Differential Equations

Most fundamental and basic equations in physics as well as frequently occurring problems appear as differential equations.

Examples:

Simple harmonic oscillator

$$m\frac{d^2x(t)}{dt^2} = -kx(t)$$

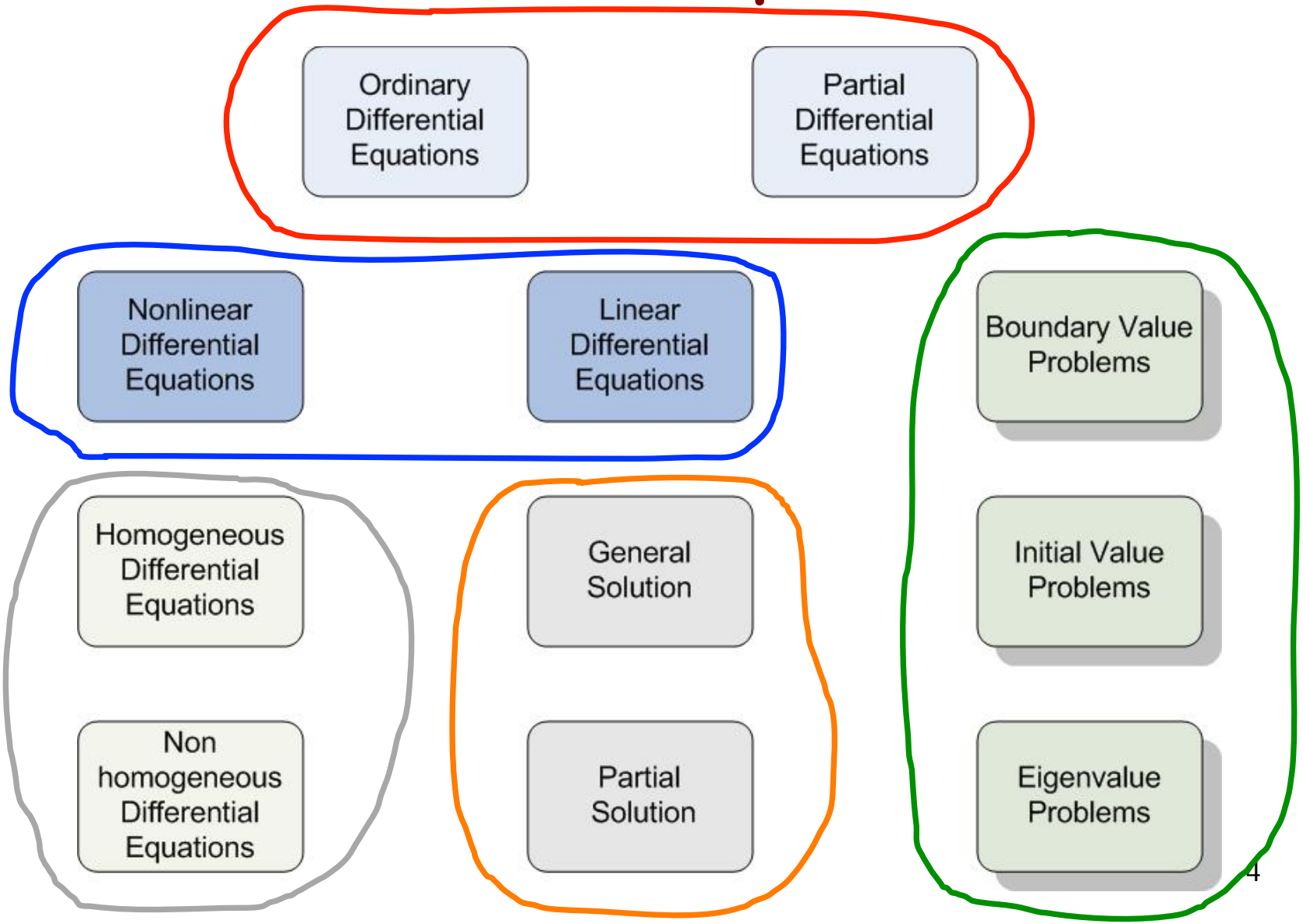
Schrödinger equation (example for 1D)

$$i\hbar\frac{d\psi(x,t)}{dt} = -\frac{1}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x)\psi(x,t)$$

Part 1

classification

Differential Equations



Simple Harmonic Oscillator

$$m\frac{d^2x(t)}{dt^2} = -kx(t)$$

$$x(t=0) = x_0$$

$$\frac{dx}{dt}(t=0) = v_0$$

is initial value problem for the second order ordinary linear homogeneous differential equation

ODE or PDE

- The ordinary differential equations (ODE) – have functions of one only independent variable

  Example: stationary Schrödinger equation

- The partial differential equations (PDE) – have functions of several independent variables

  Example: time dependent Schrödinger equation for

$$\Psi(\vec{r},t)$$

ODE: Linear or Nonlinear

- A linear differential equation – all of the derivatives appear in linear form and none of the coefficient depends on the dependent variable

$$a_0 x(t) + a_1 \frac{dx}{dt} + a_2 \frac{d^2x}{dt^2} + \ldots = c$$

example

$$m \frac{d^2x(t)}{dt^2} = -kx(t)$$

ODE: Linear or Nonlinear

- A nonlinear differential equation – if the coefficients depend on the dependent variable, OR the derivatives appear in a nonlinear form

  Examples:

$$\frac{d^2x(t)}{dt^2}\frac{dx(t)}{dt} - x(t) = 0$$

$$t^2\frac{d^2x(t)}{dt^2} - x^2(t) = 0$$

Order of ODE

The order n of an ordinary differential equation is the order of the highest derivative appearing in the differential equation

Examples:

$$t^2 \frac{d^2x(t)}{dt^2} - x(t) = 0$$ second order

$$t \frac{d^3x(t)}{dt^3} - \frac{dx(t)}{dt} = 0$$ third order

General or partial solution

Example: $\frac{dx(t)}{dt} - x(t) = 0$

- General solution:

$$x(t) = Ce^{t}$$

- Partial solutions:

$$x(t) = 2.0e^{t}$$

$$x(t) = 4.8e^{t}$$

Homogeneous and nonhomogeneous ODE

- A homogeneous equation: the each term contains either the function or its derivative, but no other functions of independent variables

$$m\frac{d^2x(t)}{dt^2} - kx(t) = 0$$

- A nonhomogeneous equation: contains additional term (source terms, forcing functions) which do not involve the dependent variable

$$m\frac{d^2x(t)}{dt^2} - kx(t) = F_0\cos(\omega t)$$

Three major categories of ODE

- Initial-value problems – involve time-dependent equations with given initial conditions:

$$m\frac{d^2x(t)}{dt^2} - kx(t) = 0, \qquad x(t=0) = x_0, \quad \frac{dx}{dt}(t=0) = v_0$$

- Boundary-value problems – involve differential equations with specified boundary conditions:

$$\frac{d^2y(x)}{dx^2} - \alpha y(x) = 0, \qquad y(x=a) = y_a, \quad y(x=b) = y_b$$

- Eigenvalue problems – involve solutions for selected parameters in the equations

In reality, a problem may have more then just one of the categories above

Three general classifications in physics

- ✓ Propagation problems - are initial value problems in open domains where the initial values are marched forward in time (or space) . The order may be one or greater. The number of initial values must be equal to the order of the differential equation.
- ✓ Equilibrium problems – are boundary-value problems in closed domains where boundary values are specified at boundaries of the solution domain. The order of ODE must be at least two.
- ✓ Eigenproblems – are a special type of problems where the solution exists only for special values of a parameter.

n-th order or a set on n linear equations

Any n^{th} order linear differential equation can be reduced to n coupled first order differential equations

Example: $$m\frac{d^2x(t)}{dt} = -kx(t)$$

is the same as $$\frac{dx(t)}{dt} = v(t)$$

$$m\frac{dv(t)}{dt} = -kx(t)$$

Part 2

Initial value problem

Initial values problems are solved by marching methods using finite difference methods.

The objective of a finite difference method for solving an ODE is to transform a calculus problem into an algebra problem by

- ✓ Discretizing the continuous physical domain into a discrete finite difference grid
- ✓ Approximating the exact derivatives in the ODE by algebraic finite difference approximations (FDAs)
- ✓ Substituting the FDA into ODE to obtain an algebraic finite difference equation (FDE)
- ✓ Solving the resulting algebraic FDE

Three groups of finite difference methods for solving initial-value ODEs

- ✓ Single point methods advance the solution from one grid point to the next grid point using only the data at a single grid point. (most significant method – 4^{th} order Runge-Kutta
- ✓ Extrapolation methods evaluate the solution at a grid point for several values of grid size and extrapolate those results to get for a more accurate solution.
- ✓ Multipoint methods advance the solution form one grid point to the next using the data at several known points (see 4^{th} order Adams-Bashforth-Moulton method)

Finite difference approximations.

Using the Taylor series for x_{n+1} using grid point n

$$x_{n+1} = x_n + x'|_n \, \Delta t + \frac{1}{2} x''|_n \, \Delta t^2 + \frac{1}{6} x'''|_n \, \Delta t^3 + \ldots$$

$$x_{n+1} = x_n + x'|_n \, \Delta t + \frac{1}{2} x''|_n \, \Delta t^2 + \ldots + \frac{1}{m!} x^{(m)}|_n \, \Delta t^m + R^{m+1}$$

$$R^{m+1} = \frac{1}{(m+1)!} x^{(m+1)}(\tau) \Delta t^{m+1} \qquad t \le \tau \le t + \Delta t$$

solving for $x'|_n$ yields

$$x'|_n = \frac{x_{n+1} - x_n}{\Delta t} - \frac{1}{2} x''|_n \, \Delta t - \frac{1}{6} x'''|_n \, \Delta t^2 - \ldots$$

Finite difference approximations.

Using the Taylor series for x_{n+1} using grid point n

a first - order finite difference approximation

$$x'|_n = \frac{x_{n+1} - x_n}{\Delta t} \quad O(\Delta t)$$

a first - order backward - difference approximation

$$x'|_{n+1} = \frac{x_{n+1} - x_n}{\Delta t} \quad O(\Delta t)$$

A second - order centered difference approximation of x' at point $n + 1/2$

$$x'|_{n+1/2} = \frac{x_{n+1} - x_n}{\Delta t} \quad O(\Delta t^2)$$

Finite difference equations.

consider the general first - order initial - value ODE

$$x'(t) = f(t,x) \quad x(0) = x_0$$

use finite difference approximations

$$x'|_n = \frac{x_{n+1} - x_n}{\Delta t} \quad \text{or} \quad x'|_{n+1} = \frac{x_{n+1} - x_n}{\Delta t}$$

substitute into the ODE above and solve for x_{n+1} :

$$x_{n+1} = x_n + f(t_n, x_n)\Delta t \qquad \text{explicit finite difference}$$

$$x_{n+1} = x_n + f(t_{n+1}, x_{n+1})\Delta t \qquad \text{implicit finite difference}$$

Smoothness

Smoothness – the continuity of a function and its derivatives.

If a problem has discontinuous derivatives at some point, then the solution may misbehave at this point.

At a discontinuity – use either single point methods or extrapolation methods (not multi-point) because the step size can be chosen to have the discontinuity at a grid point.

Errors - five types.

- ✓ Errors in the initial data
- ✓ Algebraic errors
- ✓ Truncation errors – cased by truncating the Taylor series approximation (decreases with decreasing of the step size)
- ✓ Round off errors – caused the finite word length (increases as the step size decreases: more steps and small difference between large numbers)
- ✓ Inherited errors – the sum of all accumulated errors from all previous steps (means that the initial condition for the next is incorrect)

Four important issues

- ✓ Accuracy
- ✓ Efficiency
- ✓ Stability
- ✓ Consistency

Part 2a

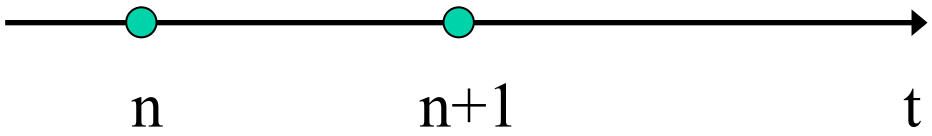
The first-order Euler method

The explicit Euler method for ODE

The explicit Euler method – first-order finite difference method for solving initial-value problem for ODE

Let's consider a general first - order ODE

$$\frac{dx}{dt} = f(t,x) \quad \text{with } x(t_0) = x_0$$



Explicit finite difference (first order)

$$x'|_n = \frac{x_{n+1} - x_n}{\Delta t} \quad O(\Delta t) \quad \text{and then}$$

$$x_{n+1} = x_n + f(t_n, x_n)\Delta t$$

The explicit Euler method for ODE

The method $x_{n+1} = x_n + f(t_n, x_n)\Delta t$

- ✓ explicit (since $f(t_n,x_n)$ does not depend on x_{n+1})
- ✓ requires only one known point (singe point method)
- ✓ the local truncation error is $O(\Delta t^2)$
- ✓ the global error accumulated after n steps $O(\Delta t)$
- ✓ problem: the method is conditionally stable for $\Delta t < \Delta t_{cr}$

example

$$\frac{dx}{dt} = -x$$

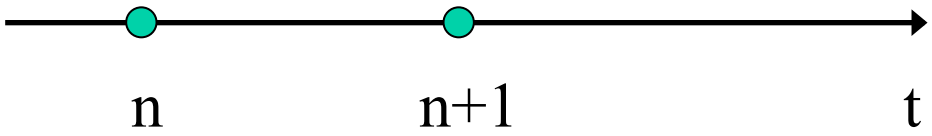
$$x_{n+1} = x_n - x_n \Delta t$$

The implicit Euler method for ODE

The implicit Euler method – first-order finite difference method for solving initial-value problem for ODE

Let's choose n+1 as the base point for

$$\frac{dx}{dt} = f(t,x) \quad \text{with} \quad x(t_0) = x_0$$



Implicit finite difference (first order)

$$x'|_{n+1} = \frac{x_{n+1} - x_n}{\Delta t} \quad O(\Delta t) \quad \text{and then}$$

$$x_{n+1} = x_n + f(t_{n+1}, x_{n+1})\Delta t$$

The implicit Euler method (more)

Let's consider an example

$$\frac{dx}{dt} = -x$$

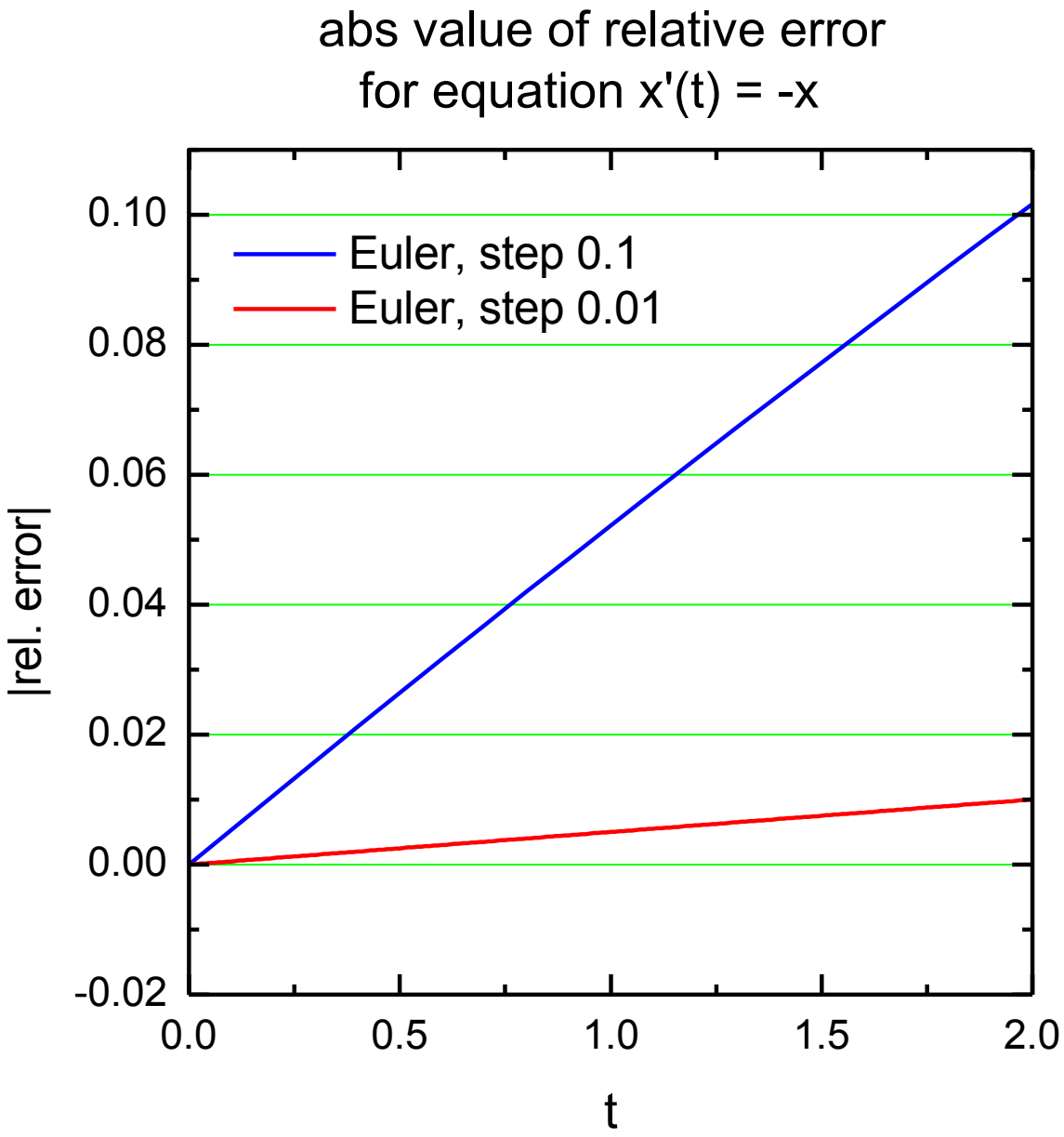
$$x_{n+1} = x_n + f(t_{n+1}, x_{n+1})\Delta t$$

$$x_{n+1} = x_n - x_{n+1}\Delta t$$

the implicit solution $x_{n+1} = \dfrac{x_n}{1+\Delta t}$

the explicit solution $x_{n+1} = x_n(1-\Delta t)$

- ✓ The implicit Euler is unconditionally stable
- ✓ however, if $f(t,x)$ is nonlinear, the we need to use one of methods for solving nonlinear equations



$$\frac{dx}{dt} = -x$$

$$x(t=0) = 1$$

```
double f1(double, double);
double euler1d(double(*)(double, double), double, double,
double);
int main()
{   double xi, ti, xf, tf, dt, tmax;
    ti = 0.0;                  // initial value for variable
    xi = 1.0;                  // initial value for function
    dt = 0.01;                 // step size for integration
    tmax = 2.0 ;               // integrate from ti till tmax
    while (ti <= tmax)
    {   tf = ti + dt;
        xf = euler1d(f1,ti,xi,tf);
        cout<< setw(12) << tf << setw(12) << xf  << endl;
        ti = tf;
        xi = xf;  }
return 0;
}
double euler1d(double(*f)(double, double), double ti,
double xi, double tf)
{  double xf;
     xf = xi + f(ti,xi)*(tf-ti);
   return xf;}
 double f1(double t, double x)
{   double dx;
    dx = (-1.0)*x;
    return dx;
}
```

example: C++

2nd order ODE: example harmonic oscillator

$$m\frac{d^2x(t)}{dt^2} = -kx(t)$$

$$\frac{dx(t)}{dt} = v(t)$$

$$m\frac{dv(t)}{dt} = -kx(t)$$

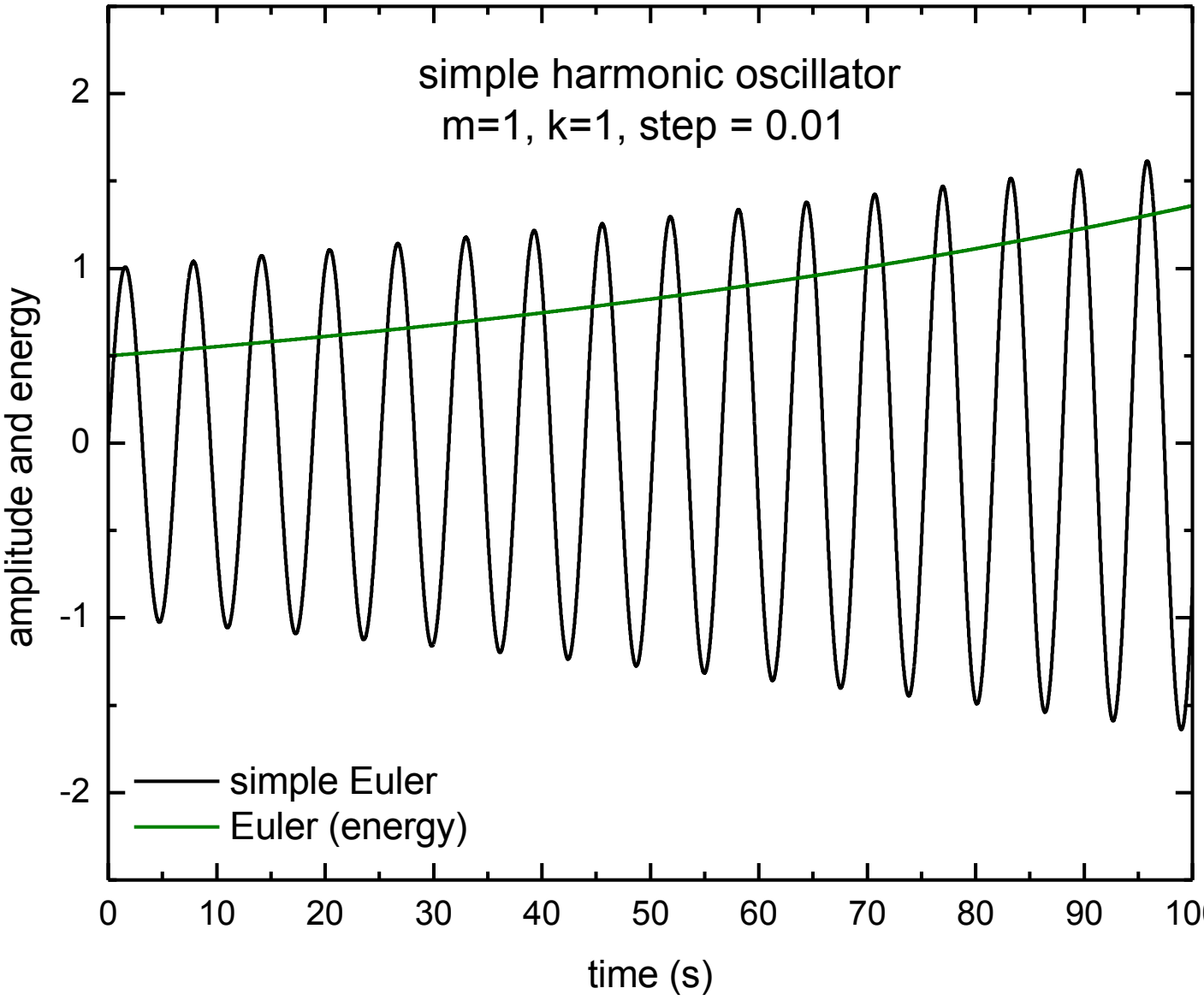
$$x(t_0 + h) \approx x(t_0) + hx'(t_0)$$

```
x(I+1)=x(I)+v(I)*h

v(I+1)=v(I)-(k/m)*x(I)*h
```

example: C++ - simple harmonic oscillator

```
double euler2d(double(*d1x)(double, double, double),
       double(*d2x)(double, double, double),
       double ti, double xi, double vi, double tf,
       double& xf, double& vf)
{
    xf = xi + d1x(ti,xi,vi)*(tf-ti);
    vf = vi + d2x(ti,xi,vi)*(tf-ti);
   return 0.0;
}

    double f1(double t, double x, double v)
{
    double d1x;
    d1x = v;
    return d1x;
}

    double f2(double t, double x, double v)
{
    double d2x;
    d2x = (-1.0)*x;  //simple harmonic oscillator
    return d2x;
}
```

problems with the simple Euler method



Problems with the Euler method

- Euler method corresponds to keeping the first two terms in the Taylor series
- Accuracy is low $x(t_0 + h) \approx x(t_0) + hx'(t_0) + O(h^2)$
- Error propagation is a problem!
  After N steps, the error is on the order $N \cdot O(h^2) \approx O(h)$
- Step size is important
- Do not use the Euler methods unless for learning!

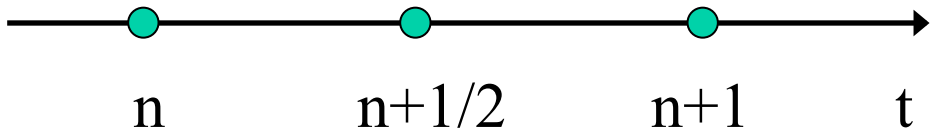
Practice

Write a program that implements Euler method to solve the simple harmonic oscillator for t=0-100 (try step sizes 0.1 and 0.01) with k=1 and m=1, x(0)=0.0 and v(0)=1.0 Compare you solutions with exact ones and check conservation of energy as a function of time.

Part 2b

Second-order single point methods

The second-order central difference



choose $n+1/2$ as a base point

$$x_{n+1} = x_{n+1/2} + x'_{n+1/2}\frac{\Delta t}{2} + \frac{1}{2}x''_{n+1/2}\left(\frac{\Delta t}{2}\right)^2 + \frac{1}{6}x'''_{n+1/2}\left(\frac{\Delta t}{2}\right)^3 + \ldots$$

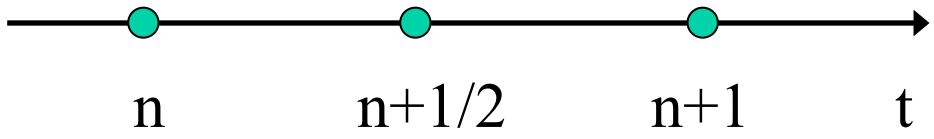
$$x_n = x_{n+1/2} + x'_{n+1/2}\left(-\frac{\Delta t}{2}\right) + \frac{1}{2}x''_{n+1/2}\left(-\frac{\Delta t}{2}\right)^2 + \frac{1}{6}x'''_{n+1/2}\left(-\frac{\Delta t}{2}\right)^3 + \ldots$$

the difference

$$x'_{n+1/2} = \frac{x_{n+1} - x_n}{\Delta t} - \frac{1}{24}x'''_{n+1/2}(\Delta t)^3$$

and for a first - order ODE $x_{n+1} = x_n + f_{n+1/2}\Delta t + O(\Delta t^3)$

Way 1 to evaluate $f_{n+1/2}$ (modified midpoint)



$$x_{n+1} = x_n + f_{n+1/2}\Delta t + O(\Delta t^3)$$

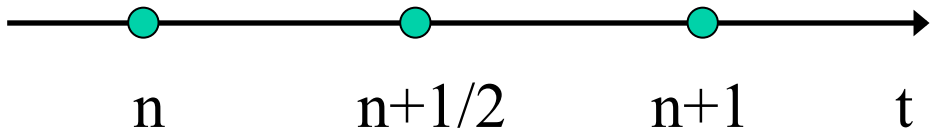
step 1:

$$x^P_{n+1/2} = x_n + \frac{\Delta t}{2} f(t_n, x_n)$$

step 2:

$$x^C_{n+1} = x_n + \Delta t\, f\left(t_n + \frac{\Delta t}{2}, x^P_{n+1/2}\right)$$

Way 2 to evaluate $f_{n+1/2}$ (modified Euler)



$$x_{n+1} = x_n + f_{n+1/2}\Delta t + O(\Delta t^3)$$

from the Taylor series

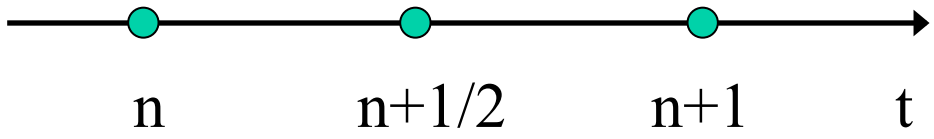
$$f_{n+1} = f_{n+1/2} + f'_{n+1/2}\left(\frac{\Delta t}{2}\right) + \cdots$$

$$f_n = f_{n+1/2} + f'_{n+1/2}\left(-\frac{\Delta t}{2}\right) + \cdots$$

$$f_{n+1/2} = \frac{1}{2}\left(f_n + f_{n+1}\right) + O(\Delta t^2)$$

then $x_{n+1} = x_n + \frac{\Delta t}{2}\left[f_n + f_{n+1}\right]$ ⟹ for linear ODEs it can be solved directly, but for non-linear ODEs must be solved iteratively

Way 2 to evaluate $f_{n+1/2}$ (modified Euler)



however for

$$x_{n+1} = x_n + \frac{\Delta t}{2}\left[f_n + f_{n+1}\right]$$

f_{n+1} can be predicted by the explicit first - order Euler

then step 1:

$$x_{n+1}^{P} = x_n + \Delta t\, f(t_n, x_n)$$

step 2:

$$x_{n+1}^{C} = x_n + \frac{\Delta t}{2}\left(f_n + f_{n+1}^{P}\right)$$

the method is also named as a "predictor - corrector" method

Summary: A modified Euler method

$$x_{n+1}^{P} = x_n + f(t_n, x_n)\Delta t$$

$$x_{n+1}^{C} = x_n + \left(f(t_n, x_n) + f(t_{n+1}, x_{n+1}^{P})\right)\frac{\Delta t}{2}$$

P – predicted value

C – corrected value

- ✓ the local truncation error is $O(\Delta t^3)$
- ✓ the global error after n steps $O(\Delta t^2)$

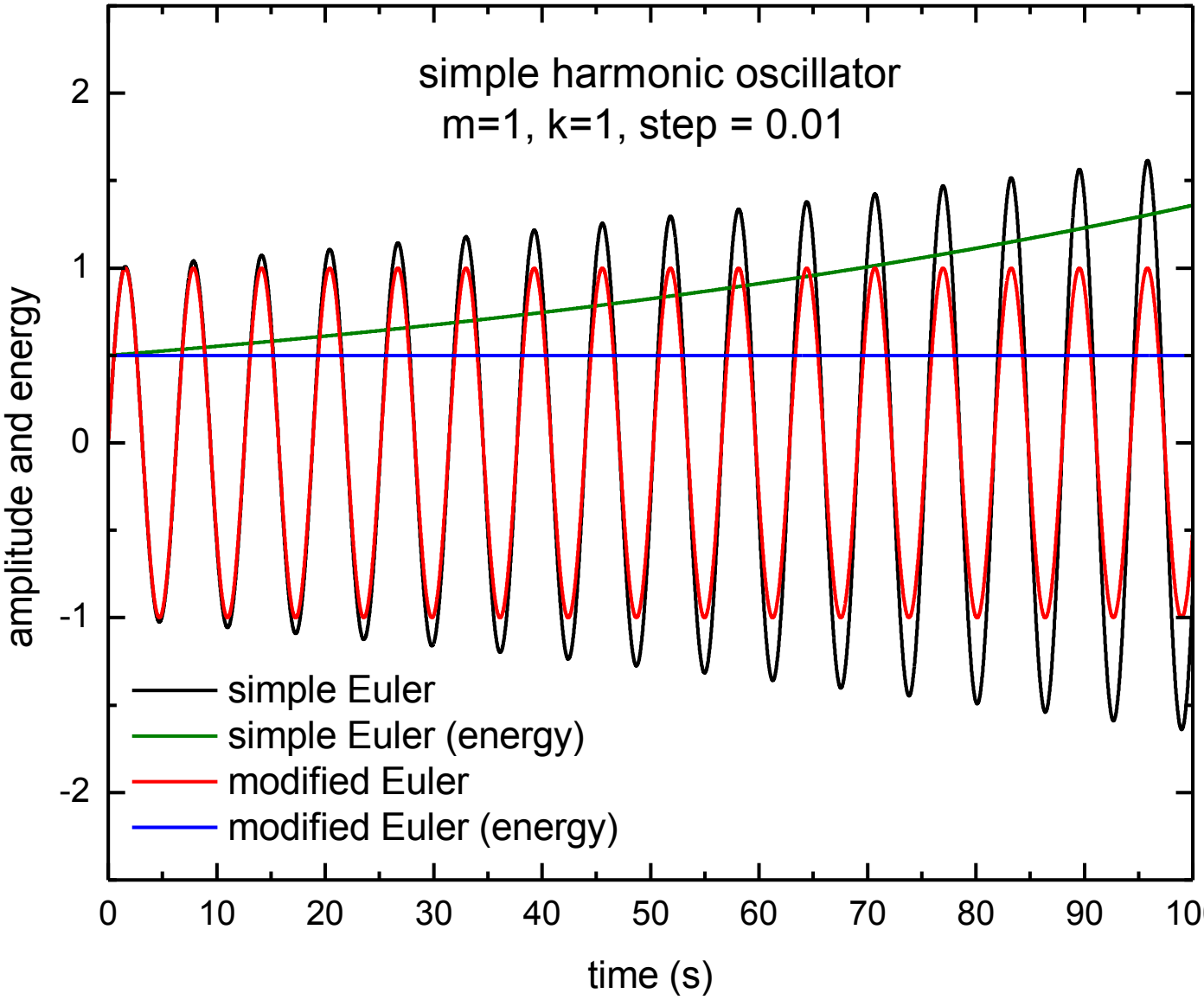
Example

```
x(I+1)=x(I)+v(I)*h

v(I+1)=v(I)-(k/m)*x(I)*h

correction

x(I+1)=x(I)+(v(I)+v(I+1))*h/2.0

v(I+1)=v(I)-(k/m)*  (x(I)+x(I+1))*h/2.0
```

example: C++ - modified Euler

```
double euler2d(double(*d1x)(double, double, double),
       double(*d2x)(double, double, double),
       double ti, double xi, double vi, double tf,
       double& xf, double& vf)
{
    xf = xi + d1x(ti,xi,vi)*(tf-ti);
    vf = vi + d2x(ti,xi,vi)*(tf-ti);
    xf = xi + (d1x(ti,xi,vi)+d1x(ti,xf,vf))*0.5*(tf-ti);
    vf = vi + (d2x(ti,xi,vi)+d2x(ti,xf,vf))*0.5*(tf-ti);
   return 0.0;
}
 double f1(double t, double x, double v)
{
    double d1x;
    d1x = v;
    return d1x;
}
    double f2(double t, double x, double v)
{
    double d2x;
    d2x = (-1.0)*x;  //simple harmonic oscillator
    return d2x;
}
```

simple harmonic oscillator (more)



Simple methods for a particle dynamics

$$m\frac{d^2x(t)}{dt^2} = F(x)$$

$$\begin{cases} \dfrac{dx(t)}{dt} = v(t) \\ m\dfrac{dv(t)}{dt} = F(x) \end{cases}$$

explicit Euler method

$$\begin{cases} x_{n+1} = x_n + v_n \Delta t \\ v_{n+1} = v_n + F(x_n)\Delta t / m \end{cases}$$

Simple methods for a particle dynamics

explicit Euler method

$$\begin{cases} x_{n+1} = x_n + v_n \Delta t \\ v_{n+1} = v_n + F(x_n)\Delta t / m \end{cases}$$

Modification 1 (Euler - Cromer method)

$$v_{n+1} = v_n + F(x_n)\Delta t / m$$

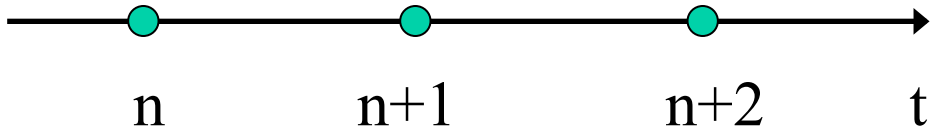
$$x_{n+1} = x_n + v_{n+1}\Delta t$$

Modification 2 (the midpoint method)

$$v_{n+1} = v_n + F(x_n)\Delta t / m$$

$$x_{n+1} = x_n + \frac{v_{n+1} + v_n}{2}\Delta t$$

Simple methods for a particle dynamics



Modification 3 (Leap-Frog method)

$$\frac{v(t+h)-v(t-h)}{2h}+O(h^2)=\frac{F}{m}$$

$$\frac{x(t+2h)-x(t)}{2h}+O(h^2)=v(t+h)$$

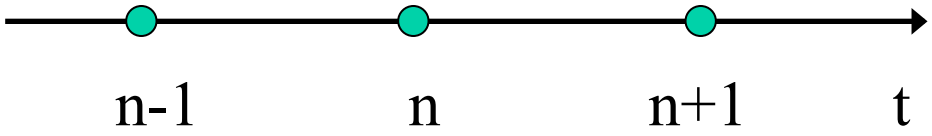
velocity - centered difference, position - forward difference

$$v_{n+1}=v_{n-1}+2F(x_n)\Delta t/m$$

$$x_{n+2}=x_n+2v_{n+1}\Delta t$$

the position evaluated at $n=1,3,5$ and the velocity $n=2,4,6$

Simple methods for a particle dynamics



Modification 4 (Verlet method)

$$\frac{x_{n+1} - x_{n-1}}{2\Delta t} + O(\Delta t^2) = v_n$$

$$\frac{x_{n+1} + x_{n-1} - 2x_n}{\Delta t^2} + O(h^2) = \frac{F(t_n, x_n)}{m}$$

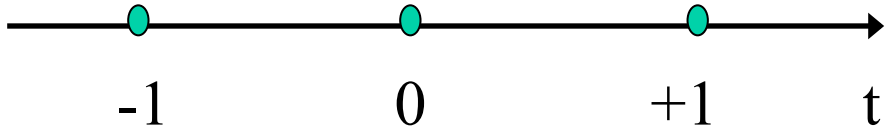
centered differences for the derivatives

$$v_n = \frac{x_{n+1} - x_{n-1}}{2\Delta t}$$

$$x_{n+1} = 2x_n - x_{n-1} + \Delta t^2 \frac{F(t_n, x_n)}{m}$$

Simple methods for a particle dynamics

Verlet and leap-frog methods are not "self-starting"



for leap-frog we need v_{-1}

for Verlet we need x_{-1}

possible solutions: use a backward Euler step

$$v_{-1} = v_0 - \Delta t \frac{F(t_0, x_0)}{m} \qquad \text{start leap-frog}$$

$$x_{-1} = x_0 - \Delta t\, v_0 + \frac{\Delta t^2}{2} \frac{F(t_0, x_0)}{m} \qquad \text{start Verlet}$$

Simple methods for a particle dynamics

Comments for Verlet and leap-frog methods

Leap-frog: energy conservation for some problems

Verlet: the method is very popular for computing trajectories in many-particle classical systems, e.g. molecular dynamics

Part 2c

Runge-Kutta methods

Runge-Kutta methods

Runge-Kutta methods are a family of single point methods.

Runge-Kutta methods propagate a solution over an interval by combining information from several Euler-style steps, and then using the information obtained to match a Taylor series expansion up to some order.

For many scientific users, fourth-order Runge-Kutta is not just the first word on solving ODE, but the last word as well

Second-order Runge-Kutta

$$x_{n+1} = x_n + C_1 \Delta x_1 + C_2 \Delta x_2 + \ldots$$

where $\Delta x_1 = h\, f(t_n, x_n)$ (h correposnds to Δt)

and $\Delta x_2 = h\, f(t_n + \alpha h, x_n + \beta \Delta x_1)$

(α and β) to be determined by matching to the Taylor series

… after some math …

$$k_1 = f(t_n, x_n)$$
$$k_2 = f(t_n + h, x_n + hk_1)$$
$$x_{n+1} = x_n + \frac{h}{2}(k_1 + k_2)$$

⇐ the iterative algorithm is identical to the modified Euler method

$$x_{n+1} = x_n + C_1 \Delta x_1 + C_2 \Delta x_2 + \ldots$$

Using explicit Euler

$$\Delta x_1 = \Delta t f(t_n, x_n)$$

and Δx_2 is based on $f(t, x)$ evaluated in the interval $t_n < t < t_{n+1}$

$$\Delta x_2 = \Delta t f(t_n + \alpha \Delta t, x_n + \beta \Delta x_1)$$

where α and β are to be determined together with C_1 and C_2

Let $\Delta t = h$, and substituting Δx_1 and Δx_2 in the first equation

$$x_{n+1} = x_n + C_1(hf_n) + C_2 hf(t_n + \alpha h, x_n + \beta h f_n)$$

Taylor series

$$f(t, x) = f_n + f|_{t_n} h + f|_{x_n} \Delta x + \ldots$$

Evaluation at $t = t_n + \alpha h$ and $x = x_n + \beta h f_n$ gives

$$f(t_n + \alpha h, x_n + \beta h f_n) = f_n + \alpha h f|_{t_n} + (\beta h f_n) f|_{x_n} + O(h^2)$$

Substituting this result into the last equation for x_{n+1} yields

$$\begin{aligned} x_{n+1} &= x_n + C_1 h f_n + C_2 h f_n + C_2 h \alpha h f|_{t_n} + C_2 h (\beta h f_n) f|_{x_n} \\ &= x_n + (C_1 + C_2) h f_n + h^2 (\alpha C_2 f|_{t_n} + \beta C_2 f_n f|_{x_n}) + O(h^3) \end{aligned}$$

The four free parameters can be determined by matching this equation to the the Taylor series through second order term
Taylor

$$\begin{aligned}
\overline{x}_{n+1} &= \overline{x}_n + \overline{x}'|_n h + \frac{1}{2}\overline{x}''|_n h^2 + \ldots \\
\overline{x}'|_n &= f(t_n, x_n) = f_n \\
\overline{x}''|_n &= (\overline{x}')'|_n = \frac{df}{dt}|_n = f_{t_n} + f_{x_n} x'|_n \\
\overline{x}_{n+1} &= \overline{x}_n + f_n h + \frac{1}{2}(f|_{t_n} + f|_{x_n} x'_n)
\end{aligned}$$

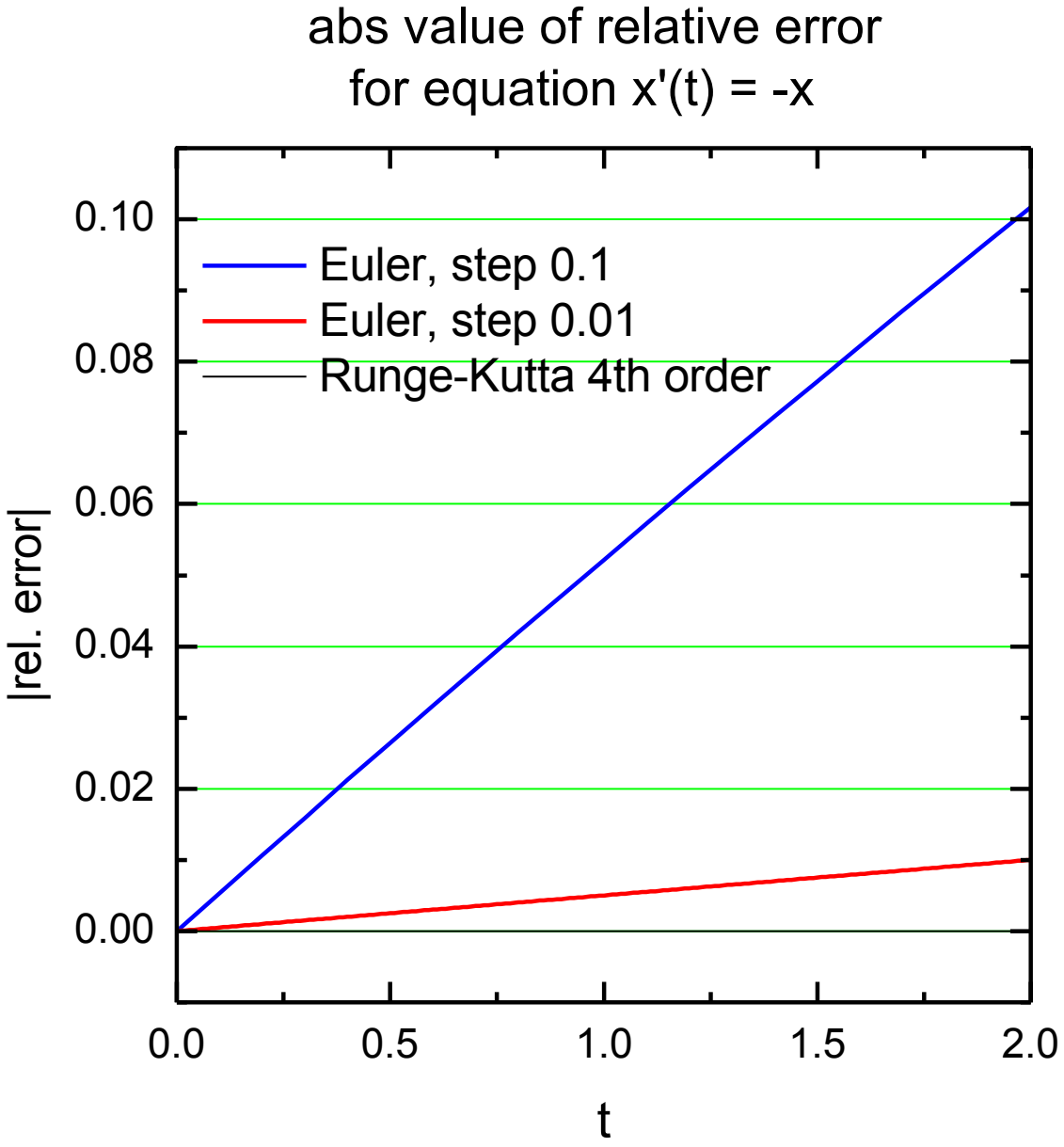
then $C_1 + C_2 = 1$, $\alpha C_2 = 1/2$ and $\beta C_2 = 1/2$.
The solution
$C_1 = C_2 = 1/2$, $\alpha = 1$, $\beta = 1$, and then

$$\begin{aligned}
x_{n+1} &= x_n + \frac{1}{2}h(k_1 + k_2) \\
k_1 &= f(t_n, x_n) = f_n \\
k_2 &= f(t_n + h, x_n + hk_1)
\end{aligned}$$

Forth-order Runge-Kutta method

2-nd order RK	$O(h^2)$
3-rd order RK	$O(h^3)$
4-th order RK	$O(h^4)$

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{hk_1}{2}\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, x_n + \frac{hk_2}{2}\right) \\
k_4 &= f(t_n + h, x_n + hk_3) \\
x_{n+1} &= x_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) + O(h^5)
\end{aligned}
$$



$$\frac{dx}{dt} = -x$$

$$x(t=0) = 1$$

example: C++ - RK method for 1st order ODE

```
/*-------------------------------------------
 4th-order Runge-Kutta method for ODE x'(t) = f(t,x)
 input ...
 f(t,x)- function supplied by a user
 ti  - initial value for an independent variable (t)
 xi  - initial value for a function x(t)
 tf  - find solution for this point t
 output ...
 xf  - solution at point tf, i.e. x(tf)
-------------------------------------------*/
double rk4_1st(double(*f)(double, double), double ti,
double xi, double tf)
{
   double xf;
   double h,k1,k2,k3,k4;
   h  = tf-ti;
   k1 = h*f(ti,xi);
   k2 = h*f(ti+h/2.0,xi+k1/2.0);
   k3 = h*f(ti+h/2.0,xi+k2/2.0);
   k4 = h*f(ti+h,xi+k3);
   xf = xi + (k1 + 2.0*(k2+k3) + k4)/6.0;
   return xf;
}
```

RK4 for second-order differential equations

The solution of two coupled 1st order ODEs

$$\frac{dx}{dt} = G(t,x,y), \qquad \frac{dy}{dt} = F(t,x,y)$$

can be obtained by RK4 method using

$$\begin{aligned}
K_1 &= G(t_n, x_n, y_n), & L_1 &= F(t_n, x_n, y_n),\\
K_2 &= G\big(t_n + \frac{h}{2}, x_n + K_1\frac{h}{2}, y_n + L_1\frac{h}{2}\big), & L_2 &= F\big(t_n + \frac{h}{2}, x_n + K_1\frac{h}{2}, y_n + L_1\frac{h}{2}\big),\\
K_3 &= G\big(t_n + \frac{h}{2}, x_n + K_2\frac{h}{2}, y_n + L_2\frac{h}{2}\big), & L_2 &= F\big(t_n + \frac{h}{2}, x_n + K_2\frac{h}{2}, y_n + L_2\frac{h}{2}\big),\\
K_4 &= G(t_n + h, x_n + K_3 h, y_n + L_3 h), & L_4 &= F(t_n + h, x_n + K_3 h, y_n + L_3 h)
\end{aligned}$$

with the incremental results being

$$\begin{aligned}
x_{n+1} &= x_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4),\\
y_{n+1} &= x_n + \frac{h}{6}(L_1 + 2L_2 + 2L_3 + L_4)
\end{aligned}$$

To solve second-order differential equation

$$\frac{d^2x}{dt^2} = F\big(t, x(t), \frac{dx(t)}{dt}\big)$$

we take

$$\frac{dx}{dt} = y \quad (\Leftrightarrow G = y) \qquad \text{and} \qquad \frac{d^2x}{dt^2} = \frac{dy}{dt} = F(t,x,y)$$

example: C++ - RK method for 2nd order ODE

```
/* input ...
   output ...  */

double rk4_2nd(double(*d1x)(double, double, double),
       double(*d2x)(double, double, double),
       double ti, double xi, double vi, double tf,
       double& xf, double& vf)
{
      double h,t,k1x,k2x,k3x,k4x,k1v,k2v,k3v,k4v;
      h = tf-ti;
      t = ti;
      k1x = h*d1x(t,xi,vi);
      k1v = h*d2x(t,xi,vi);
      k2x = h*d1x(t+h/2.0,xi+k1x/2.0,vi+k1v/2.0);
      k2v = h*d2x(t+h/2.0,xi+k1x/2.0,vi+k1v/2.0);
      k3x = h*d1x(t+h/2.0,xi+k2x/2.0,vi+k2v/2.0);
      k3v = h*d2x(t+h/2.0,xi+k2x/2.0,vi+k2v/2.0);
      k4x = h*d1x(t+h,xi+k3x,vi+k3v);
      k4v = h*d2x(t+h,xi+k3x,vi+k3v);

      xf = xi + (k1x + 2.0*(k2x+k3x) + k4x)/6.0;
      vf = vi + (k1v + 2.0*(k2v+k3v) + k4v)/6.0;
      return 0.0;
}
```

Runge-Kutta method is what you use when

- you don't know any better
- you have a computational problem where computational efficiency is of no concern

Runge-Kutta methods succeed virtually always

Practice: Apply Runge-Kutta method to the simple harmonic oscillator

- Equations
- Program
- Calculations

Error estimate and adaptive step-size

Any good program for solving ODEs should have

- ✓ an error control (accuracy)
- ✓ and adaptive step-size (efficiency)

The most intuitive way to vary the step size adaptively is the step doubling technique

do calculations for x1 with a step size h

do calculations for x2 with a step size h/2

compare the difference d= x1 – x2

if d < predefined acceptable error - use h

if d > predefined acceptable error - use h/2 …

Approach 1 (doubling technique)

do calculations for x1 with a step size h

do calculations for x2 twice with a step size h/2

$$x(t+h) = x_1 + Ch^{m+1}$$

$$x\left(t+\frac{h}{2}+\frac{h}{2}\right) = x_2 + 2C\left(\frac{h}{2}\right)^{m+1}$$

error estimate $\Delta = x_2 - x_1$

subtract the second from the first and find C

$$C = \frac{\Delta}{h^{m+1}}\frac{2^m}{2^m - 1}$$

$$x\left(t+\frac{h}{2}+\frac{h}{2}\right) = x_2 + \frac{\Delta}{2^m - 1}$$

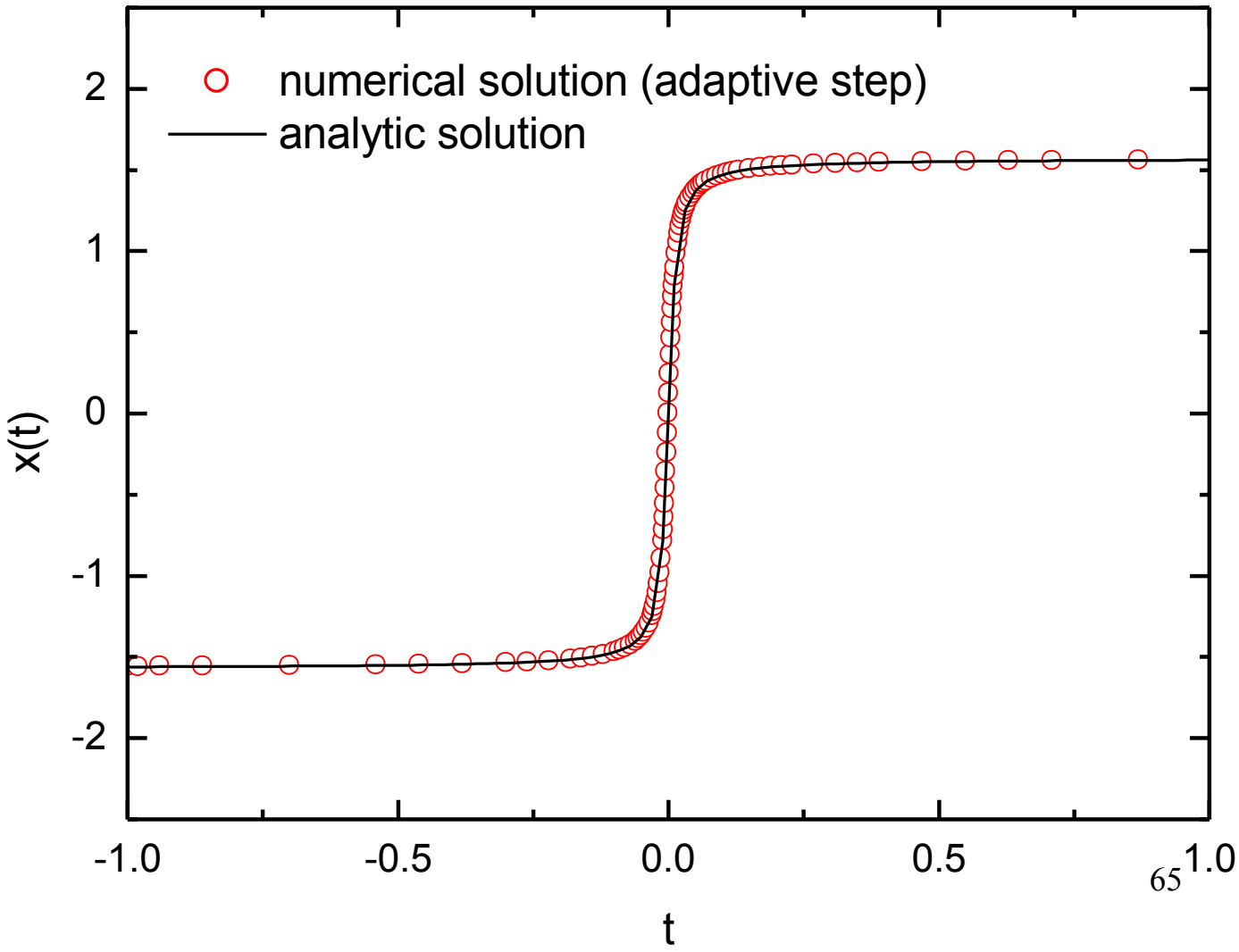
RK 4th order $x_{2c} = x_2 + \frac{\Delta}{15}$

however: needs three times more work

$$\frac{dx}{dt} = \frac{100}{1 + 10000\, t^2}$$

$$x(-1) = \arctan(-100)$$

The solution $x(t) = \arctan(100\, t)$



Approach 2 (more efficient)

Use two methods for the same interval h+t (accuracy m and m+1). Looks like still a lot work

However, Fehlberg found a fifth-order RK with six function evaluations, where forth-order RK is a combination of the first six functions (a very high efficiency for low price)

adaptive step size

Δ_1 accuracy on step h_1

Δ_2 desired (needs h_2)

$$\frac{\Delta_2}{\Delta_1} = \frac{h_2^5}{h_1^5}$$

practical

$$h_2 = \begin{cases} Sh_1 \left|\frac{\Delta_2}{\Delta_1}\right|^{0.2} & \text{if } \Delta_2 \geq \Delta_1 \\ Sh_1 \left|\frac{\Delta_2}{\Delta_1}\right|^{0.25} & \text{if } \Delta_2 < \Delta_1 \end{cases}$$

RK-Fehlberg method with error estimation

$$k_1 = hf(t_n, x_n)$$

$$k_2 = hf\left(t_n + \frac{1}{4}h, x_n + \frac{1}{4}k_1\right)$$

$$k_3 = hf\left(t_n + \frac{3}{8}h, x_n + \frac{3}{32}k_1 + \frac{9}{32}k_2\right)$$

$$k_4 = hf\left(t_n + \frac{12}{13}h, x_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right)$$

$$k_5 = hf\left(t_n + h, x_n + \frac{439}{216}k_1 - 8k_2 + \frac{3860}{513}k_3 - \frac{845}{4104}k_4\right)$$

$$k_6 = hf\left(t_n + \frac{1}{2}h, x_n - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right)$$

$$x_{n+1} = x_n + \left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right)$$

$$\text{Error} = \frac{1}{360}k_1 - \frac{128}{475}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6 + O(h^6)$$

Part 2d

Extrapolation methods

Extrapolation methods

Extrapolated methods are able to increase good accuracy with rather simple second-order algorithms.

Can be used when the calculations time is the issue.

The Bulirsch-Stoer method (1980) is a very good variation of the extrapolated mid-point method.

Part 2e

Multipoint methods

Multipoint methods

Multipoint methods use more than one point to advance the solution (i.e. points n, n-1, n-2, …)

The fourth-order Adams-Bashforth-Moulton method is a popular one

Global error - $O(h^4)$

There is a family of Adams methods for ODE.

Part 2f

Higher-order ODEs

Higher-order ODEs

In general, a higher-order ODE can be replaced by a system of first-order ODEs

$$x^{(n)} = f(t, x, x', x'', \ldots x^{(n-1)})$$

$$x(t_0) = x_0, \quad x^{(i)}(t_0) = x_0^i \ (i = 1, 2, \ldots n-1)$$

$$x_1 = x \qquad x_1' = x_2 \quad x_1(0) = x_0$$

$$x_2 = x' = x_1' \qquad x_2' = x_3 \quad x_2(0) = x'_0$$

$$x_3 = x'' = x_2' \qquad \ldots$$

$$\ldots \qquad x'_{n-1} = x_n \quad x_{n-1}(0) = x_0^{(n-2)}$$

$$x_n = x^{(n-1)} = x'_{n-1} \qquad x'_n = F(t, x_1, x_2, \ldots, x_n) \quad x_n(0) = x_0^{(n-1)}$$

Part 2g

Stiff ODEs

Stiff ODEs

Definitions of stiffness

- ✓ the step size required for stability is much smaller than the step size required for accuracy.
- ✓ if it contains some components of the solution that decay rapidly compared to other components of the solution.
- ✓ if the step size based on computational time is too large to obtain an accurate solution.

Stiff ODEs

There is a set of methods developed by Gear (1971) for solving stiff ODEs

good package: LSODE (Fortran) developed in Lawrence Livermore National Laboratory (LLNL)

see

ODEPACK - A Systematized Collection of ODE Solvers

https://computation.llnl.gov/casc/odepack/odepack_home.html

Part 2s

Summary

The explicit Euler method $x_{n+1} = x_n + f(t_n, x_n)\Delta t$

The implicit Euler method $x_{n+1} = x_n + f(t_{n+1}, x_{n+1})\Delta t$

The modified midpoint method

$$x^P_{n+1/2} = x_n + f(t_n, x_n)\Delta t/2, \qquad x^C_{n+1} = x_n + f(t_{n+1/2}, x^P_{n+1/2})\Delta t$$

The modified Euler method

$$x^P_{n+1} = x_n + f(t_n, x_n)\Delta t, \qquad x^C_{n+1} = x_n + \left(f(t_n, x_n) + f(t_{n+1}, x^P_{n+1})\right)\Delta t/2$$

The forth-order Runge-Kutta method

$$\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f(t_n + \frac{h}{2}, x_n + \frac{hk_1}{2}) \\
k_3 &= f(t_n + \frac{h}{2}, x_n + \frac{hk_2}{2}) \\
k_4 &= f(t_n + h, x_n + hk_3) \\
x_{n+1} &= x_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) + O(h^5)
\end{aligned}$$

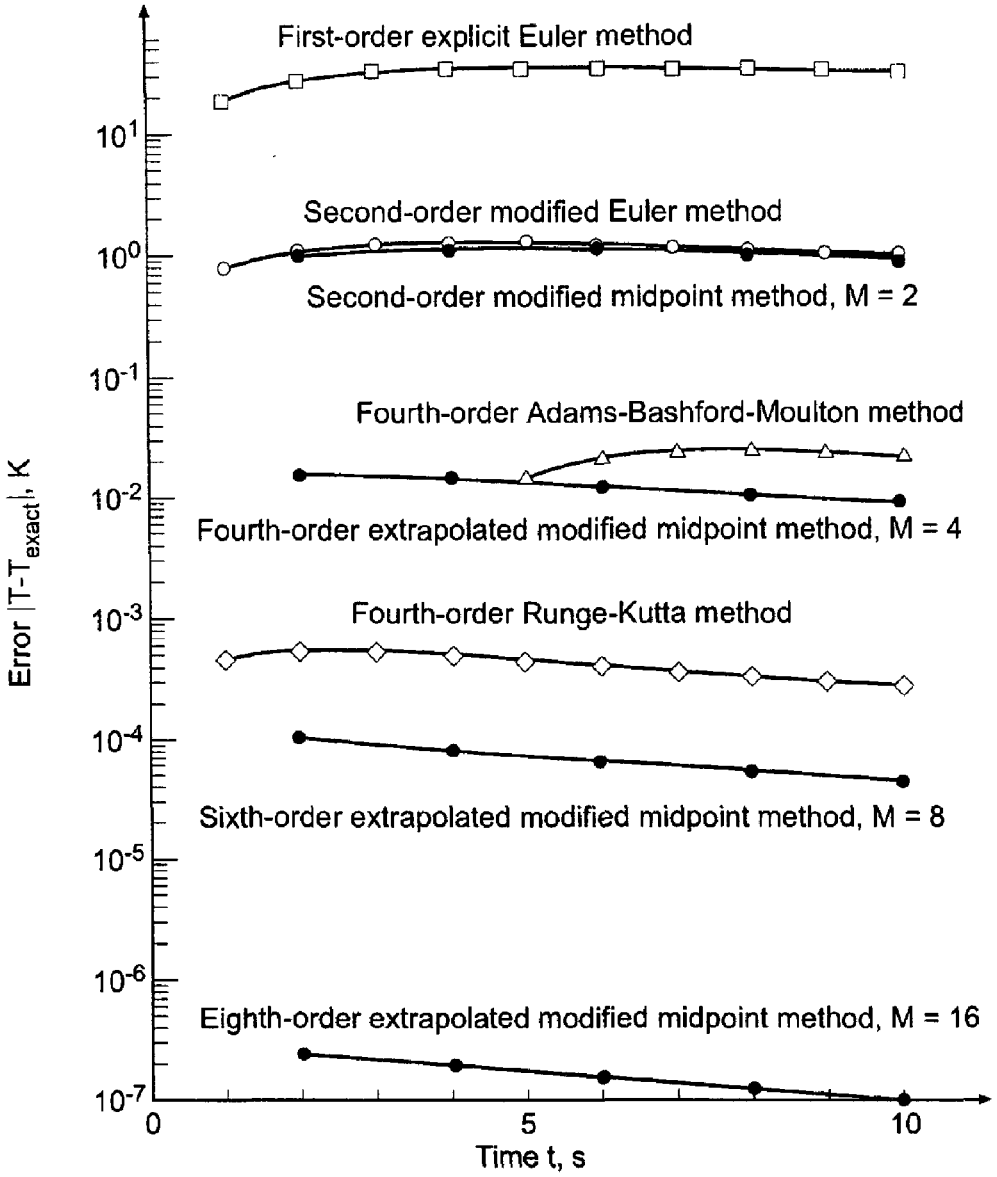


Figure 7.22 Errors in the solution of the radiation problem.

Example for errors: 1D ODE for a radiation problem (Hoffman 2001)

Packages for initial value ODEs

- ✓ ODEPACK - A Systematized Collection of ODE Solvers
  computation.llnl.gov/casc/odepack/odepack_home.html
- ✓ IMSL
- ✓ NAG
- ✓ Mathematica
- ✓ Maple
- ✓ see also http://gams.nist.gov/ (Guide to available mathematical software

Part 2x

Applications in physics
(a couple examples)

1st order ODE: examples in physics

$$\frac{dx}{dt} = f(x,t)$$

- Steady state flow of heat
- Decomposition and growth problems
- Flow of water through an orifice
- Atmospheric and oceanic pressure
- …

2nd order ODE: examples in physics

$$a\frac{d^2x}{dt^2} + b\frac{dx}{dt} + f(x,t) = 0$$

- Motion of a particle along a straight line
  vertical projectile motion
  rocket motion
  oscillatory motion (harmonic, damped, forced)
- Electric circuits
- Rolling bodies
- …

A system of two 2nd order ODEs: examples

$$\frac{d^2x}{dt^2} + f_x(x, y, x', y', t) = 0$$

$$\frac{d^2y}{dt^2} + f_y(x, y, x', y', t) = 0$$

- Projectile motion in 2D plane
- Planetary motion in 2D plane

Part 3

Boundary-value problem
(one-dimension 2nd order ODE)

Boundary-value problem

Boundary-value problems – involve differential equations with specified **boundary** conditions:

example: one-dimension second order ODE (where *P* and *Q* some constants)

$$\frac{d^2y}{dx^2} + P\frac{dy}{dx} + Qy = F(x)$$

$$y(x_1) = y_1 \quad \text{and} \quad y(x_2) = y_2$$

The boundary-value problem is more difficult to solve than the similar initial-value problem with the same differential equation

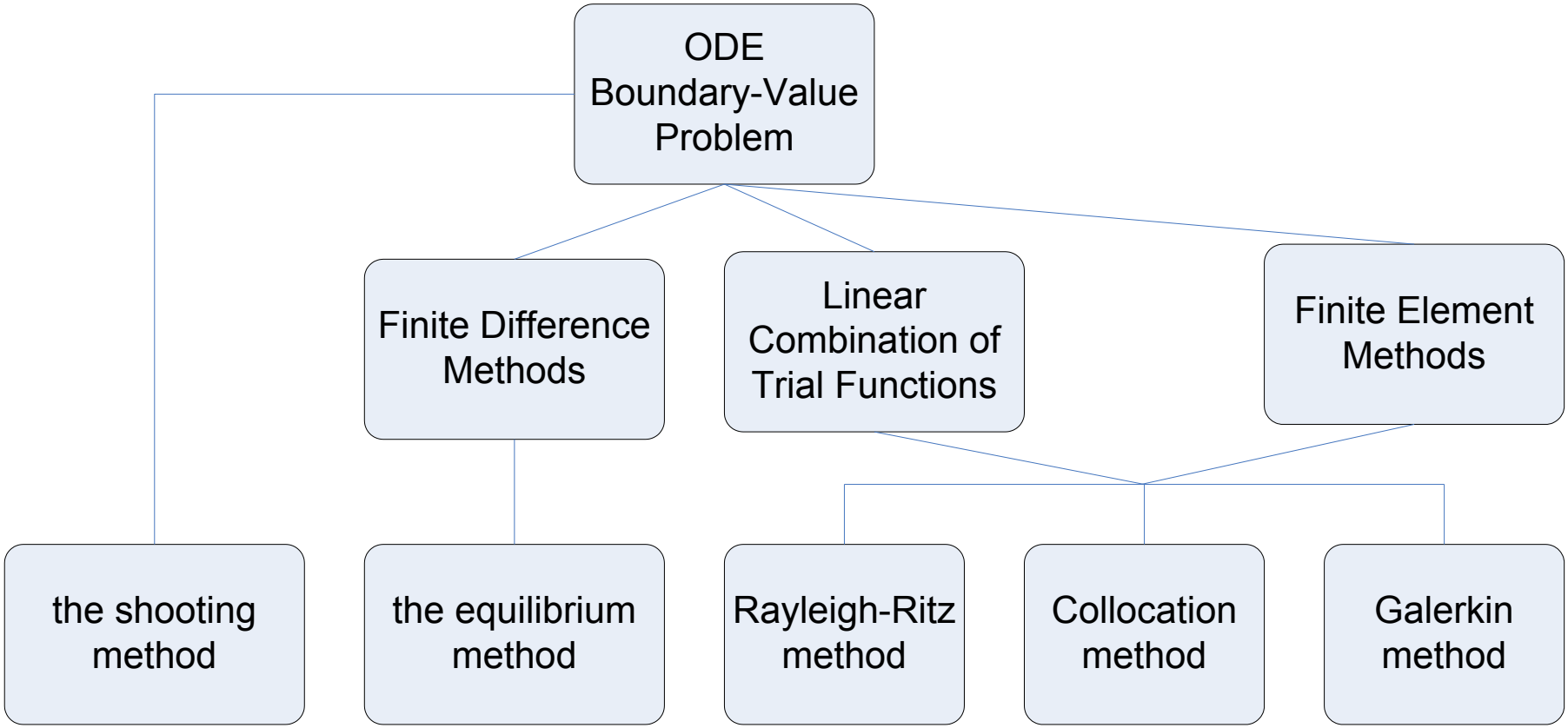
Three types of boundary conditions

the solution domain is closed $x_1 \le x \le x_2$

the solution $y(x)$ must satisfy two boundary conditions

- ✓ for the function y(x) - Dirichlet boundary conditions
- ✓ for the derivatives y’ (x) - Neumann boundary conditions
- ✓ for a combination of y(x) and y’ (x) – mixed boundary conditions

Principal methods for B-V problem



The shooting method

The key idea of the shooting method is to transform the boundary value ODE into a system of first-order ODEs and solve as an initial value problem.

Only boundary condition on one side is used as one of the initial conditions. The additional initial condition is assumed.

Then an iterative approach is used to vary the assumed initial condition till the boundary condition on the other side is satisfied.

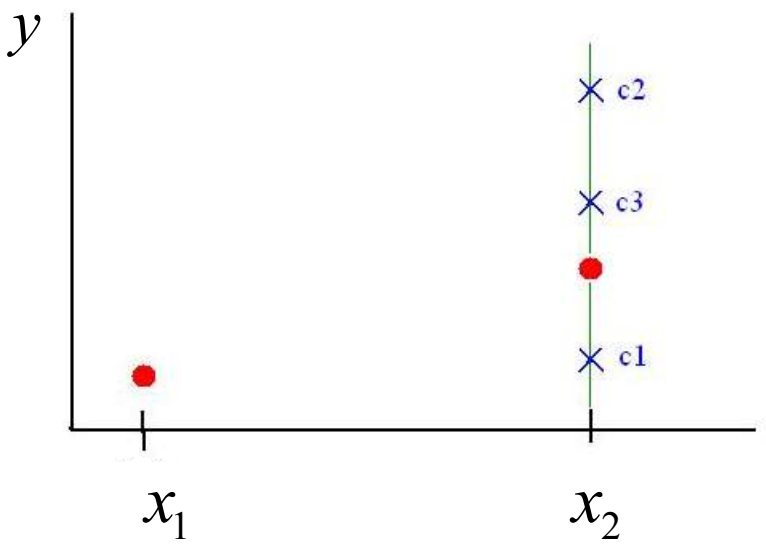
The shooting method (cont)

Assume $y(x_1) = y_1$ and $y(x_2) = y_2$

Let us consider an initial-value problem with

$$y(x_1) = y_1 \quad \text{and} \quad y'(x_1) = c$$

where c is a parameter to be adjusted



We use a root search algorithm to find c that insures $c_i - y_2 < \delta$

Quite often the fourth-order Runge-Kutta is combined with the secant method

Let's consider a general second-order ODE with Dirichlet boundary conditions

$$y"(x) + P(x,y)y'(x) + Q(x,y)y(x) = F(x), \tag{1.3}$$

where $y(x_1) = y_1$ and $y(x_2) = y_2$. Let's also introduce a new parameter $z = y'(x_1)$ that specifies the initial condition for the derivatives. Then equation (1.3) can be solved as initial value problem by some standard method. For a value of a parameter z_1 the solution would give a value $y(x_2) = c_1$ with $c_1 \neq y_2$ (unless a very lucky guess for the initial z_1). Then we assume a second value for the parameter z, and solve equation (1.3) again. For $z = z_2$ it would give $y(x_2) = c_2$. Now we may construct an iterative procedure to find such a z_n that $c_n = y_2$. Practically we deal with finding roots of a non-linear equation. The method of secant is a very common approach to solve the problem. In this method the next approximation to a solution for $f(x) = 0$ is written as

$$x_{k+1} = x_k - \frac{f(x_k)}{f_k - f_{k-1}}(x_k - x_{k-1}). \tag{1.4}$$

Then the secan method provides the next value of z_{k+1}

$$z_{k+1} = z_k - \frac{c_k - y_2}{c_k - c_{k-1}}(z_k - z_{k-1}). \tag{1.5}$$

For a linear ODE we can apply the superposition principle:

1. Compute two solutions for $z(x_1) = y_1^{(1)}$ and $z(x_1) = y_1^{(2)}$. Let these solutions be $y^{(1)}(x)$ and $y^{(2)}(x)$.
2. Form a linear combination of these two solutions

$$y(x) \;=\; a_1 y^{(1)}(x) + a_2 y^{(2)}(x)$$

and apply at $x = x_1$ and $x = x_2$

$$\begin{aligned} \text{at } \mathrm{x} = \mathrm{x}_1: \qquad y_1 &= a_1 y_1^{(1)} + a_2 y_1^{(2)} \\ \text{at } \mathrm{x} = \mathrm{x}_2: \qquad y_2 &= a_1 y_2^{(1)} + a_2 y_2^{(2)} \end{aligned}$$

3. Solve the system for c_1 and c_2:

$$a_1 + a_2 \;=\; 1 \quad \Rightarrow \quad y_2 \;=\; a_1 y_2^{(1)} + (1 - a_1) y_2^{(2)} \;=\; y_2^{(2)} + a_1 [y_2^{(1)} - y_2^{(2)}]$$

$$\Rightarrow \quad a_1 \;=\; \frac{y_2 - y_2^{(2)}}{y_2^{(1)} - y_2^{(2)}}, \qquad a_2 \;=\; \frac{-y_2 + y_2^{(1)}}{y_2^{(1)} - y_2^{(2)}}$$

We see that no further iteration in Eq. (1.5) is necessary for linear ODE because

$$y'(x_1) \;=\; a_1 y_1^{(1)} + a_2 y_1^{(2)} \;=\; y_1^{(2)} + \frac{y_2 - y_2^{(2)}}{y_2^{(2)} - y_2^{(1)}} (y_1^{(2)} - y_1^{(1)})$$

which is (1.5) since $c_1 = y_2^{(1)}$, $c_2 = y_2^{(2)}$, $z_1 = y_1^{(1)}$, and $z_2 = y_1^{(2)}$

The equilibrium B-V method

Idea: construct a finite difference approximation of the exact ODE at every point on a discrete finite difference grid. Then a system of equations must be solved simultaneously. Here are the steps:

- ✓ Discretizing the continuous solution domain into a discrete finite difference grid
- ✓ Approximating the exact derivatives in the boundary-value ODE by algebraic finite difference approximations
- ✓ Substituting the FDAs into the ODE to obtain an algebraic finite difference equation
- ✓ Solving the resulting system of algebraic FDEs (for linear ODEs – a system of linear equations)

Equilibrium method

Let us consider a BV problem for linear 2nd order ODE :

$$y''(x) + P(x)y'(x) + Q(x)y(x) = F(x) \quad (1$$

with Dirichlet boundary conditions $y(a) = y_1$, $y(b) = y_2$.

Introduce a grid in the interval $[x_1, x_2]$ by dividing the interval into N equal subintervals of size h. This produces two boundary grid points $x_0 = a$ and $x_N = b$, and $N-1$ interior grid points x_i, $i = 1, 2..., N-1$. The values $y(x_0) = y(a)$ and $y(x_N) = y(b)$ are known but the values $y(x_i)$, $i = 1, 2, ..., N-1$ must be determined. The "lattice spacing" is $\Delta x = h = \frac{b-a}{N}$. From the Taylor expansions for $y(x_{i+1})$ and $y(x_{i_1})$ we can obtain the following centered difference formula:

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2h} - \frac{h^2}{6}y'''(\eta)$$

$$y_i'' = \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + \frac{h^2}{12}y''''(\eta)$$

for some $\eta \in [a, b]$.

If the higher order terms are discarded from the above formulas and the approximations are used in the original differential equation (1), the second order accurate finite difference approximation to the BVP becomes a system of simultaneous linear algebraic equations

$$\left[\frac{y_{i-1}-2y_i+y_{i+1}}{h^2}+O(h^2)\right]+P_i\left[\frac{y_{i+1}-y_{i-1}}{2h}+O(h^2)\right]+Q_iy_i=F_i$$

Multiplying this by h^2 we get

$$(1-\frac{h}{2}P_i)y_{i-1}+(-2+h^2Q_i)y_i+(1+\frac{h}{2}P_i)y_{i+1}=h^2F_i$$

for points $i=1,2,...N$. In matrix notation the system becomes

$$\mathbf{Ay}=\mathbf{b}$$

where $\mathbf{A}$ is a tridiagonal matrix. This equation can be solved by Thomas algorithm

Theorem of Uniqueness: If p(x); q(x) and r(x) are continuous and $q(x)>0$ on [a; b] the problem $\mathbf{Ay}=\mathbf{b}$ has a unique solution provided $h<2/L$ where $L=\max_{a\le x\le b}|p(x)|$. Further, if y'''' is continuous on [a; b] the truncation error is $O(h^2)$.

Neumann boundary conditions

Neumann boundary conditions

The shooting method: same idea as for the Dirichlet boundary conditions (1) but shooting for $z'(x_2) = y'_2$ rather than for $z(x_2) = y_2$.

The equilibrium method needs some modification:
Consider

$$y''(x) + P(x)y'(x) + Q(x)y(x) = F(x)$$

with boundary conditions $y(a) = y_1$, $y'(b) = y'_2$.
Let us write the finite-difference equation for the last point:

$$(1 - \frac{h}{2}P_N)y_{N-1} + (-2 + h^2Q_N)y_N + (1 + \frac{h}{2}P_N)y_{N+1} = h^2F_N$$

The value of y_{N+1} is unknown. We can approximate it from

$$y'_N = \frac{y_{N+1} - y_{N-1}}{2h} + O(h^2)$$

and get $$2y_{N-1} + (-2 + h^2Q_N)y_N = h^2F_N - h(2 + hP_N)y'_N$$

For points inside the [a,b] interval we can use the old formula

$$(1 - \frac{h}{2}P_i)y_{i-1} + (-2 + h^2Q_i)y_i + (1 + \frac{h}{2}P_i)y_{i+1} = h^2F_i$$

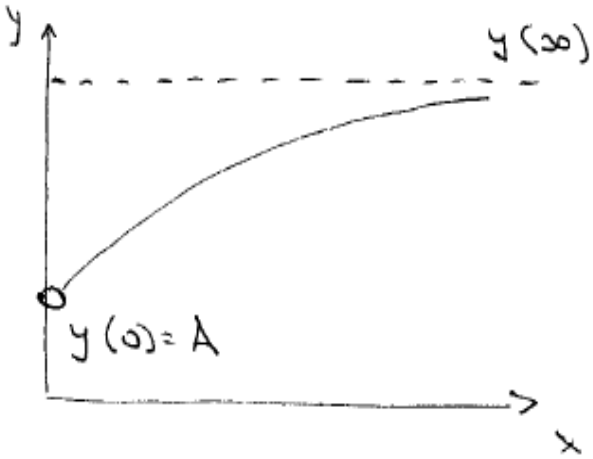
which gives again the tridiagonal system of equations.

Boundary Condition at Infinity

Two procedures for implementing boundary conditions at infinity

- ✓ Replace ∞ with a large value of x (x = X)
- ✓ Match an asymptotic solution at large values of x

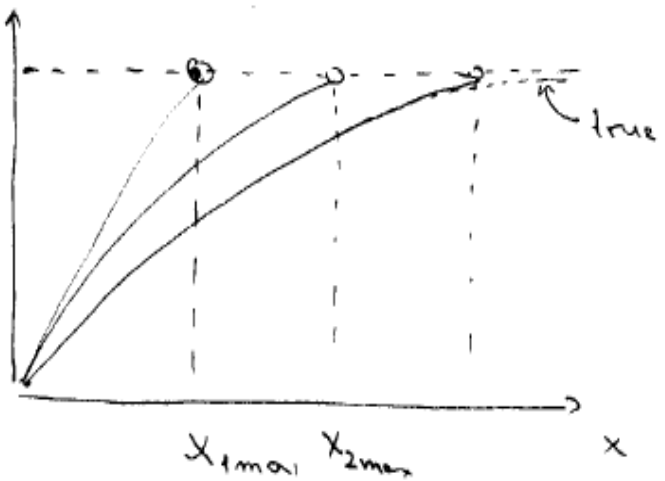
Boundary conditions at Infinity



two procedures

1. Replace ∞ with a large value of x
2. An asymptotic solution at large values of x.

for 1).



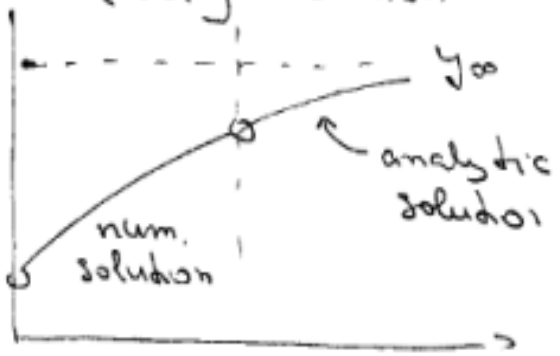
comparing solutions x_{1max}, x_{2max} ... till solutions change by less than some tolerance.

for 2) $y(x)\big|_{\text{asym}} = F(x)$ $\quad \bar{X} \leq x < \infty$

using the $\rightarrow$ $y(X) = F(X) = \bar{Y}$.

i.e. the bound. condition $y(\infty) = y_\infty$ is replaced on $y(X) = Y$

(very common methods in quantum mechanics.)



Higher-order equilibrium methods

For the shooting method it is quite easy to go to higher-order approximation (initial value problem + roots of equations).

For equlibrium method it is more complicated. Popular fourth-order methods: five-point method and three-point method.

Five-point method: using Taylor series at $y_{i-2}, y_{i-1}, y_{i+1}, y_{i+2}$ (y_i is a base point). One can show that

$$y_i' = \frac{-y_{i+2} + 8y_{i+1} - 8y_{i-1} + y_{i-2}}{12h} + O(h^4)$$

and

$$y_i'' = \frac{-y_{i+2} + 16y_{i+1} - 30y_i + 16y_{i-1} - y_{i-2}}{12h^2} + O(h^4)$$

We can use these equations for the interior points, but for the initial and final points we need to modify them. For example, one can use forward or backward finite-difference formulas (with some loss of accuracy).

Quick comment on the non-linear equations

1. For the shooting method the solution is straightforward.

2. For the equilibrium methods the situation is more complicated. One gets the system of non-linear finite-difference equations which can be solved by Newton's iteration method.

Comments

1. The system of 2nd-order boundary-value ODEs can be solved by
   - The shooting method by replacing each second-order ODE by two first-order ODEs and solving the coupled systems of first-order ODEs.
   - Each seond-order ODE can be solved by the equilibrium method and the coupling between the individual ODEs can be accomplished by relaxation

   By either approach, solving a coupled system of second-order ODEs can be quite difficult

2. Higher-order boundary-value ODEs can be solved by
   - The shooting method by replacing each higher-order ODE by a system of first-order ODEs
   - *Some* higher-order ODEs could be reduces to systems of 2nd-order ODEs which can be solved by the equilibrium method.

   Direct solution of higher-order ODEs by equilibrium method can be quite difficult

Part 4

Eigenvalue problem

Eigenvalue problem

Eigenvalue problems - equilibrium problems where the solution exists only for special values (eigenvalues) of a parameter of the problem

example:

$$\frac{d^2y}{dx^2} + k^2 y = 0 \qquad y(0) = y(1) = 0$$

solutions exists for $k = \pm \pi n$

Shooting methods are not well suited for solving eigenvalue problems

Eigenvalue problems are generally solved by the equilibrium method

Example

Let us solve $y'' + k^2 y = 0$. We take four interior points and use 2nd-order centered difference in place of $y''(x)$

$$\frac{y_{i+1} - 2y_i + y_{i+1}}{h^2} + O(h^2) + k^2 y_i = 0$$
$$\Rightarrow \; y_{i-1} - (2 - h^2 k^2) y_i + y_{i+1} = 0$$

We get

$$\begin{aligned}
0 \quad & y_1 = 0 \\
0.2 \quad & y_1 - (2 - 0.04k^2) y_2 + y_3 = 0 \\
0.4 \quad & y_2 - (2 - 0.04k^2) y_3 + y_4 = 0 \\
0.6 \quad & y_3 - (2 - 0.04k^2) y_4 + y_5 = 0 \\
0.8 \quad & y_4 - (2 - 0.04k^2) y_5 + y_6 = 0 \\
1 \quad & y_6 = 0
\end{aligned}$$

In the matrix form

$$\begin{bmatrix} 2-0.04k^2 & -1 & 0 & 0 \\ -1 & 2-0.04k^2 & -1 & 0 \\ 0 & -1 & 2-0.04k^2 & -1 \\ 0 & 0 & -1 & 2-0.04k^2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = 0 \Leftrightarrow (\mathbf{A}-0.04k^2\mathbf{I})\mathbf{y} = 0$$

where

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

The equation $(\mathbf{A}-\lambda\mathbf{I})\mathbf{y} = 0$ is a classical eigenvalue problem of linear algebra. Solutions exist if

$$\det[\mathbf{A}-\lambda\mathbf{I}] = 0 \qquad (1)$$

Denote $2-0.04k^2 \equiv z$, then the Eq. (1) reduces to

$$z^4 - 3z^2 + 1 = 0 \quad \Rightarrow \quad z_{1,2} = \pm 1.618, \quad z_{3,4} = \pm 0.618$$

1D stationary Schrödinger equation

Bound states

$$\varphi''(x) + \frac{2\mu}{\hbar^2}(E - V(x))\varphi(x) = 0$$

$$\varphi(x) \to 0 \quad |x| \to \infty$$

Solutions exist for only specific energies E (eigenstates)

The radial Schrödinger equation for a spherically symmetric potential

$$R''(\rho) + \left(\frac{2\mu}{\hbar^2}(E - V(\rho)) - \frac{l(l+1)}{r^2}\right)R(\rho) = 0$$

Numerov's method

Let's consider a second-order ODE of the form

$$y''(x) + f(x)y(x) = 0 \tag{1.7}$$

Using Taylor's expansion

$$\begin{aligned} y(x_n + h) &= y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + \frac{h^4}{4!}y''''(x_n) + \frac{h^5}{5!}y'''''(x_n) + \dots \\ y(x_n - h) &= y(x_n) - hy'(x_n) + \frac{h^2}{2!}y''(x_n) - \frac{h^3}{3!}y'''(x_n) + \frac{h^4}{4!}y''''(x_n) - \frac{h^5}{5!}y'''''(x_n) + \dots \end{aligned}$$

the sum of the two equations is

$$y_{n-1} + y_{n+1} = 2y_n + h^2 y_n'' + \frac{h^4}{12}y_n'''' + O(h^6)$$

the solving for y_n'' and using $y_n'' = -f_n y_n$

$$h^2 f_n y_n = 2y_n - y_{n+1} - y_{n-1} + \frac{h^4}{12}y_n''''$$

The forth-order derivative can be written as

$$y_n'''' = \frac{d^2}{dx^2} y_n'' = \frac{d^2}{dx^2}(-f(x)y(x))$$

Using the centered-difference formula for the second-order derivative gives

$$y_n'''' = -\frac{f_{n+1}y_{n+1} - 2f_n y_n + f_{n-1}y_{n-1}}{h^2}$$

and then the original equation can be written as

$$h^2 f_n y_n = 2y_n - y_{n+1} - y_{n-1} - \frac{h^4}{12}\frac{1}{h^2}(f_{n+1}y_{n+1} - 2f_n y_n + f_{n-1}y_{n-1})$$

Regrouping terms gives

$$\left(1 + \frac{h^2}{12} f_{n+1}\right) y_{n+1} = \left(2 - \frac{5}{6} h^2 f_n\right) y_n - \left(1 + \frac{h^2}{12} f_{n-1}\right) y_{n-1}$$

The accuracy is $O(h^4)$

Solving 1D Schrödinger equation

Key points: consider a boundary value problem

$$\varphi''(x) + 2(E - V(x))\varphi(x) = 0 \quad \varphi(x) \to 0 \text{ or } \to \exp(-\sqrt{2|E|}\, x) \;\; |x| \to \infty$$

1. define boundary conditions for two left-end and two right-end points $\varphi(x_{-n}) \quad \varphi(x_{-n+1}) \qquad \varphi(x_n) \quad \varphi(x_{n-1})$
2. assume two values for the energy $\mathrm{E}_{\min}, \mathrm{E}_{\max}$
3. for canceling numerical errors it is better to solve the equation moving to some matching point from the left and from the right (using Numerov's method)
4. at the matching point the logarithmic derivatives are continuous $\varphi_l'(x_m)/\varphi_l(x_m) = \varphi_r'(x_m)/\varphi_r(x_m)$ where index l mean the left marching solution and r – right (scaling 'right' solution on the 'left' is recommended
5. use bisectional method to find E that satisfies the cond.

Summary

The shooting method:

Good:

- any initial value ODE method can be used
- it is easy to achieve higher-order accuracy

Not good:

- shooting for more than one boundary condition is time consuming
- a nonlinear problem is to be solved

Summary

The equilibrium method:

Good:

- automatically satisfied to the boundary conditions

Not good:

- it can difficult to achieve higher than 2^{nd} order accuracy
- a system of Finite Difference Equations must be solved
- nonlinear ODEs yield a system of nonlinear FDEs.