We begin with another optimization example. What is the shortest distance from the origin to the line given by the equation $y = 2x + 1$? Let us first solve this problem without using calculus. We will then solve it using calculus.

By elementary geometry, the shortest path from the origin to the line given by $y = 2x + 1$ makes the angle of 90 degrees with the line $y = 2x + 1$. It follows that the equation of this path is $y = -\frac{x}{2}$. The intersection of the line $y = 2x + 1$ and $y = -\frac{x}{2}$ is given by $2x + 1 = -\frac{x}{2}$, so $\frac{5}{2}x = -1$, yielding $x = -\frac{2}{5}$ and $y = \frac{1}{5}$. The shortest distance is thus

$$\sqrt{\frac{4}{25} + \frac{1}{25}} = \frac{1}{\sqrt{5}}.$$

Let us now solve this problem using calculus. Every point on the line $y = 2x + 1$ can be written in the form $(x, 2x + 1)$. The square of the distance from the origin to a given point on the line given by the equation $y = 2x + 1$ is equal to

$$f(x) = x^2 + (2x+1)^2 = 5x^2 + 4x + 1.$$

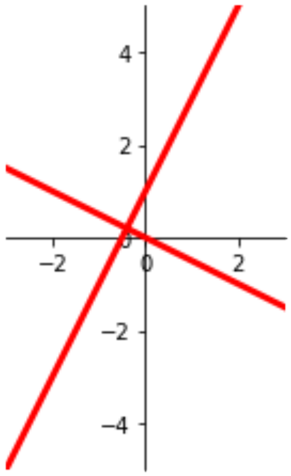
We have

$$f'(x) = 10x + 4.$$

Setting it equal to 0 yields a critical point at $x = -\frac{2}{5}$. By looking at f' to the left and to the right of this point we see that it is a local minimum. By plugging in the endpoints (±∞) we see that it is a global minimum as well. This yields the same answer as above.

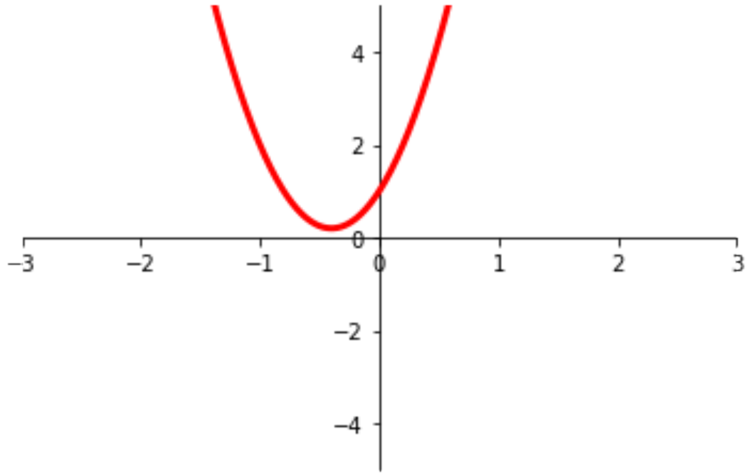
Take a look at the diagram below. One of the lines is the line $y = 2x + 1$, while the other is the line $y = -\frac{x}{2}$.

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import sympify, lambdify
from sympy.abc import x
import warnings; warnings.simplefilter('ignore')

fig = plt.figure(1)
ax = fig.add_subplot(111)


# set up axis
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# setup x and y ranges and precision
xx = np.arange(-3,3,0.01)

# draw my curve
myfunction=sympify(2*x+1)
myfunction2=sympify(-x/2)
mylambdifiedfunction=lambdify(x,myfunction,'numpy')
mylambdifiedfunction2=lambdify(x,myfunction2,'numpy')
ax.plot(xx, mylambdifiedfunction(xx),zorder=100,linewidth=3,color='red
')
ax.plot(xx, mylambdifiedfunction2(xx),zorder=100,linewidth=3,color='re
d')
plt.axes().set_aspect('equal')

#set bounds
ax.set_xbound(-3,3)
ax.set_ybound(-5,5)

plt.show()
```



In [ ]: Let us also plot the function we minimized above, namely $f(x)=5x^2+4x+1$.

In [10]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import sympify, lambdify
from sympy.abc import x
import warnings; warnings.simplefilter('ignore')

fig = plt.figure(1)
ax = fig.add_subplot(111)


# set up axis
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# setup x and y ranges and precision
xx = np.arange(-3,3,0.01)

# draw my curve
myfunction=sympify(5*x**2+4*x+1)
#myfunction2=sympify(-x/2)
mylambdifiedfunction=lambdify(x,myfunction,'numpy')
#mylambdifiedfunction2=lambdify(x,myfunction2,'numpy')
ax.plot(xx, mylambdifiedfunction(xx),zorder=100,linewidth=3,color='red
')
#ax.plot(xx, mylambdifiedfunction2(xx),zorder=100,linewidth=3,color='r
ed')
#plt.axes().set_aspect('equal')

#set bounds
ax.set_xbound(-3,3)
ax.set_ybound(-5,5)

plt.show()
```



Our next problem is a bit harder and requires us to pay close attention to the geometry involved.

A man launches his boat from point A on a bank of a straight river, 3 km wide, and wants to reach point B, 8 km downstream on the opposite bank, as quickly as possible. He could row his boat directly across the river to point C and then run to B, or he could row directly to B, or he could row to some point D between C and B and then run to B. If he can row 6 km/h and run 8 km/h, where should he land to reach B as soon as possible? (We assume that the speed of the water is negligible compared with the speed at which the man rows.)

Let x be the distance from C to D. Then the distance from A to D equals

$$\sqrt{x^2+9}$$

and the distance from D to B is $8-x$.

Let $T(x)$ denote the time it takes to reach B from A. Then

$$T(x)=\frac{\sqrt{x^2+9}}{6}+\frac{8-x}{8}.$$

Then

$$T'(x)=\frac{x}{6\sqrt{x^2+9}}-\frac{1}{8}$$

Setting $T'(x)$ equal to 0 and solving we find a critical point at $x = \frac{9}{\sqrt{7}}$. Looking at $T'(x)$ to the left and to the right of $\frac{9}{\sqrt{7}}$, we see that it is a local minimum.

To see that $x = \frac{9}{\sqrt{7}}$ gives us a global minimum, we also consider the endpoints $x = 0$ and $x = 8$. If $x = 0$,

$$T(0) = \frac{3}{2}.$$

If $x = 8$, we get

$$T(8) = \frac{\sqrt{73}}{6}.$$

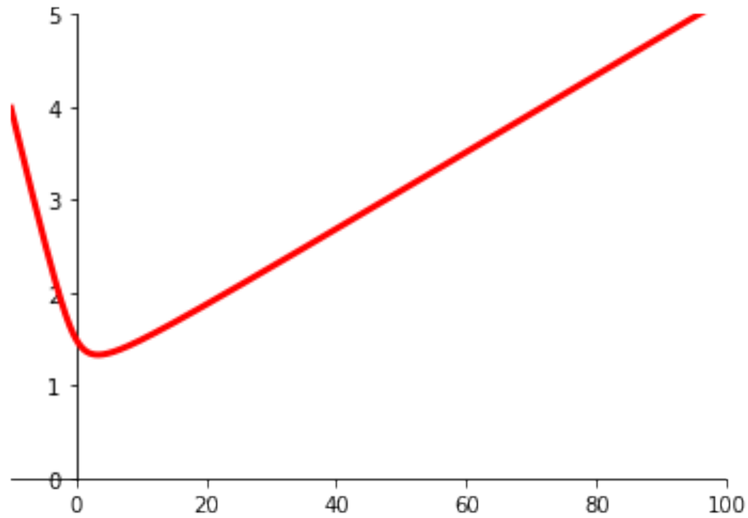
We compare this with $x = \frac{9}{\sqrt{7}}$, so

$$T\left(\frac{9}{\sqrt{7}}\right) = 1 + \frac{\sqrt{7}}{8},$$

which is the smallest value by a direct comparison.

In [ ]:
```
Let us now confirm what we computed by drawing the graph.
```

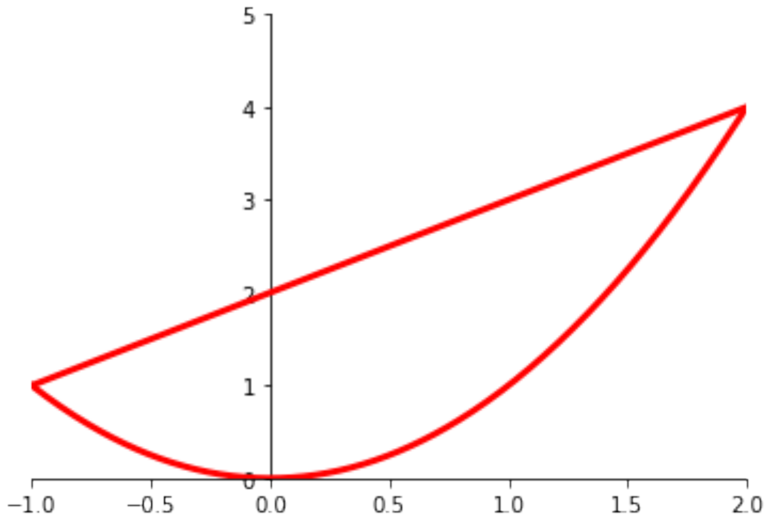
In [18]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import sympify, lambdify
from sympy.abc import x
import warnings; warnings.simplefilter('ignore')

fig = plt.figure(1)
ax = fig.add_subplot(111)


# set up axis
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# setup x and y ranges and precision
xx = np.arange(-10,100,.01)

# draw my curve
myfunction=sympify((x**2+9)**(1/2)/6+(8-x)/8)
#myfunction2=sympify(-x/2)
mylambdifiedfunction=lambdify(x,myfunction,'numpy')
#mylambdifiedfunction2=lambdify(x,myfunction2,'numpy')
ax.plot(xx, mylambdifiedfunction(xx),zorder=100,linewidth=3,color='red')
#ax.plot(xx, mylambdifiedfunction2(xx),zorder=100,linewidth=3,color='red')
#plt.axes().set_aspect('equal')

#set bounds
ax.set_xbound(-10,100)
ax.set_ybound(0,5)

plt.show()
```



Let us now go back to geometric shapes in the plane. What is the maximum vertical distance between the line $y = x + 2$ and the parabola $y = x^2$ for $-1 \leq x \leq 2$?

Let us first see what the picture looks like:

In [22]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import sympify, lambdify
from sympy.abc import x
import warnings; warnings.simplefilter('ignore')

fig = plt.figure(1)
ax = fig.add_subplot(111)


# set up axis
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# setup x and y ranges and precision
xx = np.arange(-1,2,.01)

# draw my curve
myfunction=sympify(x+2)
myfunctionalt=sympify(x**2)
#myfunction2=sympify(-x/2)
mylambdifiedfunction=lambdify(x,myfunction,'numpy')
mylambdifiedfunctionalt=lambdify(x,myfunctionalt,'numpy')

#mylambdifiedfunction2=lambdify(x,myfunction2,'numpy')
ax.plot(xx, mylambdifiedfunction(xx),zorder=100,linewidth=3,color='red')
ax.plot(xx, mylambdifiedfunctionalt(xx),zorder=100,linewidth=3,color='red')

#ax.plot(xx, mylambdifiedfunction2(xx),zorder=100,linewidth=3,color='red')
#plt.axes().set_aspect('equal')

#set bounds
ax.set_xbound(-1,2)
ax.set_ybound(0,5)

plt.show()
```



To solve this problem, we note that the line $y = x + 2$ is always above the parabola $y = x^2$ on the interval $[-1, 2]$. It follows that the vertical distance equals

$$f(x) = x + 2 - x^2.$$

Then

$$f'(x) = 1 - 2x$$

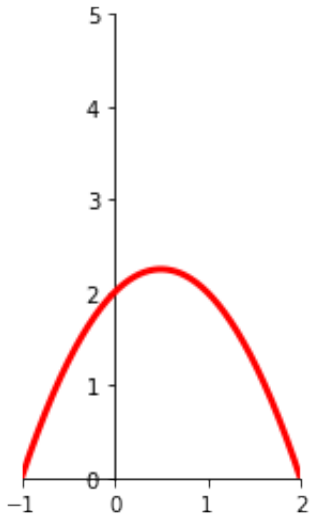
and setting it equal to 0 we obtain $x = \frac{1}{2}$. Checking to the left and to the right of $x = \frac{1}{2}$ we see that it is a local maximum. Since

$$f(-1) = f(2) = 0,$$

so $x = \frac{1}{2}$ is a global maximum.

Just for practice and to be extra sure, let's graph the function we just maximized:

In [23]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import sympify, lambdify
from sympy.abc import x
import warnings; warnings.simplefilter('ignore')

fig = plt.figure(1)
ax = fig.add_subplot(111)


# set up axis
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# setup x and y ranges and precision
xx = np.arange(-1,2,.01)

# draw my curve
myfunction=sympify(x+2-x**2)
#myfunctionalt=sympify(x**2)
#myfunction2=sympify(-x/2)
mylambdifiedfunction=lambdify(x,myfunction,'numpy')
#mylambdifiedfunctionalt=lambdify(x,myfunctionalt,'numpy')

#mylambdifiedfunction2=lambdify(x,myfunction2,'numpy')
ax.plot(xx, mylambdifiedfunction(xx),zorder=100,linewidth=3,color='red
')
#ax.plot(xx, mylambdifiedfunctionalt(xx),zorder=100,linewidth=3,color=
'red')

#ax.plot(xx, mylambdifiedfunction2(xx),zorder=100,linewidth=3,color='r
ed')
plt.axes().set_aspect('equal')

#set bounds
ax.set_xbound(-1,2)
ax.set_ybound(0,5)

plt.show()
```



It is very easy to believe from this picture that $x = \frac{1}{2}$ is in fact the global maximum. Let's go ahead and solve this problem without using calculus. We are maximizing the function

$$f(x) = x + 2 - x^2$$

on the interval $[-1, 2]$. Completing the square, we get

$$f(x) = -(x^2 - x - 2) = -\left(\left(x - \frac{1}{2}\right)^2 - \frac{1}{4} - 2\right)$$

$$= -\left(x - \frac{1}{2}\right)^2 + \frac{9}{4}.$$

It follows that the maximum is $\frac{9}{4}$ and it takes place at $x = \frac{1}{2}$, as we established above using calculus.

```
In [ ]:
```