CME 302: NUMERICAL LINEAR ALGEBRA
FALL 2005/06
LECTURE 0

GENE H. GOLUB

1. What is Numerical Analysis?

In the 1973 edition of the Webster's New Collegiate Dictionary, numerical analysis is defined to be "the study of quantitative approximations to the solutions of mathematical problems including consideration of the errors and bounds to the errors involved." A more concise yet more insightful definition was offered in 1992 by Lloyd Trefethen of Oxford University: "the study of algorithms for the problems of continuous mathematics." This definition conveys the notion that numerical analysis is primarily, though not exclusively, devoted to the development of algorithms for solving continuous problems.

While the foundations of numerical analysis were laid centuries ago through the work of Gauss and Euler, among others, the discipline took on a whole new importance by motivating the development of computers for the purpose of solving problems in ballistics, PDE's and data analysis. Pioneers in this development include Alan M. Turing and John von Neumann.

2. Numerical Linear Algebra

Numerical linear algebra (NLA) is a relatively small area of research, with less than two hundred active participants. However, it is an integral component of numerical analysis, which contributors from a wide variety of disciplines whose ideas are often helpful to research in many others.

In 2000, Jack Dongarra wrote about the ten most influential algorithms in scientific computing. From this list the influence of numerical linear algebra is readily apparent, with some selections lying exclusively within the domain of NLA research, and others strongly connected.

(1) Metropolis Algorithm for Monte Carlo
(2) Simplex Method for Linear Programming
(3) Krylov Subspace Iteration Methods
(4) The Decompositional Approach to Matrix Computations
(5) The Fortran Optimizing Compiler
(6) QR Algorithm for Computing Eigenvalues
(7) Quicksort Algorithm for Sorting
(8) Fast Fourier Transform
(9) Integer Relation Detection
(10) Fast Multipole Method

NLA consists of three basic components:

- development and analysis of numerical algorithms
- perturbation theory, which is used to evaluate the effectiveness of algorithms
- software that implements these algorithms

Date: September 29, 2005, version 1.0.

Notes originally due to James Lambers. Minor editing by Lek-Heng Lim.

3. A Fundamental Problem

The fundamental problem of NLA is the following: Given the system of linear equations

$$A\mathbf{x} = \mathbf{b} + \mathbf{r}$$

where A is an $m \times n$ matrix and $\mathbf{b}$ is a given vector, compute $\mathbf{x}$ such that $\|\mathbf{r}\|$ is minimized.

This problem has several parameters:

- The relationship between m and n. When $m = n$, the system is said to be *square*. When $m > n$, the system is said to be *overdetermined*, and when $m < n$, it is *underdetermined.*
- The *rank* of the matrix A. This parameter, in conjunction with the size of A, determines the uniqueness of the solution.
- Choice of the vector norm $\|\mathbf{r}\|$.
- The structure of A. The *sparsity* of A greatly influences the choice of the method used to solve this problem. Also, problems involving specialized matrices, such as Hankel or Toeplitz matrices, frequently arise in applications.
- The origin of the problem. Often, this context can be helpful in developing an algorithm.

4. Perturbation Theory

Perturbation theory is used to determine bounds on the error in a computed solution to $A\mathbf{x} = \mathbf{b}$. If $\mathbf{x}$ is the exact solution to this system, and y is solution computed using floating-point arithmetic, then we can view $\mathbf{y}$ as the exact solution of the "nearby" system

$$(A + \Delta A)\mathbf{y} = \mathbf{b} + \boldsymbol{\delta}$$

where

$$\frac{\|\Delta A\|}{\|A\|} \leq \epsilon, \quad \frac{\|\boldsymbol{\delta}\|}{\|\mathbf{b}\|} \leq \epsilon, \quad \rho < 1.$$

It can be shown that

$$\frac{\|\mathbf{x} - \mathbf{y}\|}{\|\mathbf{x}\|} \leq \frac{2\epsilon}{1 - \rho}\kappa(A)$$

where

$$\rho = \|\Delta\|\|A^{-1}\| = \|\Delta\|\kappa(A)/\|A\|$$

and

$$\kappa(A) = \|A\|\|A^{-1}\|$$

is the *condition number* of A.

Even if ϵ is small, the computed solution can be useless if $\kappa(A)$ is large, as $\kappa(A)$ is a measure of how an error in the system $A\mathbf{x} = \mathbf{b}$ is amplified in the solution. The relationship between A and $\mathbf{b}$ can give further information as to how much such a perturbation is actually amplified for a particular problem. A detailed theory of condition numbers was provided by John Rice in 1966.

A system $A\mathbf{x} = \mathbf{b}$ where $\kappa(A)$ is large is an example of an *ill-posed* problem. No algorithm, no matter how accurate, will be an effective tool for solving such a problem. It is important to distinguish between ill-posed, or *unstable*, problems from unstable methods. Ensuring that a problem is well-posed is the responsibility of the modeller, who formulates the mathematical problem from the original application. On the other hand, ensuring the stability of an algorithm is the responsibility of the numerical analyst. Informally, a problem or an algorithm is stable if a small change in its input yields a small change in its output. For a problem, the output is the exact solution, whereas for an algorithm, the output is the computed solution.

5. Gaussian Elimination

The initial evaluation of Gaussian elimination by Hotelling did not encourage its use for large-scale problems. However, further analysis by Goldstine and von Neumann, and later Wilkinson, led to a reversal of this view and the widespread use of Gaussian elimination, either directly or as a foundation for other methods for solving linear systems.

The error in Gaussian elimination is the result of the accumulation of round-off errors from each floating-point operation that is performed. The result of each floating-point addition of two numbers is actually the exact sum of two nearby numbers:

$$fl(\mathbf{x}+\mathbf{y}) = \mathbf{x}(1+\epsilon) + \mathbf{y}(1+\epsilon) = \bar{\mathbf{x}} + \bar{\mathbf{y}}.$$

Gaussian elimination with pivoting is equivalent to performing the decomposition

$$\Pi A = LU$$

where Π is a permutation matrix, L is a unit lower triangular matrix, and U is an upper triangular matrix. The algorithm guarantees that

$$\max_{i \geq j} |\ell_{i,j}| = 1$$

and

$$\max_{j \geq i} |u_{i,j}| \leq 2^{n-1}.$$

Wilkinson used *backward error analysis* to show that Gaussian elimination with pivoting to solve $A\mathbf{x} = \mathbf{b}$ is equivalent to solving the perturbed system $(A+E)\mathbf{y} = \mathbf{b}$ where

$$\|E\|_\infty \leq 8n^3 G \|A\|_\infty u + O(u^2),$$

$$|u_{i,j}| \leq G$$

and u is the machine roundoff unit. Using the analysis repeated earlier, it can then be shown that the error can be bounded.

6. The Simplex Algorithm

Wilkinson's work enabled development of algorithms for many classes of problems. Consider the *linear programming problem*: Given

$$A\mathbf{x} = \mathbf{b}$$

where A is $m \times n$ with $m < n$, determine $\mathbf{x}$ such that $\mathbf{x} \geq 0$ and $\mathbf{c}^\top \mathbf{x}$ is minimized for some vector $\mathbf{c}$.

The basic algorithm, due to Dantzig, is as follows: Let $A^{(k)}$ be the basis of the subspace spanned by the columns

$$A^{(0)} = [\mathbf{a}_{i_1}, \mathbf{a}_{i_2}, \ldots, \mathbf{a}_{i_m}].$$

We then form the basis $A^{(k+1)}$ as

$$A^{(k+1)} = [\mathbf{a}_{i_1}, \ldots, \mathbf{a}_{i_{p-1}}, \mathbf{a}_{i_q}, \mathbf{a}_{i_{p+1}}, \ldots, \mathbf{a}_{i_m}]$$

so that $A^{(k+1)}$ differs from $A^{(k)}$ by one column. The approximants $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$ satisfy

$$A^{(k)}\mathbf{x}^{(k)} = \mathbf{b}, \quad A^{(k+1)}\mathbf{x}^{(k+1)} = \mathbf{b}.$$

Given $\Pi^{(k)}A^{(k)} = L^{(k)}U^{(k)}$, we seek a method for computing

$$\Pi^{(k+1)}A^{(k+1)} = L^{(k+1)}U^{(k+1)}$$

within $O(m^2)$ operations. The classical method for computing such a decomposition is based on Gauss-Jordan elimination, which is only stable for limited classes of matrices. This is an example of the classical tradeoff of sparsity vs. stability. A stable algorithm was developed by Bartels and Golub.

7. Linear Algebra and Optimization

NLA plays an important role in optimization. There is a strong connection between a formulation of a linear system and a minimization formula, and even in nonlinear optimization, the majority of computing time is spent on solving linear systems.

An example of this connection is quadratic programming. Consider the problem of minimizing

$$\frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{x}^\top \mathbf{c}$$

subject to the constraint $B^\top \mathbf{x} = \mathbf{d}$. This can be approached using the Lagrange Multipliers formulation, in which we define

$$\phi(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top A\mathbf{x} - \mathbf{x}^\top \mathbf{c} + \boldsymbol{\lambda}^\top (B^\top \mathbf{x} - \mathbf{d})$$

and compute its stationary points

$$\nabla \phi = 0.$$

This can be accomplished by solving the linear system

$$\mathcal{K}\mathbf{u} = \begin{bmatrix} A & B \\ B^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}.$$

8. Updating and Downdating

The tasks of *updating* and *downdating* arise in a variety of applications. Updating involves the addition of data to an existing problem and recomputing the solution to the modified problem efficiently. Downdating is the process of efficiently extracting desired data from the solution to a problem. Applications where updating and downdating occur are data fitting, Kalman filters, and signal processing.

For instance, consider the least squares problem: determine $\mathbf{x}$ such that $\|\mathbf{b} - A\mathbf{x}\|_2$ is minimized. Frequently, a row or column of A is added or deleted. Typically, this problem is solved using the QR factorization of A, as the approach of using the normal equations $A^\top A\mathbf{x} = A^\top \mathbf{b}$, besides being less accurate, is not conducive to efficient updating or downdating. In order to update or downdate the orthogonal matrix Q, various methods can be used, such as Gram-Schmidt, Householder reflections or Givens rotations.

9. The Singular Value Decomposition

Let A be an $m \times n$ matrix. The singular value decomposition (SVD) of A is

$$A = U\Sigma V^\top,$$

where

$$U^\top U = I_m, \quad V^\top V = I_n$$

and

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ \vdots & & \ddots & \sigma_n \\ \vdots & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}.$$

The singular values are typically ordered monotonically:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$$

The non-zero singular values of A are the square roots of the non-zero eigenvalues of $A^\top A$:

$$\sigma_i(A) = (\lambda_i(A^\top A))^{1/2}.$$

In addition to its enormous importance in NLA algorithms, the SVD is useful in areas of applications of importance to the whole scientific community, and has influenced many people's lives, with applications in areas such as vision and motion analysis, signal processing, and search engines and data mining. Some examples of the use of the SVD are computing optimal low rank approximations, solving the least squares problem, and determining the rank of a matrix.

We now examine one of those applications, computing a low rank approximation. Let A be an $m \times n$ matrix of rank r. The matrix A_k minimizing $\|A - A_k\|_2$ is simply the matrix given by

$$A_k = U\Sigma_k V^\top$$

where

$$\Sigma_k = \begin{bmatrix} \sigma_1 & 0 & \cdots & \cdots & & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \sigma_k & 0 & & \vdots \\ \vdots & & \ddots & 0 & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 0 \end{bmatrix}.$$

There are many ways of computing the SVD. A popular method due to Golub and Kahan is *bi-diagonalization*: Find X such that $X^\top X = I_m$ and Y such that $Y^\top Y = I_n$, and

$$B = \begin{bmatrix} \alpha_1 & \beta_1 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \beta_{n-1} \\ 0 & \cdots & 0 & \alpha_n \end{bmatrix},$$

such that

$$X^\top AY = [B \mid 0]^\top.$$

By using a variant of the QR method, the matrix B is then diagonalized.

10. Cyclic Reduction

Consider the system

$$\begin{bmatrix} I & F \\ F^\top & I \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix}$$

The matrix of this system is said to have *Property A*. In such a system, it is easy to eliminate the unknown vector $\mathbf{u}$, resulting in the half-sized system

$$(I - F^\top F)\mathbf{v} = \mathbf{h} - F^\top \mathbf{g}.$$

The matrix $I - F^\top F$ can be reordered in some cases to yield another matrix that has Property A, and thus we can repeat this procedure, eliminating half of the unknowns at each step. The resulting algorithm is reminiscent of the FFT, and requires only $O(N^2 \log N)$ operations to solve the original system.

This method is useful for solving Poisson's equation. For example, on the interval $[0, 1]$, we have the problem

$$-u''(x) = f(x),$$

$$u(0) = a, \quad u(1) = b.$$

Discretizing using centered schemes on a uniform mesh, the matrix associated with the linear system is

$$A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

Using *red-black re-ordering*, we obtain the linear system

$$\begin{bmatrix} I & F \\ F^\top & I \end{bmatrix} \begin{bmatrix} \mathbf{u}^{(r)} \\ \mathbf{v}^{(b)} \end{bmatrix} = \begin{bmatrix} \mathbf{s}^{(r)} \\ \mathbf{s}^{(b)} \end{bmatrix}$$

and we can repeat this process to solve the original difference equation.

11. Iterative Methods

For large, sparse linear systems, Gaussian elimination may be impractical due to *fill*, which is the loss of sparsity as a result of elementary row operations.

Iterative methods are based on computing a sequence of approximations $\mathbf{x}^{(k)}$, starting with an initial guess $\mathbf{x}^{(0)}$, and ideally converging "sufficiently close" to the solution after a "reasonable" number of iterations.

11.1. **The Conjugate Gradient Method.**

The celebrated Conjugate Gradient algorithm, developed by Hestenes and Stiefel in 1952, is an optimal approximation in the following sense: at the nth iteration,

$$\mathbf{x}^{(n)} - \mathbf{x}^{(0)} \in K_n(A) := \operatorname{span}\{\mathbf{r}^{(0)}, Ar^{(0)}, A^2\mathbf{r}^{(0)}, \cdots, A^{n-1}\mathbf{r}^{(0)}\}$$

such that

$$\|\mathbf{x} - \mathbf{x}^{(n)}\|_A = \|\mathbf{b} - A\mathbf{x}^{(n)}\|_{A^{-1}} = \|\mathbf{r}^{(n)}\|_{A^{-1}} = \min_{\mathbf{u} \in K_n(A)} \|\mathbf{x} - \mathbf{u}\|_A.$$

The idea is based on picking directions $\mathbf{p}^{(k)}$ such that

$$\mathbf{p}(i)^\top A\mathbf{p}^{(j)} = 0, \quad i \neq j.$$

The iterations are computed by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}.$$

The residual satisfies

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}.$$

The CG method is *optimal* is that the *error* is minimized over the Krylov subspace in the energy norm $\|\mathbf{e}\|_A := \mathbf{e}^\top A\mathbf{e}$. The sequence of errors satisfies

$$\|\mathbf{e}_n\|_A \leq \|\mathbf{e}_{n-1}\|_A.$$

The beauty of the method is that $\mathbf{p}^{(k)}$ can be chosen so that the iterate $x^{(k+1)}$ really minimizes the error over the whole Krylov subspace, not only over $\operatorname{span}\{\mathbf{x}^{(k)}, \mathbf{p}^{(k)}\}$.

The $\mathbf{p}^{(k)}$ can be thought of as search directions in a nonlinear optimization problem. The problem is to minimize

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{x}^\top \mathbf{b}.$$

Equating the gradient of ϕ to zero takes us to the equation $A\mathbf{x} = \mathbf{b}$. The direction $\mathbf{p}^{(k)}$ and the step length α_k can be determined mathematically by the formulation of the problem.

11.2. **Splittings.** Consider the linear system $A\mathbf{x} = \mathbf{b}$. The *splitting*

$$A = M - N, \quad M\mathbf{x} = N\mathbf{x} + \mathbf{b}$$

leads to the *fundamental* iterative scheme

$$M\mathbf{x}^{k+1} = N\mathbf{x}^k + \mathbf{b}. \tag{11.1}$$

Define the *error* and the *iteration matrix*

$$\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k, \quad K = M^{-1}N$$

We obtain $\mathbf{e}^k \to 0$ as $k \to \infty$ if $\rho(K) < 1$. We assume that it is "easy" in some sense to solve (11.1), which is equivalent to

$$\begin{aligned} M\mathbf{z}^k &= \mathbf{r}^k \equiv \mathbf{b} - A\mathbf{x}^k \equiv Ae^k, \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \mathbf{z}^k. \end{aligned} \tag{11.2}$$

An example of splittings arises from domain decomposition from PDEs. A matrix of the form

$$A = \begin{bmatrix} A_1 & & & B_1 \\ & \ddots & & \vdots \\ & & A_r & B_r \\ B_1^\top & \cdots & B_r^\top & Q \end{bmatrix}$$

where the B_j, $j = 1, \ldots, r$ represent values at the interfaces between subregions, can be split into

$$M = \begin{bmatrix} A_1 & & & \\ & \ddots & & \\ & & A_r & \\ & & & Q \end{bmatrix}, \quad N = -\begin{bmatrix} & & & B_1 \\ & & & \vdots \\ & & & B_r \\ B_1^\top & \cdots & B_r^\top & 0 \end{bmatrix}.$$

Non-symmetric problems can also be tackled using splittings. Every matrix A can be split into its symmetric and skew-symmetric parts, suggesting a natural splitting:

$$A = \frac{A + A^\top}{2} + \frac{A - A^\top}{2} = M - N.$$

Concus and Golub proved that the conjugate gradient and Chebyshev semi-iterative methods will converge for such a splitting, provided that M is positive definite.

12. Software

Over the years, software that implements numerical methods like the ones previously presented have continually improved. Such software is readily available from Netlib, a numerical software distribution system.

Widely used packages are LAPACK, LINPACK, EISPACK, FISHPACK and MATLAB. Another recent addition is ATLAS, Automatically Tuned Linear Algebra Package.

In recent years, parallel algorithms have attracted great interest due to advancements in computer architecture. There are many architectural aspects to consider in designing parallel algorithms, such as mesh processing, blocking sizes, software pipelining strategies, register allocations, memory hierarchy, and memory distribution. Parallel algorithms use a variety of techniques for exploiting the diverse architectures available, including multi-color orderings, recursion, and "locality."

Another interesting aspect of parallel computing is the "resurrection" of old methods that had been considered "dead" for a long time, since they are useful for parallel architectures whereas they were not practical for serial architectures. One example of such a method is the Jacobi method for linear systems.

13. Future Directions

There are a number of hot areas of research in NLA at this time, including model reduction problems, polynomial eigenvalue problems, PDE solvers in 3D, and parallel processing. In addition to these areas, technology itself is providing direction to research in scientific computing. New devices and environments are emerging, and with them, new problems that will require new solution techniques.

Department of Computer Science, Gates Building 2B, Room 280, Stanford, CA 94305-9025
E-mail address: golub@stanford.edu