ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations Exercise 11

HS 15
Prof. M. Troyer

Problem 11.0 Convergence of the simpson integration

We are going to examine the convergence of the simpson integration with the number of bins N. In order to do that we use the function pointer version of the simpson integration. We provide the sources (`simpson.cpp`, `simpson.hpp`, `main.cpp`) for this task along with the `CMakeLists.txt` specifying the building instructions, and a `gnuplot`/`Python` script for generation of a `png`/`pdf` plot of the convergence.
Your task is to write a `Makefile` which sets the rules to

1. create the executable `simpson` by compilation of the sources; use the same compiler options as in the provided `CMakeLists.txt` and make sure that the compilation happens only when it is necessary by compiling each `cpp`-file separately and linking them at the end, optionally you may create a library `libintegrate.a`;

2. create the data file `convergence.dat` by running the executable and piping the output to the `dat`-file,

   ```
   ./simpson > convergence.dat
   ```

3. create the figure `convergence.png` (or `convergence.pdf`) by processing the `convergence.dat` in `gnuplot` (by `Python`) according to the script `plot.gp` (`plot.py`),

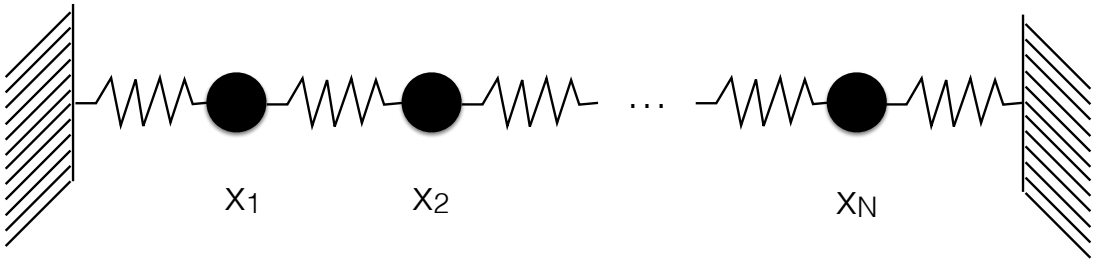
   ```
   gnuplot plot.gp
   ```
   (alternatively: `python plot.py`)

4. clean the object files `*.o`, executable `simpson`, the data file `convergence.dat`, and the figure `convergence.png` (`convergence.pdf`).

Note for Windows users: you may (remotely) use the D-PHYS machines (e.g. `login.phys.ethz.ch`) to fulfill this task.[1]

Problem 11.1 Linear algebra with libraries

In this exercise we will consider a harmonic chain with N identical masses m connected via springs with spring constants K.



The equations of motions are given by

[1] For the ssh connection you may use PuTTY (`http://www.putty.org/`).

$$m\ddot{x}_n = -K\left[(x_n - x_{n-1}) + (x_n - x_{n+1})\right], \tag{1}$$

where x_n denotes the displacement of n-th mass from its equilibrium position. Eq.1 is a set of n coupled linear differential equations with constant coefficients, therefore the solution can be written as a linear combination of eigenmodes. The eigenmodes can be obtained by plugging the Ansatz of harmonic time dependence $x_n(t) = x_n \exp(i\omega t)$ into Eq.1,

$$\omega^2 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix} = \frac{K}{m} \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix}. \tag{2}$$

Solve this eigenvalue problem and get the frequencies ω and eigenvectors of the system! As a check you may compare with the exact eigenfrequencies

$$\omega_n = \sqrt{\frac{K}{m}\left[2 - 2\cos\left(\frac{\pi n}{N+1}\right)\right]} \quad \text{for} \quad n \in \{1, \ldots, N\}. \tag{3}$$

Numerical solution: Store the matrix representation M_{ij} in a matrix and diagonalize it with LAPACK routine DSYEV (see the documentation in `http://www.netlib.org/lapack/double/dsyev.f`). Choose an appropiate system size N for the matrix.

To use the LAPACK routine, you have to link your program with some additional libraries (`-llapack -lblas` for Linux or `-framework accelerate` for Mac). As the LAPACK routines are FORTRAN routines, you might have to link your program also with some additional FORTRAN runtime libraries (flag `-lgfortran` may help).