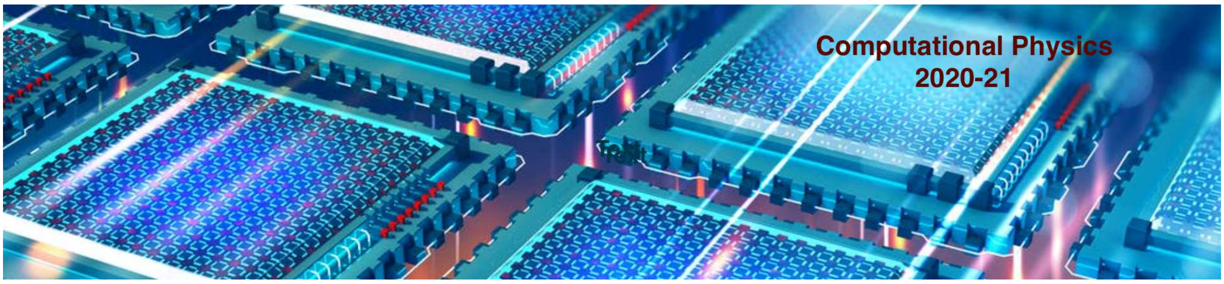
Computational Physics

numerical methods with C++ (and UNIX)

2020-21



Fernando Barao
Instituto Superior Técnico, Dep. Fisica
email: fernando.barao@tecnico.ulisboa.pt

Computational Physics

Physics problems

and Solutions

Fernando Barao, Phys Department IST (Lisbon)

ODEsolver class

```
int main() {

  //differential equations
  // x[0] = time
  // x[1] = theta
  // x[2] = omega (angular velocity)

  auto f1 = [](double* xval, double* par) { return xval[2]; };
  auto f2 = [](double* xval, double* par) { return -sin(xval[1]); };

  vector<TF1> F {
    TF1("f1", f1, 0., 1000., 0, 3),
    TF1("f2", f2, 0., 1000., 0, 3)
  };

  ODEsolver solver(F);

  vector<ODEpoint> v_euler = solver.Euler(ipoint, step, T);
}
```


ODEsolver class

```
///////////////////////////////////////////////////// Euler

vector<ODEpoint> ODEsolver::Euler( ODEpoint iPoint, double h, double T) {

  vector<ODEpoint> vP;
  vP.push_back(iPoint);

  int Nh = 1 + int(T/h); // nb steps

  ODEpoint cPoint(iPoint);
  ODEpoint next(iPoint);
  for (int i = 0; i < Nh; ++i) {
    next[0] = cPoint.T() + h;
    next[2] = cPoint.X(1) + h*F[1].EvalPar(cPoint.GetArray());
    next[1] = cPoint.X(0) + h*F[0].EvalPar(cPoint.GetArray());
    vP.push_back(next);
    cPoint=next;
  }
  return vP;
}
```

Electrostatic field lines

Let's consider N electric charges of charge Q_i located at fixed positions given by ther position vector $\vec{r}_i$, $i = 1, 2, ..., N$

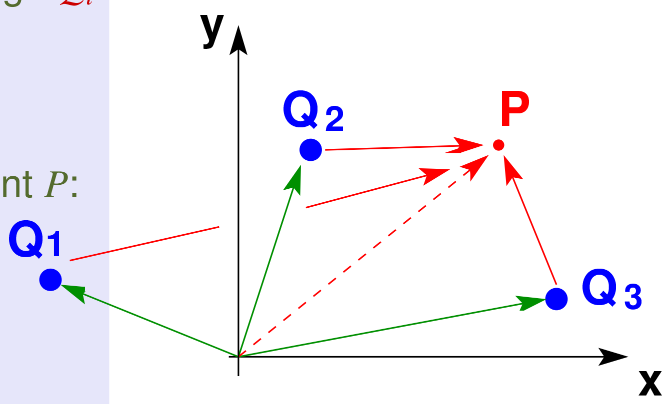
The electric field produced at point a point P:

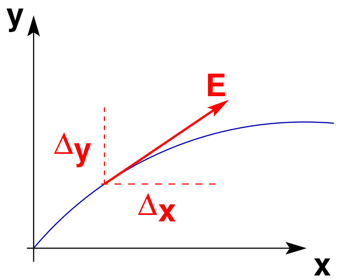
$$\vec{E}(\vec{r}) = \frac{1}{4\pi\varepsilon_0} \sum_{i=1}^{N} \frac{Q_i}{|\vec{r}-\vec{r}_i|^2} \vec{e}_{r_i}$$

The electric field components:

$$\vec{E}_x = \frac{1}{4\pi\varepsilon_0} \sum_{i=1}^{N} \frac{Q_i(x-x_i)}{[(x-x_i)^2+(y-y_i)^2]^{3/2}}$$
$$\vec{E}_y = \frac{1}{4\pi\varepsilon_0} \sum_{i=1}^{N} \frac{Q_i(y-x_i)}{[(x-x_i)^2+(y-y_i)^2]^{3/2}}$$

The field lines are curves whose tangent lines at every point are parallel to the electric field at the point.





Computing the field lines

The field line is derived from the equation:

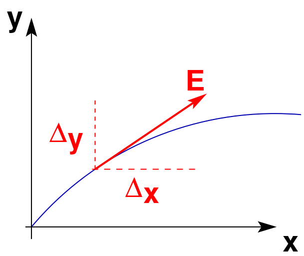
$$\boxed{d\vec{\ell} \times \vec{F} = \vec{0}}$$

$$d\vec{\ell} = (dx, dy, dz)$$
$$\vec{F} = \left(F_x, F_y, F_z\right)$$

$$d\vec{\ell} \times \vec{F} = \begin{vmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \\ dx & dy & dz \\ F_x & F_y & F_z \end{vmatrix} = \vec{e}_x \left(dyF_z - dzF_y\right) - \vec{e}_y \left(dxF_z - dzF_x\right) + \vec{e}_z \left(dxF_y - dyF_x\right)$$

Therefore, the eqs to solve:

$$\begin{pmatrix} dy\ F_z - dz\ F_y \\ dz\ F_x - dx\ F_z \\ dx\ F_y - dy\ F_x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \frac{dx}{F_x} = \frac{dy}{F_y} = \frac{dz}{F_z}$$

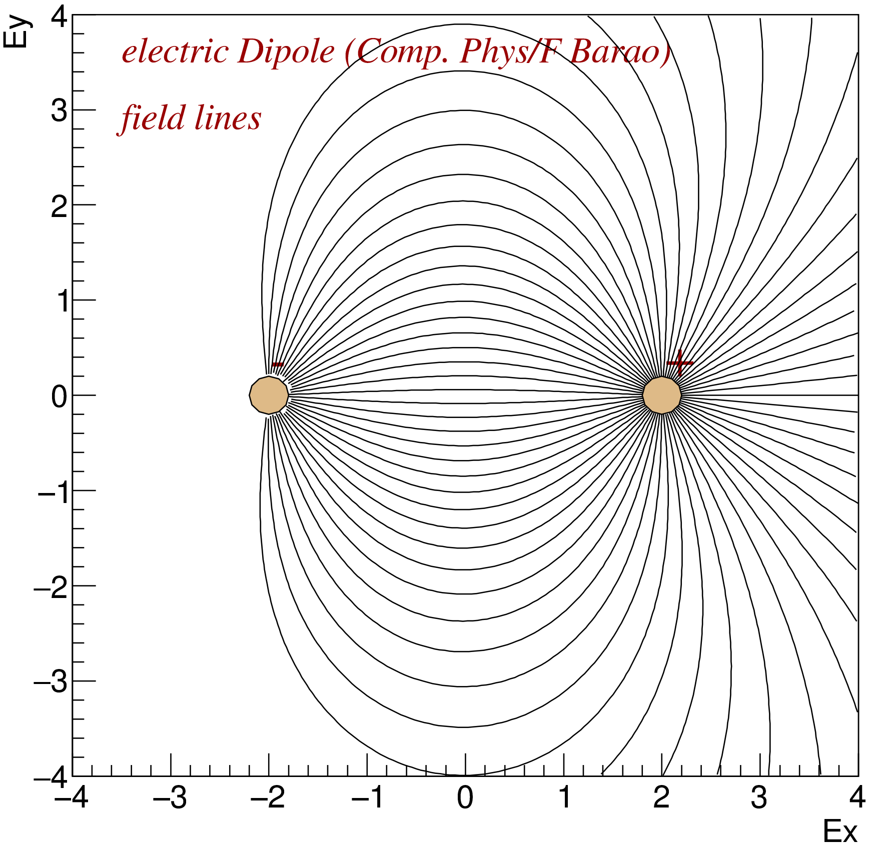


Taking the step in space to be the scalar $d\ell$ that can be expressed as:

$$\begin{aligned} d\ell &= \sqrt{(dx)^2 + (dy)^2 + (dz)^2} \\ &= \sqrt{(dx)^2 + \left(\frac{F_y}{F_x}\right)^2 (dx)^2 + \left(\frac{F_z}{F_x}\right)^2 (dx)^2} \\ &= \frac{dx}{F_x} \sqrt{F_x^2 + F_y^2 + F_z^2} \\ &= \frac{F}{F_x}\ dx \end{aligned}$$

$$\boxed{dx = \frac{F_x}{F} d\ell \quad dy = \frac{F_y}{F} d\ell \quad dz = \frac{F_z}{F} d\ell}$$

Electrostatic field lines



$$\Phi(r) = \frac{Q_1}{\sqrt{(x-x_1)^2 + (y-y_1)^2}} + \frac{Q_2}{\sqrt{(x-x_2)^2 + (y-y_2)^2}}$$

$$\vec{E} = -\, grad(\Phi)$$

```
// start from point around charge
x = ...
y = ...

LOOP OVER

 // compute gradient (TVectorD used)
 h = ...(optimal step)
 Dx = f(x+h,y) - f(x-h,y)
 Dx /=h
 Dy = f(x,y+h) - f(x,y-h)
 Dy /=h

 //electric field
 TevctorD E(2);
 E[0]=-Dx; E[1]=-Dy;

 // iteration point
 x += step*E[0]/Emag;
 y += step*E[1]/Emag;
```


ODEs: boundary value problems

✔ There are problems in physics where differential equations conditions are given at the boundaries.

Suppose for instance a rod that is conducting heat from two reservoirs at different temperatures, $T(x=0) = T_0$ and $T(x=L) = T_L$

The temperature equation in the bar:

$$\frac{\partial T}{\partial t} = \frac{k}{c\rho}\frac{\partial^2 T}{\partial x^2}$$

k, thermal conductivity (W.m^{-1}.K^{-1})
c, specific heat capacity (J.Kg^{-1})
ρ, density (Kg.m^{-3})

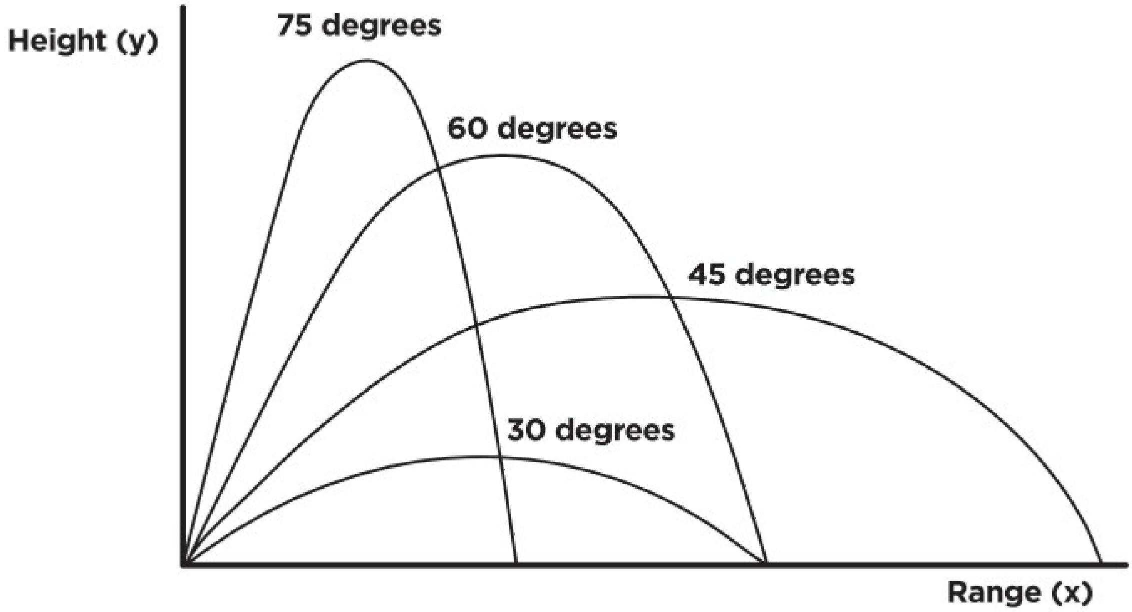
✔ In equilibrium ($\frac{\partial T}{\partial t} = 0$), the temperature equation is an example of a linear 2nd-order boundary value problem:

$$a_2(x)\, y''(x) + a_1(x)\, y'(x) + a_0(x)\, y(x) = f(x)\ , \qquad x \in [a,b]$$

$$\alpha_0 y(a) + \alpha_1 y'(a) = \lambda_1\ , \qquad |\alpha_0| + |\alpha_1| \neq 0$$

$$\beta_0 y(b) + \beta_1 y'(b) = \lambda_2\ , \qquad |\beta_0| + |\beta_1| \neq 0$$

BV problems: rockets



BV problems: shooting method

- 2nd-order differential equation with boundary values on interval $x \in [a, b]$

$$\boxed{\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right)}$$

boundary values (Dirichlet conditions):

$y(a) = \alpha$

$y(b) = \beta$

- we can transform the **boundary value problem** into an **initial value problem**

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right)$$

initial values:

$y(a) = \alpha$

$\left.\frac{dy}{dx}\right|_{(a)} = u$

- guess u and solve the initial value problem by going from $x = a$ to $x = b$
- the iterated value is a function of u, $g(u) \equiv y_{iter}(b)$
- the determination of u is a root finding problem
  the equation to solve:

$$\boxed{F(u) = y(b) - g(u) = 0}$$

- the use of the secant method requires two iterated values u_0, u_1
  low convergence speed if f' small

  next initial value iteration:

$$u_{n+1} = u_n - g(u_n)\frac{u_n - u_{n-1}}{g(u_n) - g(u_{n-1})}$$

Roots: newton-raphson

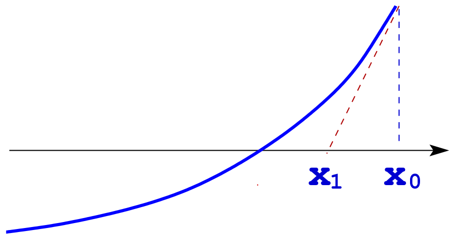
- ✔ we start from a point x_0 near the zero of the function
  the convergence of the method depend how far we are from the zero
- ✔ we can approximate the function $f(x)$ at x_0 by a 1st order polynomial and its root will first the first iteration:

$$f(x_1) = f(x_0) + (x_1 - x_0) f'(x_0) = 0$$

- ✔ the iterated values:

$$x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}$$

the analytic derivative of the function is needed!
avoid large step iterations (huge variations on derivative)!



Newton method

```
double eps = 1.e-3;
int i=0, iter = 100;
do {
  i = i + 1;
  double fx = f->Eval(x);
  double fdx = fd->Eval(x);
  xn = x - fx/fdx;
  x = xn;
  if(i >= iter) return -999.;
} while (fabs(fx) >= eps);
return xn;
```

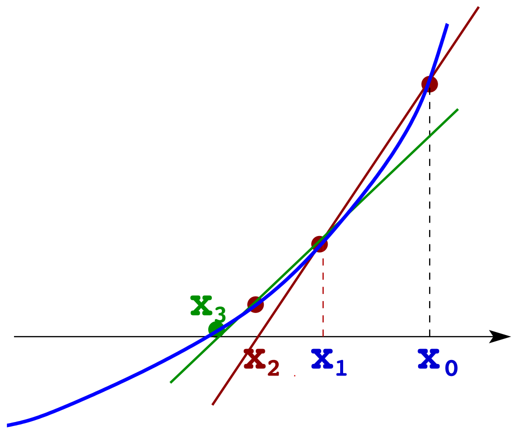

Roots: secant method

- ✔ We replace the derivative of the newton-raphson method by a numeric derivative given by:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

- ✔ The iterated values:

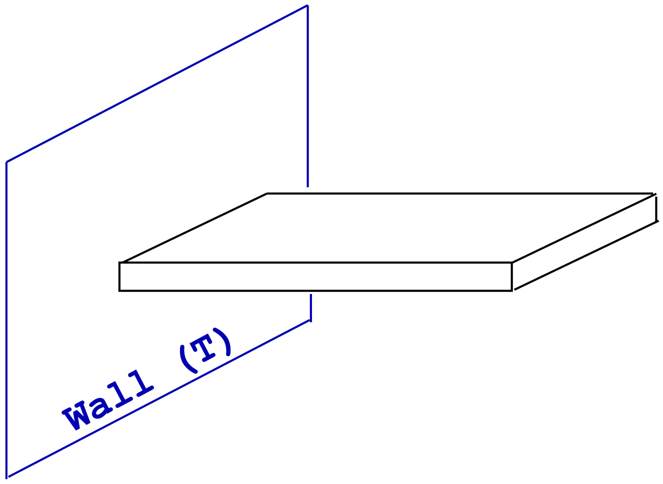
$$x_{i+1} = x_i - f(x_i) \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

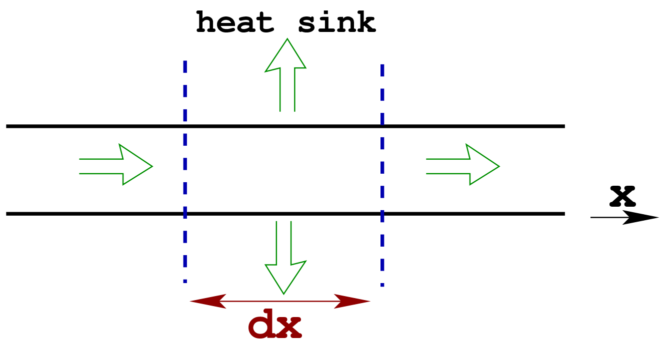


Secant method

```
double x1=XL, x2 = XU;
int iter=100;
int i = 0;
double eps = 1.e-6;
while (fabs(x2 - x1) >= eps) {
 i = i + 1;
 double fx2 = f->Eval(x2);
 double fx1 = f->Eval(x1);
 x3 = x2 - (fx2*(x2-x1))/(fx2-fx1);
 x1 = x2;
 x2 = x3;
 if(i >= iter) return -999.;
}
return x3;
```

heat transfer fin





- heat flux ($W.m^{-2}$) given by Fourier law

$$\boxed{\dot{q} = -k\frac{\partial T}{\partial x}} \qquad \text{(k, thermal conductivity)}$$

- heat flow

$$\dot{q}_{out} \equiv \dot{q}(x+dx) = \dot{q}(x) + \frac{d\dot{q}}{dx}dx$$

- convection: heat flow through pipe walls (h=convection coeff)

$$\dot{q}_{sink} = \underbrace{P\,dx}_{\text{convection area}} \quad h(T - T_\infty) \qquad \text{(P, perimeter)}$$

- heat balance

$$\dot{Q}_{in} = \dot{Q}_{out} + \dot{Q}_{sink}$$

$$\dot{q}(x)\,A = \dot{q}(x+dx)\,A + P\,dx\,h\,(T - T_\infty)$$

$$Ph(T - T_\infty)dx + \frac{d\dot{q}}{dx}Adx = 0$$

$$\boxed{\frac{\partial^2 T}{\partial x^2} - \frac{Ph}{kA}(T - T_\infty) = 0}$$

shooting method example

- the equation modeling the temperature y of a heat transfer pipe at distance x is:

$$\boxed{\frac{d^2y}{dx^2} = 2y} \qquad \begin{cases} y(x=0) = 1 \\ y'(x=1) = 0 \end{cases}$$

- analytic solution:

$$\boxed{y(x) = a_1 e^{\sqrt{2}x} + a_1 e^{-\sqrt{2}x}} \qquad \begin{array}{|l} a_1 = 0.05581 \\ a_2 = 0.94419 \end{array}$$

- shooting method

transform 2nd-order into a 1st-order system

$$\begin{cases} \frac{dy}{dx} = z \\ \frac{dz}{dx} = 2y \end{cases} \qquad \begin{array}{|l} y(0) = 1 \\ z(1) = 0 \end{array}$$

- transform to an initial value problem with a trial hypothesis to start $\mathbf{u}$

$$\begin{cases} y(0) = 1 \\ z(0) = u \end{cases}$$

- iterate equation solutions with RK4 method through all the $\mathbf{x}$ interval and retrieve the value for $\mathbf{z}$ function at the boundary

$$z_{iter}(1) \equiv g(u)$$

- solve the following equation in order to find the right initial value providing the boundary condition $\mathbf{z(1) = 0}$

$$z(1) - g(u) = 0$$

root finding algorithm will provide iterated $\mathbf{u_i}$'s for introducing on RK4

shooting method example

✔ first iteration on root finding (RK4)

$y(0) = 1, z(0) = 0$

x	y	z
0	1	0
0.2	1.04027	0.405333
0.4	1.1643	0.84331
0.6	1.3821	1.3492
0.8	1.71119	1.96373
1	2.17807	2.73641

✔ second iteration on root finding (RK4)

$y(0) = 1, z(0) = -2$

x	y	z
0	1	−2
0.2	0.634933	-1.6752
0.4	0.320993	-1.4853
0.6	0.0328983	-1.41499
0.8	-0.252549	-1.45864
1	-0.558335	−1.61974

✔ secant method: find next initial value iteration

$$\begin{cases} u_0 = 0 & g(0) = 2.73641 \\ u_1 = -2 & g(-2) = -1.61974 \end{cases}$$

$$u_2 = u_1 - g(u_1)\frac{u_1 - u_0}{g(u_1) - g(u_0)}$$

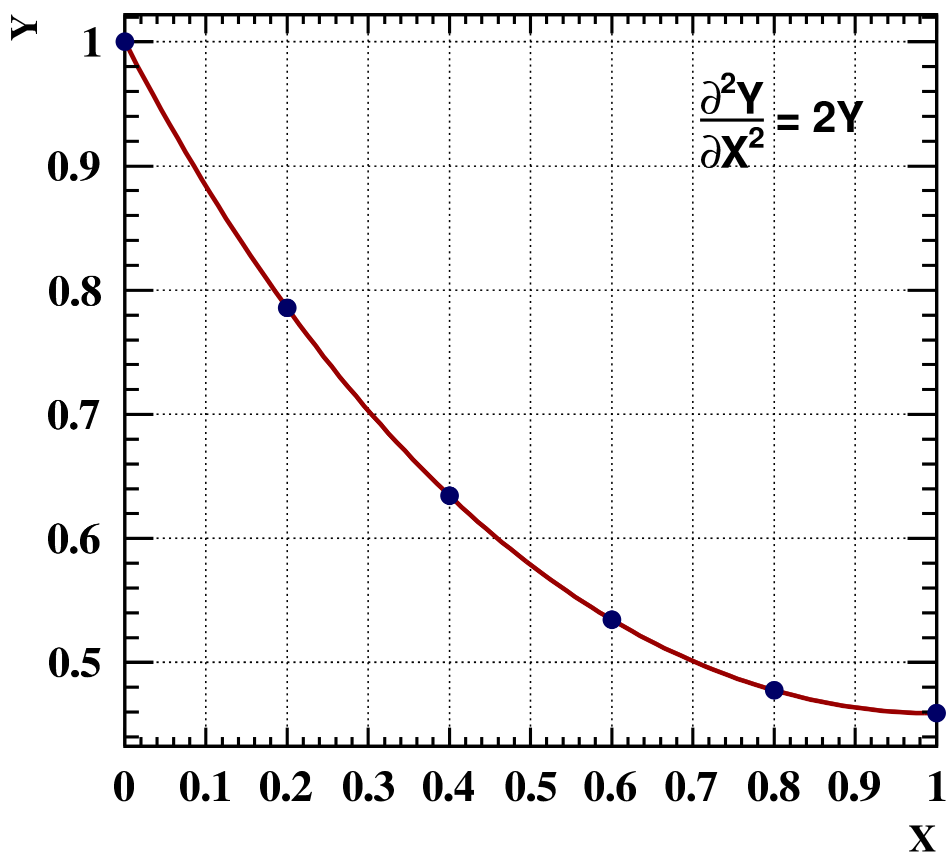
$$= -2 + 1.62\frac{-2-0}{-1.62 - -2.736} = -1.256$$

✔ third iteration on root finding (RK4)

$y(0) = 1, z(0) = -1.256$

x	y	z
0	1	−1.25634
0.2	0.785648	-0.901599
0.4	0.634559	-0.619454
0.6	0.534568	-0.38719
0.8	0.477623	-0.186103
1	0.459138	$3.05311e-16$

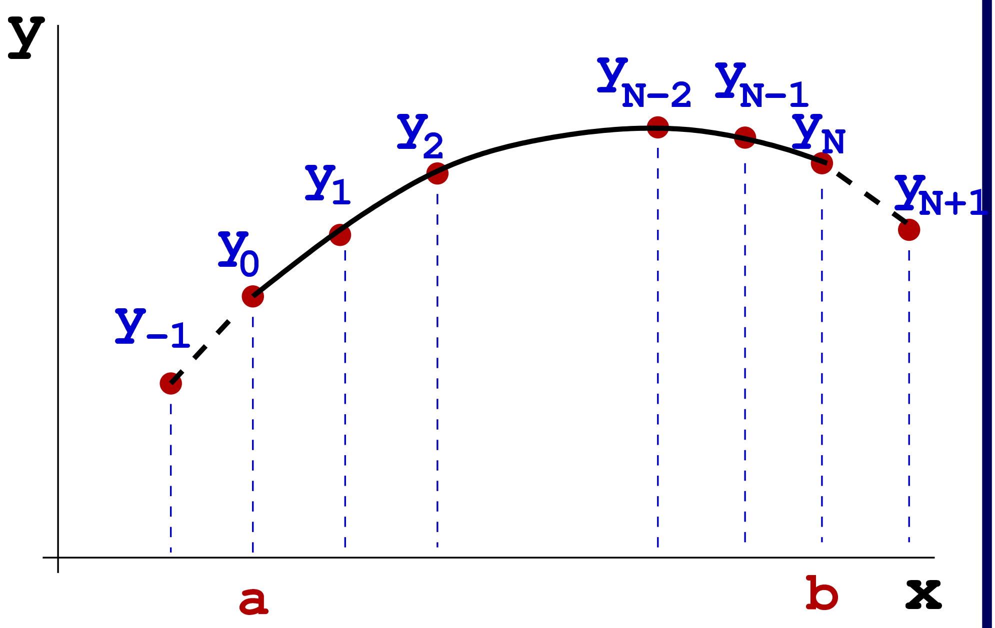
shooting method: solution



BV problems: finite-differences

In the finite-difference method, the independent variable ($\mathbf{x}$) is discretized and we transform the differential equation into a set of algebraic equations

the purpose of the equation system is to find the function value at every mesh point



BV problems: finite-differences

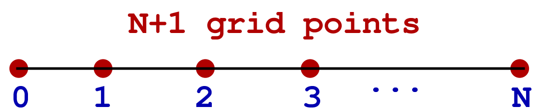
✔ discretize the interval $x \in [a, b]$ in $N+1$ grid points: $k = 0, \cdots, N$

grid spacing $h = \frac{b-a}{N}$

✔ using the central difference derivative at the grid points

$$y''(x_k) \equiv y''_k = \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}$$

$$y'(x_k) \equiv y'_k = \frac{y_{k+1} - y_{k-1}}{2h}$$



✔ the differential equation becomes for the inner grid points $k = 1, \cdots, N-1$

$$a(x)\, y''(x) + b(x)\, y'(x) + c(x)\, y(x) = f(x)$$

$$a_k \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + b_k \frac{y_{k+1} - y_{k-1}}{2h} + c_k y_k = f_k$$

sorting the y_k terms, we define a set of linear equations:

$$\boxed{\left(\frac{a_k}{h^2} - \frac{b_k}{2h}\right) y_{k-1} + \left(c_k - 2\frac{a_k}{h^2}\right) y_k + \left(\frac{a_k}{h^2} + \frac{b_k}{2h}\right) y_{k+1} = f_k} \qquad k = 1, \cdots, N-1$$

BV problems: finite-differences (cont.)

✔ the boundary problem reduces to a system of linear equations at the inner mesh points:

$$
\begin{cases}
\left(\frac{a_1}{h^2} - \frac{b_1}{2h}\right) y_0 & + & \left(c_1 - 2\frac{a_1}{h^2}\right) y_1 & + & \left(\frac{a_1}{h^2} + \frac{b_1}{2h}\right) y_2 & = & f_1 & \text{(k=1)} \\
\left(\frac{a_2}{h^2} - \frac{b_2}{2h}\right) y_1 & + & \left(c_2 - 2\frac{a_2}{h^2}\right) y_2 & + & \left(\frac{a_2}{h^2} + \frac{b_2}{2h}\right) y_3 & = & f_2 & \text{(k=2)} \\
\cdots & + & \cdots & + & \cdots & = & \cdots & \\
\left(\frac{a_{N-1}}{h^2} - \frac{b_{N-1}}{2h}\right) y_{N-2} & + & \left(c_3 - 2\frac{a_{N-1}}{h^2}\right) y_{N-1} & + & \left(\frac{a_{N-1}}{h^2} + \frac{b_{N-1}}{2h}\right) y_N & = & f_{N-1} & \text{(k=N-1)}
\end{cases}
$$

The boundary conditions:

$$
\begin{cases}
y(a) = \lambda_1 & \text{or} & y'(a) = \lambda_1 \\
y(b) = \lambda_2 & \text{or} & y'(b) = \lambda_2
\end{cases}
$$

The boundary conditions approximating the derivatives:

$$
\begin{cases}
y(a) = \lambda_1 & \text{or} & \frac{y_1 - y_{-1}}{2h} = \lambda_1 \\
y(b) = \lambda_2 & \text{or} & \frac{y_{N+1} - y_{N-1}}{2h} = \lambda_2
\end{cases}
$$

BV problems: finite-differences (cont.)

✔ Two additional equations are added at the extreme mesh points:

- either we know immediately the solutions $\mathbf{y_0}$ and $\mathbf{y_N}$ from the boundary conditions, and the two equations are:

  $y_0 = \lambda_1$

  $y_N = \lambda_2$

- or if the boundary conditions are derivatives, we introduce new equations at the extreme mesh points,

$$
\begin{array}{lccccccc}
(k=0) & \left(\frac{a_0}{h^2} - \frac{b_0}{2h}\right) y_{-1} & + & \left(c_0 - 2\frac{a_0}{h^2}\right) y_0 & + & \left(\frac{a_0}{h^2} + \frac{b_0}{2h}\right) y_1 = f_0 \\
(k=N) & \left(\frac{a_N}{h^2} - \frac{b_N}{2h}\right) y_{N-1} & + & \left(c_N - 2\frac{a_N}{h^2}\right) y_N & + & \left(\frac{a_N}{h^2} + \frac{b_N}{2h}\right) y_{N+1} = f_N
\end{array}
$$

The additional mesh points $\mathbf{y_{-1}}$ and $\mathbf{y_{N+1}}$ outside the equation range $[\mathbf{a}, \mathbf{b}]$ can be eliminated from the boundary conditions,

$$
\begin{array}{lrl}
(k=0) & \left(\frac{a_0}{h^2} - \frac{b_0}{2h}\right)(y_1 - 2h\lambda_1) + \left(c_0 - 2\frac{a_0}{h^2}\right) y_0 + \left(\frac{a_0}{h^2} + \frac{b_0}{2h}\right) y_1 & = f_0 \\
 & \left(c_0 - 2\frac{a_0}{h^2}\right) y_0 + \left(2\frac{a_0}{h^2}\right) y_1 & = f_0 + 2h\lambda_1 \left(\frac{a_0}{h^2} - \frac{b_0}{2h}\right) \\
(k=N) & \left(\frac{a_N}{h^2} - \frac{b_N}{2h}\right) y_{N-1} + \left(c_N - 2\frac{a_N}{h^2}\right) y_N + \left(\frac{a_N}{h^2} + \frac{b_N}{2h}\right)(y_{N-1} + 2h\lambda_2) & = f_N \\
 & \left(2\frac{a_N}{h^2}\right) y_{N-1} + \left(c_N - 2\frac{a_N}{h^2}\right) y_N & = f_N - 2h\lambda_2 \left(\frac{a_N}{h^2} - \frac{b_N}{2h}\right)
\end{array}
$$

finite-difference method example

- the equation modelling the temperature y of a heat transfer pipe at distance x is:

$$\boxed{\frac{d^2y}{dx^2} = 2y} \qquad \begin{cases} y(x=0) = 1 \\ y'(x=1) = 0 \end{cases}$$

- analytic solution:

$$\boxed{y(x) = a_1 e^{\sqrt{2}x} + a_1 e^{-\sqrt{2}x}} \qquad \begin{array}{|l} a_1 = 0.05581 \\ a_2 = 0.94419 \end{array}$$

- finite-difference method
  discretize $\mathbf{x}$ variable in the range where we want to solve the differential equation $[\mathbf{0}, \mathbf{1}]$

  using a step $\mathbf{h = 0.1}$, the range is divided in $\mathbf{10}$ sub-intervals and the grid points are $\mathbf{x_i = 0, \cdots, 10}$

- In the inner mesh points, $\mathbf{x_i = 1, \cdots, 9}$ the discretization of the differential equation $\mathbf{y''(x) = 2y}$ provides the following set of equations:

$$2y_i = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \qquad (i = 1, \cdots, 9)$$

$$\boxed{y_{i+1} - 2y_i(1+h^2) + y_{i-1} = 0} \qquad (i = 1, \cdots, 9)$$

till now we have $\mathbf{9}$ equations for $\mathbf{11}$ $(\mathbf{y_i = y_0, \cdots, y_{10}})$ unknowns

- an additional equation is provided by the first boundary condition

$$\boxed{y(0) = 1} \qquad (i = 0)$$

- other equation is provided by the other boundary condition

$$\boxed{y'(1) = 0} \qquad \frac{y_{11} - y_9}{2h} = 0 \quad \Rightarrow \quad y_{11} = y_9$$

and the discretized differential equation defined on the last mesh point $\mathbf{i = 10}$

$$\begin{aligned} y_{11} - 2y_{10}(1+h^2) + y_9 &= 0 \qquad (\text{as, } y_{11} = y_9) \\ y_9 - 2y_{10}(1+h^2) + y_9 &= 0 \end{aligned}$$

$$\boxed{y_9 - y_{10}(1+h^2) = 0}$$

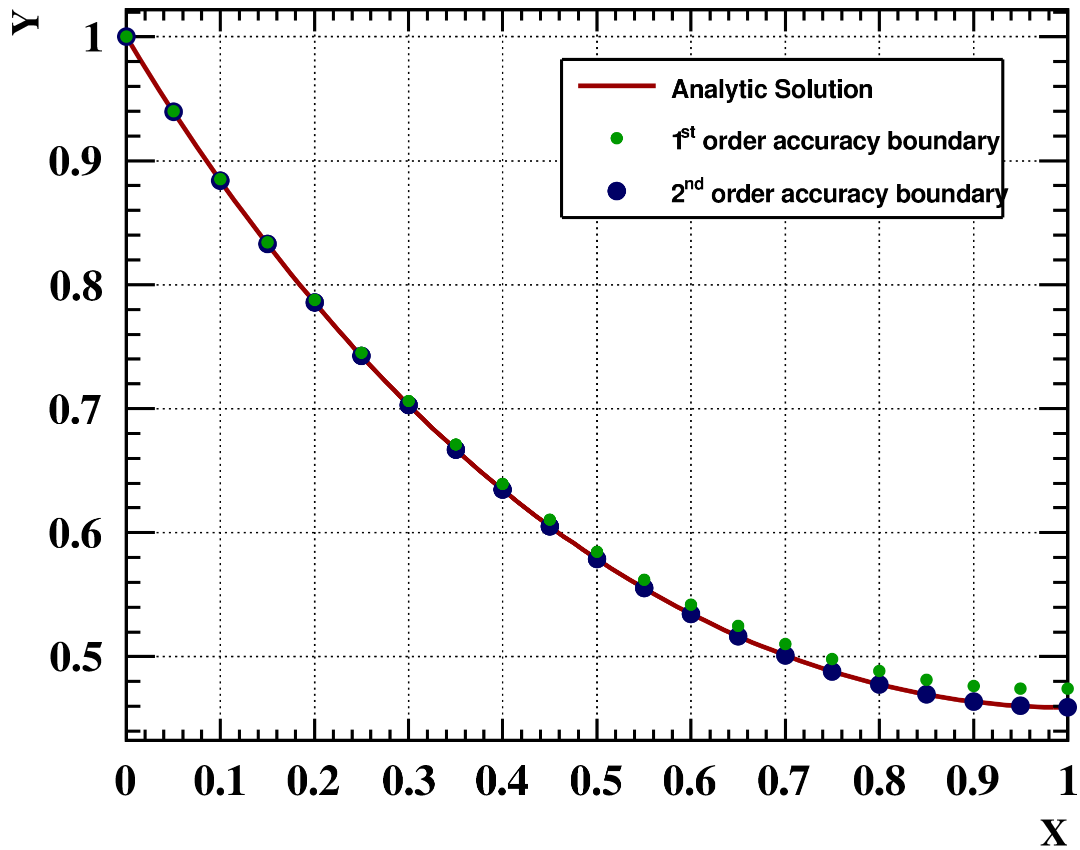
finite-difference method: example

the system of equations to solve (using for instance LU decomposition)

$$\begin{array}{lcrl} i = 1 & : & -2(1+h^2)y_1 + y_2 & = -y_0 \\ i = 2 & : & y_1 - 2(1+h^2)y_2 + y_3 & = 0 \\ i = 3 & : & y_2 - 2(1+h^2)y_3 + y_4 & = 0 \\ & & \vdots & = \vdots \\ i = 9 & : & y_8 - 2(1+h^2)y_9 + y_{10} & = 0 \\ \text{boundary condition eq} & : & y_9 - (1+h^2)y_{10} & = 0 \end{array}$$

$$\begin{pmatrix} -2(1+h^2) & +1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ +1 & -2(1+h^2) & +1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & +1 & -2(1+h^2) & +1 & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & +1 & (1+h^2) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{10} \end{pmatrix} = \begin{pmatrix} -y_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

finite-differ method: example (cont.)



Heat conduction in rod

✔ We are going to solve the stationary heat equation for a cylindrical rod of length L:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) T = 0$$

✔ it is a one-dimensional problem if the rod cylinder is perfectly isolated

$$\frac{d^2T}{dx^2} = 0 \quad x \in [0, L]$$
$$T(x=0) = T_a$$
$$T(x=L) = T_b$$

✔ Analytical solution:

$$T(x) = T_a + \frac{T_b - T_a}{L} x$$

Numerical solution

Let's use 6 grid points: $n = 0, ..., 5$

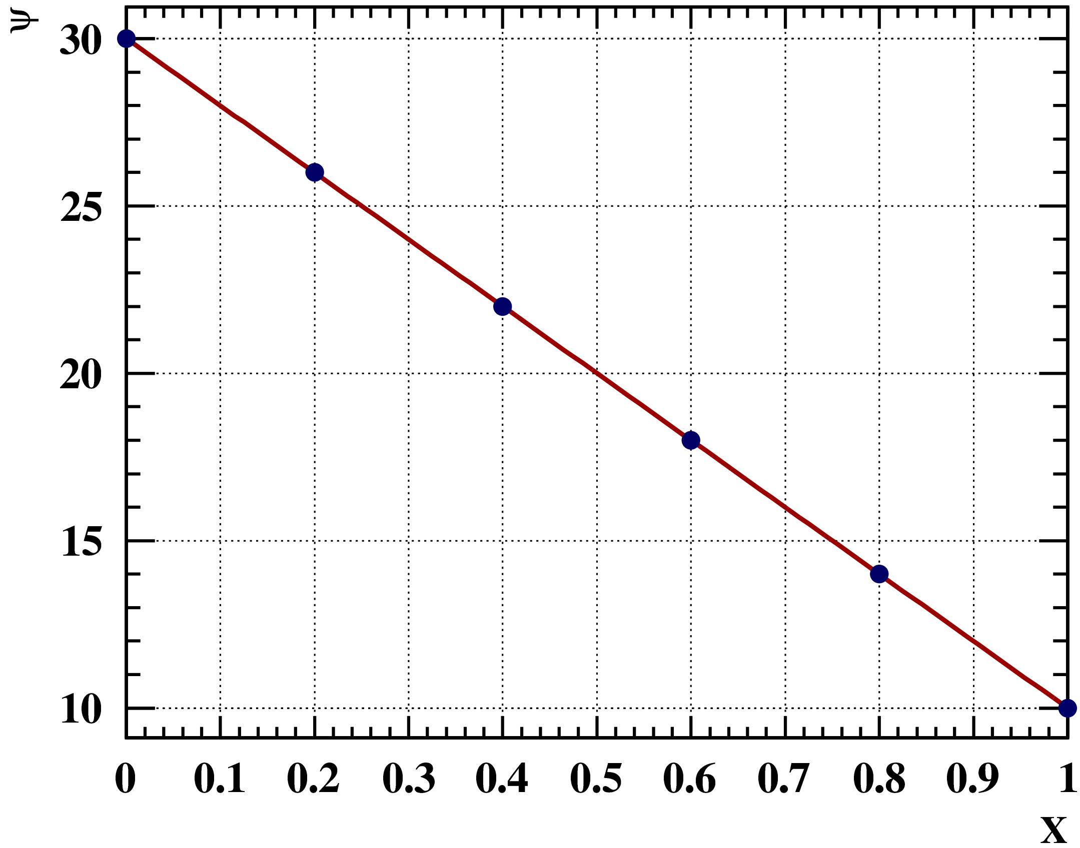
$T_{n+1} - 2T_n + T_{n-1} = 0$ (n=1,...,4)

boundary values: $T_0 = T_a$ and $T_5 = T_b$

$$\begin{aligned} T_2 - 2T_1 + T_0 &= 0 \quad \text{(n=1)} \\ T_3 - 2T_2 + T_1 &= 0 \quad \text{(n=2)} \\ T_4 - 2T_3 + T_2 &= 0 \quad \text{(n=3)} \\ T_5 - 2T_4 + T_3 &= 0 \quad \text{(n=4)} \end{aligned}$$

$$\begin{bmatrix} -2 & +1 & 0 & 0 \\ +1 & -2 & +1 & 0 \\ 0 & +1 & -2 & +1 \\ 0 & 0 & +1 & -2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} -T_a \\ 0 \\ 0 \\ -T_b \end{bmatrix}$$

Heat equation: finite-differences



other examples

✔ schrodinger equation

$$-\frac{h^2}{4\pi^2 m}\frac{d^2\psi}{dx^2} + V(x)\psi(x) = E\psi(x)$$

✔ Poisson equation

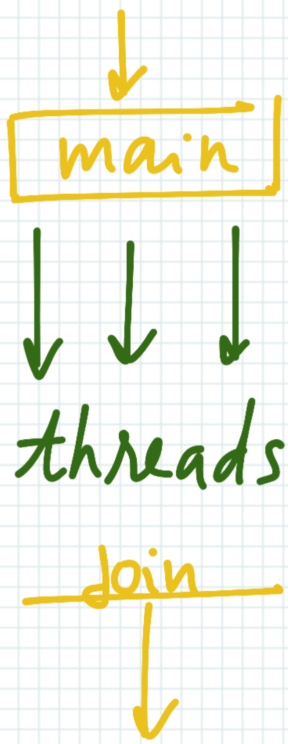
$$\frac{d^2\phi}{dx^2} = f(x)$$

Beyond this Course

Parallelism

☑ C++ 11 revision introduced threads

```
#include <thread>
```



☑ thread objects are created this way,

```
std::thread(    );
```

Lambda functions
function pointer

☑ wait for thread end `thread.join()`

Lots of C++ libraries

ROOT, STL, Boost, GSL, GTKmm, Qt, ...

Comentários sobre o curso (bem vindos...)

Obrigado pelo semestre!