Solving Systems of Linear Equations with Matrices

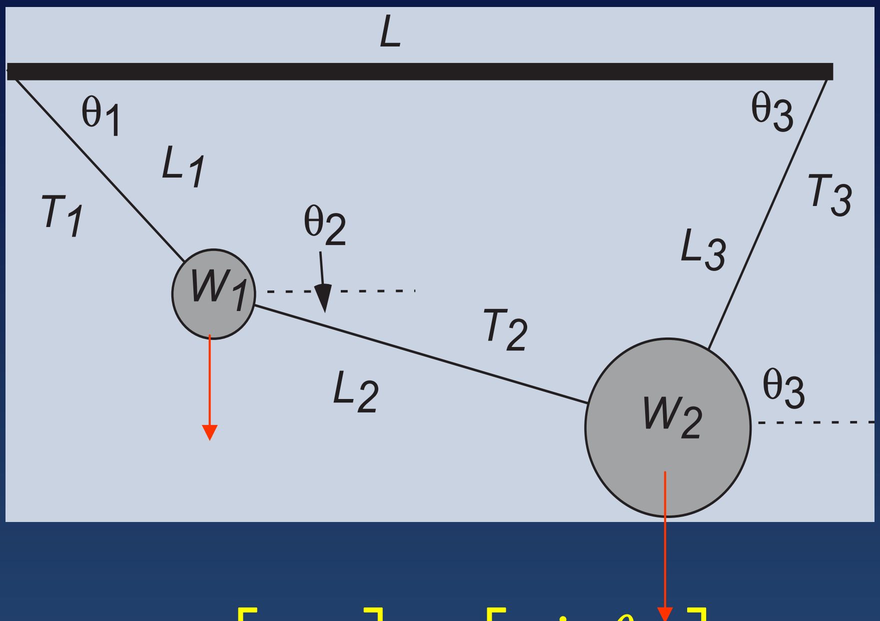
Computers are especially good for this
much of High Performance Computing (HPC)

Rubin H Landau
With
Sally Haerer

Computational Physics for Undergraduates
BS Degree Program: Oregon State University

"*Engaging People in Cyber Infrastructure*"
Support by EPICS/NSF & OSU

Recall 2 Weights on a String Problem



$$[\mathbf{x}] = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} \sin\theta_1 \\ \sin\theta_2 \\ \sin\theta_3 \\ \cos\theta_1 \\ \cos\theta_2 \\ \cos\theta_3 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix},$$

$$\begin{aligned}
f_1(\mathbf{x}) &= 3x_4 + 4x_5 + 4x_6 - 8 = 0 \\
f_2(\mathbf{x}) &= 3x_1 + 4x_2 - 4x_3 = 0 \\
f_3(\mathbf{x}) &= x_7x_1 - x_8x_2 - 10 = 0 \\
f_4(\mathbf{x}) &= x_7x_4 - x_8x_5 = 0 \\
f_5(\mathbf{x}) &= x_8x_2 + x_9x_3 - 20 = 0 \\
f_6(\mathbf{x}) &= x_8x_5 - x_9x_6 = 0 \\
f_7(\mathbf{x}) &= x_1^2 + x_4^2 - 1 = 0 \\
f_8(\mathbf{x}) &= x_2^2 + x_5^2 - 1 = 0 \\
f_9(\mathbf{x}) &= x_3^2 + x_6^2 - 1 = 0
\end{aligned}$$

Systems of Equations via Matrices

Solution

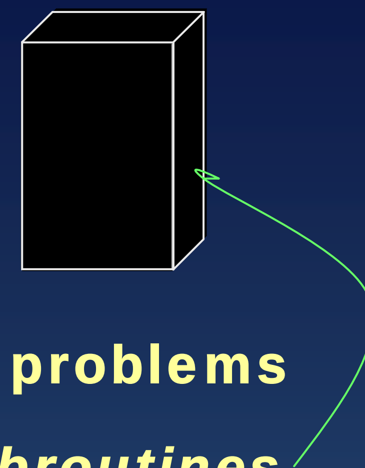
$$\begin{bmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_N \\ \partial f_2/\partial x_1 & \cdots & \partial f_2/\partial x_9 \\ & \vdots & \\ \cdots & \cdots & \partial f_9/\partial x_9 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_9 \end{bmatrix} = - \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_9 \end{bmatrix}_o \quad (1$$

[?]

$$[A]\vec{x} = \vec{b} \quad (2$$

- Many physical models $\Rightarrow$ simultaneous equations
- Place in matrix form, easier math (more abstract)
- More realistic models $\Rightarrow$ larger matrices
- Computer = excellent tool (same steps many times)

Scientific Subroutine Libraries



- **Industrial strength, matrix subroutines**
- **> 10X faster than elementary methods**
- **Minimize roundoff error, failure**
- **Robust: high chance of success, broad class of problems**
- **Recommend:** ***do not write your own matrix subroutines***
- **Also auto scales: desktop $\Rightarrow$ parallel cluster**

What's the cost?

1. **Must find them (not installed)**
2. **Must find names of all subroutines**
3. **May be Fortran only, C only**

Classes of Matrix Problems (Math)

1. **Rules of math still apply!**
2. **N unknowns > N equations (unique)?**
3. **Equations not linearly independent?**
4. **N equations > N unknowns (fitting)?**
5. **Basic problem: system linear equations (2 masses)**

$$[A]\vec{x} = \vec{b} \qquad (1$$

$$[A]_{N\times N} \times \vec{x}_{N\times 1} = \vec{b}_{N\times 1} \qquad (2$$

- [*A*] = known N x N matrix
- *x* = unknown length N vector
- *b* = known length N vector

Solution Linear Equations

$$[A]\vec{x} = \vec{b} \qquad (1$$

$$[A]_{N\times N} \times \vec{x}_{N\times 1} = \vec{b}_{N\times 1} \qquad (2$$

[?]

- **"Best" solution: Gaussian elimination**
- **Triangular decomposition: no** $[A]^{-1}$
- **Slower, less robust: compute** $[A]^{-1}$

$$[A]^{-1}[A]\vec{x} = [A]^{-1}\vec{b} \qquad [A]^{-1}\ (1$$

$$\vec{x} = [A]^{-1}\vec{b}$$

- **Both methods in libes**

Classes of Matrix Problems (cont)

Eigenvalue Problem

$$[A]\vec{x} = \lambda \vec{x} \qquad (1$$

- Different matrix equation, not $[A]\vec{x} = \vec{b}$ (2
- x (vector), λ (number) **=** unknowns RHS
- No direct solution, ∃ for some λ
- When ∃ ?

Trivial solution $$([A] - \lambda[I])\,\vec{x} \quad = \quad 0 \qquad (3$$

$$\times\ ([A] - \lambda[I])^{-1} \qquad \Rightarrow \quad \vec{x} = 0 \qquad (4$$

Nontrivial solution $$\nexists\ \ ([A] - \lambda[I])^{-1} \qquad (5$$

Secular Equation (Cramer's Rule) $$\det[\mathbf{A} - \lambda \mathbf{I}] = 0 \qquad (6$$

Evaluate det[ ] & Search

Practical Aspects of Matrix Computing

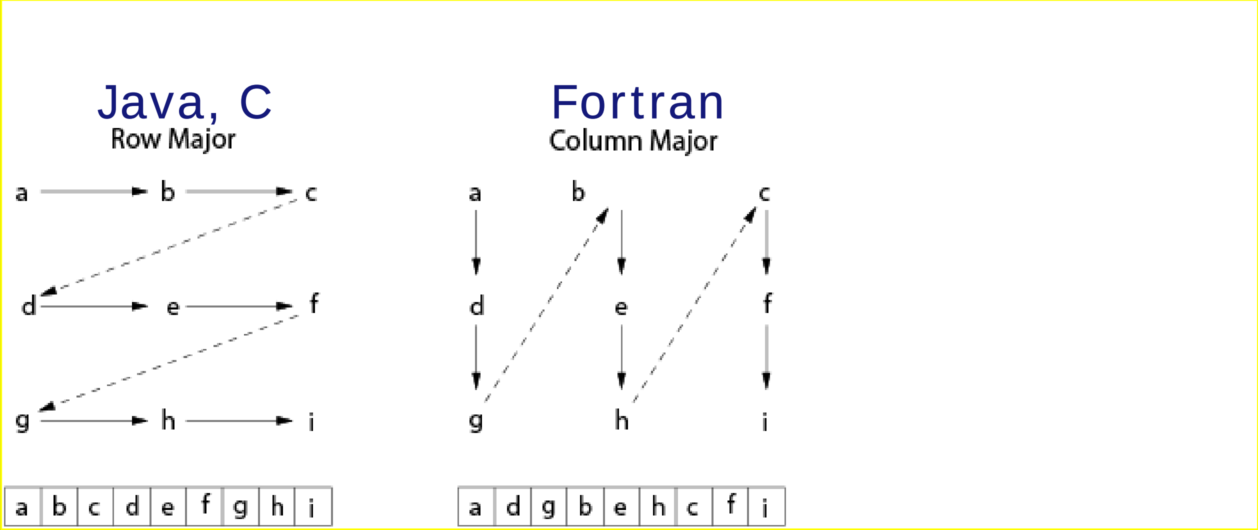
- Scientific programming bugs: often arrays
- Even vector V[N] = “array” (1-D)

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix}$$

- Rules of thumb
  - Computers are finite: size matters
  - Physical dimension of 100: **`A[100][100][100][100]`** ≈ 1GB
  - Processing time: ~ N^3 steps for **`A[N][N]`**
  - Double N ⇒ 8X time
  - Avoid page faults: 1 word → entire page

Practical Aspects: Memory

- Matrix storage: we think blocks, computer stores linear



- Avoid large "strides"

- Don't have too many indices: V[L, Nre, Nspin, k, kp, Z, A] (1
  - $\rightarrow$ V1[k, kp], V2[k, kp], V3[k, kp] (2
- Subscript 0: math must match (count from 1 or 0?)

$$(l+1)P_{l+1} - (2l+1)xP_l + lP_{l-1} = 0$$ (3

- Physical vs logical dimensions
  - declared a[3][3], defined (') up to a[2][2]

a[1][1]' a[1][2]' a[1][3] a[2][1]' a[2][2]' a[2][3] a[3][1] a[3][2] a[3][3] (4

Implementation: Scientific Libraries, WWW

NETLIB	**WWW metalib of free math libraries**	**LAPACK**	**Linear Algebra pack**
JLAPACK	**LAPACK library in Java**	**SLATEC**	**Comprehensive Math & Stats**
ESSL	**Engr & Sci Lib (IBM)**	**IMSL**	**Intl Math & Stats**
CERNLIB	**European Cntr Nuclear Res**	**BLAS**	**Basic Linear Algebra Subs**
JAMA	**Java Matrix Lib**	**NAG**	**Numerical Algorithms Group (UK)**
Lapack++	**Linear Algebra pack in C++**	**ScaLAPACK**	**Distributed Memory LAPACK**
TNT	**C++ Template Numerical Toolkit**	**GNU GSL**	**Full Scientific Libe in C & C++**

Linear algebra	Matrix operations	Interpolation, fitting
Eigensystem analysis	Signal processing	Sorting and searching
Solution of linear eqns	Differential equations	Roots, zeros & extrema
Random-number ops	Statistical functions	Numerical quadrature

JAMA: Java Matrix Library

- JAMA = basic linear algebra package for Java
- Works well, natural, non-expert, free
- Jampack: complex matrices
- True **`Matrix`** objects; linear algebra, aligned elements
- e.g. [A] x = b

```
double[][] array = { {1.,2.,3}, {4.,5.,6.}, {7.,8.,10.} };
Matrix A = new Matrix(array);
Matrix b = Matrix.random(3,1);
Matrix x = A.solve(b);
Matrix Residual = A.times(x).minus(b);
Matrix Itest = A.inverse().times(A);          // Test inverse
```

JamaEigen.java: Eigenvalue Problem

```
import Jama.*;          import java.io.*;
public class JamaEigen {

  public static void main(String[] argv) {
  double[][] I = { {2./3,-1./4,-1./4},  {-1./4,2./3,-1./4},   {-1./4,-
1./4,2./3}};
  Matrix MatI = new Matrix(I);                              // Array →matrix
  System.out.print( "Input Matrix" );
  MatI.print (10, 5);                                        // Print matrix
  EigenvalueDecomposition E = new EigenvalueDecomposition(MatI);
double[] lambdaRe =  E.getRealEigenvalues();                 //   Eigens
 System.out.println("Eigenvalues: \t lambda.Re[]="+ lambdaRe[0]);
  Matrix V = E.getV();                                       // Vectors
  System.out.print("\n Matrix with column eigenvectors ");
  V.print (10, 5);
}}
```

Run Program for Output

JamaFit.java: fit $y(x) = b_0 + b_1 x + b_2 x^2$

- Look at now, will describe math and use latter
- Let’s take a test drive before purchase