2012 Basic Numerical Analysis

Numerical integration and Ordinary Differential Equations (常微分方程式の解法)

Contents

- Basics of C language part 2
- Integration of simple functions
- Errors
  truncation error, round-off error
- Numerical stability
- Euler's method, Runge-Kutta method
- (Next week) Monte Carlo methods

Programming in C

(with some recap)

Functions

Declaration and definition

```
double factorial(int);

int main()

{

   int  i;   double fac;

   for(i=0; i <10; i++)

        fac = factorial(j);

}

double factorial(int n)

{     ......
```

Functions

Declaration and definition

```
double factorial(int);
int main()
{
   int  i;   double fac;
   for(i=0; i <10; i++)
        fac = factorial(j);
}
double factorial(int n)
{     ......
```

prototype declaration

definition

Control statements

if (expression) statement1 else ...

```
{
   double e;
   e = b*b - 4.0 * a * c;
   if(e < 0){
       cout<<“Error”<<endl;      // terminal print out
       exit(332);
    }else{
      x = - b + sqrt(e);
    }
}
```

Control statements

while (expression) statement1

```
{
   double e, q;
   e = func_ex(q);
   while(e < 10.0){
        q = pow(10.0, 0.1 * e);
        e = func_ex(q);
   }
}
```

Recall Newton's method with accuracy ε

Control statements

do statement while (expression)

```
{
   double e, q;
   do {
        q = pow(10.0, 0.1 * e);
        e = func_ex(q);
   } while(e < 10.0);
}
```

The do part execute at least once.

Control statements

switch (expression) case ...

```
{
  int  n;
  n=func_hugou(x);
  switch (n) {
      case A:   statement 1;   // if n == A
              break;
      case B:   statement 2;  // if n == B
              break;
      default:  cout<<“ Error”<<endl;  // otherwise
  }
```

Terminal I/O

Input

```
int main()
{  int  n;   double x, y;
   printf(“ Initial values ? ¥n”);
   scanf(“%d %lf %lf”, &n, &x, &y);
}
```

Output

```
{
    printf(“Results: m=%d a=%9.4f  b=%13.7f”, m, a, b);
}
```

Blackboard

Trapezoid, Simpson’s formula,

Euler’S method, Runge-Kutta

Ex. Equation of motion

Newton's equation $m\,a = f$ →

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = f$$

is a second-order differential equation.

It can also be written as

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= v,\\ \frac{\mathrm{d}v}{\mathrm{d}t} &= \frac{f}{m}\end{aligned}$$

: A set of first-order differential equations.

General problem

We'd like to solve

$$y' = f(x, y)$$

subject to the initial (boundary) condition

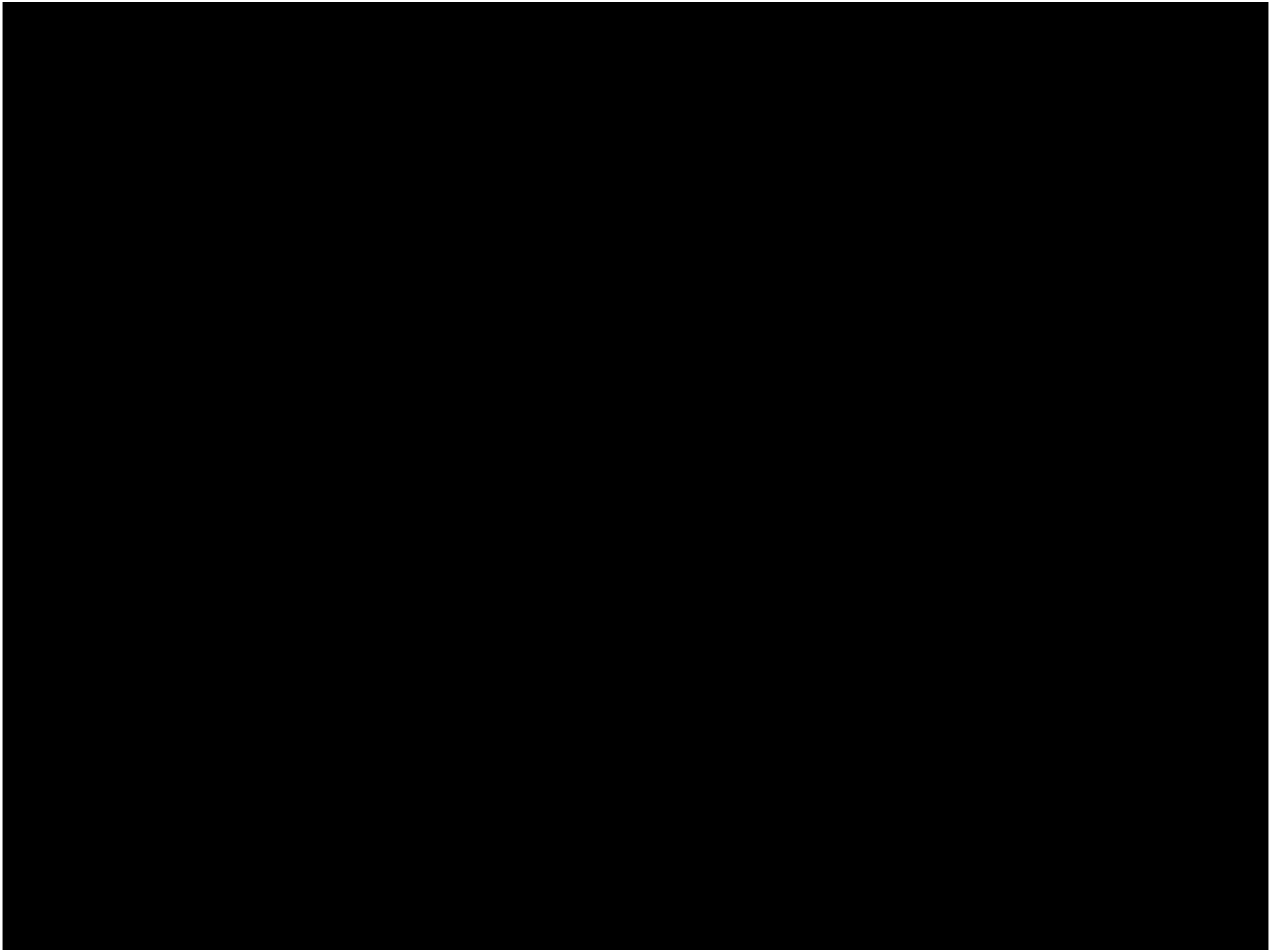
$$y(x_0) = y_0$$

By the way...

Schroedinger equation

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2}+V(x)\right]\psi=E\psi$$

is also a differential equation with some boudary conditions. Such “eigen value problems” will be introduced in the latter half of the class (by Shimizu-sensei).



Integration in multi-D

In 1-D: $$\int_0^1 f(x)\,\mathrm{d}x \sim \frac{1}{N}\sum_i^{N-1} f(x_i)$$

Sum of *N* numbers
~ *N* operations
(at least)

In 2-D: $$\int_0^1\int_0^1 f(x,y)\,\mathrm{d}x\mathrm{d}y \sim \frac{1}{NM}\sum_i^{N-1}\sum_j^{M-1} f(x_i,y_j)$$

NxM
operations

In more dimensions:

$$\int_0^1\int_0^1\cdots\int_0^1 f(x_1,x_2,...,x_d)\,\mathrm{d}x_1\mathrm{d}x_2...\mathrm{d}x_d$$
$$\sim \frac{1}{N^d}\sum_i^{N-1}\sum_j^{N-1}\cdots\sum_k^{N-1} f(x_i,x_j,...)$$

~ N^d operations
= 10^{30} for N = 1000, d=10

This would take 3000万年

using K-computer with
peta-flops speed.

Monte-Carlo method (next week)

Stiff equations

$$u' = 998\,u + 1998\,v$$
$$v' = -999\,u + 1999\,v$$

Initial values: $u(0) = 1,\ v(0) = 0$.

By the transformation $u = 2y - z,\ v = -y + z,$

we obtain

$$u = 2\,e^{-x} - e^{-1000x}$$

$$v = -e^{-x} + e^{-1000x}$$