MATH1110H-B-lab-2023-09-26-F01

October 3, 2023

[1]:

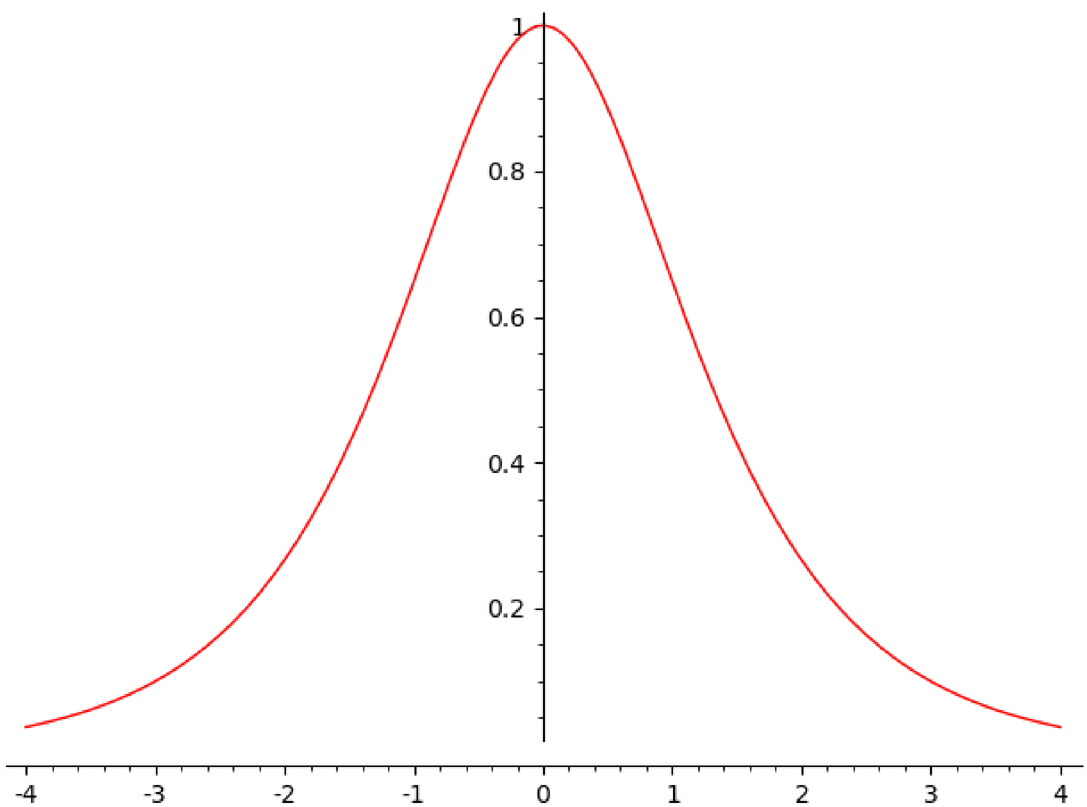
```
# MATH 1110H-B F02 Lab 2023-09-19
#
# Wherein we colour our grpahs, add up plots, and learn about the
# lim and solve commands.

p1 = plot(sech(x),-4,4,color='red')  # We can change the colour of
                                     # the graph by specifying one
                                     # other than the default blue.
                                     # We can also give a plot a
                                     # name instead of displaying
                                     # it immediately.
```

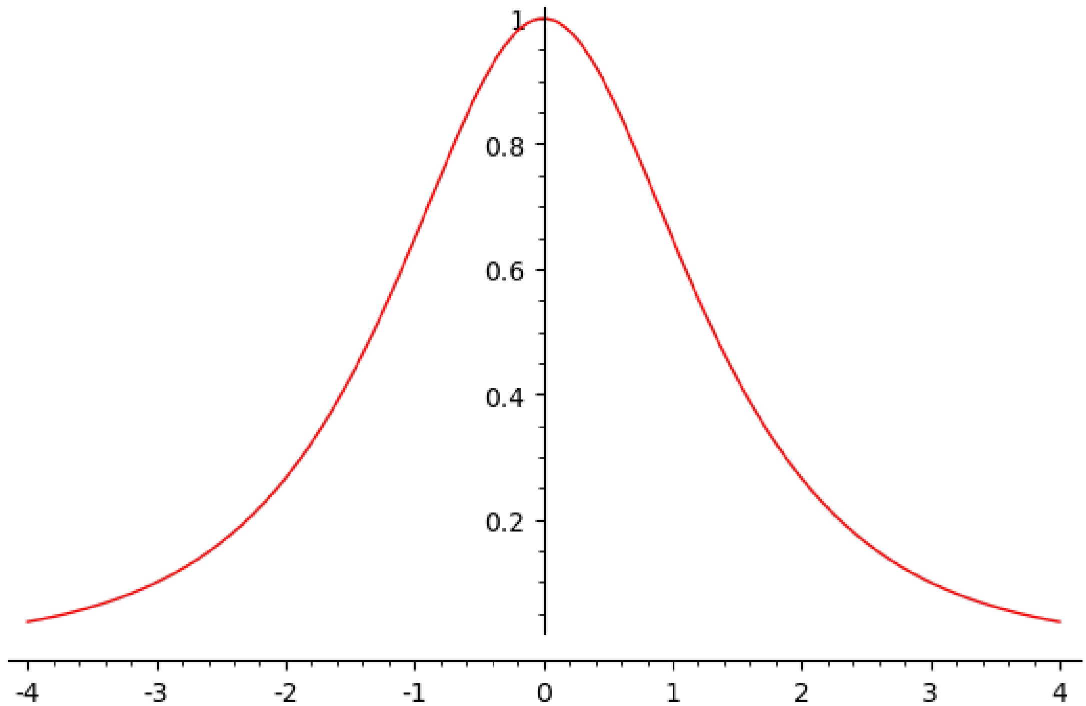
[2]:

```
p1  # We can then display the named plot just by typing its name...
```

[2]:



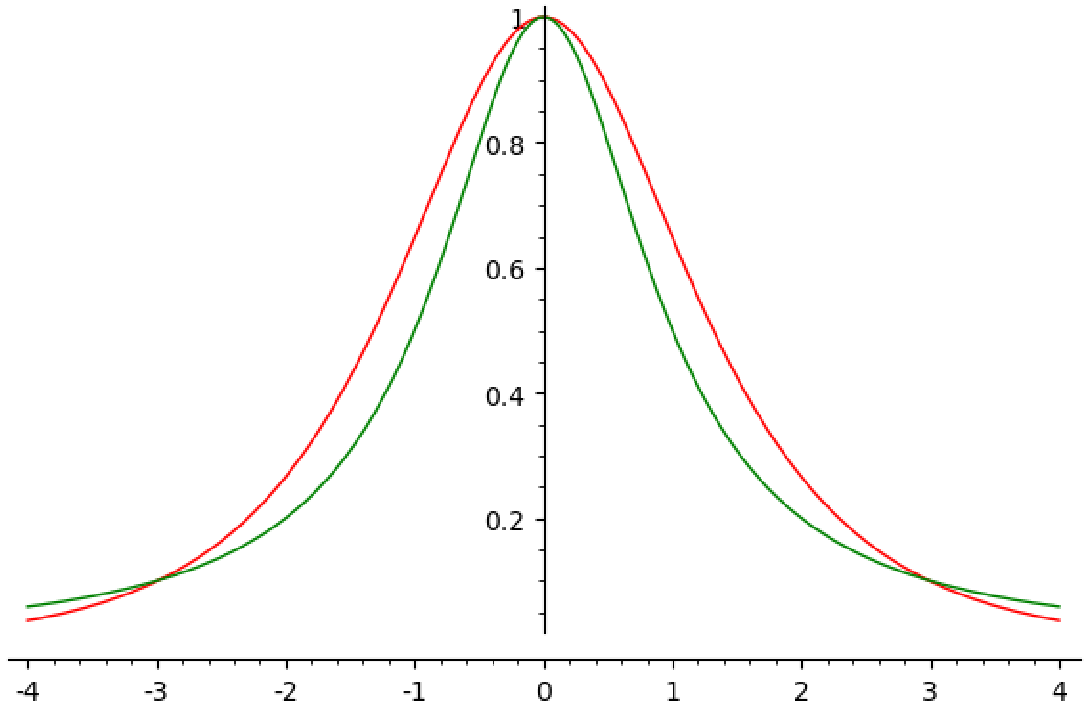
[3]:
```
show(p1)  # ... or by using the show command.
```



[4]:
```
p2 = plot(1/(1+x^2),-4,4,color='green')  # Another named plot with
                                         # another colour.
```

[5]:
```
p1 + p2  # We can display the two plots simultaneously simply by
         # adding them.
```

[5]:



[6]:
```
lim(sech(x),x=-oo)  # We can compute limits in SageMath using the
                    # lim command.  Note that one has to specify
                    # with respect to which variable the limit is
                    # taken as well as using oo for infinity.
```

[6]: 0

[7]:
```
lim(sech(x),x=56)   # One can,  of course,  also take limits at a
                    # point.
```

[7]: 2*e^56/(e^112 + 1)

[8]:
```
N(2*e^56/(e^112 + 1))  # As a small bonus,  if you want a decimal
                       # approximation to the exact answer,  the N
                       # command will do that for you.
```

[8]: 9.56178576777094e-25

[10]:
```
solve(x^2 == 30,x)  # The solve command lets you solve equations.
                    # Note that you have to specify which variable
                    # to solve for, even if there is only one...
```

[10]: [x == -sqrt(30), x == sqrt(30)]

[11]:
```
solve(x^2 + 1 == x, x)  # Solve will not hesitate to give you
                        # complex solutions.  It represents the
                        # square root of -1 by I when the solution
                        # is a complex number.
```

[11]: [x == -1/2*I*sqrt(3) + 1/2, x == 1/2*I*sqrt(3) + 1/2]

[12]:
```
solve(sqrt(x) + 1 == x, x)  # One weakness of the solve command is
                            # that it tends to give you a lazy and
                            # useless solution when the equation is
                            # not a polynomial one and x is easy to
                            # isolate.
```

[12]: [x == sqrt(x) + 1]

[13]:
```
solve(sqrt(x) + 1 - x == 0, x)  # Small rearrangements won't fix the
                                # problem...
```

[13]: [x == sqrt(x) + 1]

[15]:
```
solve( x == (x-1)^2, x )  # ... but doing some preliminary work by
                          # hand to eliminate the square root and
                          # recast the equation as a polynomial
                          # equation will do the job.
```

[15]: [x == -1/2*sqrt(5) + 3/2, x == 1/2*sqrt(5) + 3/2]

[16]:
```
var("y")  # We'll need another variable for what is to follow.
solve(y == sech(x), y)  # Asking for the solution y to y = sech(x)
                        # is kind of redundant...
```

[16]: [y == sech(x)]

[17]:
```
solve( x == sech(y), y)  # ... but asking for the solution y to
                         # x = sech(y) solves for y the as the
                         # inverse function, arcsech, to sech.
```

[17]: [y == arcsech(x)]

[18]:
```
solve( x == 1/(e^y + e^(-y)), y)  # If you actually want a formula
                                  # for arcsech,  you ask SageMath
                                  # to solve x = sech(y) using the
                                  # formula for sech.  Note that you
                                  # two possible answers - each is
                                  # the inverse of a different part
                                  # of sech.
```

[18]: [y == log(-1/2*sqrt(-4*x^2 + 1)/x + 1/2/x), y == log(1/2*sqrt(-4*x^2 + 1)/x +
1/2/x)]

[19]:
```
solve( sin(x) == 1,  x)  # One last example, using the solve command
                         # to find where sin(x) = 1.  Note that it
                         # one of the infinitely many possible values
                         # of x.
```

[19]: [x == 1/2*pi]

[ ]:
```
# That's all,  folks!
```