ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations Exercise 9

HS 15
Prof. M. Troyer

Problem 9.1 Simpson integration with virtual functions

Implement a new version of your Simpson integration routine using virtual functions.

- Define an abstract base class with a pure virtual `operator()` for function objects in double precision.
- Write a free simpson integrator function taking the integrand as a reference to the abstract base class.
- Inherit a concrete class from the abstract base class and implement the `operator()`.

Problem 9.2 Benchmarks of Simpson integrations

Benchmark the four different implementations of the Simpson integration you have already made:

- hard-coded function
- function pointer
- templated function object
- virtual function

We suggest using the optimization flags `-O3 -DNDEBUG -funroll-loops`.[1]
Repeat the benchmarks for functions of different computational complexity: $f_1(x) = 0$, $f_2(x) = 1$, $f_3(x) = x$, $f_4(x) = x^2$, $f_5(x) = \sin(x)$, $f_6(x) = \sin(5x)$.
Discuss the results.

Problem 9.3 Penna Model with Fishing

S. Moss de Oliveira et al., Physica A 215, 298, 1995

Implement a Penna simulation with fishing. The goal is to observe how a slight increase in fishing may destroy an initially stable population. The following is how you should modify your original Penna simulation:

1. At a time-step M1, when the fish population is stable, introduce the concept of fishing. Here, any fish can die due to fishing (in addition to illness) with probability p_1. At a later time-step $M_2 > M_1$ increase the fishing probability to $p_2 > p_1$.
2. Observe that the increase in fishing at time-step M_2 will destabilize the fish population; use simulational parameters from the paper: $M = 1$, $T = 3$, genome length $= 32$, $R = 7$, pregnancy probability $= 1$, $p_1 = 0.17$, $p_2 = 0.22$.
3. What will happen if fishing is only allowed for the adult species?

[1] When using `CMake`, perform the benchmarks using the `RELEASE` build type. The build type is controlled via the variable `CMAKE_BUILD_TYPE`. By default, the `RELEASE` build type adds `-O3 -DNDEBUG` to the compiler flags.