1.00 Lecture 31

Systems of Linear Equations

Reading for next time: Numerical Recipes, pp. 129-139
http://www.nrbook.com/a/bookcpdf.php

Systems of Linear Equations

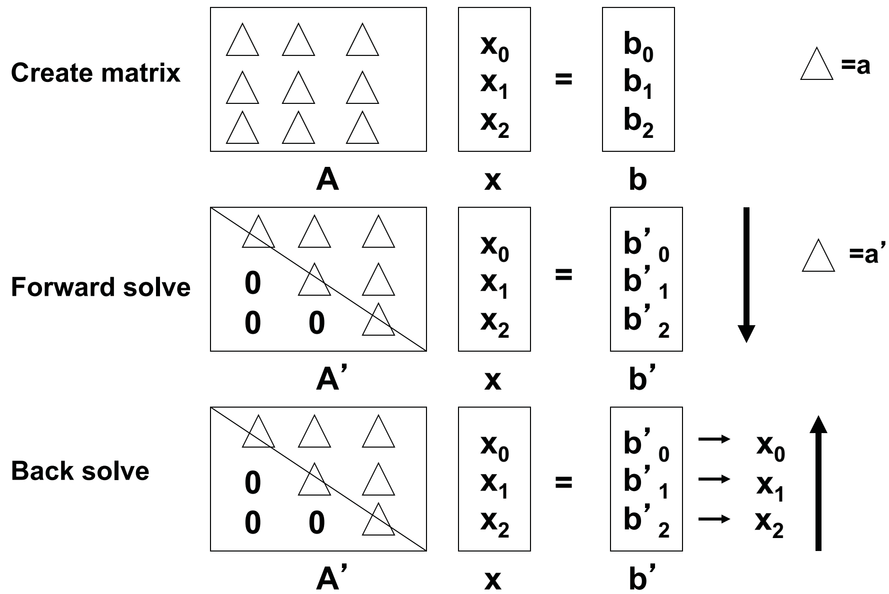
$$
\begin{aligned}
3x_0 + x_1 - 2x_2 &= 5 \\
2x_0 + 4x_1 + 3x_2 &= 35 \\
x_0 - 3x_1 &= -5
\end{aligned}
$$

$$
\begin{vmatrix} 3 & 1 & -2 \\ 2 & 4 & 3 \\ 1 & -3 & 0 \end{vmatrix}
\begin{vmatrix} x_0 \\ x_1 \\ x_2 \end{vmatrix}
=
\begin{vmatrix} 5 \\ 35 \\ -5 \end{vmatrix}
$$

A x = b

3 x 3 3 x 1 3 x 1

Algorithm to Solve Linear System



Gaussian Elimination: Forward Solve

Q=

3	1	-2	5
2	4	3	35
1	-3	0	-5
	A		b

Form Q for convenience
Do elementary row ops:
- Multiply rows
- Add/subtract rows

Make column 0 have zeros below diagonal

Pivot					
	3	1	-2	5	
Pivot= 2/3	0	10/3	13/3	95/3	Row 1' = row 1 - (2/3) row 0
Pivot= 1/3	0	-10/3	2/3	-20/3	Row 2' = row 2 - (1/3) row 0

Make column 1 have zeros below diagonal

Pivot					
	3	1	-2	5	
Pivot= -1 →	0	10/3	13/3	95/3	
	0	0	15/3	75/3	Row 2'' = row 2' + 1 * row 1

Gaussian Elimination: Back Solve

3	1	-2	5
0	10/3	13/3	95/3
0	0	15/3	75/3

$(15/3)x_2=(75/3)$ $x_2 = 5$

3	1	-2	5
0	10/3	13/3	95/3
0	0	15/3	75/3

$(10/3)x_1 + (13/3)*5= (95/3)$ $x_1 = 3$

3	1	-2	5
0	10/3	13/3	95/3
0	0	15/3	75/3

$3x_0 + 1*3 - 2*5 = 5$ $x_0 = 4$

A Complication

0	1	-2	5
2	4	3	35
1	-3	0	-5

Row 1' = row 1 - (2/0) row 0

Exchange rows: put largest pivot element in row:

2	4	3	35
0	1	-2	5
1	-3	0	-5

Do this as we process each column.

If there is no nonzero element in a column, matrix is not full rank.

Gaussian Elimination

```
// In class Matrix, add:

public static Matrix gaussian(Matrix a, Matrix b) {
    int n = a.data.length;                   // Number of unknowns
    Matrix q = new Matrix(n, n + 1);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)          // Form q matrix
            q.data[i][j]= a.data[i][j];
        q.data[i][n]= b.data[i][0];
    }

    forward_solve(q);               // Do Gaussian elimination
    back_solve(q);                  // Perform back substitution

    Matrix x= new Matrix(n, 1);
    for (int i = 0; i < n; i++)
        x.data[i][0]= q.data[i][n];
    return x;
}
```

Forward Solve

```
private static void forward_solve(Matrix q) {
  int n = q.data.length;

  for (int i = 0; i < n; i++) { // Find row w/max element in this
    int maxRow = i;             // column, at or below diagonal
    for (int k = i + 1; k < n; k++)
       if (Math.abs(q.data[k][i]) > Math.abs(q.data[maxRow][i]))
          maxRow = k;

    if (maxRow != i)      // If row not current row, swap
       for (int j = i; j <= n; j++) {
         double t = q.data[i][j];
         q.data[i][j]= q.data[maxRow][j];
         q.data[maxRow][j]= t;
       }

     for (int j = i + 1; j < n; j++) {  // Calculate pivot ratio
       double pivot = q.data[j][i] / q.data[i][i];
       for (int k = i; k <= n; k++)  // Pivot operation itself
         q.data[j][k] -= q.data[i][k] * pivot;
     }
   }
}
```

Back Solve

```
private static void back_solve(Matrix q) {
    int n = q.data.length;

    for (int j = n - 1; j >= 0; j--) {       // Start at last row
        double t = 0.0;                        // t- temporary
        for (int k = j + 1; k < n; k++)
            t += q.data[j][k] * q.data[k][n];
        q.data[j][n]= (q.data[j][n] - t) / q.data[j][j];
    }
}

// GaussTest is in your download for simple tests of this code
```

Variations

Multiple right hand sides: augment Q, solve all eqns at once

3	1	-2	5	7	87
2	4	3	35	75	-1
1	-3	0	-5	38	52

Matrix inversion:

3	1	-2	1	0	0
2	4	3	0	1	0
1	-3	0	0	0	1
A			**I**		

Q (A and I together)

⟶

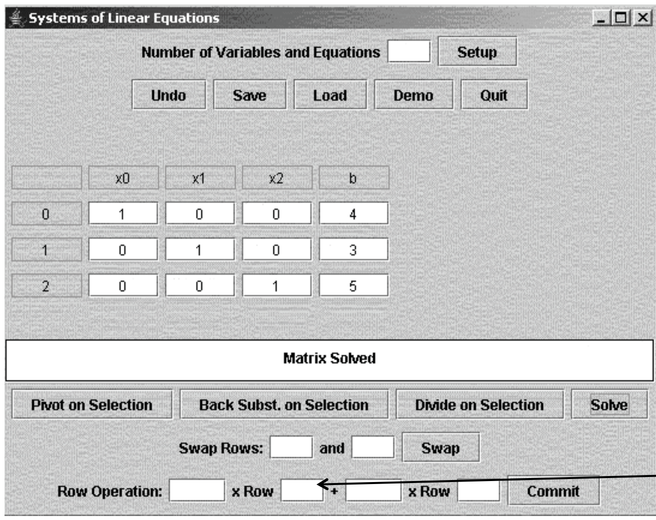
#	#	#	@	@	@
0	#	#	@	@	@
0	0	#	@	@	@
				$\mathbf{A^{-1}}$	

A • ? = I

? = $\mathbf{A^{-1}}$

Exercise

- **Download GElim and Matrix**
- **Compile and run GElim:**



Replace row with result

Exercise

- **Experiment with the following 3 systems:**
  - **Use pivot, back subst, divide on selection, etc. <u>not solve</u>**

System 1: The 3x3 matrix example in the previous slides. Click on "Demo" to load it.

System 2:

4	6	-3	10
2	5	9	12
8	8	-27	6

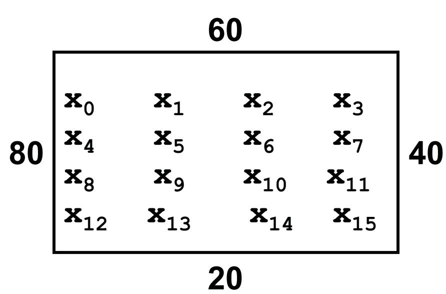
System 3:

12	-13.5	3	0.5	2.75
8	-9	4	2.5	3.5
3	6	1.5	2	4.25
2	1.5	4	12	6

Using Linear Systems

- A common pattern in engineering, scientific and other analytical software:
  - Problem generator (model, assemble matrix)
    - Customized to specific application (e.g. heat transfer)
    - Use matrix multiplication, addition, etc.
  - Problem solution (system of simultaneous linear equations)
    - Usually “canned”: either from library or written by you for a library
  - Output generator (present result in understandable format)
    - Customized to specific application (often with graphics, etc.)
- We did a pattern earlier: model-view-controller

Heat Transfer Exercise



4 by 4 grid of points on the plate produces 16 unknown temperatures x_0 through x_{15}

$$T = (T_{left} + T_{right} + T_{up} + T_{down})/4$$

Edge temperatures are known; interior temperatures are unknown
This produces a 16 by 16 matrix of linear equations

Heat Transfer Equations

- **Node 0:**
  $x_0 = (80 + x_1 + 60 + x_4)/4$ $\quad$ $4x_0 - x_1 - x_4 = 140$
- **Node 6:**
  $x_6 = (x_5 + x_7 + x_2 + x_{10})/4$ $\quad$ $4x_6 - x_5 - x_7 - x_2 - x_{10} = 0$
- **Interior node:**
  $x_i = (x_{i-1} + x_{i+1} + x_{i-n} + x_{i+n})/4$ $\quad$ $4x_i - x_{i-1} - x_{i+1} - x_{i-n} - x_{i+n} = 0$

Node	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	**4**	**-1**	**0**	**0**	**-1**	**0**	**0**	**0**
1	**-1**	**4**	**-1**	**0**	**0**	**-1**	**0**	**0**
2	**0**	**-1**	**4**	**-1**	**0**	**0**	**-1**	**0**
3	**0**	**0**	**-1**	**4**	**0**	**0**	**0**	**-1**
4	**-1**	**0**	**0**	**0**	**4**	**-1**	**0**	**0**
5	**0**	**-1**	**0**	**0**	**-1**	**4**	**-1**	**0**
6	**0**	**0**	**-1**	**0**	**0**	**-1**	**4**	**-1**
7	**0**	**0**	**0**	**-1**	**0**	**0**	**-1**	**4**

Ax= b

j

i

	A																x	b
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
0	4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	x0	140
1	-1	4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	x1	60
2	0	-1	4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	x2	60
3	0	0	-1	4	0	0	0	-1	0	0	0	0	0	0	0	0	x3	100
4	-1	0	0	0	4	-1	0	0	-1	0	0	0	0	0	0	0	x4	80
5	0	-1	0	0	-1	4	-1	0	0	-1	0	0	0	0	0	0	x5	0
6	0	0	-1	0	0	-1	4	-1	0	0	-1	0	0	0	0	0	x6	0
7	0	0	0	-1	0	0	-1	4	0	0	0	-1	0	0	0	0	x7	40
8	0	0	0	0	-1	0	0	0	4	-1	0	0	-1	0	0	0	x8	80
9	0	0	0	0	0	-1	0	0	-1	4	-1	0	0	-1	0	0	x9	0
10	0	0	0	0	0	0	-1	0	0	-1	4	-1	0	0	-1	0	x10	0
11	0	0	0	0	0	0	0	-1	0	0	-1	4	0	0	0	-1	x11	40
12	0	0	0	0	0	0	0	0	-1	0	0	0	4	-1	0	0	x12	100
13	0	0	0	0	0	0	0	0	0	-1	0	0	-1	4	-1	0	x13	20
14	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	4	-1	x14	20
15	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	4	x15	60

Heat Transfer System

16

16 $\begin{bmatrix} a_{00} & a_{01} & a_{02} & \ldots & a_{0,15} \\ a_{10} & a_{11} & a_{12} & \ldots & a_{1,15} \\ a_{20} & a_{21} & a_{22} & \ldots & a_{2,15} \\ & & \ldots & & \\ a_{15,0} & a_{15,1} & \ldots & a_{15,15} & \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \ldots \\ x_{15} \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \ldots \\ b_{15} \end{bmatrix}$

A　　x　　b

Contains 0, -1, 4 coefficients in (simple) pattern

Known temperatures (often 0 but use edge temperatures when close)

16 unknown interior temperatures

Heat Transfer Result

60

80

66	59	55	50
66	56	50	45
62	50	44	41
50	38	34	34

40

20

Exercise: Ax= b: Fill in green, orange

i=j   j=i+1   j= ___

________ j

j=i-1

j=___

______

______ i

A

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	x	b
0	4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	x0	140
1	-1	4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	x1	60
2	0	-1	4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	x2	60
3	0	0	-1	4	0	0	0	-1	0	0	0	0	0	0	0	0	x3	100
4	-1	0	0	0	4	-1	0	0	-1	0	0	0	0	0	0	0	x4	80
5	0	-1	0	0	-1	4	-1	0	0	-1	0	0	0	0	0	0	x5	0
6	0	0	-1	0	0	-1	4	-1	0	0	-1	0	0	0	0	0	x6	0
7	0	0	0	-1	0	0	-1	4	0	0	0	-1	0	0	0	0	x7	40
8	0	0	0	0	-1	0	0	0	4	-1	0	0	-1	0	0	0	x8	80
9	0	0	0	0	0	-1	0	0	-1	4	-1	0	0	-1	0	0	x9	0
10	0	0	0	0	0	0	-1	0	0	-1	4	-1	0	0	-1	0	x10	0
11	0	0	0	0	0	0	0	-1	0	0	-1	4	0	0	0	-1	x11	40
12	0	0	0	0	0	0	0	0	-1	0	0	0	4	-1	0	0	x12	100
13	0	0	0	0	0	0	0	0	0	-1	0	0	-1	4	-1	0	x13	20
14	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	4	-1	x14	20
15	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	4	x15	60

N

W

E

S

Heat Transfer Exercise, p.1

```
public class Heat {                   // Problem generator
  public static void main(String[] args) {
    double Te= 40.0, Tn=60.0, Tw=80.0, Ts=20.0; // Edge temps
    int col= 4;     // Te - east, Tn - north, Tw- west, Ts-south
    int row= 4;
    int n= col * row;
    Matrix a= new Matrix(n,n);
    for (int i=0; i < n; i++)
      for (int j=0; j < n; j++) {
        if (i==j)                  // Diagonal element (yellow)
          a.setElement(i, j, 4.0);
        else if (…)
         // Complete this code in step 1:
         //    Green elements (4, or col, away from diagonal)
         //    Blue elements (1 away from diagonal)
         //    Set blue and skip orange where we go to
         //    the next row on the actual plate
        //     Relate i and j to determine blue, orange cells
        //     using the diagram on the previous slide
    // Continued on next slide
```

Heat Transfer Exercise, p.2

```
Matrix b= new Matrix(n, 1);     // Known temps
for (int i=0; i < n; i++) {
  if (i < col)                  // Next to north edge
    b.incrElement(i, 0, Tn);    // incrElement, not setElement
  if (…) // Step 2
  // Complete this code for the other edges; no 'elses'
  // Add edge temperature to b; you may add more than one
  // Look at the Ax=b example slide to find the pattern
  // Use i, col, row to determine cells at the edge
}
Matrix x= Matrix.gaussian(a, b);           // Problem solution

System.out.println("Temperature grid:");   // Output generator
for (int i=0; i< row; i++) {
 for (int j=0; j < col; j++)
   System.out.print(Math.round(x.getElement((i*row+j),0)+" ");
 System.out.println();
}
}
```

Linear Systems

$$a_{00}x_0 + a_{01}x_1 + a_{02}x_2 + \ldots + a_{0,n-1}x_{n-1} = b_0$$

$$a_{10}x_0 + a_{11}x_1 + a_{12}x_2 + \ldots + a_{1,n-1}x_{n-1} = b_1$$

…

$$a_{m-1,0}x_0 + a_{m-1,1}x_1 + a_{m-1,2}x_2 + \ldots + a_{m-1,n-1}x_{n-1} = b_{m-1}$$

- **If n=m, we try to solve for a unique set of x. Obstacles:**
  - **If any row (equation) or column (variable) is linear combination of others, matrix is degenerate or not of full rank. No solution. Your underlying model is probably wrong; you'll need to fix it.**
  - **If rows or columns are nearly linear combinations, roundoff errors can make them linearly dependent during computations. You'll fail to find a solution, even though one may exist.**
  - **Roundoff errors can accumulate rapidly. While you may get a solution, when you substitute it into your equation system, you'll find it's not a solution. (Right sides don't quite equal left sides.)**

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012