day 1

March 26, 2019

0.1 Day 1: Of Numerical Integration, Python and Tensorflow

Welcome to Day 1! Today, we start with our discussion of what Numerical Integration is.

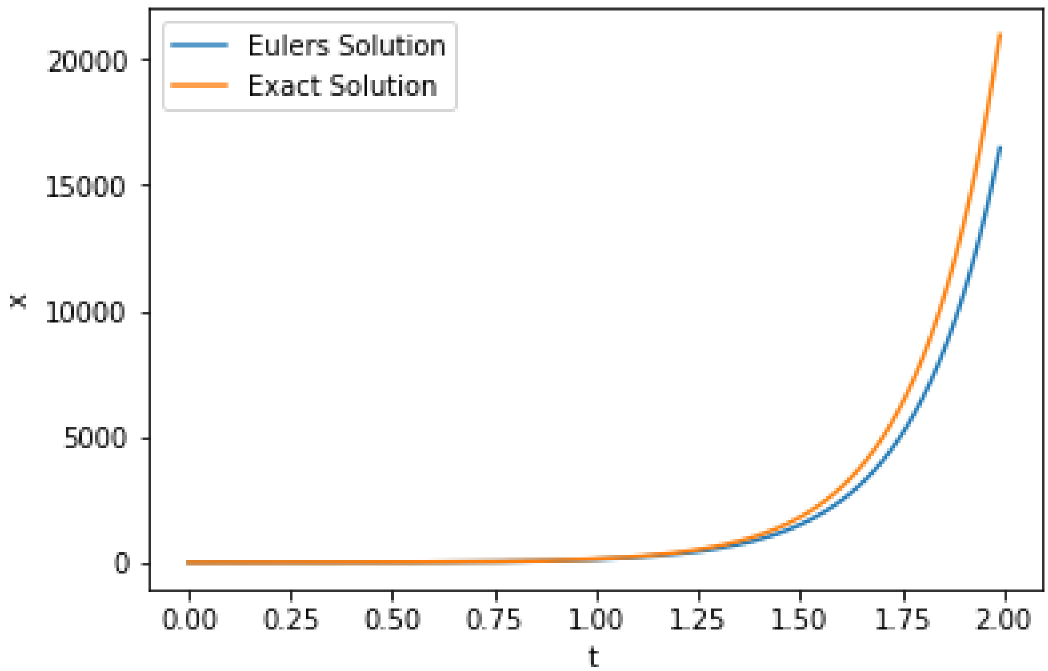
0.1.1 What is Numerical Integration?

… Insert Text Here …

Euler's Method … Insert Text Here …

Euler and Python Let $\frac{dx}{dt} = f(x,t)$, we want to find $x(t)$ over $t \in [0,2)$, given that $x(0) = 1$ and $f(x,t) = 5x$. The exact solution of this equation would be $x(t) = e^{5t}$.

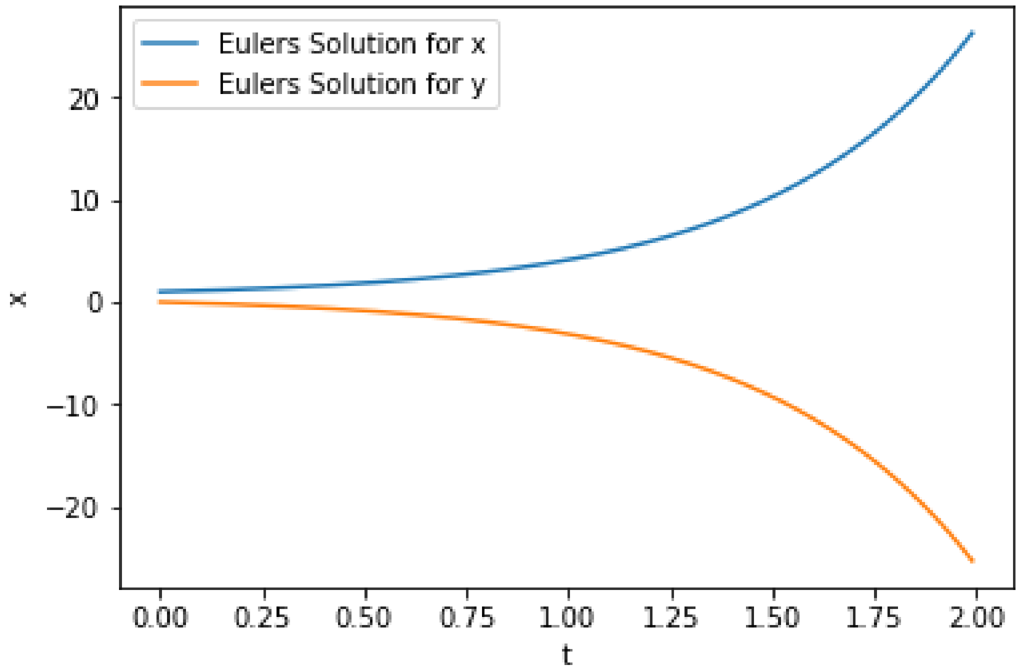
```
In [39]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline

         def f(x,t): # define the function f(x,t)
             return 5*x

         epsilon = 0.01 # define timestep

         t = np.arange(0,2,epsilon) # define an array for t
         x = np.zeros(t.shape) # define an array for x

         x[0]= 1 # set initial condition

         for i in range(1,t.shape[0]):
             x[i] = epsilon*f(x[i-1],t[i-1])+x[i-1] # Euler Integration Step

         plt.plot(t,x,label="Eulers Solution")
         plt.plot(t,np.exp(5*t),label="Exact Solution")
         plt.xlabel("t")
         plt.ylabel("x")
         plt.legend()
         plt.show()
```



Euler and Vectors Euler's Method also applies to vectors and can solve simultaneous differential equations

Let $\frac{d\vec{x}}{dt} = f(\vec{x}, t)$, we want to find $\vec{x}(t)$ over $t \in [0, 2)$, given that $\vec{x} = [x, y]$, $\vec{x}(0) = [1, 0]$ and $f(\vec{x}, t) = [x - y, y - x]$.

```
In [47]: def f(x,t): # define the function f(x,t)
             x_,y_ = x
             return np.array([x_-y_,y_-x_])

         epsilon = 0.01 # define timestep

         t = np.arange(0,2,epsilon) # define an array for t
         x = np.zeros((2,t.shape[0])) # define an array for x

         x[:,0]= [1,0] # set initial condition

         for i in range(1,t.shape[0]):
             x[:,i] = epsilon*f(x[:,i-1],t[i-1])+x[:,i-1] # Euler Integration Step

         plt.plot(t,x[0,:],label="Eulers Solution for x")
         plt.plot(t,x[1,:],label="Eulers Solution for y")
         plt.xlabel("t")
         plt.ylabel("x")
         plt.legend()
         plt.show()
```



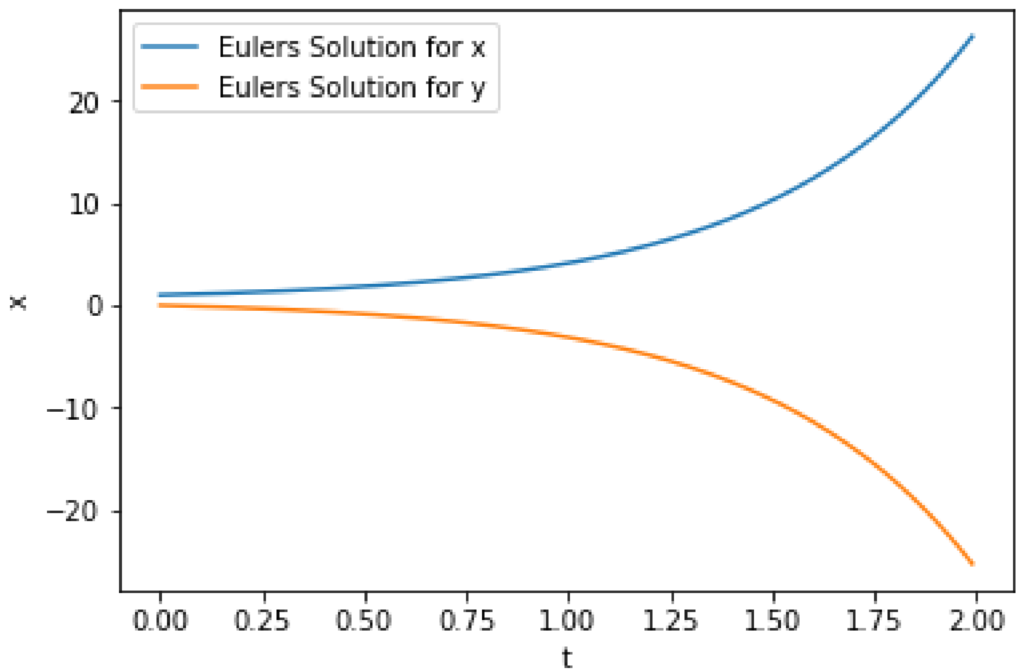
A Single function for Euler Integration Now, we create a generalized function that takes in 3 inputs ie. the function $f(\vec{y},t)$ when $\frac{d\vec{y}}{dt} = f(\vec{y},t)$, the time array, and initial vector $\vec{y_0}$.

... Insert Algorithm for Function ...

```
In [79]: def check_increasing(t): # Ensure Time is Monotonically Increasing
             return np.all(t[1:]>t[:-1])

         def check_type(y,t): # Ensure Input is Correct
             return y.dtype == np.floating and t.dtype == np.floating

         class _Integrator():

             def integrate(self,func,y0,t):
                 time_delta_grid = t[1:] - t[:-1]

                 y = np.zeros((y0.shape[0],t.shape[0]))
                 y[:,0] = y0

                 for i in range(time_delta_grid.shape[0]):
                     y[:,i+1]= time_delta_grid[i]*func(y[:,i],t[i])+y[:,i]
                 return y


         def odeint_euler(func,y0,t):
```

```
    y0 = np.array(y0)
    t = np.array(t)
    if check_increasing(t) and check_type(y0,t):
        return _Integrator().integrate(func,y0,t)
    else:
        print("error encountered")
```

```
In [78]: solution = odeint_euler(f,[1.,0.],t)

         plt.plot(t,solution[0,:],label="Eulers Solution for x")
         plt.plot(t,solution[1,:],label="Eulers Solution for y")
         plt.xlabel("t")
         plt.ylabel("x")
         plt.legend()
         plt.show()
```



0.1.2 An Introduction to TensorFlow

https://www.geeksforgeeks.org/introduction-to-tensorflow/

TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well!

Euler Integration Function in TensorFlow

In [124]:
```
import tensorflow as tf

def tf_check_type(t, y0): # Ensure Input is Correct
    if not (y0.dtype.is_floating and t.dtype.is_floating):
        raise TypeError('Error in Datatype')

def tf_check_increasing(t): # Ensure Time is Monotonically Increasing
    assert_op = tf.Assert(tf.reduce_all(t[1:]>t[:-1]),["Time must be monotonic"])
    return tf.control_dependencies([assert_op])

class _Tf_Integrator():

    def integrate(self, func, y0, t):
        time_delta_grid = t[1:] - t[:-1]
        scan_func = self._make_scan_func(func)
        y = tf.scan(scan_func, (t[:-1], time_delta_grid),y0)
        return tf.concat([[y0], y], axis=0)

    def _make_scan_func(self, func): # stepper function
        def scan_func(y, t_dt):
            t, dt = t_dt
            dy = dt*func(y,t)
            dy = tf.cast(dy, dtype=y.dtype) # Failsafe
            return y + dy
        return scan_func


def tf_odeint_euler(func, y0, t):
    t = tf.convert_to_tensor(t, preferred_dtype=tf.float64, name='t')
    y0 = tf.convert_to_tensor(y0, name='y0')
    tf_check_type(y0,t)
    with tf_check_increasing(t):
        return _Tf_Integrator().integrate(func,y0,t)
```

In [127]:
```
def f(X,t):
    x = X[:-1]
    y = X[1:]
    out = tf.concat([x-y,y-x],0)
    return out

y0 = tf.constant([1,0], dtype=tf.float64)

epsilon = 0.01
t = np.arange(0,2,epsilon)

state = tf_odeint_euler(f,y0,t)
```

```
    with tf.Session() as sess:
        state = sess.run(state)
```

```
In [128]: plt.plot(t,state.T[0,:],label="Eulers Solution for x")
          plt.plot(t,state.T[1,:],label="Eulers Solution for y")
          plt.xlabel("t")
          plt.ylabel("x")
          plt.legend()
          plt.show()
```