Course description

The course teaches reliable numerical methods for solving linear equations \((Ax = b)\) and linear and nonlinear optimization problems (LO and NLO), with a focus on sparse matrices as in the conjugate-gradient method, the simplex method, and large-scale constrained optimization.

3 units, 5 homeworks (60%), 1 project (40%), no mid-term or final.

Prerequisites: Basic numerical linear algebra, including LU, QR, and SVD factorizations, and an interest in MATLAB, sparse-matrix methods, and gradient-based algorithms for constrained optimization.

Syllabus

1. Review of dense factorizations (LU, QR, EVD, SVD, \(U^TAV = T\))
2. Overview of optimization software (problem types, NEOS, MATLAB, TOMLAB)
3. Class project: using NEOS for challenging LO problems
4. Iterative methods for symmetric \(Ax = b\) (symmetric Lanczos process, CG, SYMMLQ, MINRES, MINRES-QLP)
5. Iterative methods for unsymmetric \(Ax = b\) and least squares (Golub-Kahan process, CGLS, LSQR, LSMR, Craig, Arnoldi process, GMRES)
6. The primal simplex method (phase 1 in practice, basis factorization, updating, crash, scaling, degeneracy)
7. LUSOL: A Basis Factorization Package (the engine for MINOS, SQOPT, SNOPT, MILES, PATH, lp_solve)
8. Basis LU updates (Product-Form, Bartels-Golub, Forrest-Tomlin, Block-LU)
9. Primal-dual interior methods for LO (CPLEX, HOPDM, IPOPT, KNITRO, LOQO, MOSEK) and convex nonlinear objectives (PDCO), Basis Pursuit, BP Denoising (Lasso, LARS, Homotopy, ASP)
10. The reduced-gradient method (MINOS part 1)
11. BCL methods (Augmented Lagrangians, LANCELOT)
12. LCL methods (MINOS part 2, Knossos)
13. SQP methods (NPSOL, SQOPT, SNOPT)
14. SNOPT input/output
1 Aim and notation

We first review matrix factorizations and the elementary triangular and orthogonal matrices that are used to compute them. We also show the existence of orthogonal transformations of symmetric or rectangular matrices to tridiagonal or bidiagonal form. Systems of equations are denoted \(Ax = b\), where \(A\) and \(b\) are always real. Depending on the topic, \(A\) may be symmetric, positive definite, indefinite, or rectangular. If \(A\) is square and nonsingular, its condition number is defined to be \(\text{cond}(A) = \|A\|\|A^{-1}\|\). Wherever possible, we use Householder notation, whereby \(\{A, a, \alpha\}\) refer to \(\{\text{matrix, vector, scalar}\}\).

<table>
<thead>
<tr>
<th>(A)</th>
<th>Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric posdef</td>
<td>(LL^T) or (LDL^T)</td>
</tr>
<tr>
<td>Symmetric indefinite</td>
<td>(LDLT), (D) block-diag with (1 \times 1) and (2 \times 2) blocks</td>
</tr>
<tr>
<td>Square or tall</td>
<td>(LU) or (QR), triangular (L, U, R), orthogonal (Q)</td>
</tr>
<tr>
<td>Symmetric</td>
<td>(VTVT^T), tridiagonal (T)</td>
</tr>
<tr>
<td>Square or tall</td>
<td>(UBVT) or (UTVT^T), bidiagonal (B), tridiagonal (T)</td>
</tr>
</tbody>
</table>

**Exercise** If \(x\) solves \(Ax = b\) and \(x + \Delta x\) solves a perturbed system \((A + \Delta A)(x + \Delta x) = b\), show that 
\[
\|\Delta x\| \|x + \Delta x\| \leq \text{cond}(A) \frac{\|\Delta A\|}{\|A\|}.
\]
Thus, the condition number measures the sensitivity of the solution of \(Ax = b\). This is a property of the problem (not of a method for solving it).

A **stable** method for solving \(Ax = b\) is one that will always give a computed solution \(x + \Delta x\) for which the size of the error \(\|\Delta x\|\) is comparable to the sensitivity of \(x\) (not worse).

1.1 Elementary triangular matrices

Sometimes we make use of \(n \times n\) triangular matrices of the form 
\[
L(\mu) \equiv \begin{bmatrix} 1 & \mu \\ -\mu & 1 \end{bmatrix}, \quad L_k(\mu) \equiv \begin{bmatrix} 1 & \mu_4 & 1 \\ \mu_5 & 1 & 1 \\ \mu_6 & 1 & 1 \end{bmatrix}, \quad k = 3, n = 6,
\]
where \(|\mu| \leq \tau\) and \(|\mu_i| \leq \tau\) for some \(\tau \in [1, 100]\) say. In using such matrices to form an LU factorization of \(A\), it’s easier to think of multiplying \(A\) by matrices \(M(\mu)\) or \(M_k(\mu)\) that are their inverse:
\[
M(\mu) \equiv \begin{bmatrix} 1 & 0 \\ -\mu & 1 \end{bmatrix} = L(\mu)^{-1}, \quad M_k(\mu) \equiv \begin{bmatrix} 1 & 1 \\ -\mu_4 & 1 \\ -\mu_5 & 1 \\ -\mu_6 & 1 \end{bmatrix} = L(\mu)^{-1}.
\]

We can think of multiplying \(A\) by \(M_k(\mu)\) in order to change the first column of \(A\) to zero except for the first entry. Ultimately, \(A = LU\) is equivalent to \(MA = U\), where \(L\) and \(M\) are the product of elementary triangular matrices.

**Example** Beware of the effect of large numbers:
\[
\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -10^6 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ 10^6 & 1 \end{bmatrix} \begin{bmatrix} 10^{-6} & 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 10^{-6} & 1 \\ -10^6 & 1 \end{bmatrix}.
\]

Our aim here (and throughout numerical linear algebra) is to avoid creating large numbers, because floating-point numbers can represent both large and small numbers within a fixed number of bits (64 bits for IEEE double-precision floating-point arithmetic), but the absolute error is larger for larger numbers.
**Cholesky example** If \( A \) is symmetric positive definite (spd), the Cholesky factorization \( A = LL^T \) shows that the \( i \)th row of \( L \) satisfies \( \sum_{j=1}^i L_{ij}^2 = A_{ii} \), so that each \( |L_{ij}| \) is strongly bounded. **Cholesky does not generate large numbers.** It is extremely stable. It’s the best way to test if \( A \) is spd.

**Exercise** Show that \( \text{cond}(L(\mu)) = \text{cond}(M(\mu)) \approx \begin{cases} 1 + |\mu| & \text{if } |\mu| \ll 1, \\ 2.618 & \text{if } |\mu| = 1, \\ |\mu|^2 & \text{if } |\mu| \gg 1. \end{cases} \)

1.2 **LU factorization**

A square unsymmetric system of equations \( Ax = b \) is normally solved by LU factorization of \( A \). This is commonly called Gaussian elimination with X pivoting, where X refers to a “pivot strategy” that makes the method **stable** in the context of numerical computation. Pivot strategies permute the rows and/or columns of \( A \) to ensure that one factor (\( L \) or \( U \)) is “friendly” or **well-conditioned.** This is how we preserve numerical stability. Thus, \( P_1AP_2 = LU \) for some permutations \( P_1 \) and \( P_2 \), where \( L \) and \( U \) are lower and upper triangular and either \( L \) or \( U \) is well-conditioned. Typically, \( L \) has unit diagonals and bounded off-diagonals like \( L_k(\mu) \) above: \( |L_{ij}| \leq \tau \), where we can enforce \( \tau = 1 \) for maximum stability, but \( \tau = 2 \) or 10 or 100 allows attention to sparsity. Such an \( L \) is **likely to be** well-conditioned. The other factor \( U \) is likely to reflect the condition of \( A \).

More generally, \( P_1AP_2 = LDU \), where \( L \) and \( U \) both have unit diagonals, at least one of them has bounded off-diagonals, and the condition of \( A \) is reflected in \( D \). **Partial pivoting** means that the off-diagonals of only \( L \) (or only \( U \)) are bounded. **Rook pivoting** bounds the off-diagonals of both \( L \) and \( U \). **Complete pivoting** has the same effect.

1.3 **Elementary orthogonal matrices**

Plane rotations and Householder transformations take the form

\[
P \equiv \begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix} \in \mathbb{R}^2, \quad H_k \equiv \begin{bmatrix} I_{k-1} & I - \beta_k v_k v_k^T \\ I - \beta_k v_k v_k^T & I_{m-k} \end{bmatrix} \in \mathbb{R}^{m \times m}, \quad 1 \leq k < m,
\]

where \( \gamma^2 + \sigma^2 = 1 \) and \( \beta_k = 2/||v_k||^2 \) (with \( \beta_k = 0 \) if \( v_k = 0 \)). There are no large numbers, and for vectors \( v \) and \( w \) of suitable dimension, \( Pv \) and \( H_kw \) have the same 2-norm as \( v \) and \( w \) because \( P^TP = I \) and \( H_k^2 = I \). For example, if \( H_1 \) reduces the first column of an \( m \times n \) matrix \( A \) (\( m > n \)) to a multiple of \( e_1 \), we have the beginning of the QR factorization \( A = QR \), where \( Q = H_1H_2 \ldots H_n \) and \( R \) is upper triangular. The construction of \( \beta_k \) and the first element of \( v_k \) is intricate to ensure stability, but the remainder of \( v_k \) is proportional to the values that \( H_k \) reduces to zero.

1.4 **QR factorization**

For an \( m \times n \) matrix \( A \) with \( m \geq n \), we can multiply on the left by a product of Householder matrices to reduce \( A \) to upper triangular form (where \( H_n = I \) if \( m = n \)):

\[
Q^T A = R, \quad Q = H_1H_2 \ldots H_n, \quad R \equiv \begin{pmatrix} R_n \\ 0 \end{pmatrix}.
\]

1.5 **Two least-squares problems**

For an \( m \times n \) matrix \( A \) with \( m > n \) and \( m < n \) respectively, two types of least-squares problem are solved by normal equations of the first and second kind:

- **Overdetermined system:** \( \min \|Ax - b\|_2^2 \) \quad \( A^T Ax = A^T b \)
- **Underdetermined system:** \( \min \|x\|_2^2 \) s.t. \( Ax = b \) \quad \( A^T y = b, \quad x = A^T y \)
Assuming $A$ has full rank, we can use the QR factors of $A$ and $A^T$ to obtain $x$:

Overdetermined system:  \[(A \ b) = Q \begin{pmatrix} R & c_1 \\ 0 & c_2 \end{pmatrix}, \quad Rx = c_1\]

Underdetermined system:  \[A^T = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R^Tz = b, \quad x = Q \begin{pmatrix} z \\ 0 \end{pmatrix}\]

### 1.6 Orthogonal tridiagonalization of symmetric $A$

A symmetric $n \times n$ $A$ can be reduced to tridiagonal form by multiplying on the left and right by $H_2$, then $H_3$, ..., then $H_{n-1}$. Note that $A_{11}$ is not altered, and no large numbers are produced. Thus, $V^TAV = T$, where $V = H_2H_3 \ldots H_{n-1}$ and $T$ is tridiagonal. This is the first part of computing the eigenvalue decomposition of symmetric $A$. Note that the first row and column of $V = H_2H_3 \ldots H_{n-1}$ must be the first row and column of $I_n$.

The existence of $V^TAV = T$ for symmetric $A$ means that there exists a transformation

\[
\begin{pmatrix} 1 \\ V^T \end{pmatrix} \begin{pmatrix} 0 & b^T \\ b & A \end{pmatrix} \begin{pmatrix} 1 \\ V \end{pmatrix} = \begin{pmatrix} 0 & \beta_1 e_1^T \\ \beta_1 e_1 & T \end{pmatrix},
\]

where $b$ is a nonzero $n$-vector and $\beta_1 = \|b\|$. We generate this transformation when we apply the Lanczos process to symmetric $n \times n$ $A$ with starting vector $b$.

### 1.7 Orthogonal bidiagonalization of general $A$

For an $m \times n$ matrix $A$ with $m \geq n$, we can multiply on the left and right by different Householder transformations to reduce $A$ to upper-bidiagonal form $B$ (with $H_n = I$ if $m = n$):

\[U^TAV = B, \quad U = H_1H_2 \ldots H_n, \quad V = \bar{H}_2\bar{H}_3 \ldots \bar{H}_n.\]

This is the first part of computing the singular value decomposition of unsymmetric $A$. Note that $H_1$ depends on the first column of $A$, and then $H_2$ depends on the effect of $H_1$. Again, the first row and column of $V$ must be the first row and column of $I_n$.

The existence of $U^TAV = B$ (upper bidiagonal) means there exists a transformation

\[U^T \begin{pmatrix} b \\ A \end{pmatrix} \begin{pmatrix} 1 \\ V \end{pmatrix} = \begin{pmatrix} \beta_1 e_1 \\ B \end{pmatrix},\]

where $b$ is a nonzero $m$-vector, $\beta_1 = \|b\|$, and $B$ is lower bidiagonal. We generate this transformation when we apply the Golub-Kahan process to $m \times n$ $A$ with starting vector $b$.

### 1.8 Orthogonal tridiagonalization of general $A$

Again for an $m \times n$ matrix $A$ with $m \geq n$, we can multiply on the left and right by different Householder transformations to reduce $A$ to tridiagonal form $T$:

\[U^TAV = T, \quad U = H_2H_3 \ldots H_n, \quad V = \bar{H}_2\bar{H}_3 \ldots \bar{H}_n,\]

where $A_{11}$ is not altered. Note that $H_2$ depends on the first column of $A$ and $\bar{H}_2$ depends on the first row of $A$, independent of $H_2$.

The existence of $U^TAV = T$ for general $A$ means that there exists a transformation

\[
\begin{pmatrix} 1 \\ U^T \end{pmatrix} \begin{pmatrix} 0 & c^T \\ b & A \end{pmatrix} \begin{pmatrix} 1 \\ V \end{pmatrix} = \begin{pmatrix} 0 & \gamma_1 e_1^T \\ \gamma_1 e_1 & T \end{pmatrix},
\]

where $b$ and $c$ are nonzero $m$- and $n$-vectors, $\beta_1 = \|b\|$, and $\gamma_1 = \|c\|$. We generate this transformation when we apply the orthogonal tridiagonalization process to $m \times n$ $A$ with starting vectors $b$ and $c$.

The orthogonal reductions in sections 1.6–1.8 are existence proofs for three iterative processes that we use later for solving equations or least-squares problems.
1.9 Further reading

Golub and Van Loan [2] discuss elementary triangular matrices (Gaussian transformations) in chapter 3.2, and elementary orthogonal transformations (Householder reflections and Givens rotations) in chapter 5.1.

The Lanczos process was originally used for computing eigenvalues of symmetric $A$ [3]. Much later it was used by Paige and Saunders [4] for solving symmetric indefinite $Ax = b$.

The Golub-Kahan process was introduced almost incidentally by Golub and Kahan in their paper about computing the singular value decomposition (SVD) of dense matrices [1]. It was used by Paige and Saunders [5, 6] for solving least-squares problems $\min \|Ax - b\|_2$ (and also square $Ax = b$ and underdetermined systems $\min \|x\|_2$ st $Ax = b$).

Orthogonal tridiagonalization was introduced by Saunders, Simon, and Yip [8] for solving square $Ax = b$. Reichel and Ye [7] used the same iterative process later for solving square $Ax = b$ and rectangular $\min \|Ax - b\|_2$.

References


