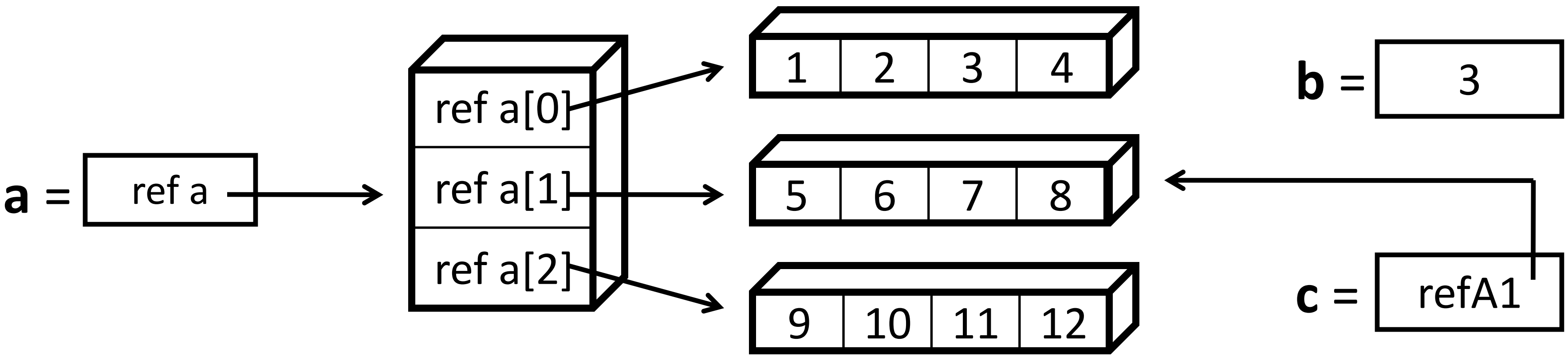
Recitation 11

Matrices, Linear Systems, Integration

Outline

- Matrices
- Linear Equations
- Integration

Matrix Representation




```
int[][] a = new int[3][4];
a[0][0] = 1;
a[0][1] = 2;
//   ...
b = a[0][3];
c = a[1];
```

No. of Columns: `a[0].length`

No. of Rows: `a.length`

Represent matrices as two dimensional arrays

`a` *is a 1-D array of references to 1-D arrays of data.*

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.

```
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

```
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

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

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

(m rows x n cols) (n x 1) = (m x 1)

Ax=b

Linear Systems

Solve using Gaussian Elimination: forward solve, backward solve

Problem Statement

$$\begin{aligned} 2x + y - z &= 8 && (L_1) \\ -3x - y + 2z &= -11 && (L_2) \\ -2x + y + 2z &= -3 && (L_3) \end{aligned}$$

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

Eliminate x from L_2, L_3

$$\begin{aligned} 2x + y - z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ 2y + z &= 5 \end{aligned}$$

Eliminate y from L_3

$$\begin{aligned} 2x + y - z &= 8 \\ \frac{1}{2}y + \frac{1}{2}z &= 1 \\ -z &= 1 \end{aligned}$$

$$\left[\begin{array}{ccc|c} 1 & \frac{1}{3} & \frac{-2}{3} & \frac{11}{3} \\ 0 & 1 & \frac{2}{5} & \frac{13}{5} \\ 0 & 0 & 1 & -1 \end{array}\right]$$

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

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?

	Flour Needed	**Sugar Needed**
Muffin	100 g	50 g
Donut	75 g	75 g

Ingredient	**Supply**
Flour	20 kg
Sugar	15 kg

Linear System Exercise

- Model as system of equations:

  100m + 75d = 20000 ← flour constraint

  50m + 75d = 15000 ← sugar constraint

- Create the matrices (in the form of Ax=b)
- Use `Matrix.setElement()` and `Matrix.gaussian()`

$$\begin{array}{ccccc} \mathbf{A} & & \mathbf{x} & = & \mathbf{b} \\ \begin{bmatrix} 100 & 75 \\ 50 & 75 \end{bmatrix} & * & \begin{bmatrix} m \\ d \end{bmatrix} & = & \begin{bmatrix} 20000 \\ 15000 \end{bmatrix} \end{array}$$

Integration

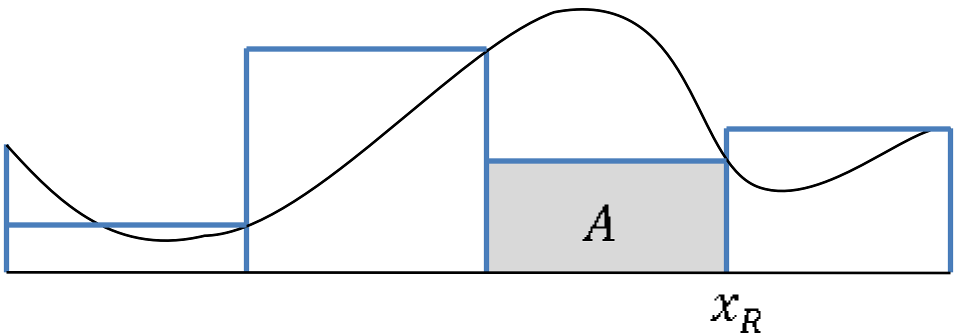
- We use objects to represent mathematical functions in Java
- Each function has its own class
- Each class implements `MathFunction`

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

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

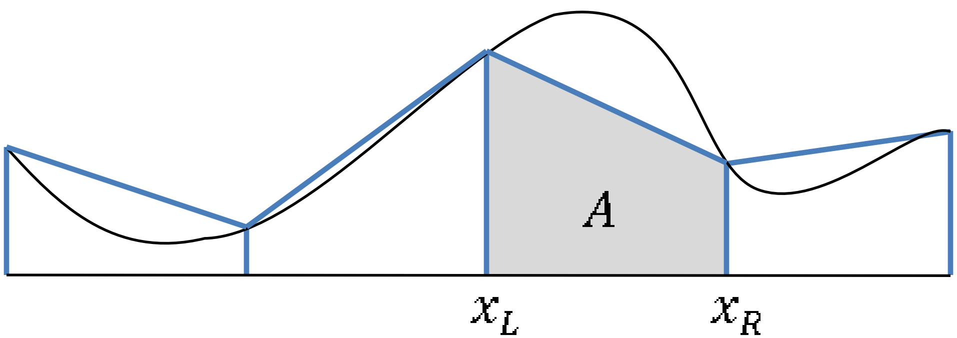
$f(x) = 2x + 3$

Integration



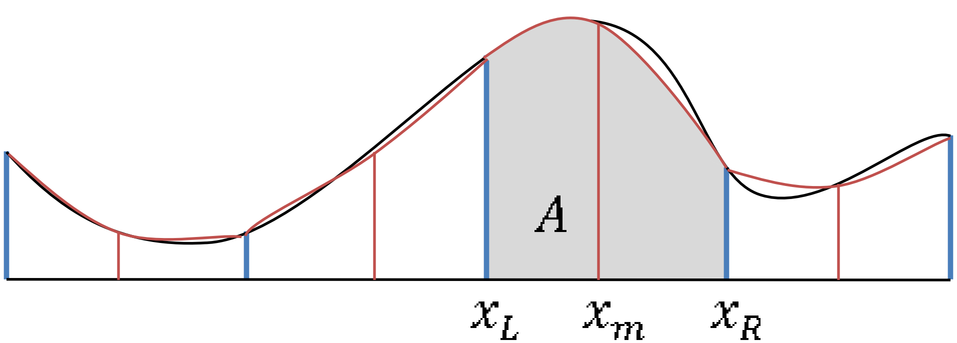
Rectangular Rule

$$A = f(x_R)\,dx$$



Trapezoidal Rule

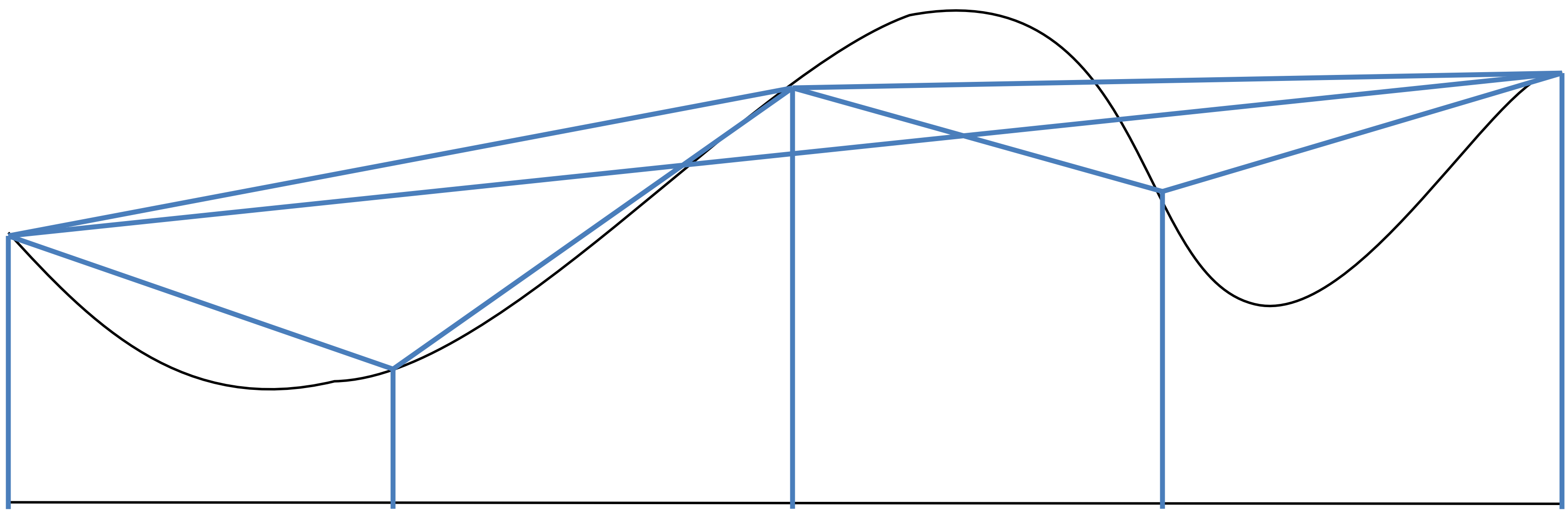
$$A = \frac{f(x_R) + f(x_L)}{2}\,dx$$



Simpson's Rule

$$A = \frac{f(x_R) + 4f(x_m) + f(x_L)}{6}\,dx$$

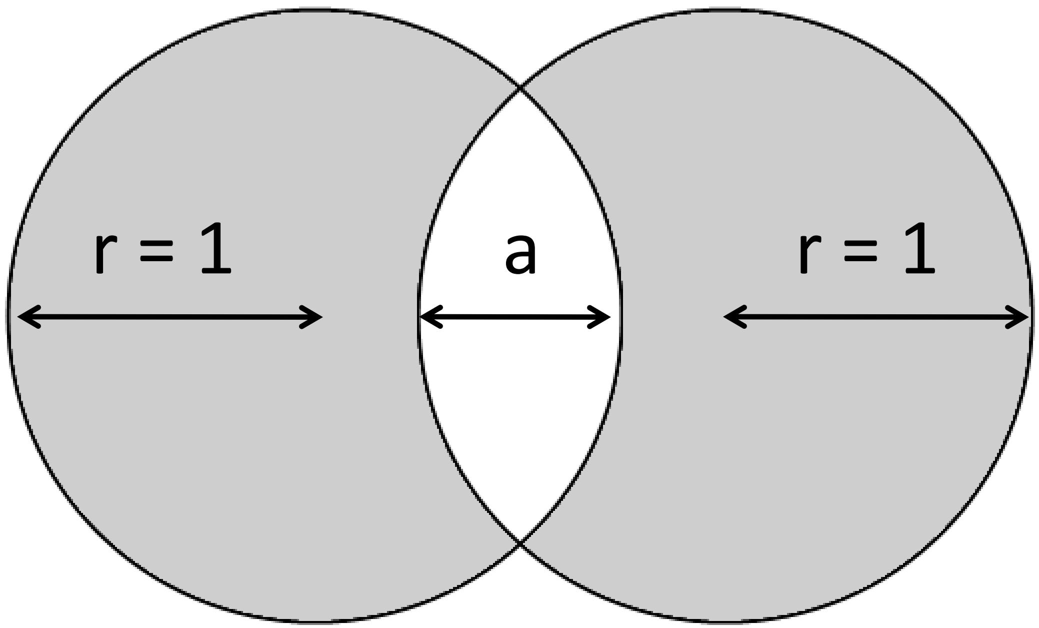
Improved Trapezoidal Rule



Keep cutting intervals in half until desired accuracy is met.

Function evaluations are stored to avoid re-computation.

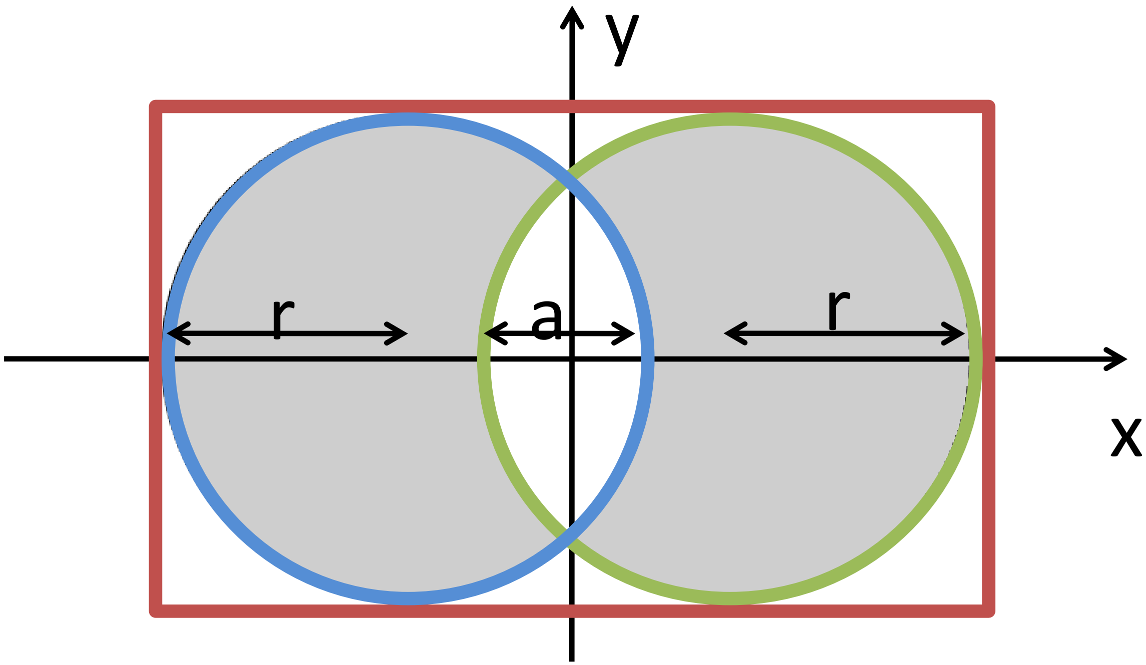
Integration Exercise



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



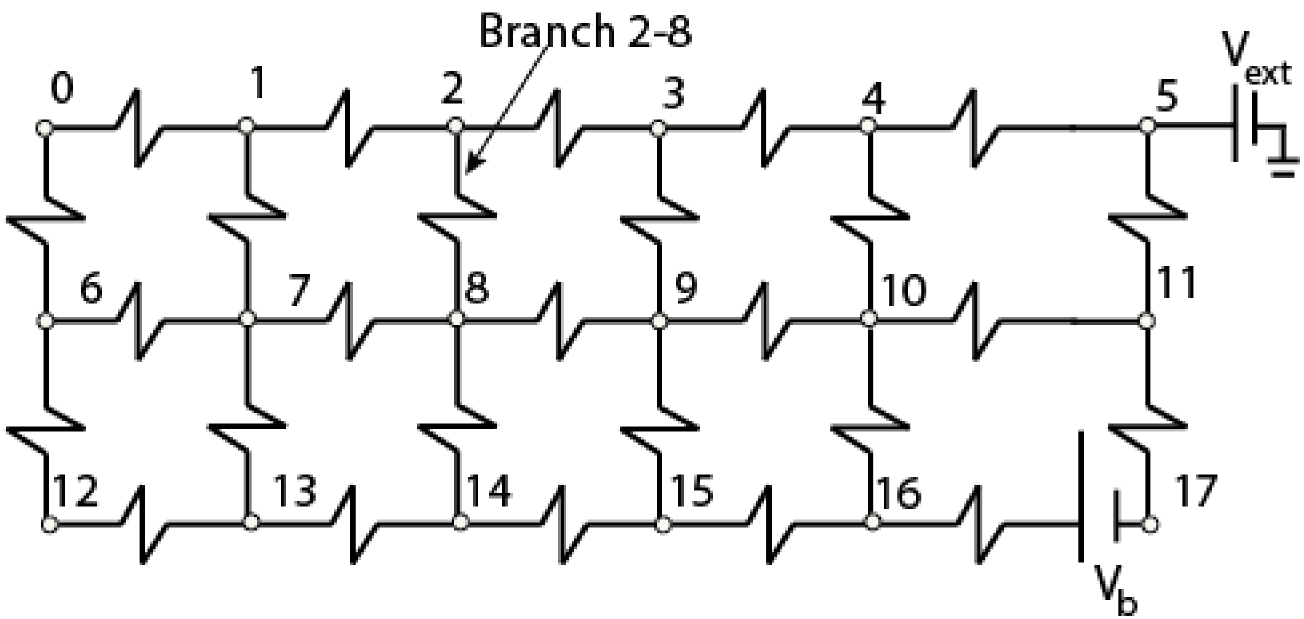
Bounding Area = 2r * (4r - a)

(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.