Lennard-Jones plot

July 8, 2015

1 Lennard-Jones potential plot

First initialize numpy and matplotlib using IPython magic.

```
In [10]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Defining our Lennard-Jones potential function, using positional and keyword arguments

```
In [11]: def lj(r, epsilon=1.0, sigma=1.0):
             if r > 0.0:
                 return epsilon*(sigma**12/r**12-sigma**6/r**6)
             else:
                 return None
```

Creating an array with function arguments. Here we use `arange` but we could also use `linspace`.

```
In [12]: x = arange(0.5, 2.5, 0.001)
```

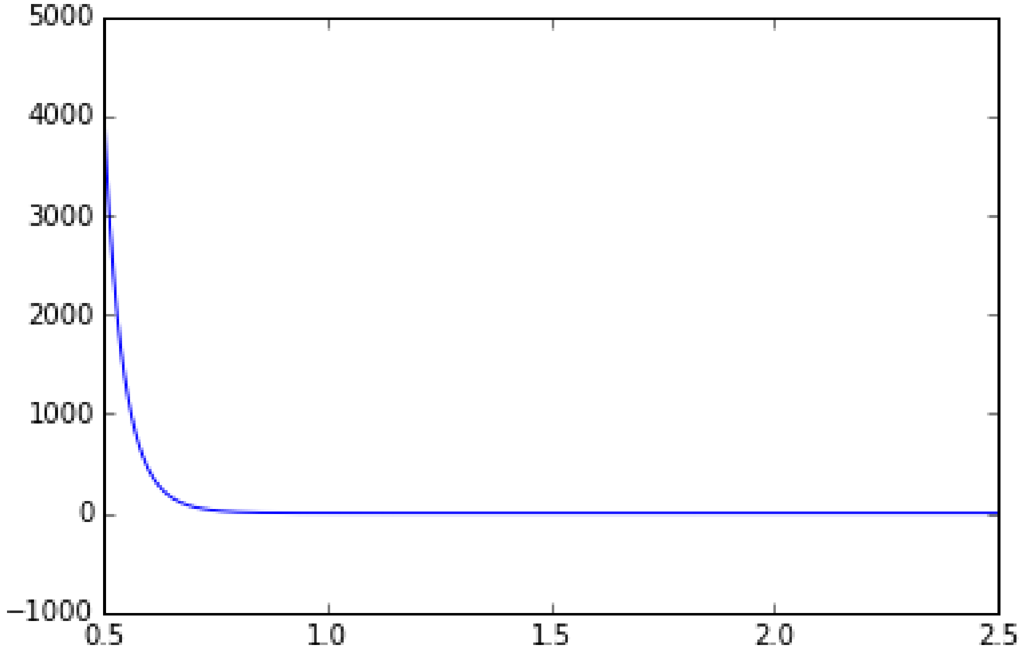
Create a vectorized version of our function, so we can call it with an array as arguments. Calling `lj(x)` would throw an error.

```
In [13]: vlj = vectorize(lj)
```

Now we plot the function.

```
In [14]: plot(x, vlj(x))
```

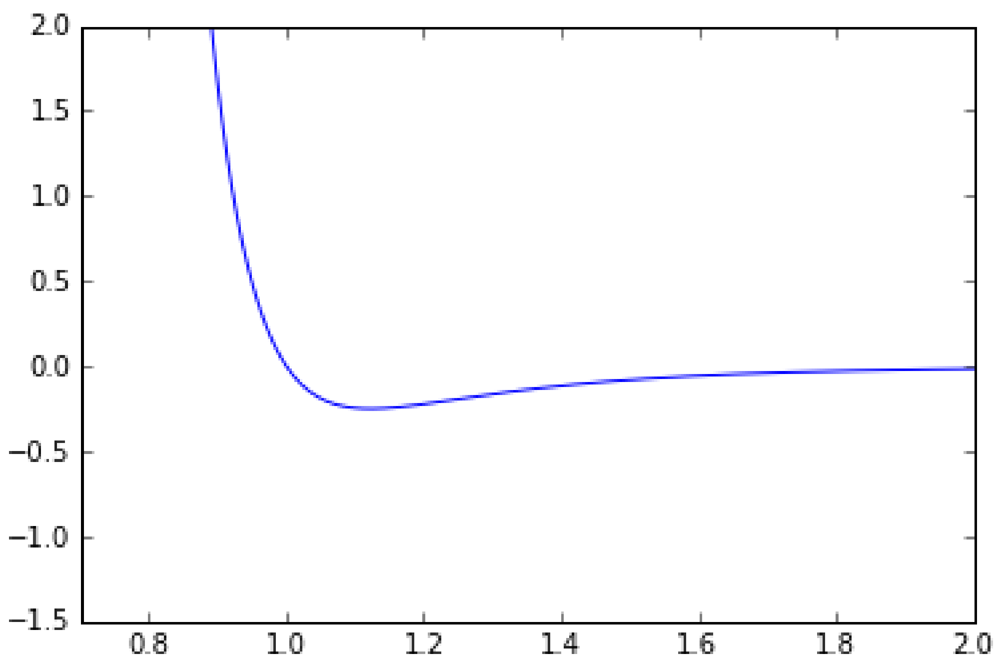
```
Out[14]: [<matplotlib.lines.Line2D at 0x104016690>]
```



Adjusting the axes limits to the interesting part of the function.

```
In [15]: xlim(0.7, 2.0)
         ylim(-1.5, 2)
         plot(x, vlj(x))
```

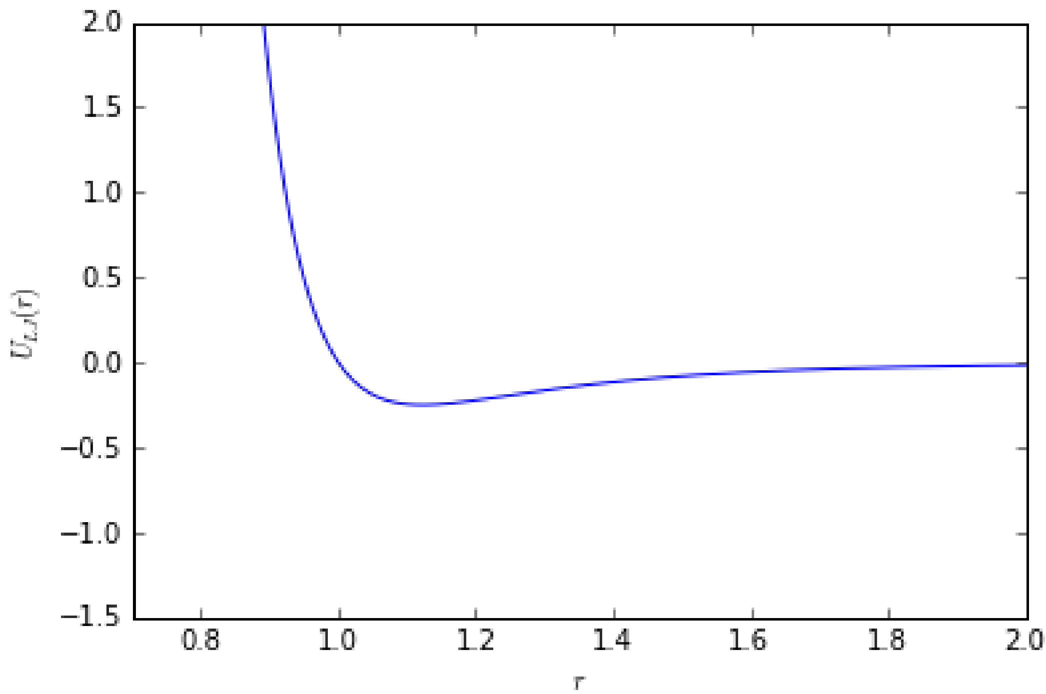
```
Out[15]: [<matplotlib.lines.Line2D at 0x1040fc850>]
```



Adding LaTeX labels to the axes.

```
In [16]: xlabel(r"$r$")
         ylabel(r"$U_{LJ}(r)$")
         xlim(0.7, 2.0)
         ylim(-1.5, 2)
         plot(x, vlj(x))
```

```
Out[16]: [<matplotlib.lines.Line2D at 0x104213710>]
```



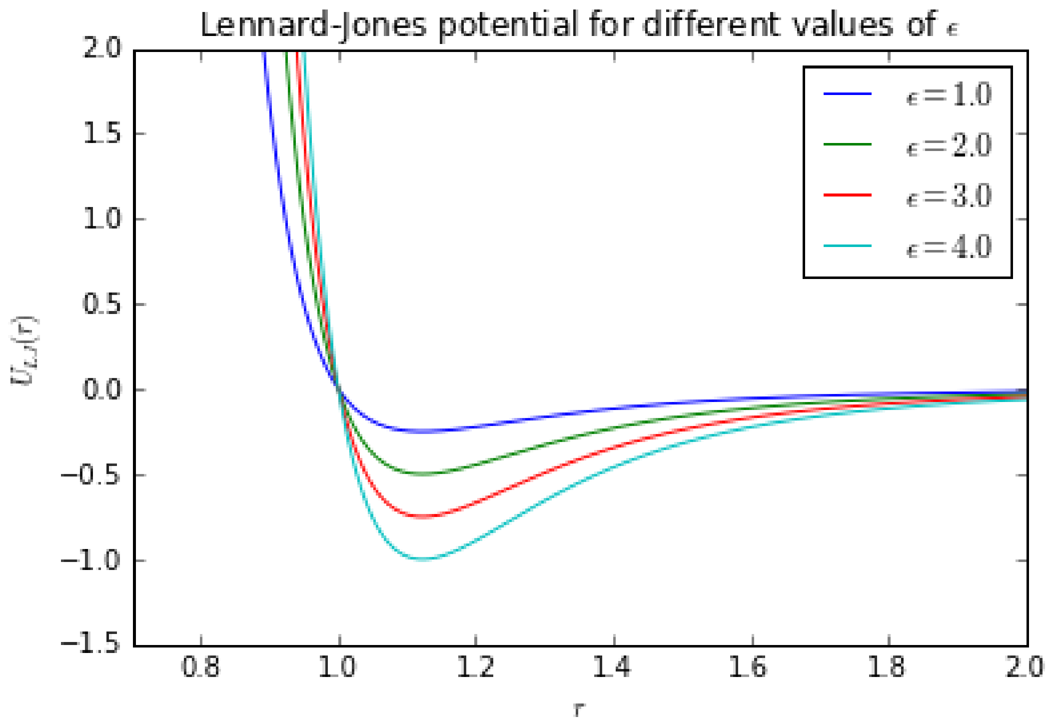

```
In [17]: epsilons = [1.0, 2.0, 3.0, 4.0] # list of different epsilons

         xlim(0.7, 2.0)
         ylim(-1.5, 2)
         xlabel(r"$r$")
         ylabel(r"$U_{LJ}(r)$")

         title(r"Lennard-Jones potential for different values of $\epsilon$")

         # loop over list of epsilons,
         # creating labels for the different plots
         for epsilon in epsilons:
             mylabel = r"$\epsilon = " + str(epsilon) + r"$"
             plot(x, vlj(x, epsilon=epsilon), label=mylabel)

         legend() # put the legend in the plot
         savefig('lj.pdf') # save plot to a PDF
```



Saving the function arguments and values into a text file.

```
In [18]: savetxt('lj.txt', transpose([x, vlj(x)]), header='x, y', delimiter='\t')
```