

2.6 Functions

Functions (Static Methods)

Java function.

- Takes zero or more input arguments.
- Returns one output value.

Applications.

- ➔ Scientists use mathematical functions to calculate formulas.
- Computer programmers use functions to build modular programs.

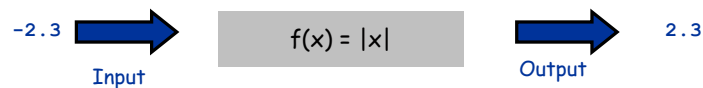
Examples.

- Built-in functions: `Math.random`, `Math.sqrt`, `Integer.parseInt`.
- COS 126 library functions: `Turtle.spot`, `StdIn.readInt`.
- User-defined functions: `main`.

Mathematical Functions

Scientists use mathematical functions to calculate formulas.

- Use built-in functions when possible.
- Build your own when not available.



```
static double abs(double x) {  
    if (x >= 0.0) return x;  
    else return -x;  
}
```

user-defined function

Build A Library of Mathematical Functions

Java makes it easy to build your own libraries.

- Can substitute `MyMath.sqrt` for `Math.sqrt`.
- Can add functionality not in `Math` library.

```
public class MyMath {  
    static double abs(double x) {   
        if (x >= 0.0) return x;   
        else return -x;   
    }   
    static double sqrt(double c) {   
        double t = c;   
        while (t - c/t > 0.0000000000000001)   
            t = (c/t + t) / 2.0;   
        return t;   
    }   
    static int random(int N) {   
        return (int) (Math.random() * N);   
    }   
}
```

Annotations in the code block:
- A green arrow points to the `abs` method with the text "absolute value".
- A green arrow points to the `sqrt` method with the text "Newton's method".
- A green arrow points to the `random` method with the text "pseudo-random integer between 0 and N-1".

Use the Library in a Program

To use the `MyMath` library.

- Put a copy of `MyMath.java` in current directory and compile it.
- Write a client program that uses it.

```
public class Die {
    public static void main(String args[]) {
        die r = 1 + MyMath.random(6);
        System.out.println(r);
    }
}
```

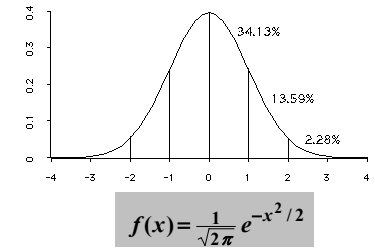
```
% javac Die.java
% java Die
5
% java Die
2
```

5

Gaussian Distribution

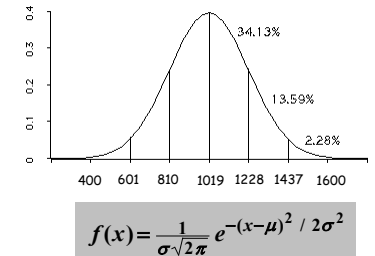
Gaussian distribution.

- "Standard Normal distribution."
- "Bell curve."
- Basis of most statistical analysis in social and physical sciences.



Ex: 2000 SAT scores.

- Follows a Gaussian distributed with:
 - mean $\mu = 1019$
 - standard deviation $\sigma = 209$
- 2.28% get above 1437.

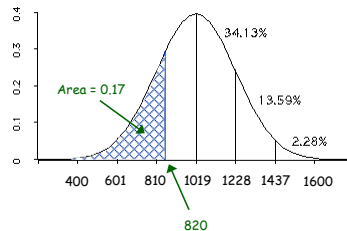


6

SAT Scores

NCAA requires at least 820 for Division I athletes. What fraction of test takers in 2000 do not qualify?

- $\Phi(820, \mu, \sigma) = 0.17051\dots$
- approximately 17%.



Challenge: write a Java program to compute this.

7

Gaussian Distribution

Why relevant in the sciences?

- Models a wide range of natural phenomena and random processes.
 - weights of humans, heights of trees in a forest, exam scores, investment returns

Why relevant in mathematics?

- Central limit theorem.
 - under very general conditions, average of any set of variables tends to the Gaussian distribution

Caveat.

- "Everybody believes in the exponential law of errors: the experimenters, because they think it can be proved by mathematics; and the mathematicians, because they believe it has been established by observation." - Lippman

8

SAT Client

```
public class SAT {
    public static void main(String args[]) {
        double mu      = 1019;
        double sigma   = 209;
        double z       = 820;
        double fraction = MyMath.Phi(z, mu, sigma);
        System.out.println(fraction);
    }
}
```

```
% java SAT
0.17050967431793962
```

9

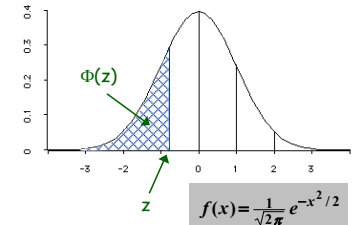
Gaussian Distribution

Compute Gaussian cumulative distribution function.

- No closed form expression and **not in Java library**.
- Express in terms of error function.

$$\Phi(z) = \int_{-\infty}^z f(x) dx$$

$$= \frac{1 + \operatorname{erf}(z/\sqrt{2})}{2}$$



Error function.

- If $z \geq 0$,

$$\operatorname{erf}(z) = \int_0^z \frac{2}{\sqrt{\pi}} e^{-x^2} dx$$

$$\approx 1 - te^{(-z^2 - 1.26551223 + 1.00002368t + 0.37409196t^2 + 0.09678418t^3 - 0.18628806t^4 + 0.27886807t^5 - 1.13520398t^6 + 1.48851587t^7 - 0.82215223t^8 + 0.17087277t^9)}$$

$$\text{where } t = \frac{1}{1+z/2}$$

- Else $\operatorname{erf}(z) = -\operatorname{erf}(-z)$.

Fractional error less than 1.2×10^{-7} .
Reference: Numerical Recipes 6.2

10

Building Layers of Abstraction

Functions enable you to build a new layer of abstraction.

- Takes you beyond pre-packaged libraries.
- You build the tools you need.
 - Ex: erf and Φ

Process.

- Step 1: identify a useful feature.
- Step 2: implement it.
- Step 3: use it.

- Step 3': re-use it in ANY of your programs.

11

Computing erf(z) and $\Phi(z)$ in Java

```
static double erf(double z) {
    double t = 1.0 / (1.0 + 0.5 * Math.abs(z));
    double ans = 1 - t * Math.exp(-z*z - 1.26551223 +
        t * ( 1.00002368 +
            t * ( 0.37409196 +
                Horner's method =>
                t * ( 0.09678418 +
                    t * (-0.18628806 +
                        t * ( 0.27886807 +
                            t * (-1.13520398 +
                                t * ( 1.48851587 +
                                    t * (-0.82215223 +
                                        t * ( 0.17087277))))))))));
    if (z >= 0) return ans;
    else return -ans;
}
```

```
static double Phi(double z) {
    return 0.5 * (1.0 + erf(z / (Math.sqrt(2.0))));
}
```

```
static double Phi(double z, double mu, double sigma) {
    return Phi((z - mu) / sigma);
}
```

overloaded
methods

12

Black-Scholes

Black-Scholes option pricing model.

- Option = right to buy stock at some future date for fixed price.
- Model stock price with stochastic differential equation.
 - stock price follows geometric Brownian motion
 - risk free interest rate is constant and known
 - no dividends
 - markets are efficient
 - no commissions
 - no arbitrage
- Won 1997 Nobel Prize in Economics.
- How much is a given option worth?

$$\frac{\partial f}{\partial t} + rS \frac{\partial f}{\partial S} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} - rf = 0$$

$$S\Phi(d_1) - Xe^{-rT}\Phi(d_2)$$

$$d_1 = \frac{\ln(S/X) + (r + \frac{1}{2}\sigma^2)T}{s\sqrt{T}}$$

$$d_2 = d_1 - s\sqrt{T}$$

Parameter	Description
S	current price of stock
X	strike price
r	risk free interest rate
σ	standard deviation of stock return (volatility)
T	time until option expires

13

Black-Scholes

Java implementation.

- Reuse `MyMath` library just like `Math` and `Turtle`.

```
public class BlackScholes {
    static double call(double S, double X, double r, double sigma, double T) {
        double d1 = (Math.log(S/X) + (r + sigma * sigma/2) * T) /
            (sigma * Math.sqrt(T));
        double d2 = d1 - sigma * Math.sqrt(T);
        return S * MyMath.Phi(d1) - X * Math.exp(-r * T) * MyMath.Phi(d2);
    }
    public static void main(String args[]) {
        double S = Double.parseDouble(args[0]);
        double X = Double.parseDouble(args[1]);
        double r = Double.parseDouble(args[2]);
        double sigma = Double.parseDouble(args[3]);
        double T = Double.parseDouble(args[4]);
        System.out.println(call(S, X, r, sigma, T));
    }
}
```

invokes library function

14

Black-Scholes

Does Black-Scholes accurately model option price?

Parameter	Microsoft (June 9, 2003)	General Electric (June 9, 2003)
S	\$23.75	\$30.14
X	\$15.00	\$15.00
r	1%	1%
σ	35.0%	33.2%
T	0.5 years	0.25 years

← historical estimate

```
% java BlackScholes 23.75 15.0 0.01 0.350 0.50
8.879159279691955 actual option price = $9.10

% java BlackScholes 30.14 15.0 0.01 0.332 0.25
15.177462481562186 actual option price = $14.50
```

15

Functions (Static Methods)

Java function.

- Takes zero or more input arguments.
- Returns one output value.

Applications.

- Scientists use mathematical functions to calculate formulas.
- Computer programmers use functions to build modular programs.

Examples.

- Built-in functions: `Math.random`, `Math.sqrt`, `Integer.parseInt`.
- COS 126 library functions: `Turtle.spot`, `StdIn.readInt`.
- User-defined functions: `main`.

16

Modular Programming

Computer programmers use functions to build modular programs.

- Divide program into self-contained pieces.
- Test each piece individually.
- Combine pieces to make program.

Ex: gambler's ruin.

- Method 1: put everything in `main`.
- Method 2: break up program into self-contained pieces.
 - flip a coin
 - play the game once
 - repeat the game and tabulate statistics

17

Gambler's Ruin Revisited

```
public class Gambler {
    static boolean flip() { ← flip fair coin
        return Math.random() < 0.5;
    }

    static boolean winsGamble(int stake, int goal) {
        while ((stake > 0) && (stake < goal))
            if (flip()) stake++;
            else stake--;
        return (stake == goal);
    }
    play the game once

    public static void main(String[] args) {
        int stake = Integer.parseInt(args[0]);
        int goal = Integer.parseInt(args[1]);
        int N = Integer.parseInt(args[2]);
        int wins = 0;
        for (int i = 0; i < N; i++)
            if (winsGamble(stake, goal)) wins++;
        System.out.println(wins + " wins of " + N);
    }
}
```

18

Craps

What is probability of winning a "pass bet" in craps?

- Roll two dice, and let x be sum.
- If x is 7 or 11, you win instantly.
- Else if x is 2, 3, or 12, you lose instantly.
- Otherwise repeatedly roll two dice until sum is 7 or x .
 - if sum is x , you win
 - if sum is 7 you lose



```
public class Craps {
    . . .
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int wins = 0;
        for (int i = 0; i < N; i++)
            if (winsPassBet()) wins++;
        System.out.println("Win % = " + 1.0 * wins / N);
    }
}
```

Monte Carlo simulation

19

Craps: Helper Functions

```
static int sumOfTwoDice(int sides) {
    int x = 1 + MyMath.random(sides);
    int y = 1 + MyMath.random(sides);
    return x + y;
}

static boolean winsPassBet() {
    int x = sumOfTwoDice(6);
    if (x == 7 || x == 11) return true;
    if (x == 2 || x == 3 || x == 12) return false;
    while (true) {
        int y = sumOfTwoDice(6);
        if (y == 7) return false;
        if (y == x) return true;
    }
}
```

```
% java Craps 1000000
Win % = 0.493396
```

```
% java Craps 1000000
Win % = 0.492537
```

20

Numerically Solving a System of Differential Equations

Lorenz attractor.

- Idealized atmospheric model to describe turbulent flow.
- Convective rolls: warm fluid at bottom, rises to top, cools off, and falls down.
- Rayleigh-Bernard cell.

$$\begin{aligned}\frac{dx}{dt} &= -\alpha x + \sigma y \\ \frac{dy}{dt} &= -xz + rx - y \\ \frac{dz}{dt} &= xy - bz\end{aligned}$$

x = fluid flow velocity
 y = temperature difference between ascending and descending currents
 z = distortion of vertical temperature profile from linearity

$\sigma = 10$ = Prandtl number
 $b = 8 / 3$ = width-to-height ratio of convective layer
 $r = 28$ = temperature difference between top and bottom

21

Euler's Method

Euler's method: to numerically solve differential equations.

- Choose Δt small.
- Update.
- Repeat.

$$\begin{aligned}x_{t+1} &= x_t + \Delta t \frac{dx}{dt}(x_t, y_t, z_t) \\ y_{t+1} &= y_t + \Delta t \frac{dy}{dt}(x_t, y_t, z_t) \\ z_{t+1} &= z_t + \Delta t \frac{dz}{dt}(x_t, y_t, z_t)\end{aligned}$$

4th order Runge-Kutta method: more accurate approximation.

- See COS 323.

22

Animation of the Lorenz Attractor

```
public class Lorenz {
    public static double dx(double x, double y, double z)
    { return -10*(x - y); }
    public static double dy(double x, double y, double z)
    { return -x*z + 28*x - y; }
    public static double dz(double x, double y, double z)
    { return x*y - 8*z/3; }

    public static void main(String[] args) {
        double x = 0.0, y = 20.0, z = 25.0;
        double dt = 0.001;
        Turtle.create(512, 512);
        while (true) {
            double xnew = x + dt * dx(x, y, z);
            double ynew = y + dt * dy(x, y, z);
            double znew = z + dt * dz(x, y, z);
            x = xnew; y = ynew; z = znew;
            Turtle.pixel(10*(x + 25), 10*z);
            Turtle.pause(10);
        }
    }
}
```

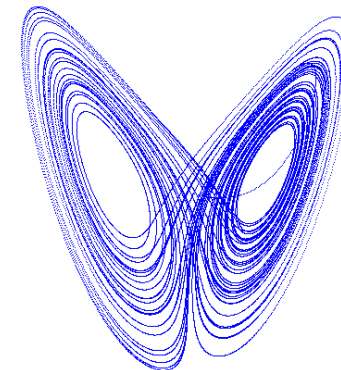
← Euler's method

← plot x vs. z, scaled appropriately

23

The Lorenz Attractor

```
% java Lorenz
```



24

Butterfly Effect

Experiment.

- Initialize $y = 20.01$ instead of $y = 20$.
- Plot original trajectory in blue, perturbed one in magenta.
- What happens?

Chaos.

- Sensitive dependence on initial conditions.
- Unpredictability of aperiodic systems like the weather.
- Lorenz attractor never repeats itself, but produces orderly pattern.

Does the flap of a butterfly's wings in Brazil set off a tornado in Texas?

-- Edward Lorenz, 1972

Functions

Why use functions?

- Makes code easier to understand.
- Makes code easier to debug.
- Makes code easier to maintain.
- Makes code easier to re-use.