I worked with my John Pharo on this worksheet. It took about four hours.

Solution of a Simple Boundary-Value Problem

My code for this problem is in `ws10.py`. I ran the BVP solver for various resolutions and with FE and RK4 integration; the results are in Figures 1 - 3.

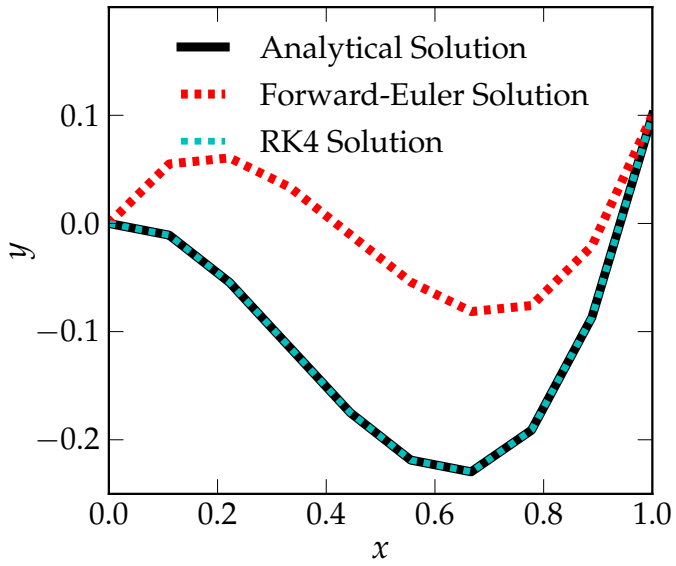

Figure 1: BVP solutions for 10 grid points. The actual solution is $y(x) = 2x^3 - 2x^2 + 0.1x$.

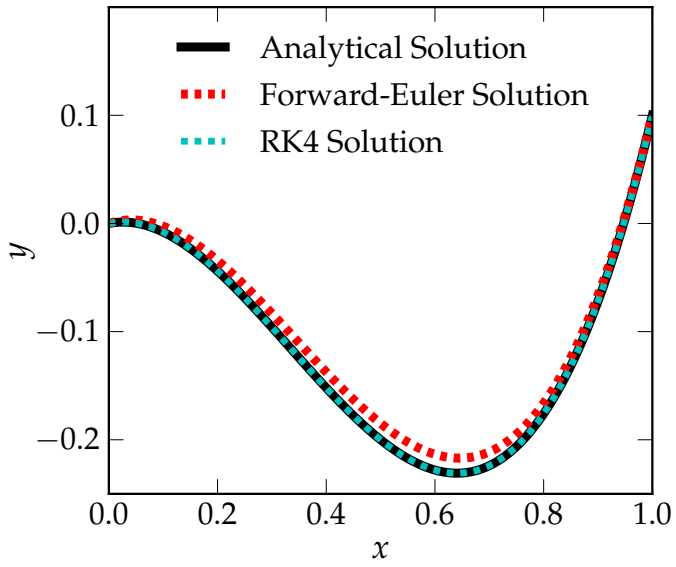

Figure 2: BVP solutions for 100 grid points. The actual solution is $y(x) = 2x^3 - 2x^2 + 0.1x$

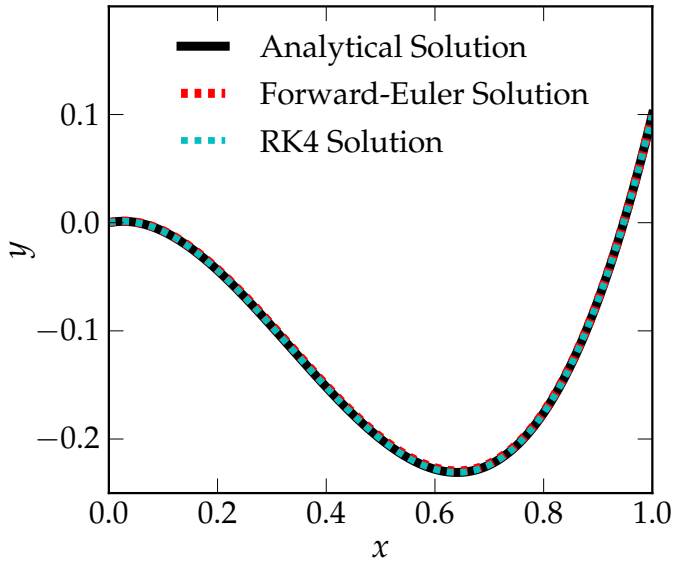

Figure 3: BVP solutions for 1000 grid points. The actual solution is $y(x) = 2x^3 - 2x^2 + 0.1x$

I'm not really sure how to check convergence for this problem. Note that the algorithm as it is written will iterate until the error falls below some threshold, so you can't just compare errors between two different resolutions without fixing the number of iterations and removing the error threshold criterion. So I did that, and ran each solver for three iterations, but the error $(y_{\text{numer}}(1) - B)$ does not go down as the resolution is increased. Then I thought to look at the error between the analytical and numerical solutions somewhere other than the endpoint–for example, the middle of the interval, $x = 0.5$. My results follow.

Table 1: Convergence of Various Integration Methods

Number of Points	$y_{\text{FE}}(0.5) - y(0.5)$	$y_{\text{RK4}}(0.5) - y(0.5)$
4	0.444	1.11×10^{-16}
8	0.209912536443	1.39×10^{-16}

One can see that FE doubles in accuracy when the resolution is doubled, as expected of a first-order approximation. However, RK4 only increases in accuracy by a factor of 25% when the resolution is doubled, whereas we expect 16-fold increase in accuracy because it is a fourth-order approximation. I think this is weird and can't really explain it.