# Computational Physics Lab

# Numerical Differentiation & Simple Differential Equations

03/05/2009

# Outline

# Homework Assignment

1. Read Chapter 9, 12 (7 pages), and 15 (5 pages)
   - 9 *"Basic function properties"*
   - 12 *"Numerical error analysis - derivatives"*
   - 15 *"Differential equations"*
2. Assignments of Section 9.14: (1) - (13)
   - Due next Tuesday, March 17
     - ➜ Hand in a paper copy or a piece of paper stating that you have posted the homework to your `comphy` web site!

# Outline

# Default Function Parameters

Default parameters must be placed last in the function.

```
# include <iostream.h>

void myFunction(int a, int b);

main() {
    myFunction(2,3);
    ...
}
```

# Default Function Parameters

Default parameters must be placed last in the function.

```
# include <iostream.h>

void myFunction(int a, int b = 1);

main() {
    myFunction(2,3);
    myFunction(2);
    ...
}
```

Okay

# Default Function Parameters

Default parameters must be placed last in the function.

```
# include <iostream.h>

void myFunction(int a = 1, int b);

main() {
    myFunction(2,3);
    myFunction(2);
    ...
}
```

Compiler error

# Inline Functions

The body of a function that is declared *inline* is automatically substituted into each function call before compilation.

- Faster executable code, but increased compile times
- Functions defined within a class definition are inline functions

```
inline int myFunction(int a, int b) {...};

main() {
    myFunction(2,3);
    ...
}
```

Does not allocate and deallocate memory upon each call to it!

# Outline

# Numerical Differentiation

- It is often possible to find derivatives given an analytic expression for a function.
- Not always the case!
  Sometimes numerical determination of the derivative is the only alternative:
  - Functions available only as a set of discrete data points
  - Determination of a function from non-linear differential equation and some initial conditions

# The Derivative Operator

## Limit-Based Determination

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \left( \frac{f(x + \Delta x) - f(x)}{\Delta x} \right)$$

Numerical approximation to the derivative generated from its original definition as the limit of a discrete expression

- Can be used to formulate numerical techniques

# The Derivative Operator

## Limit-Based Determination

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \left( \frac{f(x + \Delta x) - f(x)}{\Delta x} \right)$$

## Two methods of computing differences

1. Discrete Forward Finite Difference

$$D^{+}_{\Delta x}(f(x)) = \left( \frac{f(x + \Delta x) - f(x)}{\Delta x} \right)$$

# The Derivative Operator

## Limit-Based Determination

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \left( \frac{f(x + \Delta x) - f(x)}{\Delta x} \right) = \lim_{\Delta x \to 0} D^{+}_{\Delta x}(f(x))$$

## Two methods of computing differences

1. Discrete Forward Finite Difference

$$D^{+}_{\Delta x}(f(x)) = \left( \frac{f(x + \Delta x) - f(x)}{\Delta x} \right)$$

2. Centered Finite Difference

$$D^{c}_{\Delta x}(f(x)) = \left( \frac{f(x + \Delta x/2) - f(x - \Delta x/2)}{\Delta x} \right)$$

# Derivatives & Errors

Taylor Series Expansion:

$$f(x+\Delta x)=f(x)+\Delta x\frac{df(x)}{dx}+\frac{\Delta x^2}{2}\frac{d^2f(x)}{dx^2}+\ldots$$

$O(\Delta x^2)$ remaining term

**Forward Difference**

$$\left[\frac{f(x+\Delta x)-f(x)}{\Delta x}\right]=\frac{df(x)}{dx}+O(\Delta x)$$

(global error) $\propto N_\tau$ * (local error)

$N_\tau$: number of time steps

truncation error term is specified by its order in $\Delta x$

# Derivatives & Errors

Improved Approximation:

$$f(x+\Delta x/2)=f(x)+(\Delta x/2)f'(x)+\frac{(\Delta x/2)^2}{2}f''(x)+\frac{(\Delta x/2)^3}{6}f'''(x)+\ldots$$

$$-\quad f(x-\Delta x/2)=f(x)+(-\Delta x/2)f'(x)+\frac{(\Delta x/2)^2}{2}f''(x)+\frac{(-\Delta x/2)^3}{6}f'''(x)+\ldots$$

$$f(x+\Delta x/2)-f(x-\Delta x/2) \;=\; \Delta x f'(x)+\frac{\Delta x^3}{24}f'''(x)+\ldots$$

Central Difference

$$\left[\frac{f(x+\Delta x/2)-f(x-\Delta x/2)}{\Delta x}\right] = \frac{df(x)}{dx} + O(\Delta x^2)$$

truncation error term is of order in $\Delta x^2$

# Outline

# Euler's Method

An $n$ th-order ordinary differential equation can always be replaced by a system of $n$ first-order equations that require $n$ independent initial or boundary conditions to specify a unique solution.

# Euler's Method

An $n$ th-order ordinary differential equation can always be replaced by a system of $n$ first-order equations that require $n$ independent initial or boundary conditions to specify a unique solution.

## Example:

Single massive particle with mass $m$ attached to a spring with force constant $k$:

$$a = \frac{d^2x}{dt^2} = -\frac{k}{m}x$$

# Euler's Method

An $n$ th-order ordinary differential equation can always be replaced by a system of $n$ first-order equations that require $n$ independent initial or boundary conditions to specify a unique solution.

## Example:

Single massive particle with mass $m$ attached to a spring with force constant $k$:

$$a = \frac{d^2x}{dt^2} = -\frac{k}{m}x$$

$$a = \frac{dv}{dt} = -\frac{k}{m}x$$

# Outline

# This Week's Project

## Radioactive Decays

$$\frac{dN(t)}{dt} = \frac{-N(t)}{\tau}$$

$$Set \; \frac{dN(t)}{dt} = D^{+}_{\Delta t}(N(t))$$

and solve for the incremental equation of state

$$N(t+\Delta) = \_\_\_\_\_\_\_\_\_\_\_\_$$

# Example: Euler Method for Motion of Point Particle

## Equations of Motion

$$\frac{dv}{dt} = a(r, v) \qquad \frac{dr}{dt} = v$$

## Using the forward difference:

$$\frac{v(t+\Delta t) - v(t)}{\Delta t} + O(t) = a(r(t), v(t)) \qquad \frac{r(t+\Delta t) - r(t)}{\Delta t} + O(t) = v(t)$$

$$v(t+\Delta t) = v(t) + \Delta t\, a(t) + O(\Delta t)^2 \qquad r(t+\Delta t) = r(t) + \Delta t\, v(t) + O(\Delta t)^2$$

## Dropping the error term

$$v_{n+1} = v_n + \Delta t\; a_n \qquad r_{n+1} = r_n + \Delta t\; v_n$$

# Example: Euler Method Procedure

## Calculation of Trajectory (the incremental equation)

1. Specify the initial conditions: $r_1$ & $v_1$.
2. Choose a time step $\Delta t$.
3. Calculate the acceleration given the current $r$ and $v$.
4. Use $v_{n+1} = v_n + \Delta t\ a_n$ and $r_{n+1} = r_n + \Delta t\ v_n$ to compute new $r$ and $v$.
5. Go to step 3 until enough trajectory points have been computed.

## Euler-Cromer Method

$$v_{n+1} = v_n + \Delta t\ a_n \qquad r_{n+1} = r_n + \Delta t\ v_n$$