lec8

September 12, 2016

0.1 7.5 Numerical methods for definite integrals

Here are the examples from class on 9/9/16.

Numerical approximations of integrals. Let's start with

$$f(x) = \frac{1}{x}$$

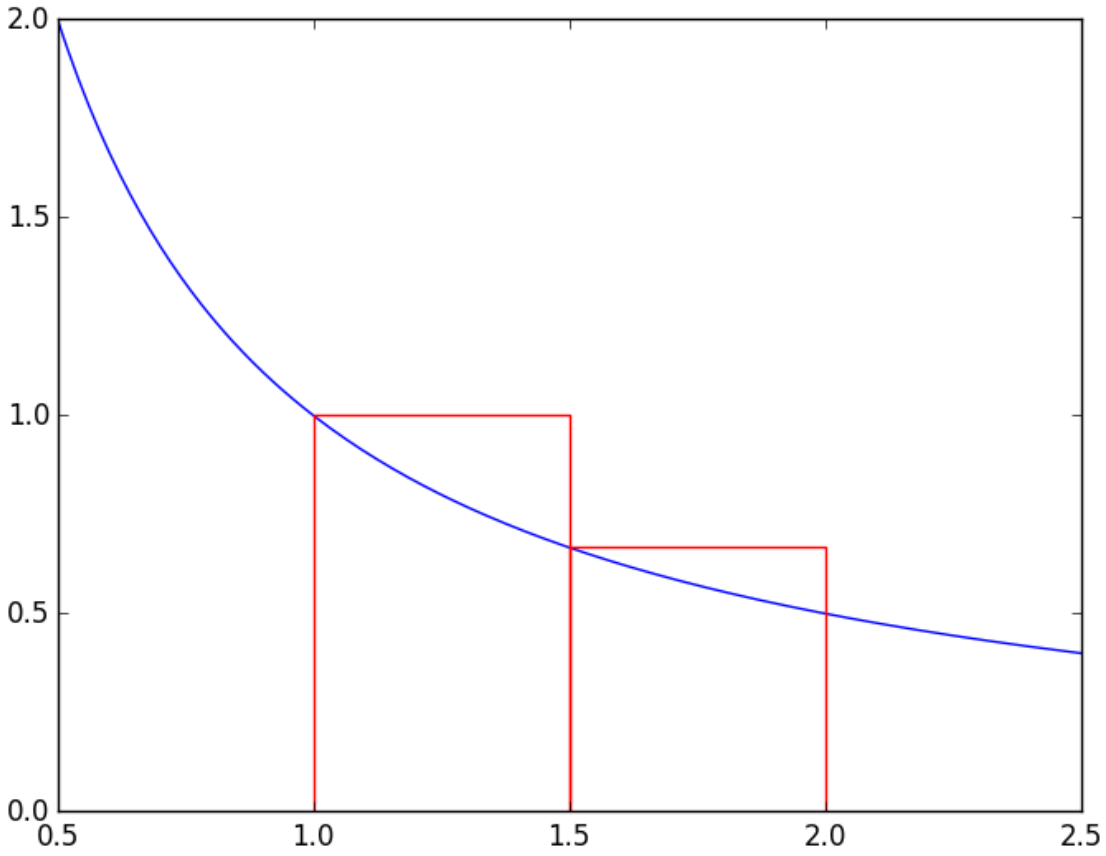
and try to estimate

$$\int_1^2 f(x)\ dx.$$

In this case, we know the answer is $\ln(2)$.

```
In [1]: ## You can ignore this.
        using lec8,PyPlot
```

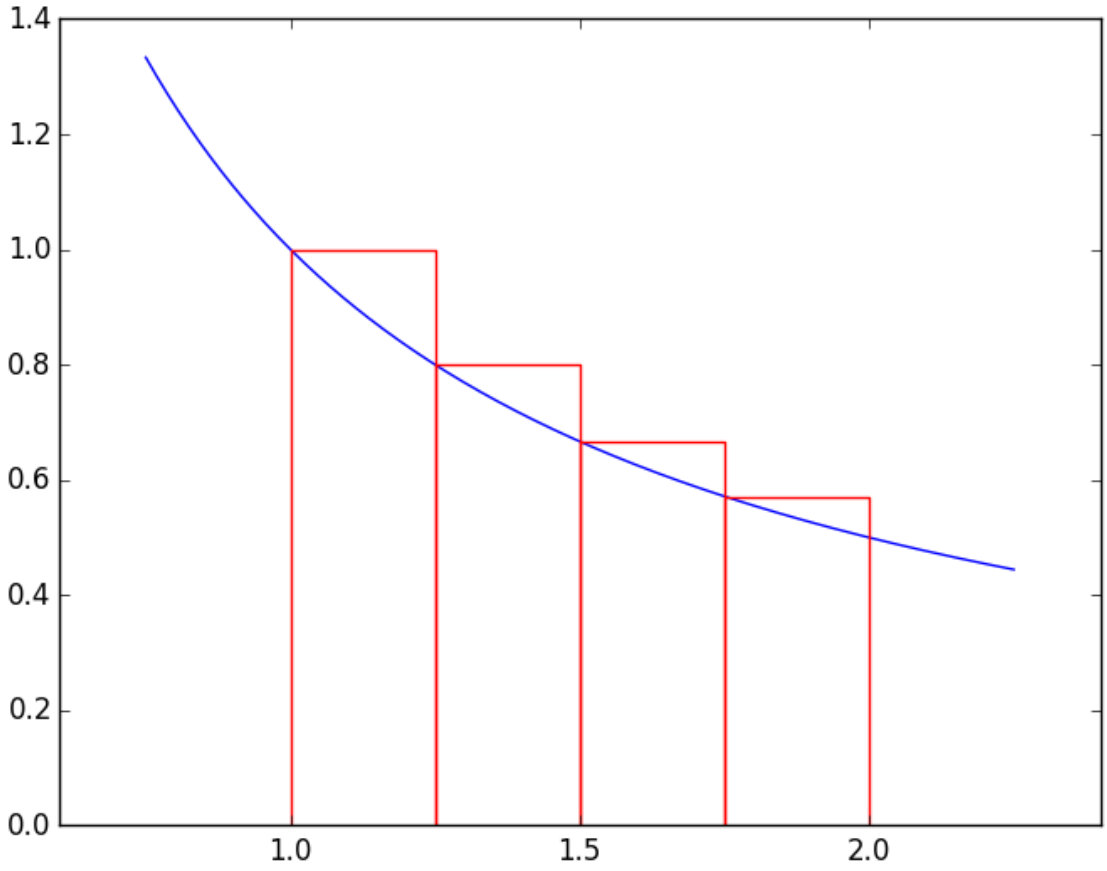
```
In [2]: ## LEFT(2) approximation of the integral above.
        left(x->1/x,a=1,b=2,n=2)
```



```
LEFT(2) = f(1.0)*0.5 + f(1.5)*0.5
```

```
Out[2]: 0.8333333333333333
```

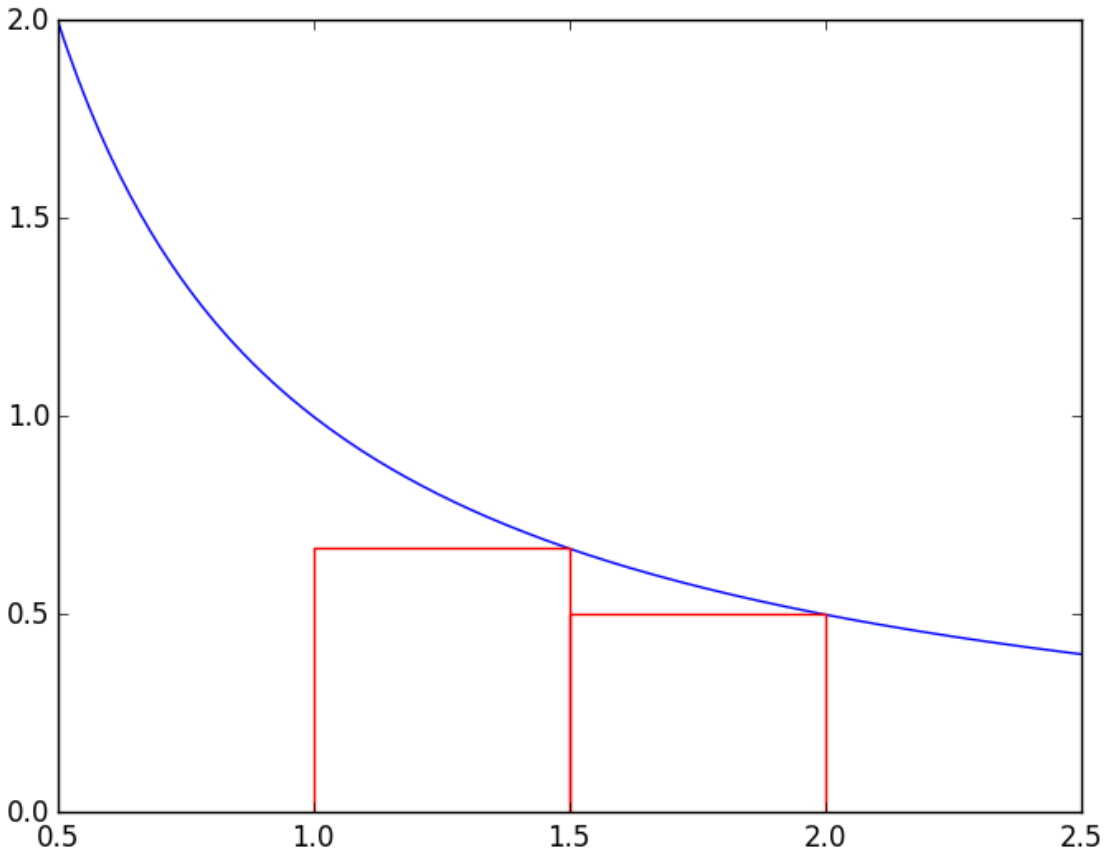
```
In [3]: ## LEFT(4)
        left(x->1/x,a=1,b=2,n=4)
```




```
LEFT(4) = f(1.0)*0.25 + f(1.25)*0.25 + f(1.5)*0.25 + f(1.75)*0.25
```

```
Out[3]: 0.7595238095238095
```

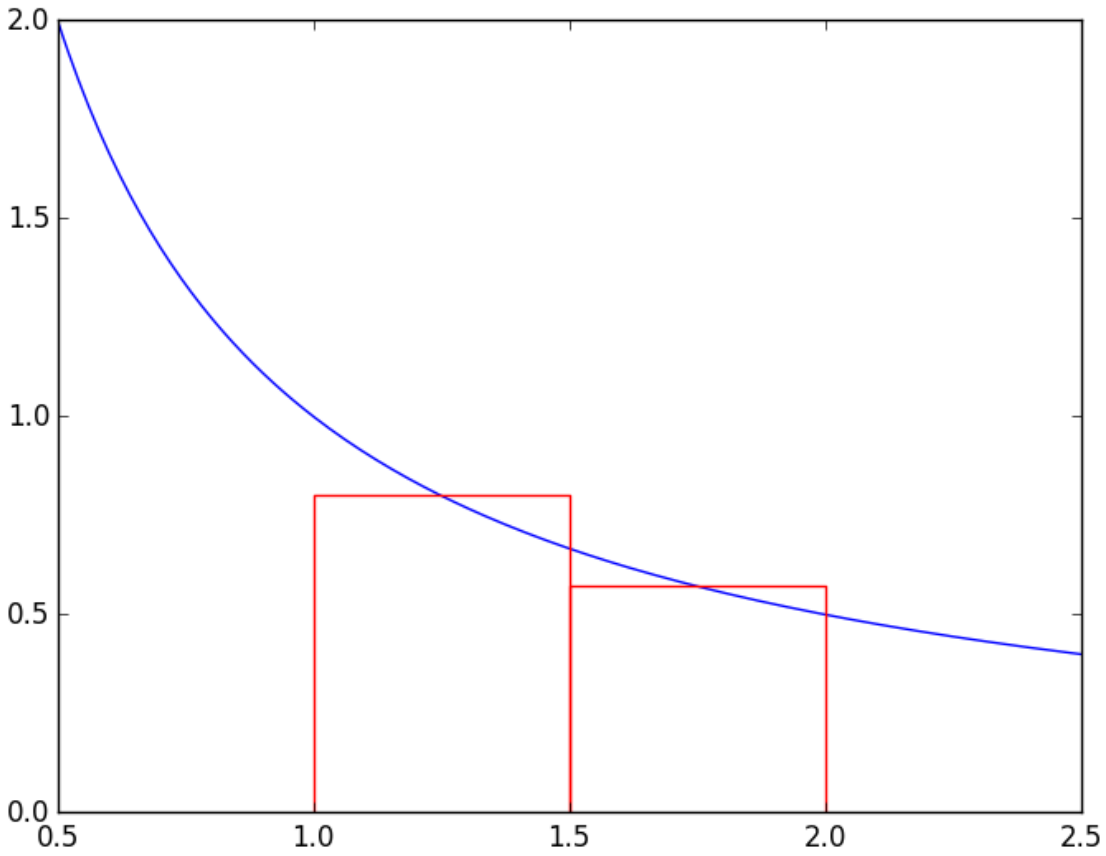
```
In [4]: ## RIGHT(2)
        right(x->1/x,a=1,b=2,n=2)
```




```
RIGHT(2) = f(1.5)*0.5 + f(2.0)*0.5
```

```
Out[4]: 0.5833333333333333
```

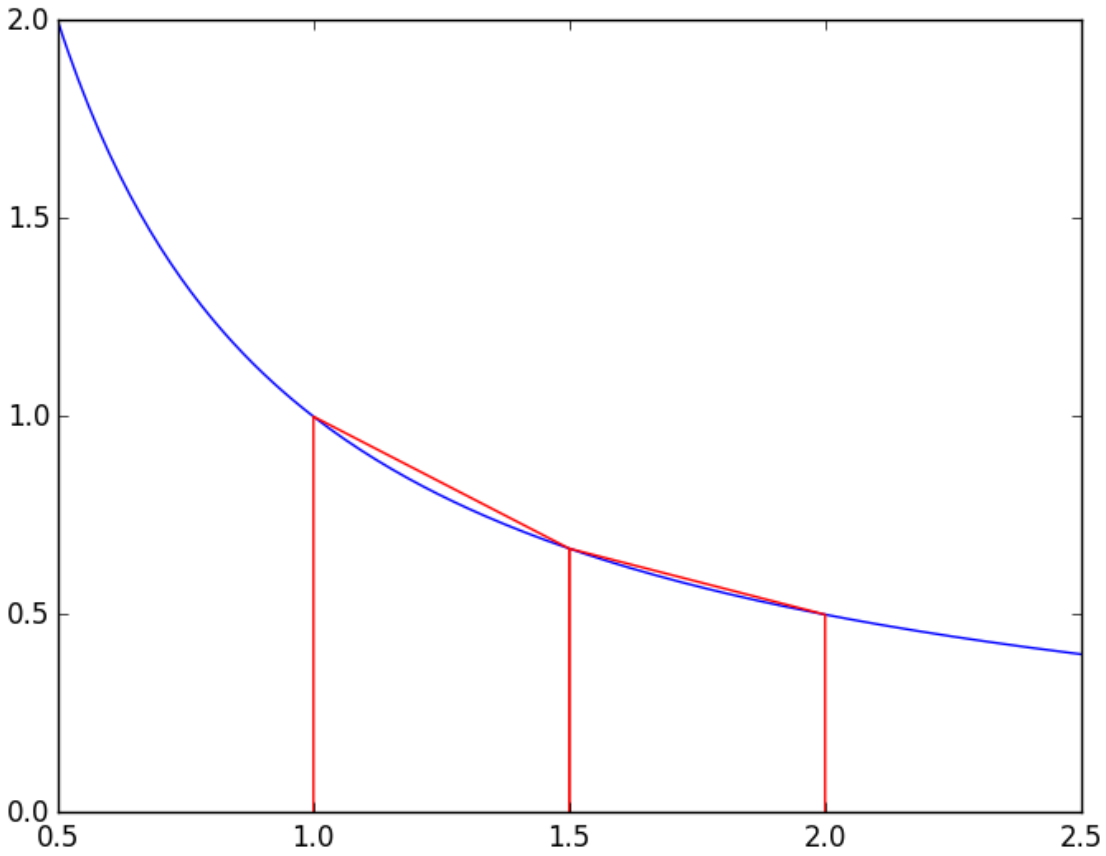
```
In [5]: ## MID(2)
        mid(x->1/x,a=1,b=2,n=2)
```




```
MID(2) = f(1.25)*0.5 + f(1.75)*0.5
```

```
Out[5]: 0.6857142857142857
```

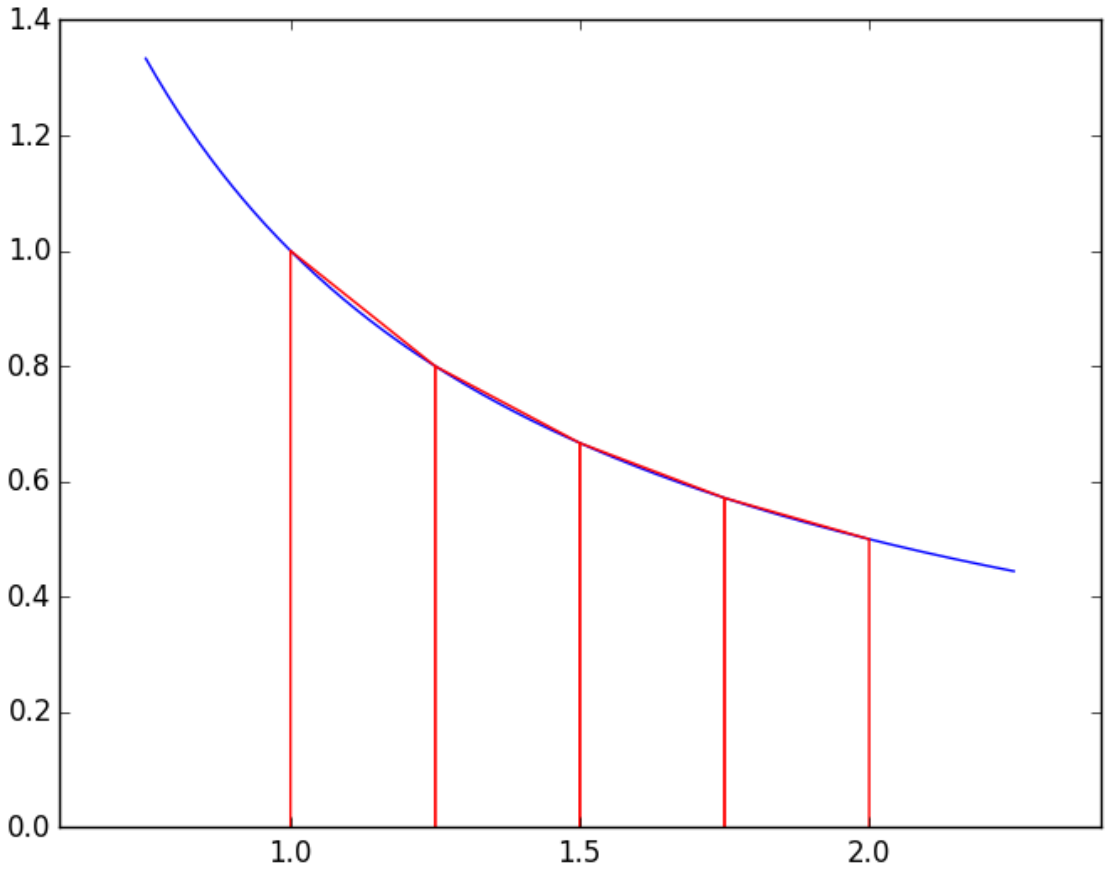
```
In [6]: ## Trapezoid rule TRAP(2)
        trap(x->1/x,a=1,b=2,n=2)
```




```
TRAP(2) = 0.5*[f(1.0)+f(1.5)]*0.5 + 0.5*[f(1.5)+f(2.0)]*0.5
```

```
Out[6]: 0.7083333333333333
```

```
In [7]: ## TRAP(4)
        trap(x->1/x,a=1,b=2,n=4)
```




```
TRAP(4) = 0.5*[f(1.0)+f(1.25)]*0.25 + 0.5*[f(1.25)+f(1.5)]*0.25 + 0.5*[f(1.5)+f(1.75)]*0.25 + 0.5*[f(1.
```

```
Out[7]: 0.6970238095238095
```

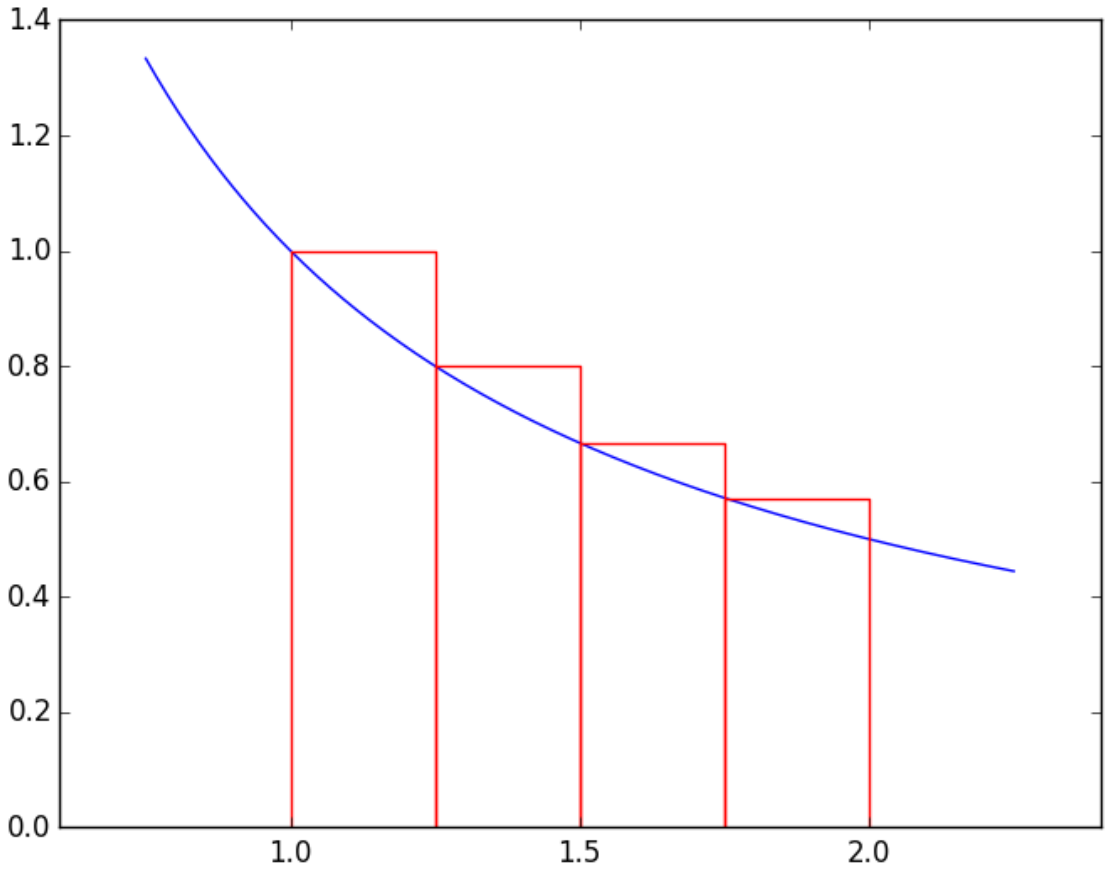
```
In [8]: ## the truth
        log(2)
```

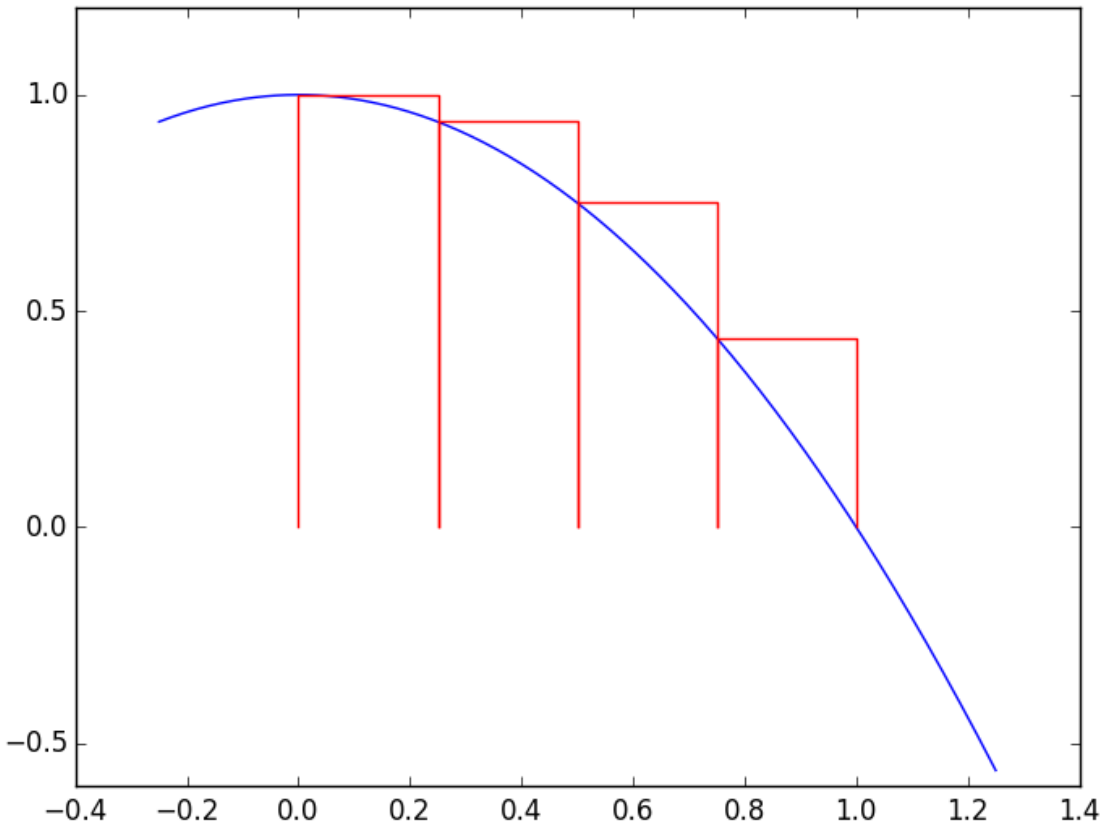
```
Out[8]: 0.6931471805599453
```

Over or under-estimate?

```
In [10]: ## For any decreasing function (regardless of concavity), LEFT(n) is an overestimate:
         figure()
         left(x->1/x,a=1,b=2,n=4)
         figure()
         left(x->1-x^2,a=0,b=1,n=4)
```

```
LEFT(4) = f(1.0)*0.25 + f(1.25)*0.25 + f(1.5)*0.25 + f(1.75)*0.25
LEFT(4) =
```

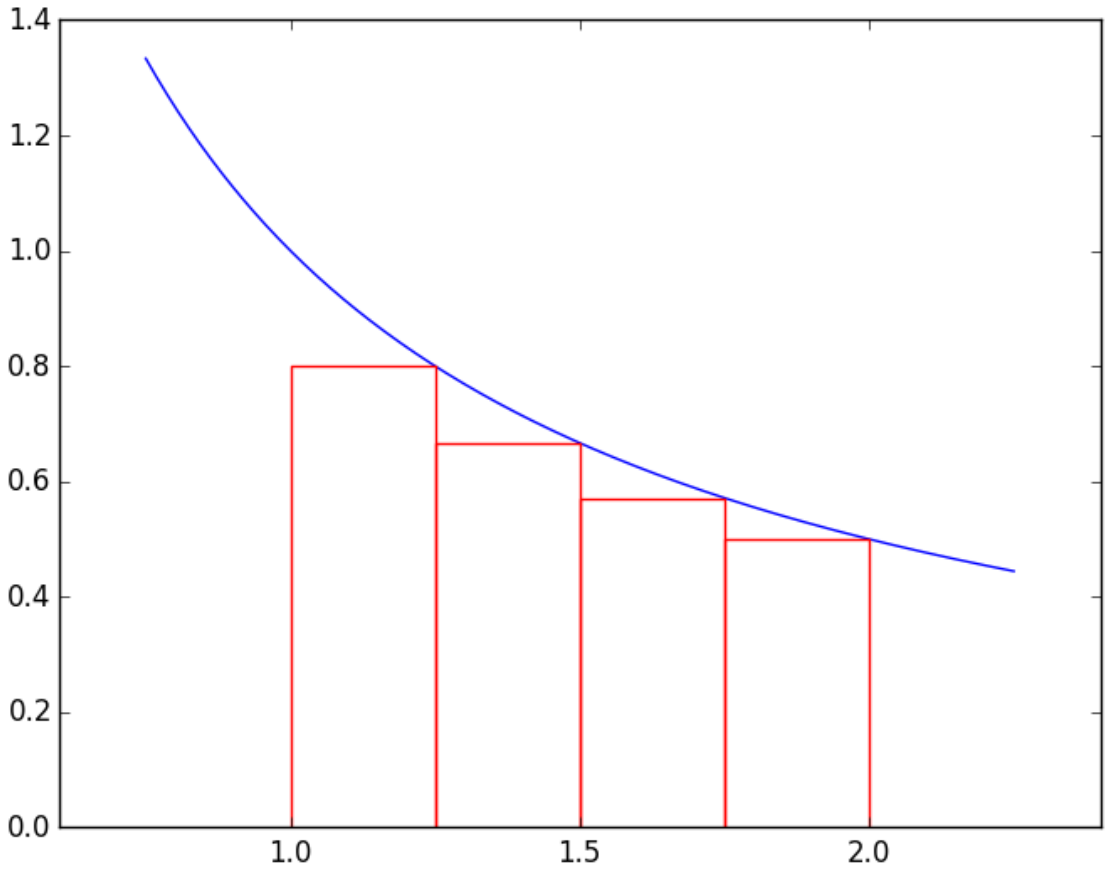

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0

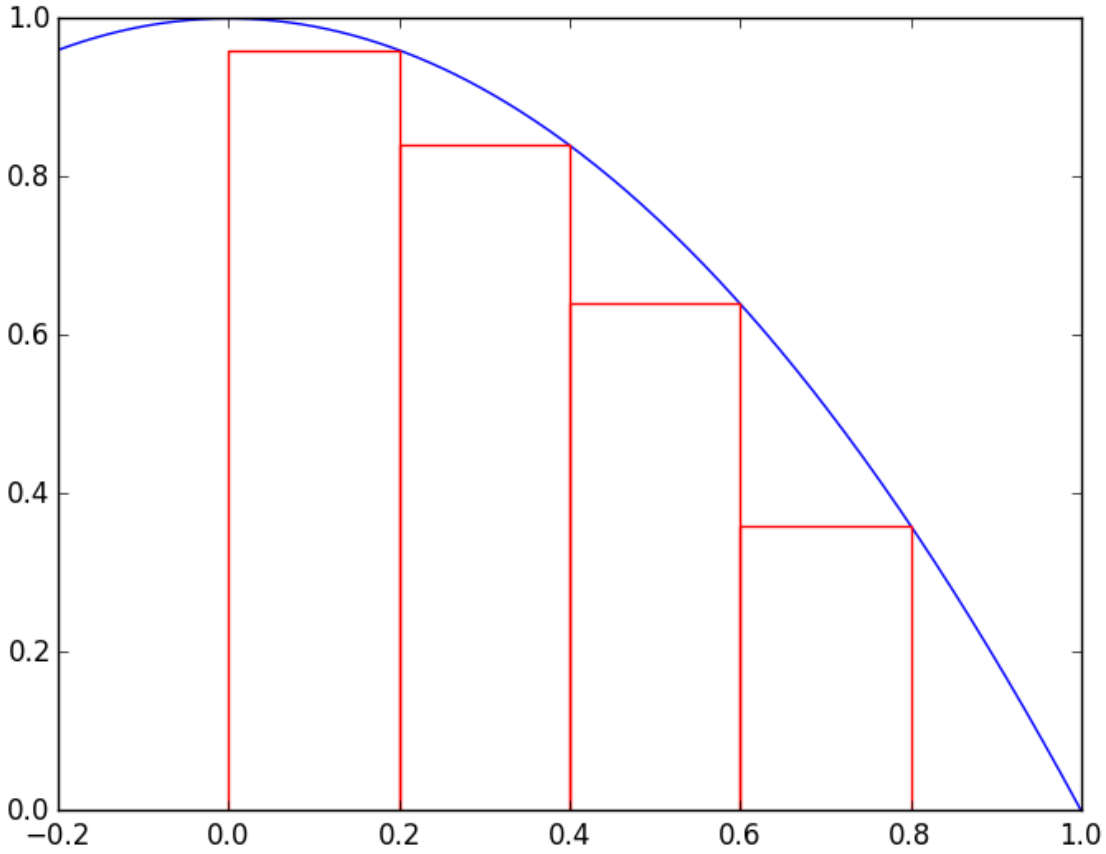



```
f(0.0)*0.25 + f(0.25)*0.25 + f(0.5)*0.25 + f(0.75)*0.25
```

Out[10]: 0.78125

```
In [11]: ## And for any decreasing function (regardless of concavity), RIGHT(n) is an underestimate:
         figure()
         right(x->1/x,a=1,b=2,n=4)
         figure()
         right(x->1-x^2,a=0,b=0.8,n=4)
```

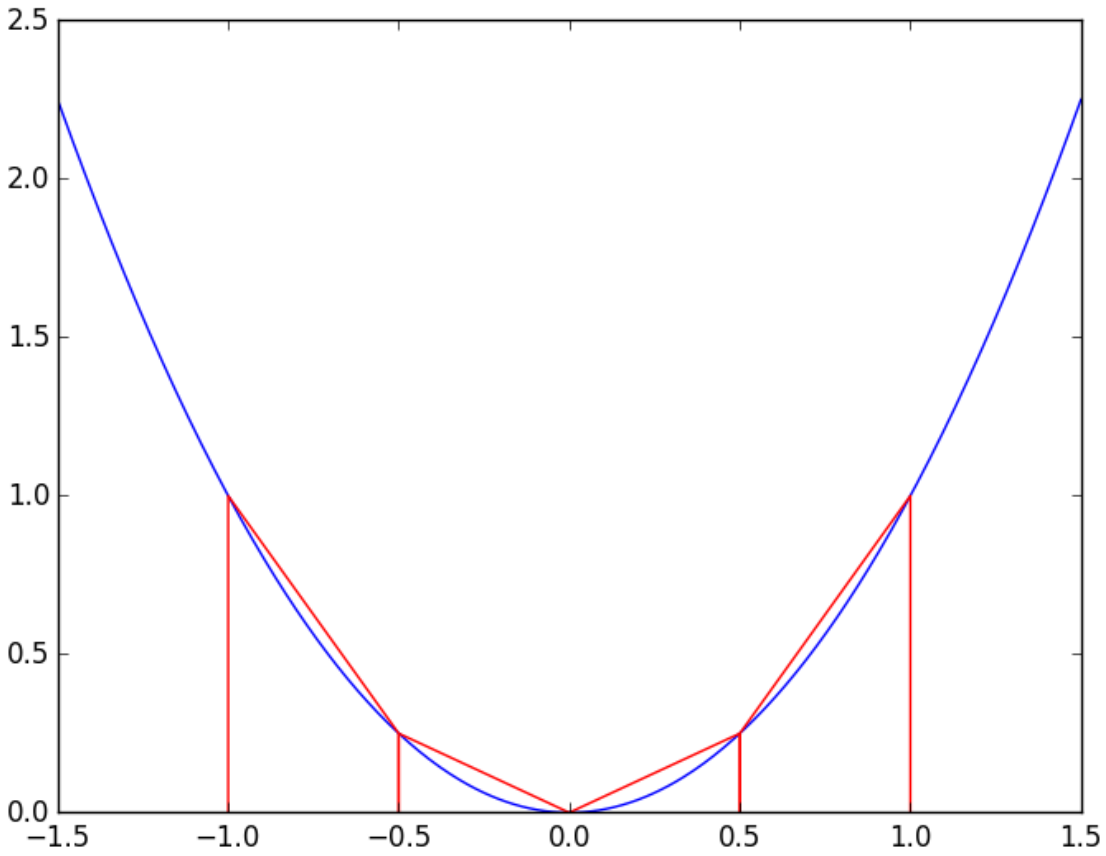

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0




```
RIGHT(4) = f(1.25)*0.25 + f(1.5)*0.25 + f(1.75)*0.25 + f(2.0)*0.25
RIGHT(4) = f(0.2)*0.2 + f(0.4)*0.2 + f(0.6000000000000001)*0.2 + f(0.8)*0.2
```

```
Out[11]: 0.5599999999999999
```

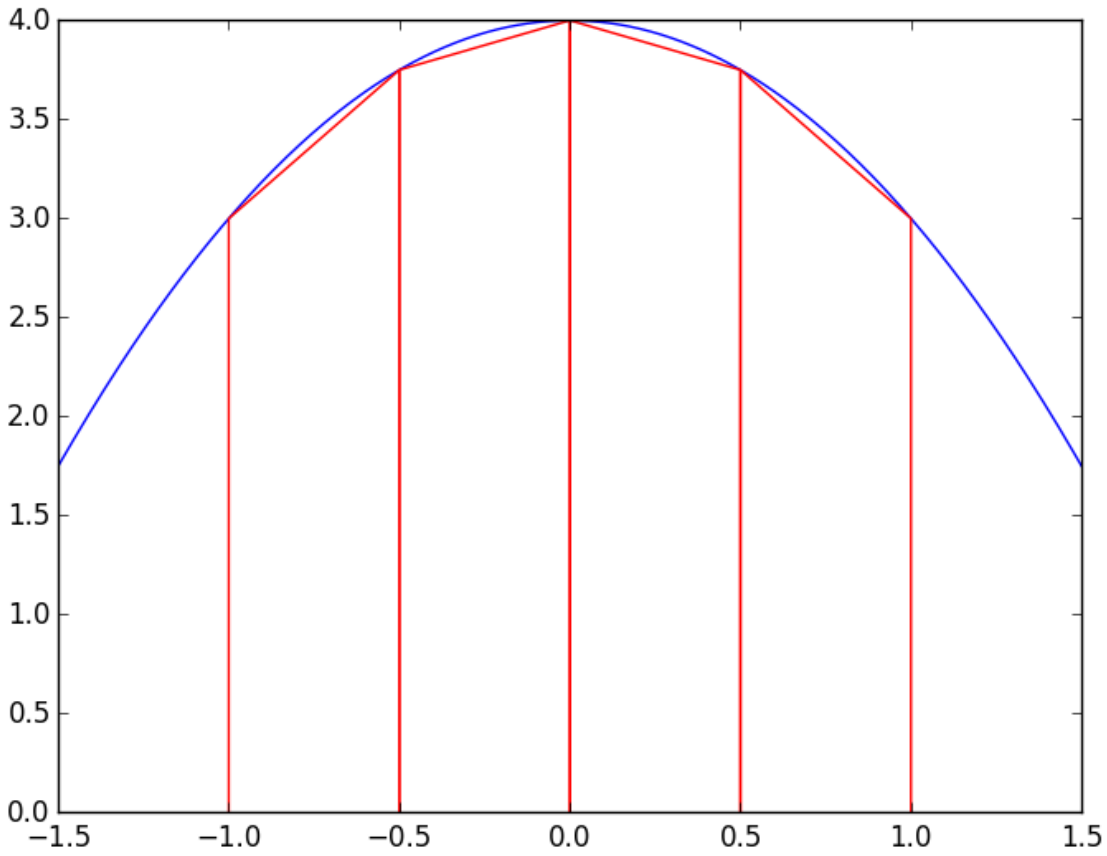
```
In [12]: ## TRAP(n) is an overestimate if the function is concave up (regardless of increasing or decre
         trap(x->x^2,a=-1,b=1,n=4)
```




```
TRAP(4) = 0.5*[f(-1.0)+f(-0.5)]*0.5 + 0.5*[f(-0.5)+f(0.0)]*0.5 + 0.5*[f(0.0)+f(0.5)]*0.5 + 0.5*[f(0.5)+
```

Out[12]: 0.75

In [13]:
```
## TRAP(n) is an underestimate if the function is concave down (regardless of increasing or de
trap(x->4-x^2,a=-1,b=1,n=4)
```




```
TRAP(4) = 0.5*[f(-1.0)+f(-0.5)]*0.5 + 0.5*[f(-0.5)+f(0.0)]*0.5 + 0.5*[f(0.0)+f(0.5)]*0.5 + 0.5*[f(0.5)+
```

Out[13]: 7.25

```
In [14]: ## MID(n) is the other way, but it is harder to see.  The book (Section 7.5) explains this.
```

In [ ]: