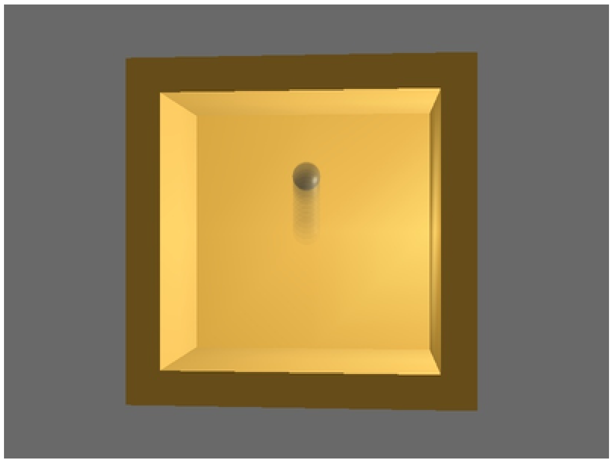
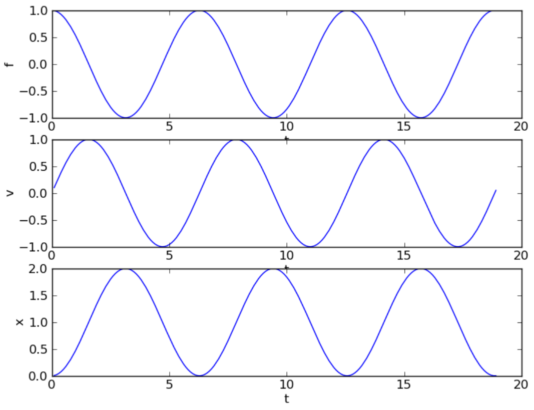
THE FUNDAMENTAL THEOREM OF CALCULUS

KATARINA GUSTAVSSON JOHAN JANSSON





1. To read

- Newton's Laws of Motion (ch. 17)
- Particle-Spring System (ch. 18)
- The Fundamental Theorem of Calculus (ch. 61)
- Proof of the Fundamental Theorem (ch. 73)

2. Goals

2.1. **Understanding.**

- Structuring your program. Strict interfaces with documentation.
- The concept of "time"-stepping.
- Connection between integral and derivative (the fundamental theorem of calculus)
- Using time-stepping to compute primitive functions (accumulating sum), compare to symbolic expressions.
- Error estimation and order of convergence by halving time-step.
- Different time-step methods (Euler, Midpoint/Trapezoid)

2.2. **Skills.**

- Symbolically show the error estimate for halving the time-step, expand to sum for all time-steps. Verify experimentally with your own time-stepper.

Simulation Technology Module.

- Implement Python functions for time-stepping, primitive functions and error estimation according to the given interface specification below.

3. Software Interfaces

Follow the interface specifications in the template Python file in the "python" subdirectory. Copy the templates into a new file named "fundamental.py" and continue adding your own code. When you're finished or want to check how well you're doing, submit your "fundamental.py" (must be named exactly that) to the Web-CAT system at http://webcat.csc.kth.se.

4. Questions

4.1. **Game/interactive simulation.** Enter the "game" directory and run: `python particle_simulator.py`. Note: pressing the middle mouse button and moving the mouse rotates the scene. Play around with different parameters and right-hand sides. One of the right-hand sides (`f_simple()`) models an oscillating force on a particle. Try both Euler and Midpoint/Trapezoid methods with different time steps (0.1, 0.05, 0.01), which method do you think is better? Why? What does the other right-hand side model? If you have time, construct your own right-hand side, how do you simulate several particles?

4.2. **Timestepping (`timestep`).**

(1) Implement `timestep` following the given interface in the template (both Euler and Midpoint/Trapezoid), using what you did in the game. Verify against reference test.

4.3. **Primtive functions (`primitive`).**

(1) Derive symbolic primitive functions of some example polynomials and rational functions ($1/x$, x^2)
(2) Implement `primitive` using `timestep` by storing the u values in an array.
(3) Compute primitive functions of polynomials (possibly rational), compare against symbolic derivations.

4.4. **Error estimation (`error_est`).**

(1) Derive an estimate symbolically of the difference of computing with time-step k and time-step k/2.
(2) Implement `error_est` and compare against reference test and symbolic estimate. Additionally, choose your own function and data and be able to explain how the estimate works.

4.5. **Estimate order of convergence (`convergence_est`).**

(1) The function `convergence_est` estimates the convergence rate by halving the time-step twice and making a plot of computed values. Verify that your implementation of

the forward Euler and Midpoint/Trapezoid methods converge with the correct order. Examine the function `convergence_est` and make sure you understand and can explain what it does and how to interpret the output.