8d. Hermite Polynomial

- **hermite.cpp**: find the solution of the following 2nd order ODE (Hermite's polynomial)

$$\frac{d^2u}{dx^2} - 2x\frac{du}{dx} = -2\lambda u$$

with $\lambda = 5$ on the domain $x_L \le x \le x_R$ and boundary conditions given by

$$u_L = +8 \quad (\text{for } x = x_L = -1)$$
$$u_R = -8 \quad (\text{for } x = x_R = 1).$$

- Solve the problem using two different algorithms:
  1. Shooting method, by integrating the previous ODE with the 4th order Runge-Kutta algorithm and `NSTEPS = 100` (*Hint*: $u'(x_L) \ll 0$). Use a root-finder of your choice (`xtol=1.e-8`). Report, in the comments at the beginning of the C++ code, the value of $u'(x_L)$ that you obtain.

  2. Finite difference method, with a grid of `(NSTEPS+1)` points (inclusive of boundary values). *Hint*: write the tridiagonal system resulting from a finite difference discretization of the 2nd and 1st derivatives and obtain the coefficients `a[], b[], c[] and r[]` by imposing the correct boundary conditions.

- Use the analytical solution:

$$H(x) = 32x^5 - 160x^3 + 120x$$

to compute the L1 norm errors* for the two methods.

$* \ \epsilon = \frac{1}{N}\sum |u_i - u_i^{ex}|$

8d. Hermite (cont)

- Upload your code with your last name and the output inserted in the comment at the beginning of the file and the *necessary* library functions at the end:

```
// Last name: ...
// u'(xL)          = ...
// L1 err(Shooting)     = ...
// L1 err(Finite Diff) = ...
#include ...
...
int main()
{
  // code here
}

void RK4Step (...){
...
}

void Residual (...){
...
}

void RHS (...){
...
}
```

- Also, upload a png (or jpeg or pdf) plot showing the solution u(x) obtained with one of the two methods.