AST 381 Homework 1: Basics

Nina Filippova

January 24, 2022

1. Floating away...

(a) *Create a floating point variable and initialize it to 0.1. Print it out to full precision. What is the degree of floating point error you find? How does this floating point error change if you the declare the variable as "single precision" vs. "double precision"?*

To full precision (using Python 3.7 on Ubuntu 18.04; Intel Core i7-6700HQ CPU),

$$x = 0.1000000000000000005551115123125782702118158340454101562 5 \quad \text{(double precision)},$$
$$x = 0.100000001490116119384765625 \quad \text{(single precision)},$$

so the floating point error is $\sim 5.6 \times 10^{-18}$ for double precision and $\sim 1.5 \times 10^{-9}$ for single precision.

(b) *Using the method outlined in class, determine the roundoff error, ϵ, for your machine.*

The roundoff error is $\sim 2.22045 \times 10^{-16}$ for double precision and $\sim 1.19209 \times 10^{-7}$ for single precision.

(c) *Compare your answer with classmates who have different hardware and/or operating systems. What do you find?*

2. Integral processes *Two methods for numerical integration are the Rectangle Rule and the Trapezoid rule:*

$$I = \int_a^b f(x)\,dx = \sum_{i=1}^{N} f(x_i)\,\Delta x, \quad \text{(Rectangle Rule)},$$
$$= \Delta x \sum_{i=1}^{N} \frac{f(x_i) + f(x_{i+1})}{2}, \quad \text{(Trapezoid Rule)},$$

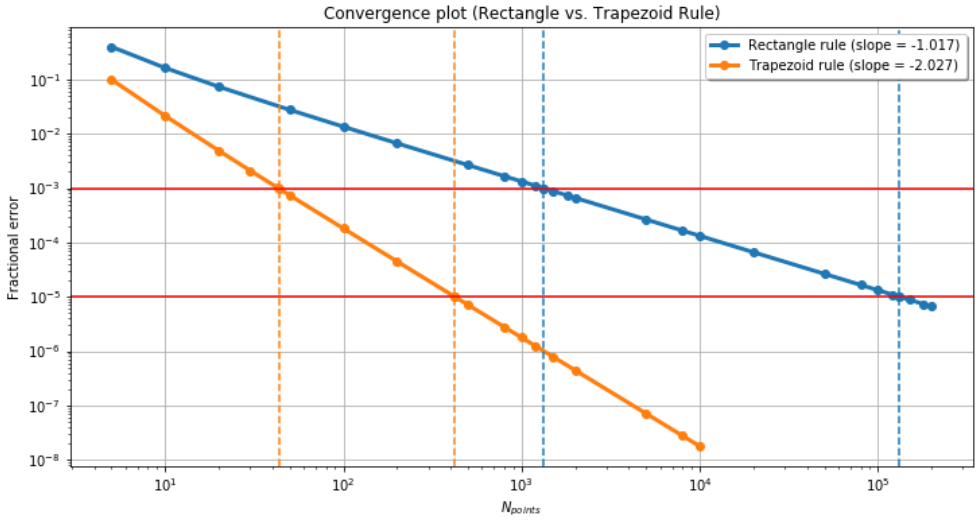
where N is the number of (equal-sized) integration steps and $\Delta x = (b-a)/N$, $x_i = a + (i-1)\Delta x$.

(a) *Numerically compute the integral*

$$\int_1^5 \frac{1}{x^{3/2}} dx$$

with both methods above, and plot the error in the numerical integral against the step size Δx for both methods. Approximately how many steps are required to get an answer with fractional error $|I - I_{\rm exact}|/I_{\rm exact} < 10^{-3}$ for both methods? What about 10^{-5}? What did you learn about the trade-off between method, accuracy, and calculation speed?

The rectangle rule is a first-order method; the trapezoid rule is a second-order method (as demonstrated by the slopes of each method in the convergence plot). Using the rectangle rule, 1326 and 132364 steps are required to achieve sub-10^{-3} and sub-10^{-5} accuracy, respectively. Using the trapezoid rule, 44 and 423 steps are required to achieve sub-10^{-3} and sub-10^{-5} accuracy, respectively.

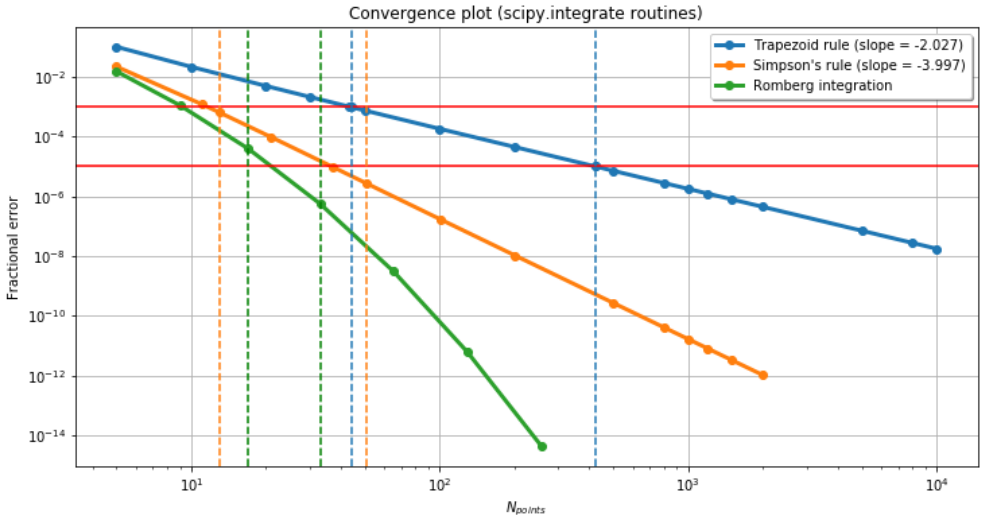


(b) *Compare the results of your two integration routines to a built-in function in your programming language of choice. What is the order of accuracy of the black-box method? What is the default approach to integration? What step sizes do you need above to obtain a similar result?*

SciPy has three built-in routines for integrating functions given fixed samples:

- `scipy.integrate.trapz` for the trapezoid rule (approximates the function as a straight line between adjacent points);
- `scipy.integrate.simps` for Simpson's rule (which approximates the function between three adjacent points as a parabola); and
- `scipy.integrate.romb` for Romberg integration.

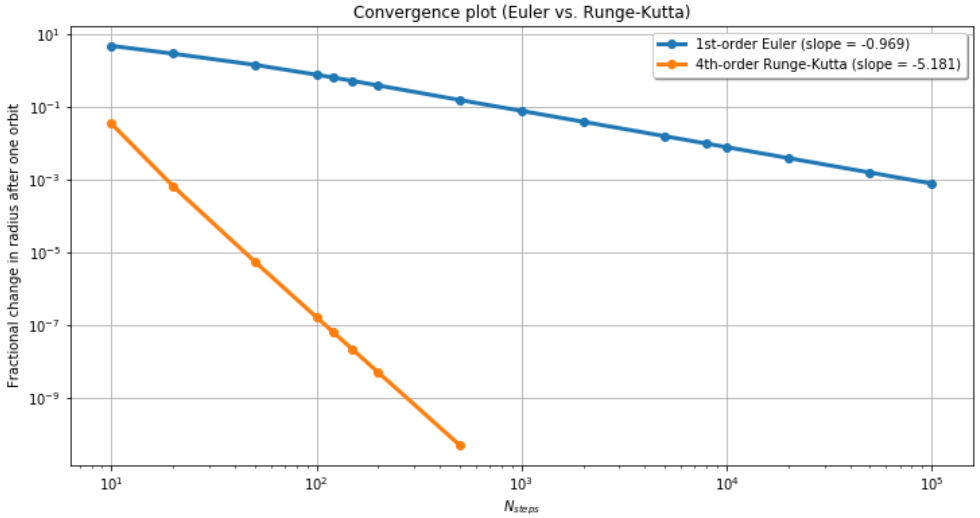
The trapezoid rule is second-order and, as before, requires 44 and 423 steps in order to achieve sub-10^{-3} and sub-10^{-5} accuracy, respectively. Simpson's rule is fourth-order and requires 13 and 37 steps in order to achieve sub-10^{-3} and sub-10^{-5} accuracy. Romberg integration does not appear to be linear in order. Romberg integration requires the number of samples to be $2^k + 1$ for some integer k, and, according to the documentation, uses the trapezoid rule at step-sizes related by a power of two and then performs Richardson extrapolation on these estimates to approximate the integral with a higher degree of accuracy. Romberg integration achieves sub-10^{-3} accuracy with 17 steps and sub-10^{-5} accuracy with 33 steps.



Shooting for the stars

(a) *Write a program to compute the orbit of the Earth around the Sun (follow Z ex. 1.8).*

Convergence plots for first-order Euler and fourth-order Runge-Kutta methods for calculating the orbit (using fractional change in radius after one orbit as a measure of error):

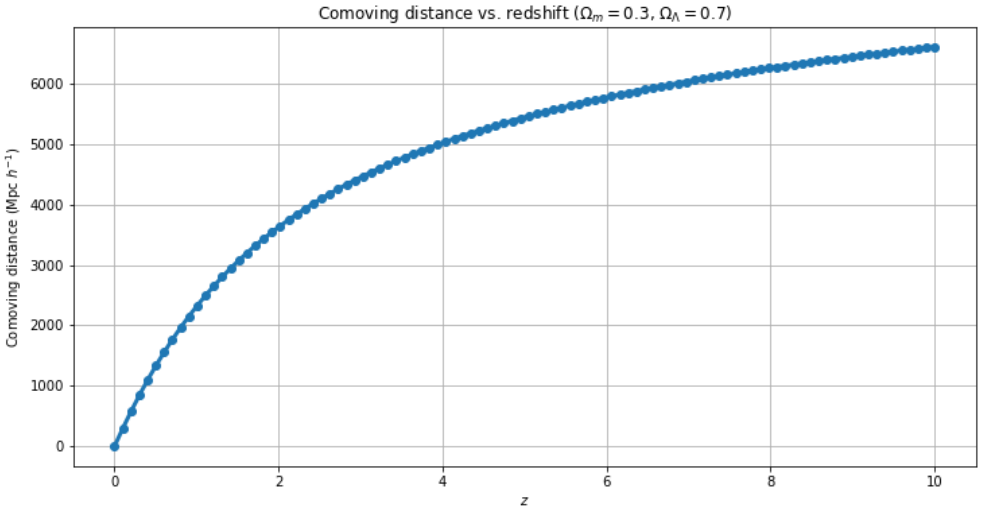


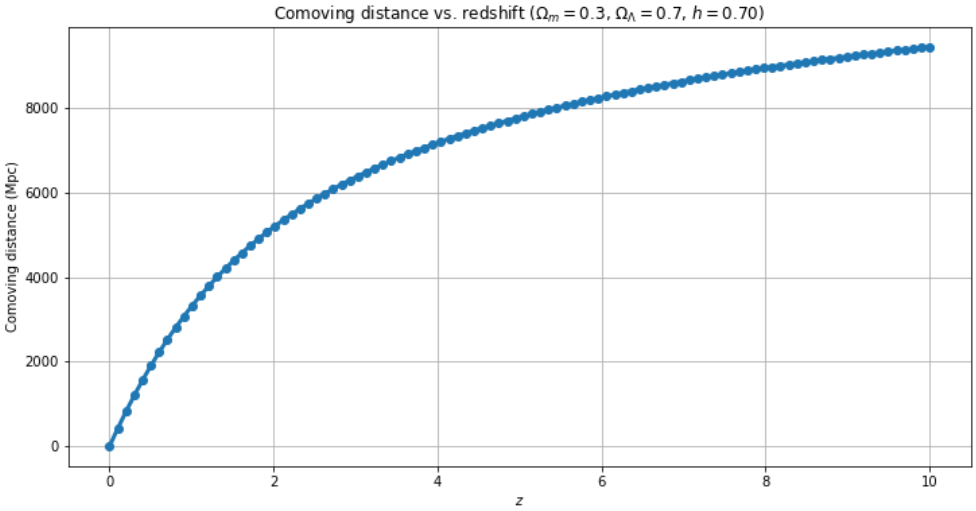
(b) *Write a program that computes the integral*

$$D_c = \int_0^z dz' \, [\Omega_m(1+z')^3 + (1-\Omega_m-\Omega_\Lambda)(1+z')^2 + \Omega_\Lambda]^{-1/2}$$

given input values of Ω_m, Ω_Λ, *and* z. *This integral, multiplied by* $cH_0^{-1} = 3000h^{-1}$ *Mpc, gives the "comoving distance" to an object at redshift* z *in a universe with matter density parameter* Ω_m *and cosmological constant* Ω_Λ. *What is the comoving distance to* $z = 2$ *in a universe with* $\Omega_m = 0.3$ *and* $\Omega_\Lambda = 0.7$*? Make a plot of the comoving distance in Mpc versus redshift for* $z \in [0, 10]$.

The comoving distance to redshift $z = 2$ is 3628.41 Mpc h^{-1} (= 5183.45 Mpc using $h = 0.70$).




Comoving distance vs. redshift ($\Omega_m = 0.3$, $\Omega_\Lambda = 0.7$, $h = 0.70$)
Comoving distance (Mpc)
0
2000
4000
6000
8000
0
2
4
6
8
10
z