Computersimulation in der Materialwissenschaft

WS 2014-2015

Diplom Werkstoffwissenschaft + andere Interessenten

Alejandro Santana und Leonardo Medrano
Institut für Werkstoffwissenschaft
Professur Werkstoffwissenschaft und Nanotechnologie

ORT:
Scilab allows us to write codes for:

1.) Numerical Differentiation

2.) Numerical integration

3.) Ordinary differential equations
Numerical Differentiation

Geometrical meaning of the derivative

\[ f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} \]

The derivative is the best linear approximation at a given point of a curve

Numerical Derivatives: Implementation

Geometrical intuition:

1.) Forward difference:

\[ f'(x_k) \approx \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} \]

2.) Central difference:

\[ f'(x_k) \approx \frac{f(x_{k+1}) - f(x_{k-1})}{x_{k+1} - x_{k-1}} \]

3.) Backward difference:

\[ f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \]

Approximation written in Scilab

\[ f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]
Numerical Integration

Geometrical meaning of the Integration

The integral is the **AREA** enclosed by a function $F(x)$ in the $xy$-plane (also is the inverse of the differentiation !)

How can we estimate this area ?

$$S = \sum_{i=1}^{n} f(y_i)(x_i - x_{i-1})$$

Geometrical meaning of the numerical Integration

The integral is an **AREA** that can be approximated by using already known Areas:

1.) Squares
2.) Rectangles
3.) Trapezoids
4.) Parabolas

\[ \int_{a}^{b} f(x) \, dx = \left( \frac{b-a}{n} \right) \left[ y_{m1} + y_{m2} + \ldots + y_{mn} \right] \]

Numerical Integration: Scilab implementation

\[ \int_a^b f(x) \, dx = \left( \frac{b-a}{n} \right) \left[ y_{m1} + y_{m2} + \ldots + y_{mn} \right] \]

Area of the rectangle

Summation over the interval of integration

**Usage:**
```
// r = midp(f,a,b,n)
// Composite midpoint rule (1-point open Newton-Cotes)
//
// Input:
// f - Matlab inline function
// a,b - integration interval
// n - number of subintervals (panels)
//
// Output:
// r - computed value of the integral
//
// Examples:
// r=midp(@sin,0,1,10);
// r=midp(@myfunc,0,1,10); here 'myfunc' is any user-defined function in M-file
// r=midp(inline('sin(x)'),0,1,10);
// r=midp(inline('sin(x)-cos(x)'),0,1,10);
```

```matlab
function r = midp(f,a,b,n)
h = (b - a) / n;
y = a + h * 0.5;
r = 0;
for i=1:n
    r = r + f(x);
y = y + h;
end
r = r * h;
endfunction
```

An ordinary differential equation (1-d) follows an equation like:

\[ \frac{dy}{dx} = f(x) \]

Can be solved by integrating both sides with respect to x, like:

\[ y = \int f(x) \, dx \]

This is a technique called a **DIRECT INTEGRATION** and is the basic idea for many algorithms.
We are approaching, at every step, by the slope given by the function

\[ x(t + \Delta t) = x(t) + \frac{d_1 + 2d_2 + 2d_3 + d_4}{6} \]

Estimators:

\[
\begin{align*}
    d_1 &= f(x(t)) \Delta t \\
    d_2 &= f(x(t) + d_1/2) \Delta t \\
    d_3 &= f(x(t) + d_2/2) \Delta t \\
    d_4 &= f(x(t) + d_3) \Delta t 
\end{align*}
\]

Next step prediction:
Ordinary differential equations (ode)

**Syntaxis of the ODE function:**

\[ y = \text{ode}(y_0, t_0, t, f) \]

- **Y0** = initial condition
- **T0** = initial time
- **T** = series of to be computed

**ODE is an in-built function that solves ordinary differential equations**

**Example:**

```plaintext
-->t = 0:100;

-->k = 0.2;

-->u = ode(1/10000, 0.0, t, f);

-->plot2d(t, u)
```

**Code snippet:**

```plaintext
// usage of function udott
// parameters t (time) and u is a variable
// ordinary differential equation to solve
// in this case is u' = k*u*(u-1)

function udott=f(t,u)
    udot = k*u.*(1.-u);
endfunction
```
Modify the numerical derivative code in order to implement the following approximations using the sine function:

a.) Two-point backward difference formula for first derivative varying \( h \)

b.) Two-point centered-difference formula for first derivative, varying \( h \)
Create a code that implements the following numerical integration scheme

$$\int_a^b f(x) \, dx \approx \frac{b-a}{6} \left[ f(a) + 4f \left( \frac{a+b}{2} \right) + f(b) \right]$$

Test your implementation by calculating:

1.) $X^2$ interval $[0,1]$
2.) Cos(x) interval $[0,90^\circ]$
Create a code that implements the following numerical integration scheme

\[
\int_a^b f(x) \, dx \approx \frac{b-a}{6} \left[ f(a) + 4f \left( \frac{a+b}{2} \right) + f(b) \right]
\]

Test your implementation by calculating:

1.) \( x^2 \) interval \([0,1]\)
2.) Cos(x) interval \([0,90°]\)

Create a code that solves the following ordinary differential equation

\[ \frac{dx}{dt} = \sin(2t) \]

1.) Enhance your code by receiving any function to solve the same class of Equation

2.) Plot the solutions by using the appropriated commands within your code
Create a code that solves the following set of coupled differential equations

\[ \frac{du}{dt} = v(t) \quad \text{and} \quad \frac{dv}{dt} = -\sin(u(t)) \]

1.) What physical system describes these set of coupled ordinary differential equations

2.) Plot the solutions by using the appropriated commands within your code