Numerical linear algebra

Purdue University
CS 51500

Fall 2017
David Gleich
David F. Gleich

Call me ...
“Prof Gleich”
“Dr. Gleich”

Please not
“Hey matrix guy!”
Huda Nassar

Call me ...
“Huda”
“Ms. Huda”

Please not
“Matrix girl”
Numerical linear algebra

Or

Matrix computations
Purpose

Matrix computations underlie much (most?) of applied computations.

It’s the language of computational algorithms.
PageRank (from the paper)

\[
\begin{align*}
R_0 & \leftarrow S \\
\text{loop:} & \\
R_{i+1} & \leftarrow AR_i \\
d & \leftarrow ||R_i||_1 - ||R_{i+1}||_1 \\
R_{i+1} & \leftarrow R_{i+1} + dE \\
\delta & \leftarrow ||R_{i+1} - R_i||_1 \\
\text{while } \delta > \epsilon
\end{align*}
\]
3. The Generalized Method

In this method (Broyden, 1967) the vector $p_i$ is given by

$$p_i = -H_if_i,$$  \hspace{1cm} (3.1)

where $H_i$ is positive definite. $H_1$ is chosen to be an arbitrary positive definite matrix (often the unit matrix) and $H_{i+1}$ is given by

$$H_{i+1} = H_i - H_iz_iw_i^T + p_it_iq_i^T, \hspace{1cm} i = 1, 2, ...,$$  \hspace{1cm} (3.2a)

where

$$y_i = f_{i+1} - f_i,$$  \hspace{1cm} (3.2b)

$$q_i^T = \alpha_ip_i^T - \beta_igy_i^TH_i,$$  \hspace{1cm} (3.2c)

$$w_i^T = \gamma_iy_i^TH_i + \beta_it_ip_i^T,$$  \hspace{1cm} (3.2d)

$$\alpha_i = (1 + \beta_igy_i^TH_iy_i)/p_i^Ty_i,$$  \hspace{1cm} (3.2e)

$$\gamma_i = (1 - \beta_it_ip_i^Ty_i)/y_i^TH_iy_i.$$  \hspace{1cm} (3.2f)

The parameter $\beta_i$ is arbitrary and setting it equal to zero gives the DFP method (Fletcher & Powell, 1963). It was shown by Broyden (1967) that the matrices $H_i$ constructed in this way are always positive definite if $\beta_i \geq 0$. 
Circular antennae design

B. Port Description of Array

Since relatively few of the elements of \( V \) are nonzero some reduction in (3) is possible. Only those columns of \( Y \) which correspond to indices of triangles centered at the dipole mid-points need be retained in (3); denote as \( Y_R \) the rectangular matrix obtained by deleting all columns of \( Y \) not having such a column index. Then,

\[
I = Y_R V_T
\]

where \( V_T \) is the \( N \) vector formed by deleting all identically zero elements of \( V \). \( Y_R \) is denoted the "reduced admittance matrix."

Furthermore, if only the feed-point currents are of interest, a similar reduction may be performed on rows of \( Y_R \) and \( I \) to yield,

\[
I_T = Y_T V_T
\]
Dynamic mode decomposition

1. Split the time series of data in $V_1^N$ into the two matrices $V_1^{N-1}$ and $V_2^N$.
2. Compute the SVD of $V_1^{N-1} = U \Sigma W^T$.
3. Form the matrix $\tilde{S} = U^T V_2^N W \Sigma^{-1}$, and compute its eigenvalues $\lambda_i$ and eigenvectors $y_i$.
4. The $i$-th DMD eigenvalues is the $\lambda_i$ and the $i$-th DMD mode is the $Uy_i$. 
Electrical circuits

“A matrix version of Kirchhoff’s circuit law is the basis of most circuit simulation software”

-- Wikipedia
Other applications

Biology
PDEs/Mechanical Engineering/AeroAstro
Machine learning
Statistics
Graphics
Purpose

The purpose this class is to teach you how to “speak matrix computations like a native” so that you can understand, implement, interpret, and extend work that uses them.
Examples

Why should we avoid the “normal equations”? 

Why do I get strange looks if I talk about the SVD of a symmetric positive definite matrix? 

Why not write things element-wise?
Overview of the material

**Basics** subspaces, rank, inverses, inner-products, norms, orthogonality, permutations

**Dense matrix computations** linear systems, triangular systems, QR factorization, Householder matrices, LU decomposition, Cholesky decomposition, condition numbers, linear least squares, eigenvalues

**Sparse matrix computations** Jacobi, Gauss-Seidel, SOR, basic convergence theory, CG, Lanczos, Arnoldi, GMRES, restarted GMRES, symmetric solvers, non-symmetric solvers, preconditioners
Textbooks

No best reference.

Golub and van Loan – “The Bible” – but sometimes a bit terse

Trefethen & Bau, Numerical Linear Algebra
Demmel, Applied Numerical Linear Algebra
Saad, Iterative Methods for Sparse Linear Systems
Background books

Strang, Linear Algebra and its Applications
Meyer, Matrix Analysis
Why I like Julia & Matlab

**Julia** Designed as a technical computing language

**Matlab** it’s a modeling language for matrix methods!

The power method described in Wikipedia

\[
b_{k+1} = \frac{Ab_k}{\|Ab_k\|}.
\]

```plaintext
while 1
    a = b;
    b = A*b;
    b = b/norm(b);
    if test_converge(a,b); break; end
end
```

**Matlab & Julia code**

```plaintext
x = b  # make a reference to A
y = zeros(length(b))  # allocate
while 1
    A_mul_B!(y,A,x)  # y = Ax
    scale!(y,1/norm(x))  # scale
    if test_converge(x,y); break; end
    x,y == y,x  # swap pointers, not
end
```

Super efficient Julia code
Software

You will have to write matrix programs in class.

**Julia & Atom** my recommendation
**Julia & Jupyter** notebook my 2\(^{nd}\) recommendation
**SciPy, NumPy** okay (look at spyder/pythonxy)
**Matlab** what I used to use

**R** not recommended, best to avoid
**Scilab** you’re on your own
**C/C++ with LAPACK** okay, but ill-advised; this is the “macho” approach
One more thing

• This is a distance class!

• Help me to remember that “important” stuff should be on projector 1!

• Also, help remind me to repeat questions.
THE SYLLABUS
Cut to website!