Numerical Astrophysics

Assignment 1: Introduction

Christian Wiskott, 01505394.

October 26, 2020

Task 1

To compute an approximation of the definite integral from 0 to π of f(x) (see eq. 1) *Simpson's 1/3-rule* was employed. This method divides the space between the limits into n evenly spaced subdivisions and approximates f(x) via a quadratic polynomial within each subdivision. In it's general form Simpson's rule looks like the RHS of equation 2 with n being an even number and $h = \frac{b-a}{n}$.

$$f(x) = 2sin(x) + 1 \tag{1}$$

$$\int_a^b f(x)\ dx \approx \int_a^b P(x)\ dx = \frac{h}{3}\Big(f_0 + f_n + 4\sum_{i=1}^{n-1/2} f_{2i} + 2\sum_{i=1}^{n/2} f_{2i-1}\Big) \tag{2}$$

Fig. 1 shows the logarithmic plot of number of bins vs. absolute difference between the analytic solution of the integral (given by π+4) and the computed value for each number of bins. This shows the rapid convergence of this method to the true value. After 100 iterations the difference lies in the 10^{-8} range and reduces further as the number of bins increases. For the y-axis the eq. 3 has been computed, with F(x) being the true solution and F'(x) being the result of Simpson's method. The logarithmic scale was chosen to show the exponential decay more clearly.

As the difference gets increasingly smaller, numerical instabilities get more apparent. Around a bin-size of 2000 the result seems to start fluctuating and becomes more unstable. This is probably due to *cancellation* as the trailing digits of the two increasingly equal numbers become more relevant and add a certain "randomness" to the result.

$$y(x) = \log(|F(x) - F'(x)|) \tag{3}$$

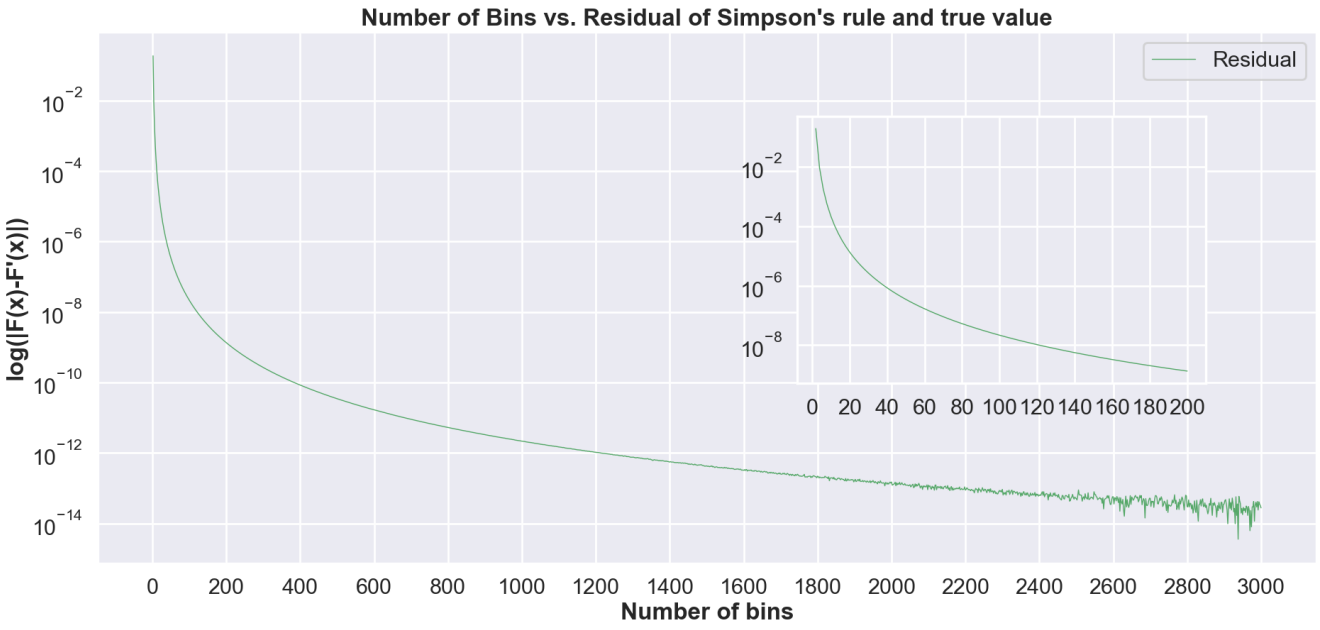

Figure 1: Plots the absolute difference between the analytic solution of the integral against the approximated value via Simpson's rule. Visible numerical instabilities starting at 2000 bins.

Task 2

Newton's method aims to find the root of a function like eq. 4 via the given iterative scheme (eq. 5) which uses the derivative of the function.

$$\begin{aligned} g(x) &= x^3 - x + 1 = 0 \\ h(x) &= \cos(x) - 2x = 0 \end{aligned} \tag{4}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5}$$

Fig. 2 shows the result of the computation. The plot of the computed root and the visual x-intercept seem to be in accordance with each other. From the convergence plot is can be seen, that in the case of the first function (green) the value seems to fluctuate quite significantly within the first 15 iterations but converges very quickly during the last few steps before clearing the convergence criterion with a value of -2.75×10^{-12} after 21 iterations.

function	root
g(x)	-1.3247
h(x)	0.4502

Table 1: Computed root of both functions. Rounded to four decimal places.

The convergence behaviour of the second function seems to be far more stable. It reaches a value of -6.67×10^{-16} after just 4 iterations.

Task 3

$$v \exp\left(-\frac{v^2}{2{c_s}^2}\right) = c_s \left(\frac{r_c}{r}\right)^2 \exp\left(-\frac{2r_c}{r} + \frac{3}{2}\right) \tag{6}$$

The objective of this task was to calculate the solution of the parker-wind-equation (see eq. 6) for different temperatures over a radii interval of 2 $R_\odot$ to 1 AU. For this purpose the given function was transformed into f(v)=0 and numerically calculated via bisection. For each chosen temperature the bisection solved f(v)=0 along all 100 radii bins. The results are plotted in fig. 3. Here we can see that higher temperatures lead to higher wind speeds, which seem to grow according to a logarithmic function. The choice of the boundary points a and b, came from the knowledge that if $r < r_c$, the velocity is subsonic and thus below the speed of sound c_s and vice versa for $r > r_c$.

The details of the computation can be found in the documentation of the .py-file.

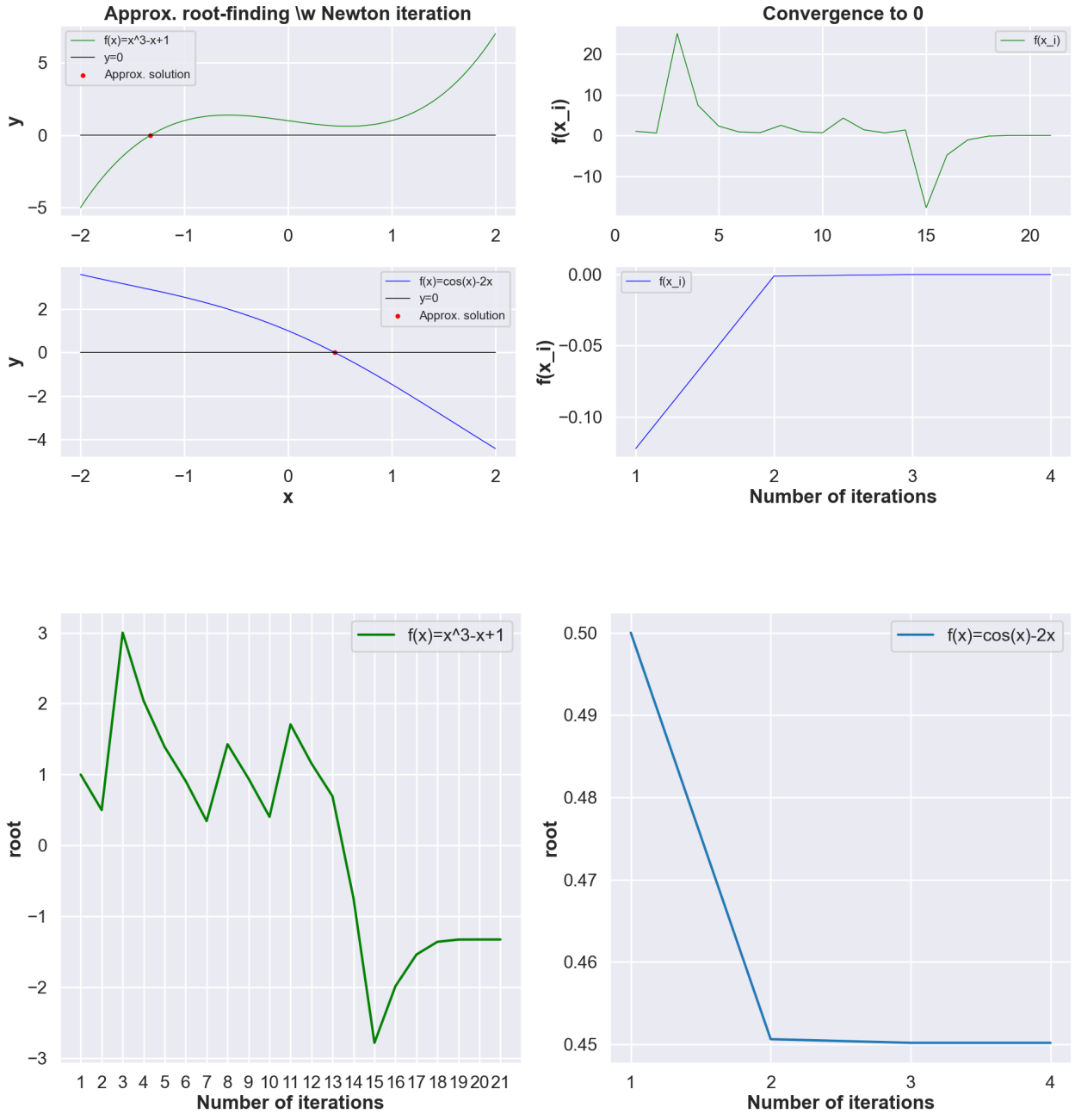

Figure 2: Above: Results of the root-finding scheme. The right column shows the visual and the computed x-intercept of the given functions while the left column shows the convergence of the method to 0, as $f(x_i)$ is evaluated after each iteration. Below: The computed root vs. the iteration steps.

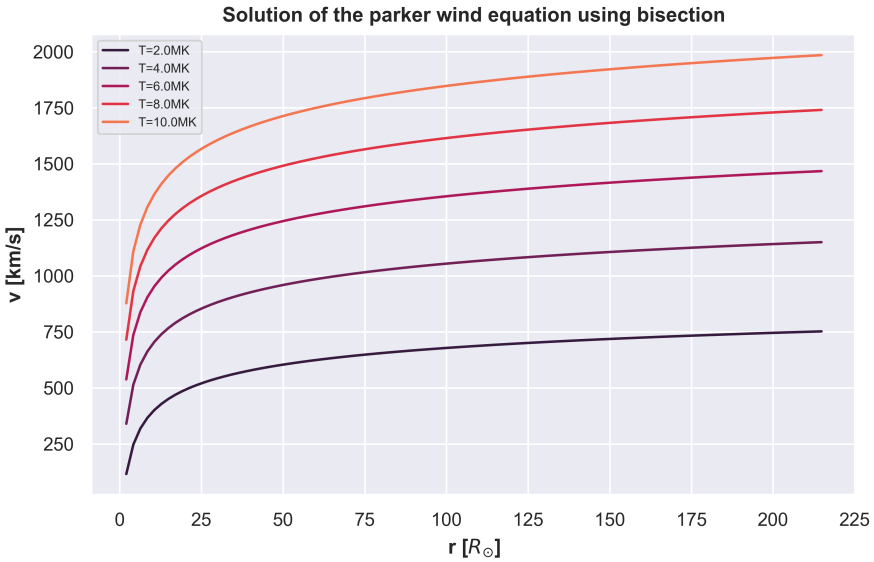

Figure 3: Solar wind speed using bisection.