

Computers in Engineering
COMP 208

Initial Value Problems
Michael A. Hawker

Differential Equations

- Differential Equations are based on an unknown function involving one or more variables
- An Ordinary Differential Equation is a DE that depends on one unknown variable and is usually in the form $F(x, f(x), f'(x), \dots, f^n(x)) = 0$

2007 Initial Value Problems 2

Ordinary Differential Equations

- In many areas of application we can measure how things change in some process
- From these measurements we would like to find the function that describes the process
- Examples:
 - Change in concentrations during chemical reaction
 - Heating or cooling of objects
 - Current flow in electrical circuits
 - Population dynamics

2007 Initial Value Problems 3

Simple ODE

$$dy / dx = x$$

- We want to find the original function
- We need to change things around and integrate:

$$dy = dx * x$$

$$y = x^2/2 + C$$

C is a some unknown constant

2007 Initial Value Problems 4

Initial Value Problems

- We need to have some way to determine the constant C
- We call this the initial value problem since we must know the value of the function, y_0 , at some initial point x_0
- Then we can determine the actual function

2007 Initial Value Problems 5

Example Cont.

- $y(x) = x^2/2 + C$
- Extra Data: $y(4) = 3$
- $3 = 4^2/2 + C$
- $3 - 8 = C$
- $C = -5$
- $y(x) = x^2/2 - 5$

2007 Initial Value Problems 6

ODE's

- We let $y(x)$ be the function we would like to study; however, we are only able to observe the rate of growth of the function
- This leads to equations of the form $dy/dx = f(x, y)$
- As we have just seen we can formulate a solution

2007 Initial Value Problems 7

Analytic Solutions

Some ODE's have analytic solutions

$$dy/dx = x + y - 1$$

Has the solution

$$y(x) = e^x - x$$

Others have no analytic solution. For example:

$$dy/dx = x^2 + y^2$$

2007 Initial Value Problems 8

What do we do?

- If we can't find a closed form of the function
- We turn back to numerical methods to approximate the solution

2007 Initial Value Problems 9

The Euler Method

- We “grow” the function from the starting value one step at a time
- Think of the dy/dx in terms of discrete steps, Δy and Δx
- Then the derivative approximates the ratio of these two values for small values of Δx

2007 Initial Value Problems 10

The Euler Method

- Multiplying dy/dx by Δx gives an approximate value for Δy
- The Euler method increments the independent variable by one stepsize, Δx , at a time
- Using the derivative, we approximate Δy and then the value of the function at the next step

2007 Initial Value Problems 11

Euler Method

We want to find an approximate solution to:

$$dy/dx = f(x, y)$$

$$y(x_0) = y_0$$

Now $f(x_0, y_0)$ is the slope of the function at (x_0, y_0)

Approximate the function value at x_0+h by

$$y_0 + h \cdot f(x_0, y_0)$$

Repeat this process so that

$$x_{n+1} = x_n + h$$

$$y_{n+1} = y_n + h f(x_n, y_n)$$

2007 Initial Value Problems 12

A Simple Example

```

#include <stdio.h>
#include <math.h>
#define H 0.1
int main() {
    double x,newx;
    double y,newy;
    int i,steps = 10;

    x = 0;
    y = 1;
    for (i = 0; i < steps; i++)
    {
        newx = x + H;
        newy = y + H * (x + y - 1);
        x = newx; y = newy;
        printf (" %f %f %f\n", newx, newy,exp(x)-x);
    }
    return 0;
}
    
```

2007
Initial Value Problems
13

#define

- After the #include commands, we can also define names using #define
- In the program, the name is replaced by the expression
- Note that the name is not a variable (there is no memory location for it)
- The name is just an alias for the value
 - i.e. Find and Replace

2007
Initial Value Problems
14

#define

- Syntax
 - No ; at end
 - No = between name and value
- This can cause problems if complex expressions are used

2007
Initial Value Problems
15

#define macros

- We can create simple functions to replace basic concepts
- i.e. adding 5 to a number
 - #define ADD_5(x) x+5
- Syntax:


```
#define MACRO_NAME(parameters) exp
```

2007 Initial Value Problems 16

#define Example

- #define ADD_5(x) x+5
...6+ADD_5(7)*4...
- Becomes: 6+7+5*4
Which Outputs: 33 not 54 as expected
- We need parenthesis for correct behavior


```
#define ADD_5(x) ((x) + 5)
```
- Why the extra parenthesis around x?

2007 Initial Value Problems 17

Example Revisited

```
void main(){
double x, y;
FILE * myfile = fopen("test2.csv","w+");
int i,steps = 100;
if (myfile){
x = 0;
y = 1;
for (i = 0; i < steps; i++){
fprintf (myfile," %f, %f, %f\n", x, y,exp(x)-x);
Euler_step(H,x,&y,func);
x += H;
}
fclose(myfile);
}
else printf("Could not open the file");
return;
}
```

2007 Initial Value Problems 18

Writing to a File – A Review

- We declare it to be of type FILE and use fopen to open it.

```
FILE * myfile = fopen("test2.csv","w+");
```
- Then we can write to the file using fprintf

```
fprintf (myfile," %f, %f, %f\n",
        x, y,exp(x)-x);
```
- When we are finished we close the file

```
fclose(myfile);
```

2007 Initial Value Problems 19

Writing to a File

- When we open a file, the filename is assigned a nonzero value if the operation is successful and a value of zero if it fails.

```
if (myfile) {
    ...
}
else printf("Could not open the file");
```

2007 Initial Value Problems 20

Function Pointers – A Review

- We can pass a function as an argument (via a pointer) to a function that computes the next iteration from the previous one.
- First give a name (fun) to the **type** of function we want to use

```
typedef double (*fun) (double,double);
```

2007 Initial Value Problems 21

Computing the Next Value

- Then define a function that uses this argument to compute the next value.
- We compute a new value for y , so we pass a pointer to it

```
void Euler_step(double h, double x,
               double *y, fun f){
    *y += h * f(x,*y);
    return;
}
```

2007

Initial Value Problems

22

Computing the Next Value

- We can define a function that we want to pass to the Euler function:

```
double func(double x, double y){
    return x+y-1;
}
```

- Functions are stored in memory at a specific address. We can pass a pointer to our function by using just its name (as we did with arrays):

```
Euler_step(H,x,&y,func);
```

2007

Initial Value Problems

23

The Final Program (a)

```
#include <stdio.h>
#include <math.h>

#define H    0.1

typedef double (*fun) (double,double);

double func(double x, double y){
    return x+y-1;
}

void Euler_step(double h, double x, double *y, fun f){
    *y += h * f(x,*y);
    return;
}
```

2007

Initial Value Problems

24

The Final Program (b)

```

int main(){
double x, y;
FILE * myfile = fopen("test2.csv","w+");
int i,steps = 100;
if (myfile){
x = 0; y = 1;
for (i = 0; i < steps; i++){
fprintf (myfile," %f, %f, %f\n", x, y,exp(x)-x);
Euler_step(H,x,&y,func);
x += H;
}
fclose(myfile);
}
else printf("Could not open the file");
return 0;
}
    
```

2007 Initial Value Problems 25

Part of the Output File

0	1	1
0.1	1.01	1.005171
0.2	1.031	1.021403
0.3	1.0641	1.049859
0.4	1.11051	1.091825
0.5	1.171561	1.148721
0.6	1.248717	1.222119
0.7	1.343589	1.313753
0.8	1.457948	1.425541
0.9	1.593742	1.559603
1	1.753117	1.718282
1.1	1.938428	1.904166
1.2	2.152271	2.120117
1.3	2.397498	2.369297
1.4	2.677248	2.6552
1.5	2.994973	2.981689

2007 Initial Value Problems 26

Excel Generated Graph

2007 Initial Value Problems 27

Runge-Kutta Method

The Euler method is not very accurate since the error tends to keep growing.

In the (fourth-order) Runge_Kutta method the derivative is evaluated four times

1. At the initial point
2. Twice at a trial midpoint
3. At a trial endpoint

2007

Initial Value Problems

28

Runge-Kutta Formula

Use the following to compute the next step

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n+h/2, y_n+k_1/2) \\ k_3 &= f(x_n+h/2, y_n+k_2/2) \\ k_4 &= f(x_n+h, y_n+k_3) \end{aligned}$$

$$y_{n+1} = y_n + (k_1+2k_2+2k_3+k_4)h/6$$

2007

Initial Value Problems

29

Implementing Runge-Kutta

```
void RK_step(double h, double x, double *y,
             fun f){
    double k1, k2, k3, k4, half = h/2.0;

    k1 = f(x, *y);
    k2 = f(x + half, *y + half * k1);
    k3 = f(x + half, *y + half * k2);
    k4 = f(x + h, *y + h*k3);

    *y += (h/6.0) * (k1 + 2.0*k2 + 2.0*k3 + k4);
}
```

2007

Initial Value Problems

30
