Recitation 11

Matrices, Linear Systems, Integration
Outline

• Matrices
• Linear Equations
• Integration
Represent matrices as two dimensional arrays

\[ a \text{ is a 1-D array of references to 1-D arrays of data.} \]

\[
\begin{align*}
\text{int}[][] & \quad \text{a} = \text{new int}[3][4]; \\
& \quad \text{a}[0][0] = 1; \\
& \quad \text{a}[0][1] = 2; \\
& \quad \text{// ...} \\
& \quad \text{b} = \text{a}[0][3]; \\
& \quad \text{c} = \text{a}[1]; \\
\end{align*}
\]

No. of Columns: \( \text{a}[0].\text{length} \)

No. of Rows: \( \text{a}.\text{length} \)
Matrix Representation

• You can create 2-D arrays manually or use Matrix class

• The Matrix class has methods for setting elements, adding, subtracting, and multiplying matrices, and forming an identity matrix.

```java
public static void main(...) {
    int[][] a = new int[3][4];
    a[0][0] = 1;
    a[0][1] = 2;
    ...
    a[2][3] = 12;
    int b = a[0][2];
    int[] c = a[1];
}
```
Matrix Exercise

• Add a method to `Matrix` to compute the transpose of a matrix

```java
public class Matrix {
    private double[][] data;
    public Matrix(int m, int n) {data = new double[m][n];}
    public int getNumRows() {return data.length;}
    public int getNumCols() {return data[0].length;}
    public double getElement(int i, int j) {
        return data[i][j];
    }
    public void setElement(int i, int j, double val) {
        data[i][j] = val;
    }
}
```
Linear Systems

- Matrices used to represent systems of linear equations
- Assume coefficients $a$ and $b$ are known, $x$ is unknown
- There $n$ unknowns ($x_0$ to $x_{n-1}$) and $m$ equations

$$
\begin{align*}
  a_{00}x_0 + a_{01}x_1 + a_{02}x_2 + \ldots + a_{0,n-1}x_{n-1} &= b_0 \\
  a_{10}x_0 + a_{11}x_1 + a_{12}x_2 + \ldots + a_{1,n-1}x_{n-1} &= b_1 \\
  \vdots \\
  a_{m-1,0}x_0 + a_{m-1,1}x_1 + a_{m-1,2}x_2 + \ldots + a_{m-1,n-1}x_{n-1} &= b_{m-1}
\end{align*}
$$

$$
\begin{pmatrix}
  a_{00} & a_{01} & a_{02} & a_{03} & \ldots & a_{0,n-1} \\
  a_{10} & a_{11} & a_{12} & a_{13} & \ldots & a_{1,n-1} \\
  a_{20} & a_{21} & a_{22} & a_{23} & \ldots & a_{2,n-1} \\
  \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  a_{m-1,0} & a_{m-1,1} & a_{m-1,2} & a_{m-1,3} & \ldots & a_{m-1,n-1}
\end{pmatrix}
\begin{pmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  \vdots \\
  x_{n-1}
\end{pmatrix}
= 
\begin{pmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  \vdots \\
  b_{m-1}
\end{pmatrix}
$$

$$
A x = b
$$
Linear Systems

Solve using Gaussian Elimination: forward solve, backward solve

Problem Statement

Eliminate $x$ from $L_2, L_3$

Eliminate $y$ from $L_3$

Now backward solve: find $z$ from $L_3$, $y$ from $L_2$, $x$ from $L_1$

Source: Wikipedia, "Gaussian elimination" License CC BY-SA. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/fairuse.
Linear System Exercise

• La Verde’s bakes muffins and donuts using flour and sugar.
• Same profit for one muffin and one donut.

How many donuts, muffins to maximize profit?

<table>
<thead>
<tr>
<th>Ingredient</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flour</td>
<td>20 kg</td>
</tr>
<tr>
<td>Sugar</td>
<td>15 kg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Flour Needed</th>
<th>Sugar Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muffin</td>
<td>100 g</td>
<td>50 g</td>
</tr>
<tr>
<td>Donut</td>
<td>75 g</td>
<td>75 g</td>
</tr>
</tbody>
</table>
Linear System Exercise

• Model as system of equations:

\[100m + 75d = 20000 \quad \leftarrow \text{flour constraint}\]
\[50m + 75d = 15000 \quad \leftarrow \text{sugar constraint}\]

• Create the matrices (in the form of \(Ax=b\))

• **Use** `Matrix.setElement()` and `Matrix.gaussian()`

\[
\begin{bmatrix}
100 & 75 \\
50 & 75
\end{bmatrix}
\begin{bmatrix}
m \\
d
\end{bmatrix}
=
\begin{bmatrix}
20000 \\
15000
\end{bmatrix}
\]
Integration

• We use objects to represent mathematical functions in Java
• Each function has its own class
• Each class implements MathFunction

```java
public interface MathFunction{
    public double f(double x);
}

public class LinearF implements MathFunction {
    public double f(double x){
        return 2 * x + 3;
    }
}
```

\[ f(x) = 2x + 3 \]
Integration

Rectangular Rule

\[ A = f(x_R) \, dx \]

Trapezoidal Rule

\[ A = \frac{f(x_R) + f(x_L)}{2} \, dx \]

Simpson's Rule

\[ A = \frac{f(x_R) + 4f(x_m) + f(x_L)}{6} \, dx \]
Improved Trapezoidal Rule

Keep cutting intervals in half until desired accuracy is met.

Function evaluations are stored to avoid re-computation.
Integration Exercise

Compute the shaded surface area using Monte Carlo integration with 1,000,000 random points.

Use `MonteCarloIntegration.java` from lecture as a starting point.
Integration Exercise

Bounding Area = 2r * (4r - a)

(x,y) is in left circle if \((x + r - a/2)^2 + y^2 < 1\)

(x,y) is in right circle if \((x - 1 + a/2)^2 + y^2 < 1\)

Write a method to find area when \(r = 1\)
Problem Set 10

- Find currents and voltages in resistor/battery network
- Build a matrices for resistor values, voltages
- Solve for currents. Use Matrix class from lecture.