TEACHING NUMERICAL LINEAR ALGEBRA

AT THE UNDERGRADUATE LEVEL

by

Biswa Nath Datta

Department of Mathematical Sciences

Northern Illinois University

DeKalb, IL 60115

E-mail: dattab@math.niu.edu

What is Numerical Linear Algebra?

Numerical Linear Algebra deals with **numerical** solutions of linear algebra problems.

Importance of NLA

NLA is no longer a subtopic of NA, it has grown into an independent topic for research and teaching in recent years. The reason, is, of course, obvious.

NLA techniques are essential ingredients in scientific computing that are routinely used to solve practical-life problems.

(Signal Processing, Control Theory, Heat Transfer, Fluid Dynamics, Biomedical Engineering, Vibration, Statistics, Bioscience, Economics)

NLA should be introduced to our students early on.

It is really simple. The prerequisites are:

1. An elementary course in linear algebra

2. **(Desirable but not necessary)** Some knowledge of computer programming.

(Because of the existence of MATLAB or MATHEMATICA, the entire course can be taught without using FORTRAN or C). Unfortunately, only a few major schools in this country have courses in NLA (as such) at the undergraduate levels. In most schools, NLA is still taught within a NA course. **But things are changing!!**

My Personal Views of How NLA Should Be Taught

**I. Discussions of Computational Difficulties with
Theoretical Linear Algebra Methods**

- Spend one lecture at the very outset of the course discussing the difficulties that one faces in computational setting when trying to solve the linear algebra problems using some of the <u>obvious</u> approaches, such as solving a linear system using **Cramer's rule** or by explicitly finding the **inverse of the matrix;** finding **the eigenvalues by computing the zeros of the characteristic polynomial**; solving the **least-squares problem** by **normal equations**; finding the **singular values** by **computing the eigenvalues of the matrix** $A^T A$; etc.

II. Early Introduction of Basic Concepts

- Introduce the **basic** but very **important** concepts such as **round-off errors, catastrophic cancellation, conditioning of a problem, stability of an algorithm** and their **joint effects on the accuracy** of solutions, etc, and then return to the discussions of these concepts with respect to each problem and the associated algorithms.

- Use plenty of simple but motivating examples. (**The students like examples!!**)

- Some **Benchmark Examples** in numerical linear algebra

An Important Fact:

Conditioning is a property of the problem, stability is a property of the algorithm, and **both have effects on the accuracy of the solution.**

Thus, if an answer is not right, the algorithm should not be blamed automatically; the condition of the problem may be bad - **ill-conditioned problem**.

If the problem is ill-conditioned, no matter what algorithm is used, accuracy can not be guaranteed. Unfortunately, there are text books which are a little fuzzy about these concepts. I have seen text books talking about **ill-conditioned algorithms rather than problems.**

Tip # 1

Explain the basic concepts of stability, conditioning, etc , with simple motivating examples, as early as possible in the course, and then return to the discussions of these concepts with respect to every problem and associated algorithms you discuss in the class.

Teaching of How to Write An Algorithm and Discussing Properties of Numerical Effectiveness.

For most students an algorithm means describing a procedure using a computer language. (This is only the implementation part of an algorithm.)

An **Algorithm** is an ordered set of operations, logical and arithmetic, which - when applied to a computational problem, defined by a given set of data, called input data - produces a solution to the problem.

Algorithms can be described by means of pseudocodes.

It is important that a student understands how to write an algorithm (using step-wise description) in a clear and concise manner so that a programmer can code it effectively, even without being a mathematician.

Again simple examples can be used. (Examples should be chosen from the problems the students are already familiar with).

- Computing the norm of a vector.

- Computing the inner product or the outer product.

- Solving the triangular system.

- Computing the inverse of an upper triangular matrix.

- Gaussian elimination.

At this point, it is a good idea to draw the attention of the students to some of the properties that make an algorithm **numerically effective.**

Purpose: A student should be able to distinguish between a bad and a good algorithm.

- Efficiency and storage - economy

- Stability

- Easiness to use

- Reliability, etc.

Again simple examples should be used to illustrate these properties.

A good example of a numerically inefficient algorithm: Cramer's Rule for $Ax = b$.

It might take more than a million years to solve a 20×20 problem using the usual definition of the determinant.

A common misconception is: An efficient algorithm is a good algorithm. It is hardly true. **An efficient algorithm may be highly unstable**. Example: Gaussian elimination (without pivoting)

Tip # 2

Explain clearly how to write an algorithm and discuss the properties of a **numerically effective algorithm** with plenty of examples. Then return to the discussion of each of these properties when describing a particular algorithm.

Algorithm Presentation: A major part of a NLA **course** is about describing various algorithms for linear algebra problems and discuss their properties and implementational aspects. The most important part of NLA **teaching** is to figure out how an algorithm should be presented to the students.

Describing an algorithm using a terse Pascal-like language is hardly helpful for a student to understand the algorithm. What is important is to **explain the mechanism of the development of the algorithm, to make the student think about how the algorithm was developed in the first place.**

What I normally do is the following:

1. Write down the goal of the algorithm at the outset.

2. Tell what are the available tools for developing the algorithm.

3. Talk about the strategy of development. If it is a direct algorithm, how many steps are there, and what

is expected out of each step. If it is an iterative algorithm, what is the stopping criterion?

4. Give a simple illustrative example following exactly the pattern of the development.

5. Discuss the issues of **efficiency, stability, accuracy, implementation, etc**.

Example: Finding QR factorization of A using Householder Matrices.

1. **Goal:** Given an matrix A, find an orthogonal matrix Q and an upper triangular matrix R such that $A = QR$, or in other words: $Q^T A = R$.

2. **Tools:** Householder matrices.
   A Householder matrix is orthogonal and can be used to create zeros in a vector.

3. **Strategy:** Starting from the first column, create zeros successively in columns 1 through $(n-1)$ of A (in appropriate places) using one Householder matrix

at a time. At step k, the zeros are created using the Householder matrix H_k such that $H_k A$ (where A is the current matrix) has zero below the diagonal entry.

I do the first step; that is I show, how to create a Householder matrix H_1 such that

$$H_1 A = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \cdots & \vdots \\ 0 & * & \cdots & * \end{pmatrix}$$

I then ask **John** to do the **second step**:

$$H_2 H_1 A = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ 0 & 0 & & \vdots \\ \vdots & \vdots & & * \\ 0 & 0 & \cdots & * \end{pmatrix}$$

Then **Linda** to do the **third step**. Then **Tom** to do the **k-th step** and, finally **Eric** to do the $(n-1)$th step.

At the end of $(n-1)$th step, we have an upper triangular matrix R on the right hand side, and Q is obtained as $Q = H_1 H_2 \cdots H_{n-1}$.

Thus. I, John, Linda, Tom, and Eric have successfully developed the Householder algorithm for QR factorization. **This approach really helps stimulate the creativity of the students.** The students feel good that they can develop an algorithm too!!

I have heard students making such remarks: **If I were born thirty years ago, maybe I could have developed this algorithm, it is really simple!!**

Tip # 3

Pay more attention to explaining the mechanism of the development of the algorithm, rather than writing the algorithm using some programming language.

Software

The students should be made aware of the important softwares such as MATLAB, MATHEMATICA, LINPACK, EISPACK, IMSL, LAPACK, etc that are available for numerical linear algebra problems.

Most softwares have implemented only the best algorithm(s) for a problem. These softwares, in general, are of not much help for students who want to see the differences between a bad algorithm and a good algorithm.

The MATLAB-based software "MATCOM" (that comes with the book "**Numerical Linear Algebra and Applications**" has implemented all the major algorithms in NLA, so that the students can study the different algorithms for the same problem, and compare efficiency, stability, and accuracy, etc.

What about having the students write their own software?

This, of course, has to be done. However, what I recommend to do here is to provide students with codes for some basic algorithms so students can use them as **templates** for writing codes for more advanced algorithms.

Motivating Practical Life Applications

Linear Systems Problems: An Electric Circuit Problem, Finite Differences and Finite Element Techniques for Laplace's and Poisson's Equations arising in Heat Transfer, Fluid Mechanics, Approximation of Functions by Polynomials. (Hilbert Systems).

Eigenvalue Problems: Stability of Differential and Difference Equations: European Arms Race; Vibrations of Structures: Resonance leading to Collapse of Tacoma Bridge, Broughton Suspension Bridge in England, etc. Buckling Problems: Simulating Transient Current of an Electric Cir-

cuit; Principal Component Analysis in Statistics: Stock Market Example. The Singular Value Decomposition (SVD): Biomedical Engineering: Detection of Fetal Electro Cardiogram (FECG) while simultaneously suppressing the Maternal Electro Cardiogram (MECG) of the pregnant mother. Digital Image Processing, etc.

Recommended one-semester syllabus for an undergraduate numerical linear algebra course.

1. An overview of numerical linear algebra problems: their importance and computational difficulties. (1/2 week)

2. A review of required linear algebra concepts (Emphasis on norms and norm-preserving properties of orthogonal matrices). (1 week)

3. Floating-point numbers and errors in computations. (1 week)

4. Basic concepts of stability, conditioning, accuracy, etc. (1 week)

5. Discussions on numerical effectiveness of algorithms and associated mathematical software. (1 week)

6. Tools of numerical linear algebra and their applications: Elementary, Householder, and Givens matrices with applications to LU and QR factorizations and Hessenberg-reductions. (2-1/2 weeks)

7. Numerical Solutions of Linear Systems: Some motivating applications, solutions via Gaussian eliminations and QR factorizations, growth factors and stability of Gaussian elimination, Hessenberg, tridiagonal, and positive definite systems, iterative refinement, perturbation analysis and conditioning, conditioning and accuracy, estimation of the condition numbers, inverses and determinants, Jacobi and Gauss-Seidel methods and discussion on convergence. (3 weeks)

8. Least-squares Solutions:

   Some motivating applications, existence and unique-

ness, Normal equations, Generalized Inverse, QR methods for overdetermined systems. (2 weeks)

9. Numerical Matrix Eigenvalue Problems: Some motivating applications, the Gersgorin disk theorem, the Power and Inverse Power methods, Rayleigh Quotient Iteration,, the Bauer-Fike theorem, Difficulties of eigenvalue computations using the characteristic polynomial, Basic and Hessenberg QR iterations. (2 weeks)

10. The Singular Value Decomposition:

    Basic properties and applications of the singular values. (1 week)

Note: The above syllabus roughly corresponds to Chapters 0-4, 5.1-5.4, 5.5.1, 6.3, 6.4, 6.5.1, 6.5.3, 6.5.4, 6.6.1, 6.7, 6.9, 6.10.1-6.10.4, 7.2-7-6, 7.8.1-7.8.2, 8.3, 8.4.1, 8.5.3, 8.6, 10.2-10.6 of my present book **"Numerical Linear Algebra and Applications"** by Biswa Nath Datta, Brooks/Cole Publications, Co., Pacific Grove, California, 1995.

A paperback version of the book is currently available from

Brooks/Cole. New ISBN for the Paperback Version : 0759322546

Toll-Free Number for Ordering the Customized Version :

(Brooks/Cole)

1-800-355-9983

The REVISED SECOND EDITION should be available in

2006.