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

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

\[
\begin{align*}
f_1(x) &= 3x_4 + 4x_5 + 4x_6 - 8 = 0 \\
f_2(x) &= 3x_1 + 4x_2 - 4x_3 = 0 \\
f_3(x) &= x_7x_1 - x_8x_2 - 10 = 0 \\
f_4(x) &= x_7x_4 - x_8x_5 = 0 \\
f_5(x) &= x_8x_2 + x_9x_3 - 20 = 0 \\
f_6(x) &= x_8x_5 - x_9x_6 = 0 \\
f_7(x) &= x_1^2 + x_4^2 - 1 = 0 \\
f_8(x) &= x_2^2 + x_5^2 - 1 = 0 \\
f_9(x) &= x_3^2 + x_6^2 - 1 = 0
\end{align*}
\]
Systems of Equations via Matrices

- Many physical models ⇒ simultaneous equations
- Place in matrix form, easier math (more abstract)
- More realistic models ⇒ 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 ⇒ 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)

\[
\begin{align*}
[A] \vec{x} &= \vec{b} \\
[A]_{N \times N} \times \vec{x}_{N \times 1} &= \vec{b}_{N \times 1}
\end{align*}
\]

- \([A]\) = known N x N matrix
- \(x\) = unknown length N vector
- \(b\) = known length N vector
Solution Linear Equations

\[ [A] \vec{x} = \vec{b} \]

\[ [A]_{N \times N} \times \vec{x}_{N \times 1} = \vec{b}_{N \times 1} \]

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

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

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

- Both methods in libes
Classes of Matrix Problems (cont)

Eigenvalue Problem

\[
[A] \vec{x} = \lambda \vec{x}
\]

1. Different matrix equation, not \([A]\vec{x} = \vec{b}\)
2. \(\vec{x}\) (vector), \(\lambda\) (number) = unknowns RHS
3. No direct solution, \(\exists\) for some \(\lambda\)
4. When \(\exists\) ?

Trivial solution

\[
([A] - \lambda[I]) \vec{x} = 0
\]

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

Nontrivial solution

\[
\not\exists ([A] - \lambda[I])^{-1}
\]

Secular Equation (Cramer’s Rule)

\[
\det[A - \lambda I] = 0
\]

Evaluate \(\det[\ ]\) & Search
Practical Aspects of Matrix Computing

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

Rules of thumb

- Computers are finite: size matters
- Physical dimension of 100: $A[100][100][100][100] \approx 1\text{GB}$
- Processing time: $\sim N^3$ steps for $A[N][N]$
- Double $N \Rightarrow 8X$ time
- Avoid page faults: 1 word $\rightarrow$ entire page
Practical Aspects: Memory

- Matrix storage: we think blocks, computer stores linear.

Java, C
Row Major

Fortran
Column Major

- Avoid large “strides”

- Don't have too many indices: \( V[L, \, Nre, \, Nspin, \, k, \, kp, \, Z, \, A] \)  

  \[ \rightarrow \quad V1[k, \, kp], \quad V2[k, \, kp], \quad V3[k, \, kp] \]  

- Subscript 0: math must match (count from 1 or 0?)

  \[ (l + 1)P_{l+1} - (2l + 1)xP_{l} + lP_{l-1} = 0 \]  

- Physical vs logical dimensions

  - declared \( a[3][3] \), defined (′) up to \( a[2][2] \)

  \[ a[1][1]' \quad a[1][2]' \quad a[1][3] \quad a[2][1]' \quad a[2][2]' \quad a[2][3] \quad a[3][1] \quad a[3][2] \quad a[3][3] \]
### Implementation: Scientific Libraries, WWW

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
<th>Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETLIB</td>
<td>WWW metalib of free math libraries</td>
<td>LAPACK</td>
<td>Linear Algebra pack</td>
</tr>
<tr>
<td>JLAPACK</td>
<td>LAPACK library in Java</td>
<td>SLATEC</td>
<td>Comprehensive Math &amp; Stats</td>
</tr>
<tr>
<td>ESSL</td>
<td>Engr &amp; Sci Lib (IBM)</td>
<td>IMSL</td>
<td>Intl Math &amp; Stats</td>
</tr>
<tr>
<td>CERNLIB</td>
<td>European Cntr Nuclear Res</td>
<td>BLAS</td>
<td>Basic Linear Algebra Subs</td>
</tr>
<tr>
<td>JAMA</td>
<td>Java Matrix Lib</td>
<td>NAG</td>
<td>Numerical Algorithms Group (UK)</td>
</tr>
<tr>
<td>Lapack++</td>
<td>Linear Algebra pack in C++</td>
<td>ScaLAPACK</td>
<td>Distributed Memory LAPACK</td>
</tr>
<tr>
<td>TNT</td>
<td>C++ Template Numerical Toolkit</td>
<td>GNU GSL</td>
<td>Full Scientific Lib in C &amp; C++</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear algebra</td>
<td>Compressed</td>
</tr>
<tr>
<td>Eigensystem analysis</td>
<td>Comprehensive Math &amp; Stats</td>
</tr>
<tr>
<td>Solution of linear eqns</td>
<td>Distributed Memory LAPACK</td>
</tr>
<tr>
<td>Random-number ops</td>
<td>Full Scientific Lib in C &amp; C++</td>
</tr>
<tr>
<td>Interpolation, fitting</td>
<td>Interpolation, fitting</td>
</tr>
<tr>
<td>Signal processing</td>
<td>Interpolation, fitting</td>
</tr>
<tr>
<td>Sorting and searching</td>
<td>Interpolation, fitting</td>
</tr>
<tr>
<td>Differential equations</td>
<td>Interpolation, fitting</td>
</tr>
<tr>
<td>Roots, zeros &amp; extrema</td>
<td>Interpolation, fitting</td>
</tr>
<tr>
<td>Statistical functions</td>
<td>Interpolation, fitting</td>
</tr>
<tr>
<td>Numerical quadrature</td>
<td>Interpolation, fitting</td>
</tr>
</tbody>
</table>
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

```java
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
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