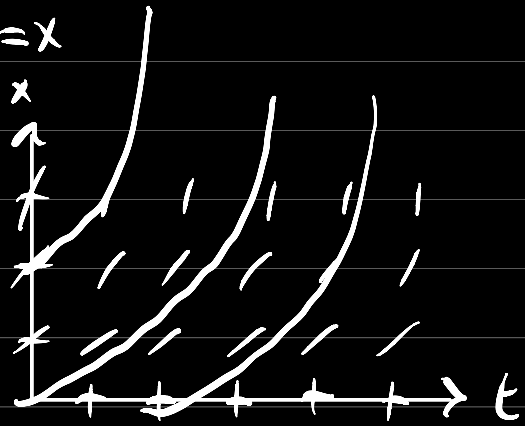
2.8.1

The slope field is constant along horizontal lines because $\dot{x} = f(x)$ does not explicitly depend on t.

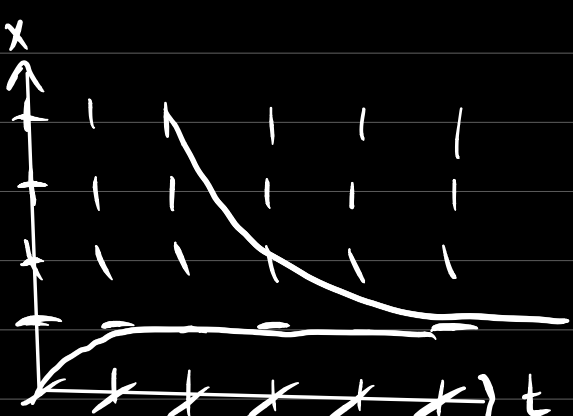
2.8.2

We sketch the slope field and several sample solutions for the following systems:

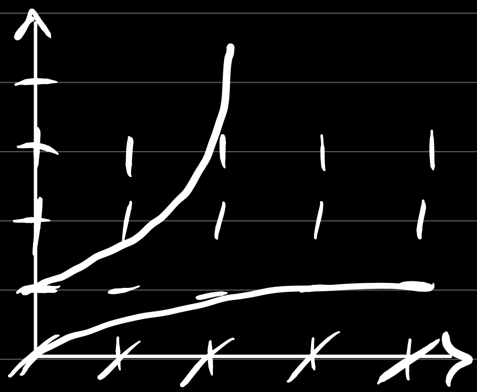
a.) $\dot{x} = x$



b.) $\dot{x} = 1 - x^2$



c.) $\dot{x} = 1-4x(1-x)$



$\vdots$

2.8.3

In this problem, we learn the Euler Method.

Let $\dot{x} = -x$, $x(0)=1$.

a.) We solve the problem analytically and find $x(1)$.

$$\frac{dx}{dt} = -x \Rightarrow \int \frac{dx}{x} = -\int dt \Rightarrow \ln x = -t + C_1$$

$$\Rightarrow x(t) = Ce^{-t}.\ x(0) = 1 = C \Rightarrow C = 1 \Rightarrow$$

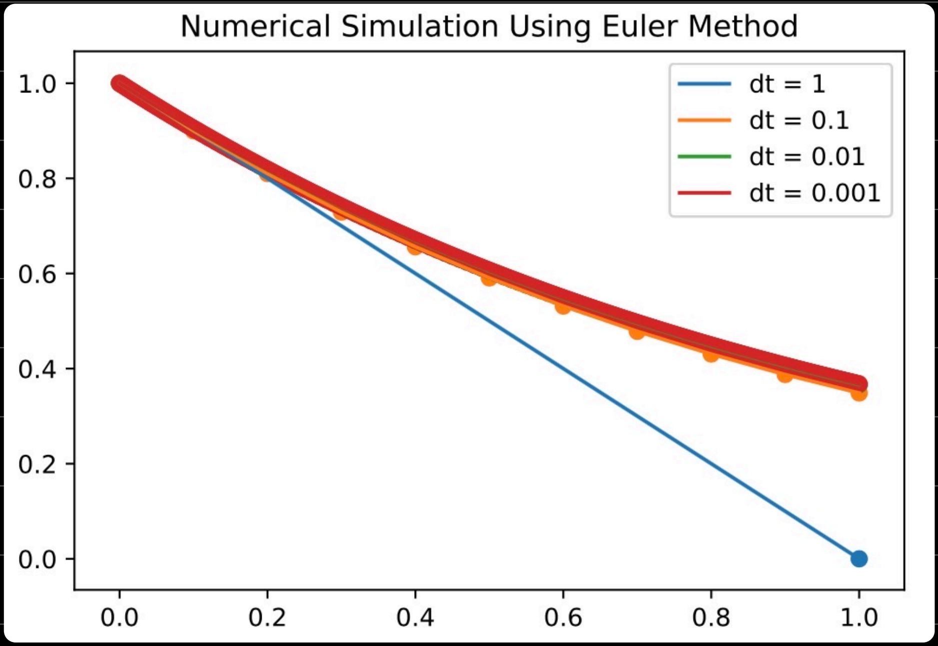
$$\boxed{x(t) = e^{-t}} \Rightarrow \boxed{x(1) = \frac{1}{e}}$$

b.) We now use the Euler method to numerically integrate this system.

We program our simulation in Python.

For $\Delta t = 1$, $\hat{x}(1) = 0$
$\Delta t = 0.1$, $\hat{x}(1) = 0.348$
$\Delta t = 0.01$, $\hat{x}(1) = 0.366$
$\Delta t = 0.001$, $\hat{x}(1) = 0.367$

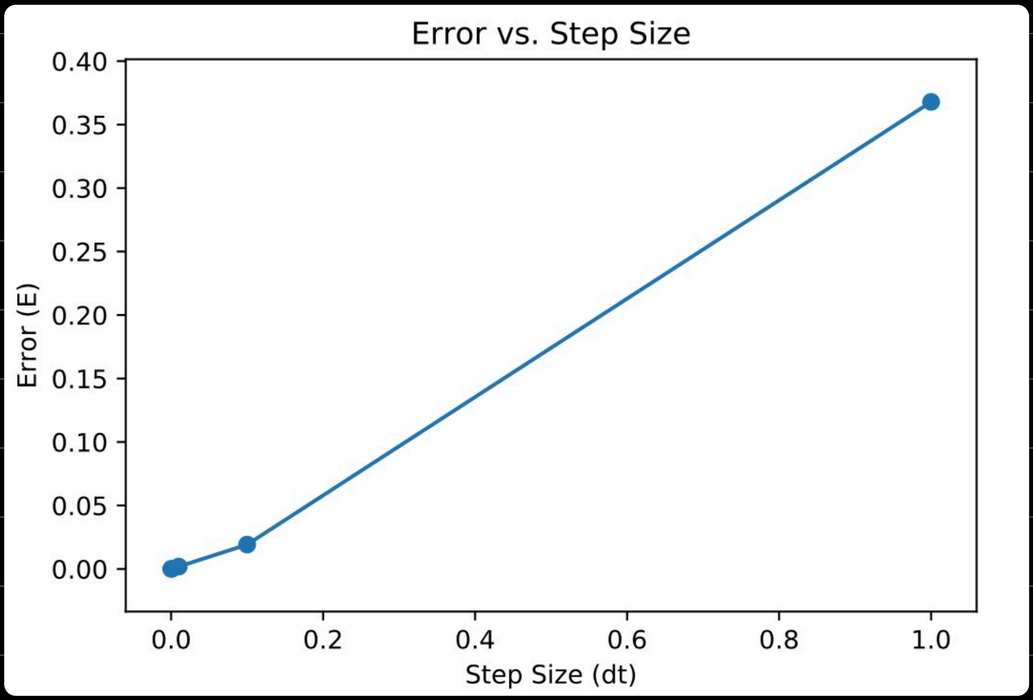
These results are quite good by $\Delta t = 0.01$, given that $1/e \approx 0.367$.



c.) Next, we plot E v. Δt. We recall that

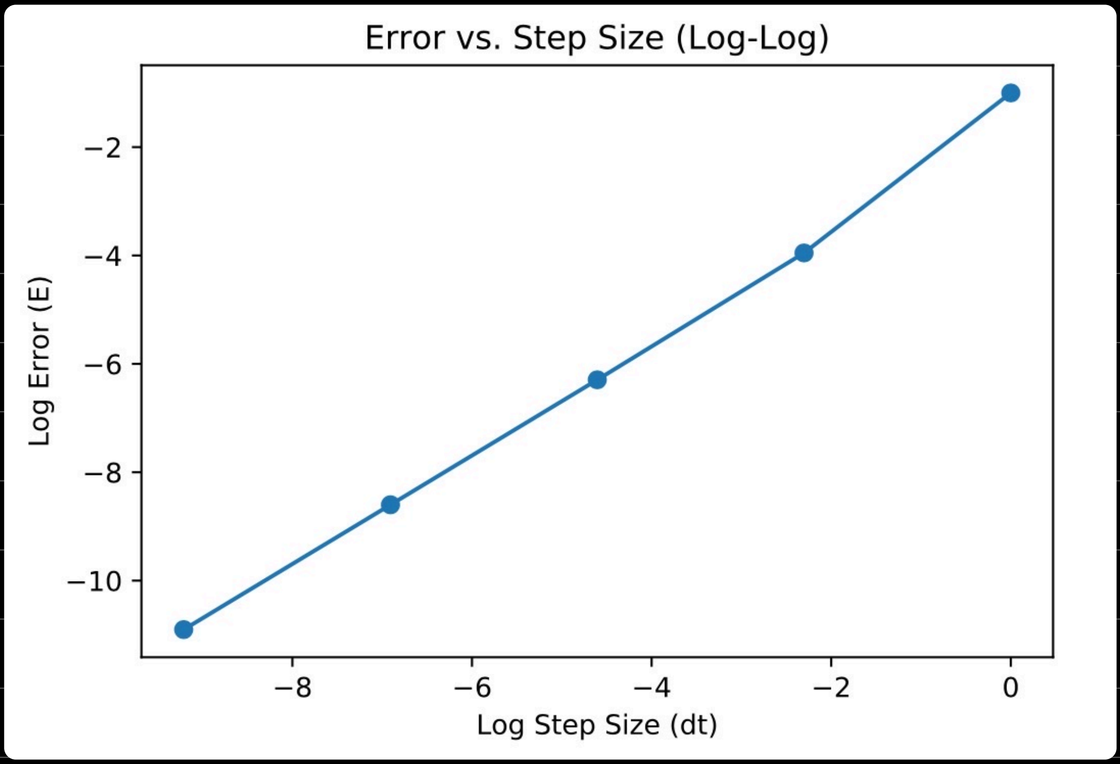
$$E = |x(1) - \hat{x}(1)| = |1/e - \hat{x}(1)|$$

Then, we have:



Clearly, the error grows with step size, but it is difficult to see how. We thus make a log-log plot (next page).

What this plot roughly tells us is that, as we increase our step size by a factor of 10, our error also increases roughly by a factor of 10.



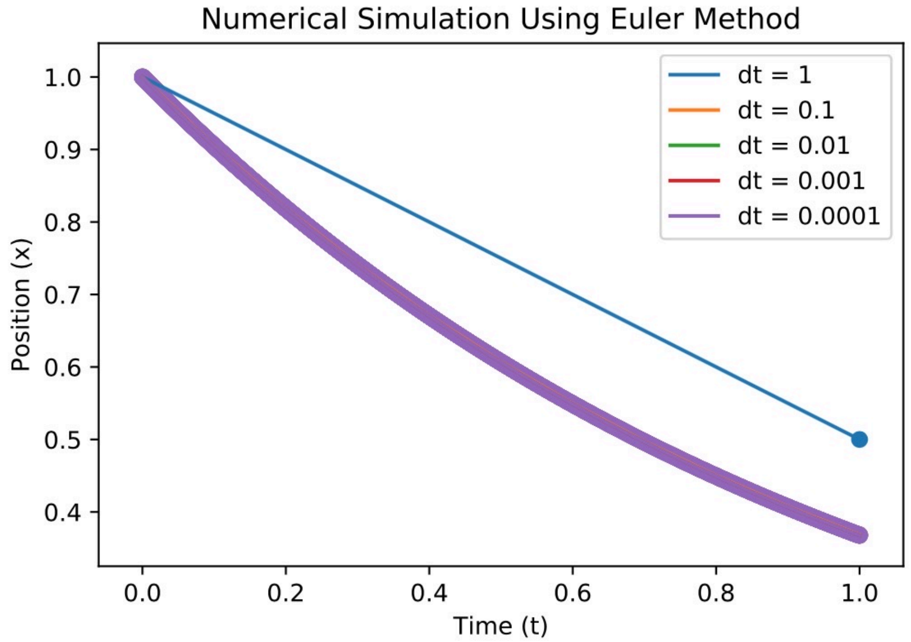
2.8.4

We now numerically simulate the same system using the improved Euler method. We recall that this method computes x_{n+1} from averaging the derivative over the interval $[t_n, t_{n+1}]$, rather than simply using the derivative at t_n.

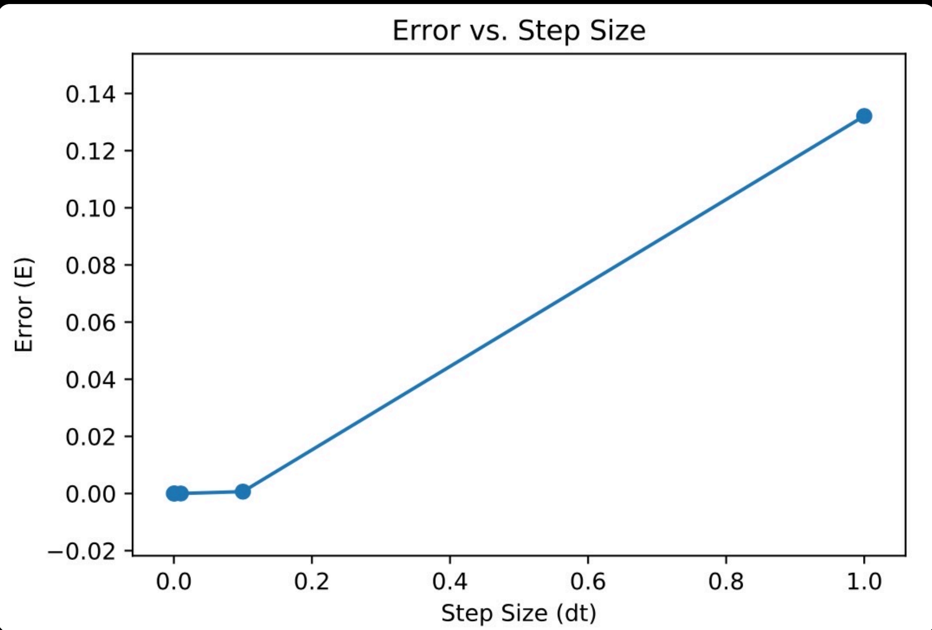
$$\tilde{x}_{n+1} = x_n + f(x_n)\Delta t$$
$$x_{n+1} = x_n + \frac{1}{2}\left[f(x_n) + f(\tilde{x}_{n+1})\right]\Delta t$$

We plot the results of our simulation below:

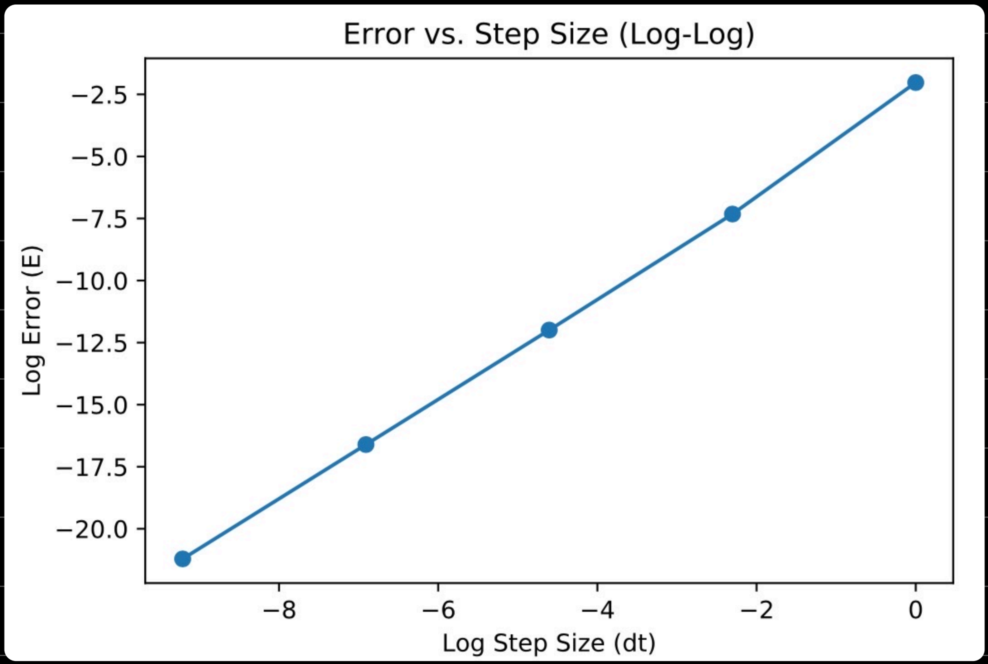
```
For dt = 1, x(1) = 0.5
For dt = 0.1, x(1) = 0.36854098483355185
For dt = 0.01, x(1) = 0.367885618716192
For dt = 0.001, x(1) = 0.36787950253069096
For dt = 0.0001, x(1) = 0.36787944178461873
```



Clearly, even for relatively large time steps, this method creates an excellent fit to the analytical solution. We illustrate this further by plotting the errors.



As we can see, the errors are very small for all but $dt = 1$. In the log-log plot, we can distinguish more structure:



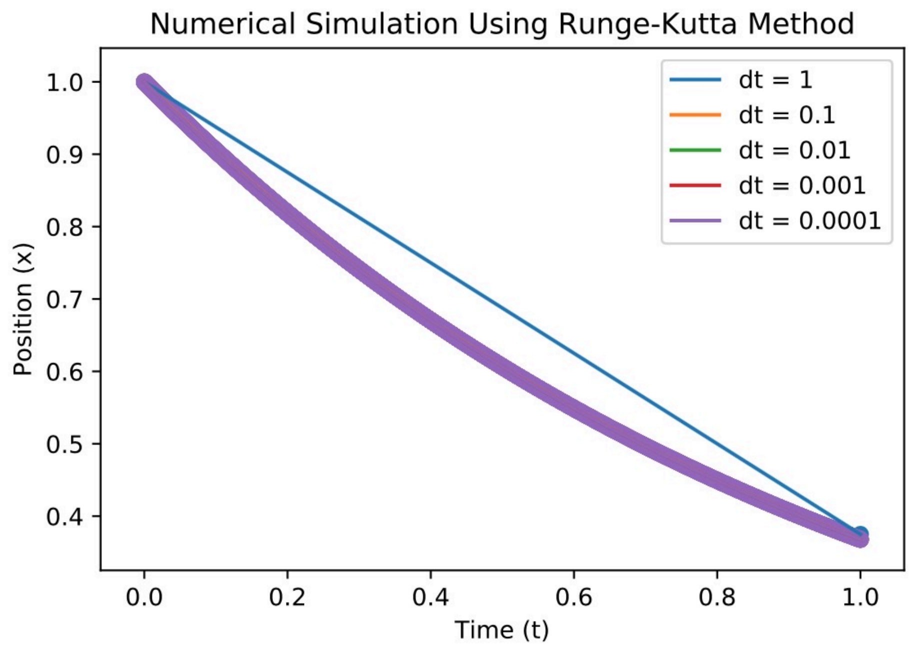
We see that the slope is very roughly $5/2$. That is, as the step size decreases by an order of magnitude, the error decreases by around 2.5 orders of magnitude. This is much faster than the standard Euler method.

2.8.5

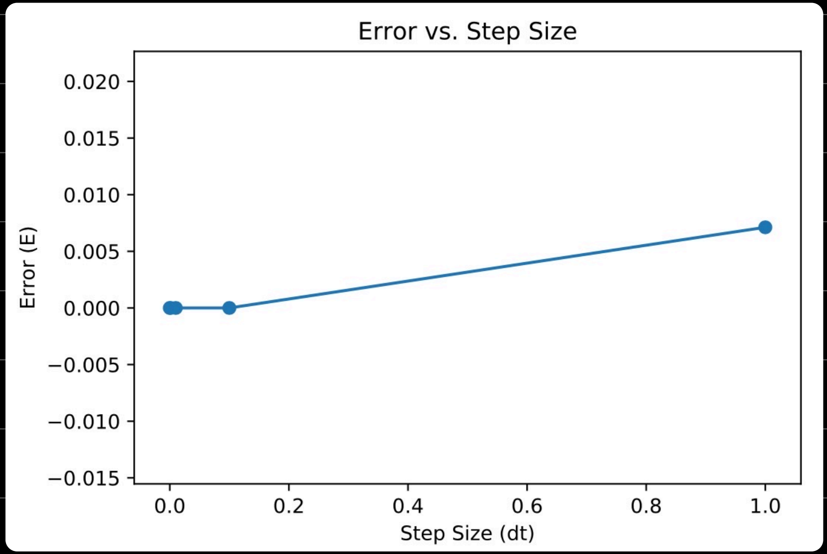
We now simulate this system using the Runge-Kutta method.

Runge-Kutta uses a Taylor Series expansion to generate a numerical solution that converges well even for relatively large Δt!

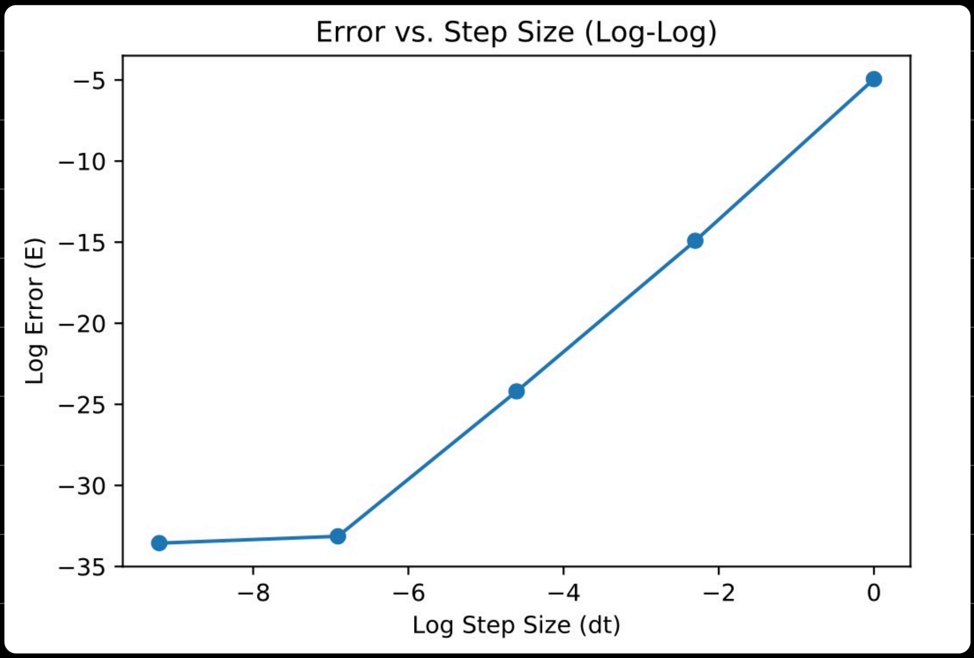
```
For dt = 1, x(1) = 0.375
For dt = 0.1, x(1) = 0.3678797744124984
For dt = 0.01, x(1) = 0.3678794412023554
For dt = 0.001, x(1) = 0.3678794411714464
For dt = 0.0001, x(1) = 0.367879441171445
```



In contrast to our other methods, Runge-Kutta even converges fairly well when $\Delta t = 1$. Beyond that, the fits are indistinguishable from the analytic solution.



We confirm this observation with the above plot. The error is small even for large dt, and is indistinguishable from 0 for the others.



In log-log, we see a slope of nearly 5, which means that errors improve by ~100,000 for each factor of 10 step size decrease. Interestingly, this levels off for smallest two steps. However, at this size, the error is so small it hardly matters.