

Floating Point

Floating point numbers are like piles of sand; every time you move them around, you lose a little sand and pick up a little dirt. - *Kernighan and Plauger*

Floating Point: Computing the Standard Deviation

Standard deviation. Two mathematically equivalent textbook formulas.

$$\begin{aligned}\sigma &= \sqrt{\frac{1}{n} \sum_i (x_i - \mu)^2}, \text{ where } \mu = \frac{1}{n} \sum_i x_i \\ &= \sqrt{\frac{1}{n} \sum_i x_i^2 - \mu^2}\end{aligned}$$

Ex: $x_0 = 0.5000000000000002$, $x_1 = 0.5000000000000001$.

Formula 1: 1.11E-16.

Formula 2: takes square root of negative number!

Caveat: Broken formula used in MS Excel 2002.

5

6

Linear System of Equations

Linear system of equations.

- N linear equations in N unknowns.
- Matrix notation: find x such that $Ax = b$.

$$\begin{aligned}0x_0 + 1x_1 + 1x_2 &= 4 \\ 2x_0 + 4x_1 - 2x_2 &= 2 \\ 0x_0 + 3x_1 + 15x_2 &= 36\end{aligned}$$

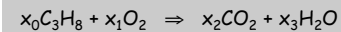
$$A = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$$

Among most fundamental problems in science and engineering.

- Linear regression.
- Chemical equilibrium.
- Linear and nonlinear optimization.
- Polynomial and spline interpolation.
- Kirchoff's current and voltage laws.
- Leontief model of economic equilibrium.
- Numerical solution to differential equations.
- Analysis of risk of portfolio of financial derivatives.

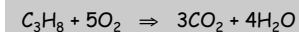
Chemical Equilibrium

Ex: combustion of propane.



Stoichiometric constraints.

- Carbon: $3x_0 = x_2$.
 - Hydrogen: $8x_0 = 2x_3$.
 - Oxygen: $2x_1 = 2x_2 + x_3$.
 - Normalize: $x_0 = 1$.
- } conservation of mass



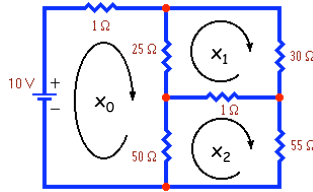
Remark. Stoichiometric coefficients tend to be small integers; this was among first hints suggesting the atomic nature of matter.

7

8

Kirchoff's Current Law

Ex: find current flowing in each branch of a circuit.



Kirchoff's current law.

- $10 = 1x_0 + 25(x_0 - x_1) + 50(x_0 - x_2).$
 - $0 = 25(x_1 - x_0) + 30x_1 + 1(x_1 - x_2).$
 - $0 = 50(x_2 - x_0) + 1(x_2 - x_1) + 55x_2.$
- } conservation of electrical charge

Solution: $x_0 = 0.2449$, $x_1 = 0.1114$, $x_2 = 0.1166$.

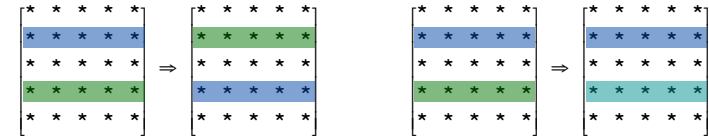
Gaussian Elimination

Gaussian elimination.

- Among oldest and most widely used solutions.
- Repeatedly apply row operations until system is *upper triangular*.
- Solve "trivial" upper triangular system.

Row operations.

- Exchange any two rows.
- Add a multiple of one row to another.



Key invariant. Row operations preserve solutions.

Gaussian Elimination: Row Operations

Row operations.

$$\begin{array}{r} 0x_0 + 1x_1 + 1x_2 = 4 \\ 2x_0 + 4x_1 - 2x_2 = 2 \\ 0x_0 + 3x_1 + 15x_2 = 36 \end{array}$$

↓ (interchange row 0 and 1)

$$\begin{array}{r} 2x_0 + 4x_1 - 2x_2 = 2 \\ 0x_0 + 1x_1 + 1x_2 = 4 \\ 0x_0 + 3x_1 + 15x_2 = 36 \end{array}$$

↓ (subtract 3x row 1 from row 2)

$$\begin{array}{r} 2x_0 + 4x_1 - 2x_2 = 2 \\ 0x_0 + 1x_1 + 1x_2 = 4 \\ 0x_0 + 0x_1 + 12x_2 = 24 \end{array}$$

Gaussian Elimination: Back Substitution

Back substitution. Upper triangular systems are easy to solve by examining equations in reverse order.

$$\begin{array}{r} 2x_0 + 4x_1 - 2x_2 = 2 \\ 0x_0 + 1x_1 + 1x_2 = 4 \\ 0x_0 + 0x_1 + 12x_2 = 24 \end{array}$$

- Equation 2: $x_2 = 24/12 = 2$.
- Equation 1: $x_1 = 4 - x_2 = 2$.
- Equation 0: $x_0 = (2 - 4x_1 + 2x_2) / 2 = -1$.

```
for (int i = N-1; i >= 0; i--) {
    double sum = 0.0;
    for (int j = i+1; j < N; j++)
        sum += A[i][j] * x[j];
    x[i] = (b[i] - sum) / A[i][i];
}
```

$$x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{j=i+1}^{N-1} a_{ij} x_j \right]$$

Gaussian Elimination: Forward Elimination

Forward elimination. Apply row operations to make upper triangular.

Pivot. Zero out entries below pivot a_{pp} .

$$a_{ij} = a_{ij} - \frac{a_{ip}}{a_{pp}} a_{jk}$$

$$b_i = b_i - \frac{a_{ip}}{a_{pp}} b_p$$

$$\begin{bmatrix} * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ p & 0 & 0 & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & * & * & * & * \end{bmatrix} \Rightarrow \begin{bmatrix} * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * & * \end{bmatrix}$$

```
for (int p = 0; p < N; p++) {
    for (int i = p + 1; i < N; i++) {
        double alpha = A[i][p] / A[p][p];
        b[i] -= alpha * b[p];
        for (int j = p; j < N; j++)
            A[i][j] -= alpha * A[p][j];
    }
}
```

13

Gaussian Elimination Example

$$\begin{array}{rclclcl} 1x_0 & + & 0x_1 & + & 1x_2 & + & 4x_3 & = & 1 \\ 2x_0 & + & -1x_1 & + & 1x_2 & + & 7x_3 & = & 2 \\ -2x_0 & + & 1x_1 & + & 0x_2 & + & -6x_3 & = & 3 \\ 1x_0 & + & 1x_1 & + & 1x_2 & + & 9x_3 & = & 4 \end{array}$$

14

Gaussian Elimination Example

$$\begin{array}{rclclcl} 1x_0 & + & 0x_1 & + & 1x_2 & + & 4x_3 & = & 1 \\ 0x_0 & + & -1x_1 & + & -1x_2 & + & -1x_3 & = & 0 \\ 0x_0 & + & 1x_1 & + & 2x_2 & + & 2x_3 & = & 5 \\ 0x_0 & + & 1x_1 & + & 0x_2 & + & 5x_3 & = & 3 \end{array}$$

15

Gaussian Elimination Example

$$\begin{array}{rclclcl} 1x_0 & + & 0x_1 & + & 1x_2 & + & 4x_3 & = & 1 \\ 0x_0 & + & -1x_1 & + & -1x_2 & + & -1x_3 & = & 0 \\ 0x_0 & + & 0x_1 & + & 1x_2 & + & 1x_3 & = & 5 \\ 0x_0 & + & 0x_1 & + & -1x_2 & + & 4x_3 & = & 3 \end{array}$$

16

Gaussian Elimination Example

$$\begin{array}{r}
 1x_0 + 0x_1 + 1x_2 + 4x_3 = 1 \\
 0x_0 + -1x_1 + -1x_2 + -1x_3 = 0 \\
 0x_0 + 0x_1 + 1x_2 + 1x_3 = 5 \\
 0x_0 + 0x_1 + 0x_2 + 5x_3 = 8
 \end{array}$$

Gaussian Elimination Example

$$\begin{array}{r}
 1x_0 + 0x_1 + 1x_2 + 4x_3 = 1 \\
 0x_0 + -1x_1 + -1x_2 + -1x_3 = 0 \\
 0x_0 + 0x_1 + 1x_2 + 1x_3 = 5 \\
 0x_0 + 0x_1 + 0x_2 + 5x_3 = 8
 \end{array}$$

$$\begin{array}{r}
 x_3 = 8/5 \\
 x_2 = 5 - x_3 = 17/5 \\
 x_1 = 0 - x_2 - x_3 = -25/5 \\
 x_0 = 1 - x_2 - 4x_3 = -44/5
 \end{array}$$

17

18

Gaussian Elimination: Partial Pivoting

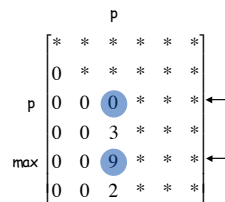
Remark. Previous code fails spectacularly if pivot $a_{pp} = 0$.

Partial pivoting. Swap row p with the row that has biggest entry in column p among unreduced rows $i > p$.

```

// find pivot row
int max = p;
for (int i = p + 1; i < N; i++)
if (Math.abs(A[i][p]) > Math.abs(A[max][p]))
    max = i;

// swap rows i and max of A and b
double[] T = A[p]; A[p] = A[max]; A[max] = T;
double t = b[p]; b[p] = b[max]; b[max] = t;
    
```



Q. What if pivot $a_{pp} = 0$ with partial pivoting?

A. System is over-determined or under-determined.

19

Gaussian Elimination with Partial Pivoting

```

for (int p = 0; p < N; p++) {
    // partial pivoting
    int max = p;
    for (int i = p + 1; i < N; i++)
    if (Math.abs(A[i][p]) > Math.abs(A[max][p]))
        max = i;
    double[] T = A[p]; A[p] = A[max]; A[max] = T;
    double t = b[p]; b[p] = b[max]; b[max] = t;

    // zero out entries of A and b using pivot A[p][p]
    for (int i = p + 1; i < N; i++) {
        double alpha = A[i][p] / A[p][p];
        b[i] -= alpha * b[p];
        for (int j = p; j < N; j++)
            A[i][j] -= alpha * A[p][j];
    }

    // back substitution
    for (int i = N - 1; i >= 0; i--) {
        double sum = 0.0;
        for (int j = i + 1; j < N; j++)
            sum += A[i][j] * x[j];
        x[i] = (b[i] - sum) / A[i][i];
    }
}
    
```

$n^3/3$ additions,
 $n^3/3$ multiplications

$n^2/2$ additions,
 $n^2/2$ multiplications

20

Numerical Unstable Algorithms

Stability. Algorithm $f_1(x)$ for computing $f(x)$ is **numerically stable** if $f_1(x) \approx f(x+\epsilon)$ for *some* small perturbation ϵ .

Nearly the right answer to nearly the right problem.

Ex: Gaussian elimination can fail spectacularly.

$a = 10^{-17}$

$$\begin{aligned} a x_0 + 1 x_1 &= 1 \\ 1 x_0 + 2 x_1 &= 3 \end{aligned}$$

Algorithm	x_0	x_1
No pivoting	0.0	1.0
Partial pivoting	1.0	1.0
Exact	$\frac{1}{1-2a} \approx 1$	$\frac{1-3a}{1-2a} \approx 1$

Lesson: use partial pivoting to improve numerical stability.

21

Stability and Conditioning

Stability. Algorithm $f_1(x)$ for computing $f(x)$ is numerically stable if $f_1(x) \approx f(x+\epsilon)$ for *some* small perturbation ϵ .

Conditioning. Problem is well-conditioned if $f(x) \approx f(x+\epsilon)$ for *all* small perturbation ϵ .

Accuracy depends on stability and conditioning.

- Guaranteed if apply stable algorithm to well-conditioned problem.
- Danger: apply unstable algorithm to well-conditioned problem.
- Danger: apply stable algorithm to ill-conditioned problem.

Numerical analysis. Art and science of designing numerically stable algorithms for well-conditioned problems.

23

Ill-Conditioned Problems

Conditioning. Problem is **well-conditioned** if $f(x) \approx f(x+\epsilon)$ for *all* small perturbation ϵ .

Solution varies gradually as problem varies.

Matrix condition number (Turing, 1948).

- $\kappa(A) = \|A\| \|A^{-1}\|$.
- # bits accuracy in solution \approx # bits in data $- \log_2 \kappa(A)$

Ex: Hilbert matrix.

- H is invertible, but nearly singular.
- $\kappa(H_4) = 28,375$; $\kappa(H_{10}) \geq 2^{45}$.
- Difficult to solve $Hx = b$.

$$H_4 = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

Lesson: must strive to avoid ill-posed problems in practice.

22

Numerical Catastrophes

Ariane 5 rocket (June 4, 1996).

- 10 year, \$7 billion ESA project exploded after launch.
- 64-bit float converted to 16 bit signed int.
- Unanticipated overflow.



Copyright, Arianespace

Vancouver stock exchange (November, 1983).

- Index undervalued by 44%.
- Recalculated index after each trade by adding change in price.
- 22 months of accumulated truncation error.

Patriot missile accident (February 25, 1991).

- Failed to track scud; hit Army barracks, killed 28.
- Inaccuracy in measuring time in 1/20 of a second since using 24 bit binary floating point.



24

Java Matrix Package

JAMA: library for numerical linear algebra.

- Linear systems of equations, least squares.
- Eigenvalues, singular values, matrix decompositions.
- Same algorithms used in LINPACK, EISPACK, MATLAB.

```
import Jama.Matrix;
public class JamaTest {
    public static void main(String[] args) {
        double[][] Adata = { { 0, 1, 1 },
                              { 2, 4, -2 },
                              { 0, 3, 15 } };
        double[][] bdata = { { 4 }, { 2 }, { 36 } };
        Matrix A = new Matrix(Adata);
        Matrix b = new Matrix(bdata);
        Matrix x = A.solve(b);
        x.print(9, 6);
    }
}
```

25

Matrix Laboratory

MATLAB. Numerical linear algebra.

```
fedora.Princeton.EDU% matlab -nodisplay
< M A T L A B > Copyright 1984-2004 The MathWorks, Inc.
>> A = [0, 1, 1; 2, 4, -2; 0, 3, 15]
A =
     0     1     1
     2     4    -2
     0     3    15
>> b = [4; 2; 36]
b =
     4
     2
    36
>> x = A \ b
x =
    -1
     2
     2
>> norm(A*x - b)
ans = 0
```

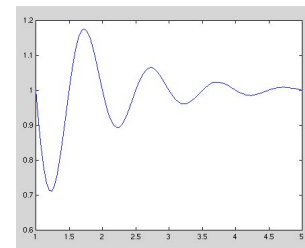
26

MATLAB

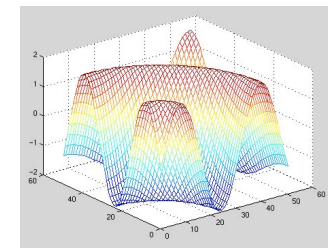
```
>> eig(A)'
ans = -0.3401 4.8812 14.4588
>> [V, D] = eig(A)
V =
   -0.9184   -0.1369    0.0557
    0.3882   -0.9497   -0.1772
   -0.0759    0.2816    0.9826
D =
   -0.3401         0         0
         0    4.8812         0
         0         0   14.4588
>> svd(A)'
ans = 15.3908 4.7978 0.3250
>> t = 0:0.2:0.6
t =
     0     0.2000     0.4000     0.6000
>> x = sin(2*pi*5*t) + sin(2*pi*12*t)
x =
     0     0.5878   -0.9511     0.9511
>> fft(x)
ans =
  0.5878+0i  0.9511+0.3633i -2.4899+0i  0.9511-0.3633i
```

27

MATLAB



```
>> x=linspace(1,5);
>> y=1-exp(-x).*sin(2*pi*x);
>> plot(x,y)
```



```
>> g = 0:0.2:10;
>> [x,y] = meshgrid(g);
>> z = 2*sin(sqrt(x.^2 + y.^2));
>> mesh(z);
```

28

Maple

Maple: symbolic algebra.

```
fedora.Princeton.EDU% maple
Maple 9.5 (IBM INTEL LINUX) Copyright (c) Maplesoft, 2004
> 2^100;
1267650600228229401496703205376
> f := int(x^3 * sin(x), x);
f := -x^3 cos(x) + 3 x^2 sin(x) - 6 sin(x) + 6 x cos(x)
> diff(f, x);
x^3 sin(x)
> int(exp(-x^2), x = 0..infinity);
1/2 Pi^2
> factor(sum(i, i = 1..N));
1/2 N (N + 1)
> fsolve(x^4 * sin(x) + x^3*exp(x) - 1 = 0);
.7306279509
```

Maple

```
> f := x -> x^3 + x^2 - 2;
f := x -> x^3 + x^2 - 2
> simplify(f(f(y)));
y^9 + 3y^8 + 3y^7 - 4y^6 - 10y^5 - 5y^4 + 8y^3 + 8y^2 - 6
> f(8/3);
650/27
> ifactor(123456789876543212345678987654321);
(19) (379) (17144395205741315420869183121)
> A := array([[0, 1, 1], [2, 4, -2], [0, 3, 15]]);
A := [ 0 1 1
       2 4 -2
       0 3 15]
> b := array([4, 3, 17]);
b := [4, 3, 17]
> linsolve(A, b);
[-21/4, 43/12, 5/12]
```

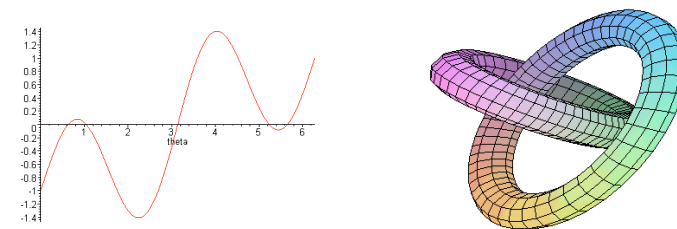
29

30

Maple

```
> (6 + 5*I)^4;
-3479 + 1320 I
> BesselK(1.0, -3.0);
-.04015643113 - 12.41987883 I
> factor(sum(i, i = 1..N));
1/2 N (N + 1)
> ode := diff(y(x), x, x) + 2*diff(y(x), x) + y(x) = exp(-x);
ode := d^2y(x)/dx^2 + 2dy(x)/dx + y(x) = exp(-x)
> dsolve({ode, y(0) = 1, D(y)(0) = 0}, y(x));
y(x) = 1/2 x^2 exp(-x) + exp(-x) + exp(-x) x
> for i from 1 to 600 do
  if isprime(2^i - 1) then print(i);
  fi;
od;
2 3 5 7 13 17 19 31 61 89 107 127 521
> mygcd := proc(p, q) if q = 0 then p
  else mygcd(q, p mod q)
  fi; end;
> mygcd(1440, 408);
24
```

Maple



```
> plot(sin(2*theta) - cos(theta/2),
theta=0..2*Pi);
> K1:=[sin(alpha), cos(alpha), sin(alpha), cos(alpha)];
> K2:=[sin(alpha), cos(alpha), -sin(alpha), -cos(alpha)];
> P:=w->(w-v)[2..4]/(w-v)[1];
> PK1:=P(K1);
> PK2:=P(K2);
> with(plots):
> R1:=tubeplot(PK1,alpha=0..2*Pi,radius=0.1);
> R2:=tubeplot(PK2,alpha=0..2*Pi,radius=0.1);
> display([R1,R2]);
```

31

32

Java, MATLAB, and Maple

Common features.

- Variables.
- Arrays.
- Conditionals and loops.
- Functions and recursion.
- User-defined data types.

Built-in types.

- Java: integer, floating point, boolean, string.
- Matlab: matrices, sparse matrices, floating point.
- Maple: functions, equations, series, operators, matrices.

Differences.

- Compiled vs. interpreted.
- Control over interfaces to outside world.

What To Do When Faced With a New Problem?

What primitive objects are important?

- Numbers, matrices, polynomials, polygons, plots.
- Text, files, programs, pictures, sounds, video.

Which tool should I use? Stick with Java or learn new tool.

Maple: symbolic computing.

Matlab: numerical linear algebra.

TeX: scientific writing.

Perl: string processing.

Gnuplot: scientific plotting.

C: systems level programming.

PHP: web page design.

Upon completion of this course, you are prepared to learn how to use these tools **on your own**.