{A}\|_\infty = \max \sum_{j=1}^{M} |a_{ij}|$
   $\|\mathbf{A}\|_1 = \max \sum_{i=1}^{N} |a_{ij}|$
   (e) We also used several properties of matrix norms:
   Cauchy-Schwarz: $\|\mathbf{AB}\|_p \leq \|\mathbf{A}\|_p \|\mathbf{B}\|_p$.
   Triangle inequality: $\|\mathbf{A} + \mathbf{B}\|_p \leq \|\mathbf{A}\|_p + \|\mathbf{B}\|_p$.

3. Condition Number

   (a) The condition number, $\kappa(\mathbf{A})$, is a measure of how numerical error is magnified in solutions of linear equations; $\log_{10}\kappa(\mathbf{A})$ give the number of digits lost.
   (b) For invertible matrices, $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$. In the 2-norm, $\kappa(\mathbf{A}) = \frac{\sigma_{max}}{\sigma_{min}}$
   (c) For a unitary matrix, $\mathbf{Q}$, $\kappa(\mathbf{Q}) = 1$.

4. Singular Matrices A matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$, that is nonsingular is invertible. There are many equivalent descriptions of a nonsingular matrix, some of which are listed below.

   (a) $\text{rank}(\mathbf{A}) = N$ e.g. the matrix has full rank

   (b) $\det(\mathbf{A}) \neq 0$

   (c) 0 is not an eigenvalue of $\mathbf{A}$

   (d) 0 is not a singular value of $\mathbf{A}$

   (e) $\mathbf{A}$ has a trivial null space

   (f) The dimension of the range space of $\mathbf{A}$ is N

   (g) The columns of $\mathbf{A}$ of linearly independent

5. Vector spaces and the four fundamental subspaces

   (a) A vector space is a "special" set of vectors that satisfy the following properties:
   Closed under addition: $\mathbf{x}, \mathbf{y} \in S \implies \mathbf{x} + \mathbf{y} \in S$.
   Closed under scalar multiplication: $\mathbf{x} \in S \implies c\mathbf{x} \in S$.
   Contains the null vector: $\mathbf{0}$.
   Has an additive inverse: $\mathbf{x} \in S \implies -\mathbf{x} \in S : \mathbf{x} + (-\mathbf{x}) = 0$.

   (b) Column/range space: $\mathcal{R}(\mathbf{A}) = \text{span}\{\mathbf{A}_1^c, \mathbf{A}_2^c, \ldots, \mathbf{A}_M^c\} = \left\{ \sum_{i=1}^{M} \lambda_i \mathbf{A}_i^c \;\; | \;\; \lambda_i \in \mathbb{R}, \;\; \mathbf{A}_i^c \in \mathbb{R}^N \right\}$.

   (c) Null space: $\mathcal{N}(\mathbf{A}) = \left\{ \mathbf{x} \in \mathbb{R}^M : \mathbf{A}\mathbf{x} = \mathbf{0} \right\}$.

   (d) Row space: $\mathcal{R}(\mathbf{A}^T) = \text{span}\{\mathbf{A}_1^r, \mathbf{A}_2^r, \ldots, \mathbf{A}_M^r\} = \left\{ \sum_{i=1}^{N} \lambda_i \mathbf{A}_i^r \;\; | \;\; \lambda_i \in \mathbb{R}, \;\; \mathbf{A}_i^r \in \mathbb{R}^M \right\}$.

   (e) Left null space: $\mathcal{N}(\mathbf{A}^T) = \left\{ \mathbf{x} \in \mathbb{R}^N : \mathbf{A}^T\mathbf{x} = \mathbf{0} \right\}$.

   (f) The rank of a matrix is the dimension of its column space, $r = \dim \mathcal{R}(\mathbf{A})$, $r \leq \min(N, M)$.

   (g) The rank nullity theorem states, $\dim \mathcal{N}(\mathbf{A}) = M - r$

Linear equations, $\mathbf{Ax} = \mathbf{b}$

We spent a lot of time analyzing the linear equation, $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{N \times M}$, $\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{b} \in \mathbb{R}^N$.

1. Existence and uniqueness of solutions

   (a) A solution exists iff $\mathbf{b} \in \mathcal{R}(\mathbf{A})$

   (b) A solution is unique iff $\dim \mathcal{N}(\mathbf{A}) = 0$

   (c) From our list of nonsingular synonyms, we know this means that the matrix is square and invertible if the solution exists and is unique.

2. Gaussian elimination transforms a matrix into its upper triangular form. It takes $O(N^3)$ operations. Pivoting of the rows may be required to avoid pivots equaling zero or to help with numerical stability. The upper triangular system is solved using back substitution which takes $O(N^2)$ operations.

   (a) When applying Gaussian elimination to sparse systems, we want to try and exploit the sparsity pattern. At a minimum, we can use a special storage matrix to decrease the memory requires (this was what we did in HW1). However, applying Gaussian elimination will lead to fill-in, which is undesirable. One way to avoid fill-in is to re-order the matrix by using permutations. Note that this is a type of preconditioning. A permutation matrix looks like

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

   (b) Eventually, N is too large and Gaussian elimination is not reasonable. At this point, you will consider an iterative method such as Jacobi iterations or Gauss-Seidel iterations. The idea of these methods is to turn a hard problem into many easier problems.

      i. Jacobi iterations split $\mathbf{A}$ in $\mathbf{D}+\mathbf{R}$ where $\mathbf{D}$ are the diagonal elements and $\mathbf{R}$ are the off-diagonal elements. Jacobi iterations work well for diagonally dominant systems, i.e. $\|\mathbf{D}^{-1}\mathbf{R}\| < 1$.

      ii. Gauss-Seidel iterations split $\mathbf{A}$ in $\mathbf{L}+\mathbf{U}$ where $\mathbf{L}$ are the lower triangular elements and $\mathbf{U}$ are the upper triangular elements (excluding the diagonal). Gauss-Seidel converges if $\|\mathbf{L}^{-1}\mathbf{U}\| < 1$.

3. Singular Value Decomposition

   (a) We can write $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger$ where $\mathbf{V}^\dagger = \bar{\mathbf{V}}^T$ denotes the conjugate transpose. Here $\mathbf{\Sigma}$ is a diagonal matrix with elements $\Sigma_{ii}^2 = \lambda_i(\mathbf{A}^\dagger\mathbf{A})$, $\mathbf{V}$ is a matrix whose columns contain the eigenvectors of $\mathbf{A}^\dagger\mathbf{A}$, and $\mathbf{U}$ is a matrix whose columns contain the eigenvectors of $\mathbf{A}\mathbf{A}^\dagger$.

   (b) $\sigma_i = \Sigma_{ii}$ are the singular values of $\mathbf{A}$.

   (c) Like an eigendecomposition for non-square matrices $\mathbf{A} \in \mathbb{R}^{N\times M}$.

   (d) Can be useful for things such as data compression/matrix approximation.

Eigenvalue problems, $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

The eigenvectors of a matrix are special vectors that are "stretched" on multiplication by the matrix. They solve the equation $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{v} \in \mathbf{C}^N$ and $\lambda \in \mathbb{C}$. This is a nonlinear system of equations with N equations and $N+1$ unknowns, therefore eigenvectors are not unique.

1. To solve the eigenvalue problem, we solve $\det(\mathbf{A} - \lambda\mathbf{I}) = 0 = p^N(\lambda)$ where $p^N(\lambda)$ is the characteristic polynomial. The roots of this polynomial are the eigenvalues.

2. Complex eigenvalues also appear in conjugate pairs.

3. For a diagonal or triangular matrix, the eigenvalues are the diagonal elements.

4. The algebraic multiplicity of an eigenvalue is the number of times it is repeated.

5. The geometric multiplicity of an eigenvalue is the number of linearly independent eigenvectors that correspond to the eigenvalue, i.e. $\dim \mathcal{N}(\mathbf{A}-\lambda\mathbf{I})$. If the eigenvalue is unique, there is only one corresponding eigenvector but if algebraic multiplicity is M, $1 \leq \dim \mathcal{N}(\mathbf{A}-\lambda\mathbf{I}) \leq M$.

6. $\mathbf{A}$ can be written $\mathbf{AV} = \mathbf{V\Lambda}$

$$\mathbf{A}\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N\end{bmatrix} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N\end{bmatrix}\begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N\end{bmatrix}$$

   If $\mathbf{A}$ has a complete set of eigenvectors, $\mathbf{V}$ is invertible and $\mathbf{A}$ is diagonalizable, $\mathbf{A} = \mathbf{V\Lambda V}^{-1}$.

7. If $\mathbf{A}$ is real and symmetric, it has a complete set of orthonormal eigenvectors and the matrix of eigenvectors is unitary.

Systems of Nonlinear Equations

We've really already seen this before because we solved eigenvalue problems however that was only for polynomials. Let's consider a general system of nonlinear equations.

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

where $\mathbf{f} : \mathbb{R}^N \mapsto \mathbb{R}^N$ and $\mathbf{x} \in \mathbb{R}^N$. Given some $\mathbf{f}(\mathbf{x})$, we want to find roots $\mathbf{x}^\star$ such that $\mathbf{f}(\mathbf{x}^\star) = \mathbf{0}$. There could no solutions, one locally unique solution, many locally unique solutions or infinite solutions.

1. We will assume that in the neighborhood close to a root, the function is approximately linear. Linearizing a general nonlinear function around $\mathbf{x}_0$ using the Taylor series method gives,

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) + O(\|\mathbf{x}-\mathbf{x}_0\|_2^2)$$

   where $\mathbf{J}(x)$ is the Jacobian and is defined as

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N}\end{bmatrix}$$

As analytic forms of the Jacobian $\mathbf{J}(\mathbf{x})$ are difficult (or impossible) to come by, we might have to turn to an alternative method for approximating $\mathbf{J}(\mathbf{x})$. One simple method is "finite differencing" such that the columns of the Jacobian can be evaluated as,

$$\mathbf{J}_j^C(\mathbf{x}) = \frac{\mathbf{f}(\mathbf{x} + \epsilon \mathbf{e}_j) - \mathbf{f}(\mathbf{x})}{\epsilon}$$

where $\mathbf{e}_j = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T$ is all zeros except for a 1 in the j^{th} element. This requires $2N$ function evaluations to compute (2 for the difference in the numerator and N for each column).

2. Convergence criteria are needed to determine when to stop.

   (a) $\|\mathbf{f}(\mathbf{x}_{i+1})\|_p \leq \epsilon$

   (b) $\|\mathbf{x}_{i+1} - \mathbf{x}_i\|_p \leq \epsilon_R \|\mathbf{x}_{i+1}\|_p + \epsilon_A$

   (c) The relative tolerance will dominate when the norm of $\mathbf{x}$ is large while the absolute tolerance will dominant when the norm of $\mathbf{x}$ is small.

3. The rate of convergence can be examined based on,

$$\lim_{k \to \infty} \frac{\|\mathbf{x}_{i+1} - \mathbf{x}^\star\|_p}{\|\mathbf{x}_i - \mathbf{x}^\star\|_p^q} = C$$

   $q = 1$ and $0 < C < 1$ is linear convergence, $q > 1$ (or $q = 1$ and $C = 0$) is super-linear convergence, and $q = 2$ is quadratic convergence.

4. Newton-Raphson method

   (a) The iterative map is,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - [\mathbf{J}(\mathbf{x}_i)]^{-1} \mathbf{f}(\mathbf{x}_i)$$

   $\mathbf{J}(\mathbf{x}_i)$ is not actually inverted! You solve a linear system of equations and should take advantage of sparsity when possible.

   (b) Newton-Raphson converges quadratically when the Jacobian is non-singular and quadratic convergence is guaranteed only when the iterates are sufficiently near the root. This means good initial guesses are essential to the success of Netwon-Raphson. Bad initial guesses can lead to very chaotic behavior in your iterates and may not converge at all.

   (c) Broyden's method is a special case of Newton-Raphson in which the secant method is used to develop a coarse approximation of the derivative. This method uses rank-1 approximation for the Jacobian,

$$\mathbf{J}(\mathbf{x}_i) = \mathbf{J}(\mathbf{x}_i) + \frac{\mathbf{f}(\mathbf{x}_i)(\mathbf{x}_i - x_{i-1})^T}{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|_2^2}$$

   This method is called rank-one because the outer product of two vectors, $\mathbf{f}(\mathbf{x}_i)$ and $(\mathbf{x}_i - \mathbf{x}_{i-1})$, has rank of one. In fact, every column of $\mathbf{f}(\mathbf{x}_i)(\mathbf{x}_i - \mathbf{x}_{i-1})^T$ is a scalar multiple of $\mathbf{f}(\mathbf{x}_i)$. This method allows us to generate an iterative formula for the Jacobian inverse,

$$\mathbf{J}(\mathbf{x}_i)^{-1} = \mathbf{J}(\mathbf{x}_{i-1})^{-1} - \frac{\mathbf{J}(\mathbf{x}_{i-1})^{-1} \mathbf{f}(\mathbf{x}_i)(\mathbf{x}_i - \mathbf{x}_{i-1})^T \mathbf{J}(\mathbf{x}_{i-1})^{-1}}{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2^2 + (\mathbf{x}_i - \mathbf{x}_{i-1})^T \mathbf{J}(\mathbf{x}_{i-1})^{-1} \mathbf{f}(\mathbf{x}_i)}.$$

   that saves us computation time at the cost of accuracy.

(d) Damped Newton-Raphson introduces a scaling factor to the NR iterative map,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha[\mathbf{J}(\mathbf{x}_i)]^{-1}\mathbf{f}(\mathbf{x}_i)$$

where $\alpha = \underset{0\leq\alpha\leq 1}{\arg\min} \|\mathbf{f}(\mathbf{x}_i - \alpha[\mathbf{J}(\mathbf{x}_i)]^{-1}\mathbf{f}(\mathbf{x}_i))\|_p$ (find the optimal α value that decreases the function value as much as possible in a single step). Finding this damping factor is as hard as finding the root. An approximate method is to use a line search that continually reduces α by 2 until the step decreases the function value. The damped Newton-Raphson is globally convergent but may converge to roots or local minima/maxima.

5. Continuation, Homotopy, and Bifurcation

(a) Continuation transforms the hard problem we want to solve to an easier one by introducing/varying a parameter. Knowing that you found all the roots is not always easy/possible.

(b) Homotopy is the transformation from one problem to another. We seek the roots $\mathbf{x}^\star(\lambda)$ to the following system of equations,

$$\mathbf{h}(\mathbf{x}, \lambda) = \lambda\mathbf{f}(\mathbf{x}) + (1-\lambda)\mathbf{g}(\mathbf{x})$$

where $\mathbf{h}(\mathbf{x}, 0) = \mathbf{g}(\mathbf{x})$ such that $\mathbf{x}^\star(0)$ are the roots of $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x}, 1) = \mathbf{f}(\mathbf{x})$ such that $\mathbf{x}^\star(1)$ are the roots of $\mathbf{f}(\mathbf{x})$. We create a smooth transition from $\mathbf{g}(\mathbf{x})$ to $\mathbf{f}(\mathbf{x})$ by varying λ in small discrete increments $\{\lambda_i\}$ from 0 to 1 where the solution $\mathbf{x}^\star(\lambda_i)$ is used as the initial guess for obtaining $\mathbf{x}^\star(\lambda_{i+1})$.

(c) During the homotopy procedure, the Jacobian of $\mathbf{h}(\mathbf{x}^\star, \lambda)$ at some λ, denoted $\mathbf{J}_h(\mathbf{x}^\star(\lambda), \lambda)$, can become singular such that the $\det(\mathbf{J}_h(\mathbf{x}^\star(\lambda), \lambda)) = 0$. This can be indicative of two phenomena: turning points and bifurcations. A turning point is when the solution branch begins to curve back such that the branch was being traced with increasing λ suddenly needs to be traced with a decreasing λ. One way to account for this is to parameterize the roots and homotopy parameter in terms of distance s traveled along the solution curve/branch i.e., $\mathbf{x}^\star(\lambda(s))$ and $\lambda(s)$. We can use the arclength constraint to determine how to change the homotopy parameter,

$$\left\| \frac{d}{ds}\mathbf{x}^\star(\lambda(s)) \right\|_2^2 + \left(\frac{d}{ds}\lambda(s) \right)^2 = 1$$

Note that the change in the parameter $\frac{d}{ds}\lambda(s)$ appears as squared so we need some policy for determining what sign (positive/increasing or negative/decreasing) to take.

(d) A bifurcation is when additional solutions appear continuously at some λ, i.e., a problem switches from having 1 solution to many solutions as λ is varied. This happens at a λ when $\det(\mathbf{J}_h(\mathbf{x}^\star(\lambda), \lambda)) = 0$.

Optimization

Optimization problems consider

$$\min_{\mathbf{x}\in D} \ f(\mathbf{x}) \qquad\qquad \underset{\mathbf{x}\in D}{\operatorname{argmin}} \ f(\mathbf{x})$$

where $f(x)$ is the objective function, x are the "design alternatives" and D is the feasible set.

1. Maximizing $f(\mathbf{x})$ is equivalent to minimizing $-f(\mathbf{x})$ so we only need to look at one (commonly minimization).

2. The goal is to find $\mathbf{x}^\star \in D$ such that $f(\mathbf{x}^\star) < f(\mathbf{x})$ for all $\mathbf{x} \in D$. Solution is not necessarily unique as there could be multiple $x^\star$ in D. If $f(x)$ is convex, it has a single global minimum.

3. If D is a closed set, the problem of finding the minimum is called constrained optimization. If D is an open set $\mathbb{R}^N$, the problem of finding the minimum is called unconstrained optimization ($\mathbf{x}$ can take on any values in $\mathbb{R}^N$; no restriction).

4. Unconstrained optimization problems have a critical point when the gradient $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{0}$ (assuming the function is differentiable).

5. For a point to be a minimum, all the eigenvalues of the Hessian at the minimum $\mathbf{H}(\mathbf{x}^\star)$ must be positive. If the eigenvalues are all negative, $\mathbf{x}^\star$ is a local maximum. If the eigenvalues are both positive and negative, then $\mathbf{x}^\star$ is a saddle point. If none of these are satisfied, the test is inconclusive and higher derivatives must be checked to characterize the critical point.

6. Steepest Descent

   (a) The direction of steepest descent is: $\mathbf{d}_i = -\mathbf{g}(\mathbf{x}_i)$. This makes you go downhill fastest.

   (b) Iterative map: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha_i \mathbf{g}(\mathbf{x}_i)$.

   (c) For small values of α_i, the iterates continue to reduce the function value until $\mathbf{g}(\mathbf{x}_i) = \mathbf{0}$ (near zero within a norm tolerance).

   (d) Ideally, we would pick α_i to lead to the smallest value of $f(\mathbf{x}_{i+1})$ but this is its own optimization that is not easy. We can estimate an optimal α_i using a Taylor series expansion of $f(\mathbf{x})$ about $\mathbf{x}_i$ evaluated at $\mathbf{x}_{i+1}$,

   $$\alpha_i = \frac{\mathbf{g}(\mathbf{x}_i)^T \mathbf{g}(\mathbf{x}_i)}{\mathbf{g}(\mathbf{x}_i)^T \mathbf{H}(\mathbf{x}_i) \mathbf{g}(\mathbf{x}_i)}$$

   This will be the exact optimal α_i if f is quadratic with respect to $\mathbf{x}_i$.

   (e) Converges to local minima and saddle points so need to check Hessian to be sure critical point is minima.

7. Conjugate Gradient Method

   (a) Considers the following minimization

   $$\min_{\mathbf{x}} \; f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T \mathbf{x} = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

   which can be derived analytically to be $\mathbf{g}(\mathbf{x}^\star) = \mathbf{A}\mathbf{x}^\star - \mathbf{b} = 0$ with $\mathbf{H}(\mathbf{x}) = \mathbf{A}$. Nice way to solve linear equations using optimization instead of direct methods like Gaussian elimination.

   (b) Iterative map: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha_i \mathbf{p}(\mathbf{x}_i)$.

   (c) Chooses descent directions (not necessarily steepest descent), $p_1, p_2, .., p_N$, that are said to be conjugate, i.e. $\mathbf{p}_{i+1}^T \mathbf{A}\mathbf{p}_i = 0$.

   (d) Used to solve linear equations in $O(N)$ operations (only for symmetric positive definite matrices). The actual matrix is never needed as we only need to compute its action on vectors $\mathbf{A}\mathbf{y}$.

(e) More sophisticated variations of the conjugate graident methods exists for non-symmetric matrices (e.g,. biconjugate gradient method) and non-linear equations.

8. Newton-Raphson for Optimization

   (a) Finding local minima in optimization is equivalent to finding the roots of the gradient $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{0}$. We can imagine applying the exact same Newton-Raphson techniques discussed above for the gradient. In this case, the Jacobian of the gradient is the Hessian of the function. The Newton-Raphson map for optimization is then,

$$\mathbf{x}_{i+1} = \mathbf{x}_i - [\mathbf{H}(\mathbf{x}_i)]^{-1}\mathbf{g}(\mathbf{x}_i)$$

   (b) This is locally convergent and the accuracy of the iterates improves quadratically (just as before)!

9. Trust-Region Methods

   (a) Trust-region methods choose the Newton-Raphson direction when the quadratic approximation is good and the steepest descent direction when it is not.

   (b) The size of the trust region radius can be set arbitrarily (MATLAB®starts with 1)
   This radius grows or shrinks depending on which of the two steps we choose. If Newton-Raphson is chosen, GROW the trust-radius when the function was smaller than predicted, otherwise, SHRINK the trust-radius. If steepest descent was chose, keep the trust-radius the same.

   (c) Can use a "dog-leg step" when the steepest descent direction is chosen. Here the steepest decent move is taken and when it is within the trust-radius, an additional step in the Newton-Raphson direction is taken to touch the boundary of the trust-region.

10. Lagrange multipliers

   (a) All of the methods up to this point haven't consider constraints. Lagrange multipliers are one way to handle the problem

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{c}(\mathbf{x}) = 0 \end{aligned}$$

   (b) A solution to this problem satisfies

$$\begin{pmatrix} \mathbf{g}(\mathbf{x}) - \mathbf{J}_c\mathbf{x}^{\mathrm{T}}\lambda \\ \mathbf{c}(\mathbf{x}) = 0 \end{pmatrix} = 0$$

   where λ is a vector of "Lagrange multipliers".

11. Interior point methods

   (a) Our optimization problem could also have inequality constaints

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{h}(\mathbf{x}) \geq 0 \end{aligned}$$

(b) Interior point methods consider the problem

$$\min f(\mathbf{x}) - \mu \sum_{i=1}^{N} \log(h_i(\mathbf{x}))$$

as $\mu \to 0^+$

(c) The log function is chosen as the barrier because it is easy to find the gradient.

(d) The parameter μ can be varied using a homotopy procedure.

Miscellaneous

1. "Big O" Notation for denoting computational complexity
2. Taylor series expansion: $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{6}f'''(a)(x-a)^3 + \cdots$
3. Basic MATLAB syntax

*Disclaimer: This document in not the gospel of numerical. There may be typos and you should always review your notes! Good luck with the exam!

10.34 Numerical Methods Applied to Chemical Engineering

Fall 2015