Graphical analysis

Introduction to Numerical Problem Solving

 Spring 2017, Sakari Lukkarinen

Helsinki Metropolia University of Applied Sciences

Initial commands

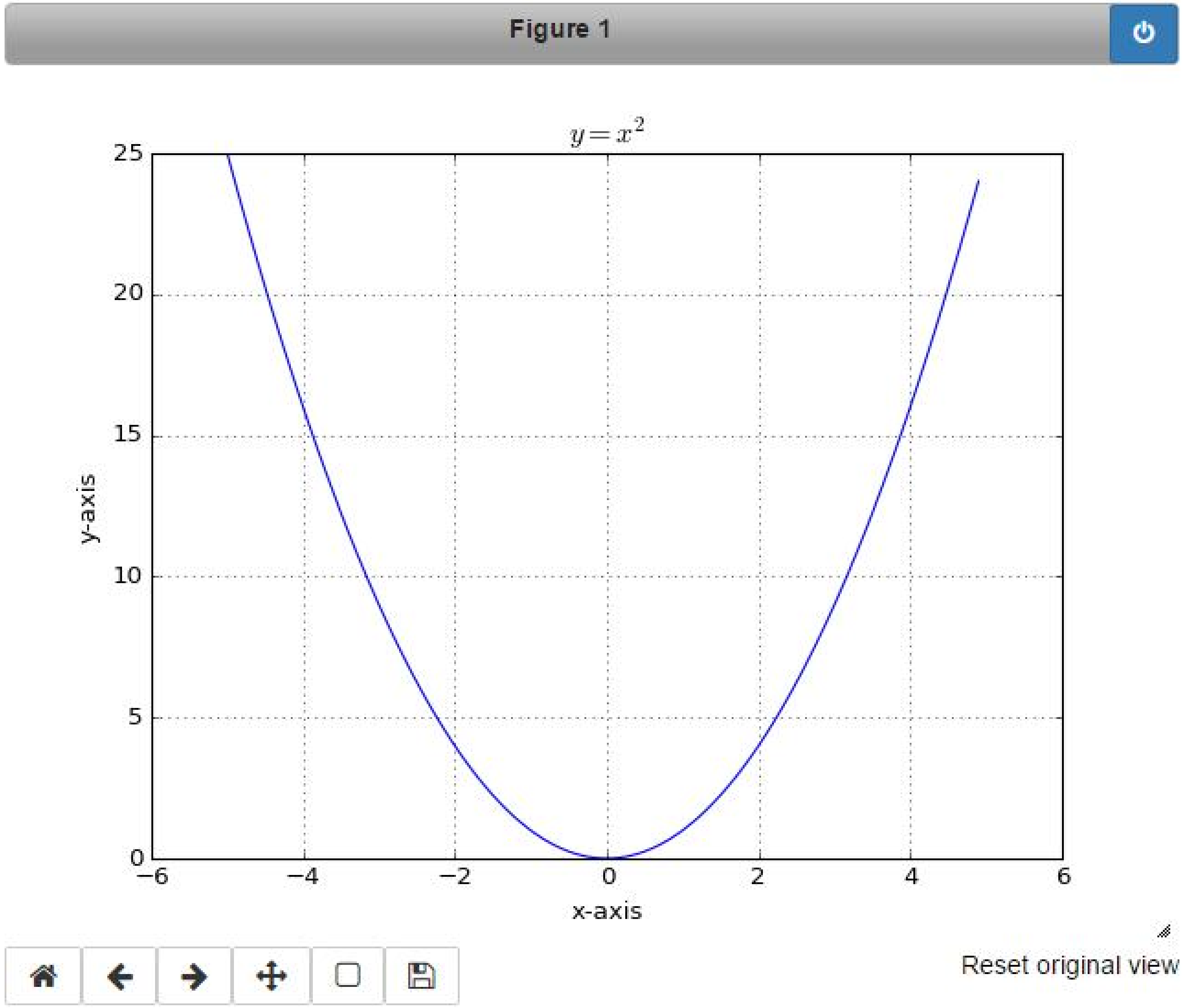
```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt
        import numpy as np
        from matplotlib.pyplot import *
        from numpy import *
```

When you need to freeze (noninteractive) graphics, for example, for creating documents (HTML, PDF, ...), then use

```
In [ ]: %matplotlib inline
```

First graph

In [5]:
```
x = arange(-5, 5, 0.1)
y = x**2
figure()
plot(x, y)
xlabel("x-axis")
ylabel("y-axis")
title("$y = x^2$")
grid()
```



Plot options

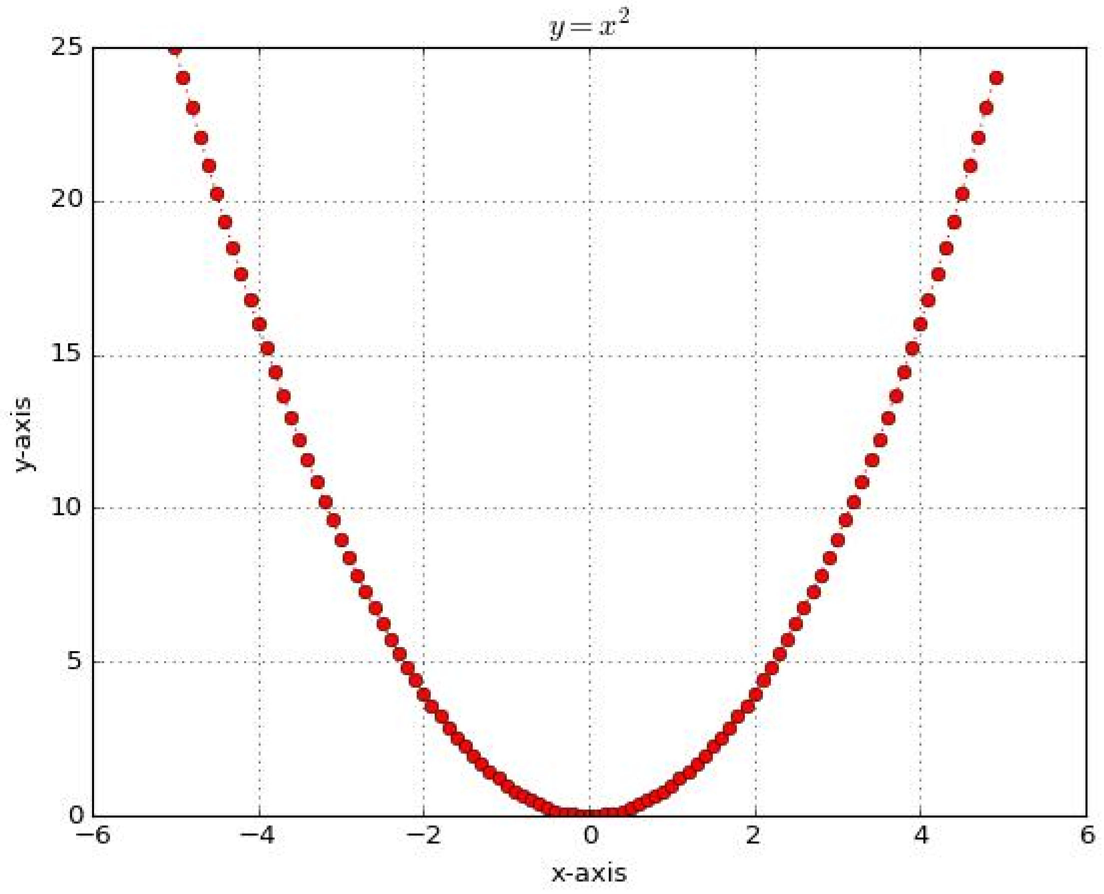
```
In [7]: ?plot
```

```
Signature: plot(*args, **kwargs)
Docstring:
Plot lines and/or markers to the
:class:`~matplotlib.axes.Axes`.  *args* is a variable length
argument, allowing for multiple *x*, *y* pairs with an
optional format string.  For example, each of the following is
legal::

    plot(x, y)        # plot x and y using default line style and color
    plot(x, y, 'bo')  # plot x and y using blue circle markers
    plot(y)           # plot y using x as index array 0..N-1
    plot(y, 'r+')     # ditto, but with red plusses

If *x* and/or *y* is 2-dimensional, then the corresponding columns
will be plotted.

If used with labeled data, make sure that the color spec is not
included as an element in data, as otherwise the last case
``plot("v","r", data={"v":..., "r":...)``
can be interpreted as the first case which would do ``plot(v, r)``
using the default line style and color.

If not used with labeled data (i.e., without a data argument),
an arbitrary number of *x*, *y*, *fmt* groups can be specified, as in::
```

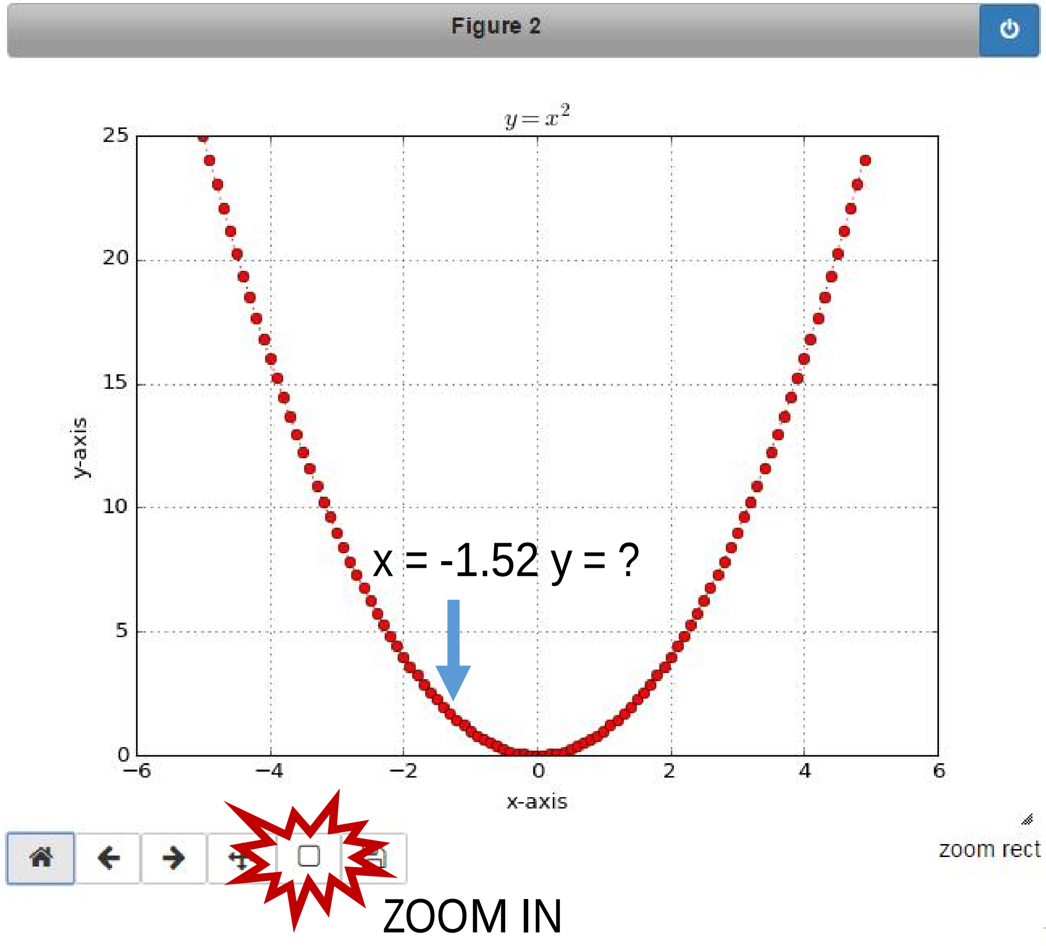
```
================    ===============================
character           description
================    ===============================
``'-'``             solid line style
``'--'``            dashed line style
``'-.'``            dash-dot line style
``':'``             dotted line style
``'.'``             point marker
``','``             pixel marker
``'o'``             circle marker
``'v'``             triangle_down marker
``'^'``             triangle_up marker
``'<'``             triangle_left marker
``'>'``             triangle_right marker
``'1'``             tri_down marker
``'2'``             tri_up marker
``'3'``             tri_left marker
``'4'``             tri_right marker
``'s'``             square marker
``'p'``             pentagon marker
``'*'``             star marker
``'h'``             hexagon1 marker
``'H'``             hexagon2 marker
``'+'``             plus marker
``'x'``             x marker
``'D'``             diamond marker
``'d'``             thin_diamond marker
``'|'``             vline marker
``'_'``             hline marker
================    ===============================
```

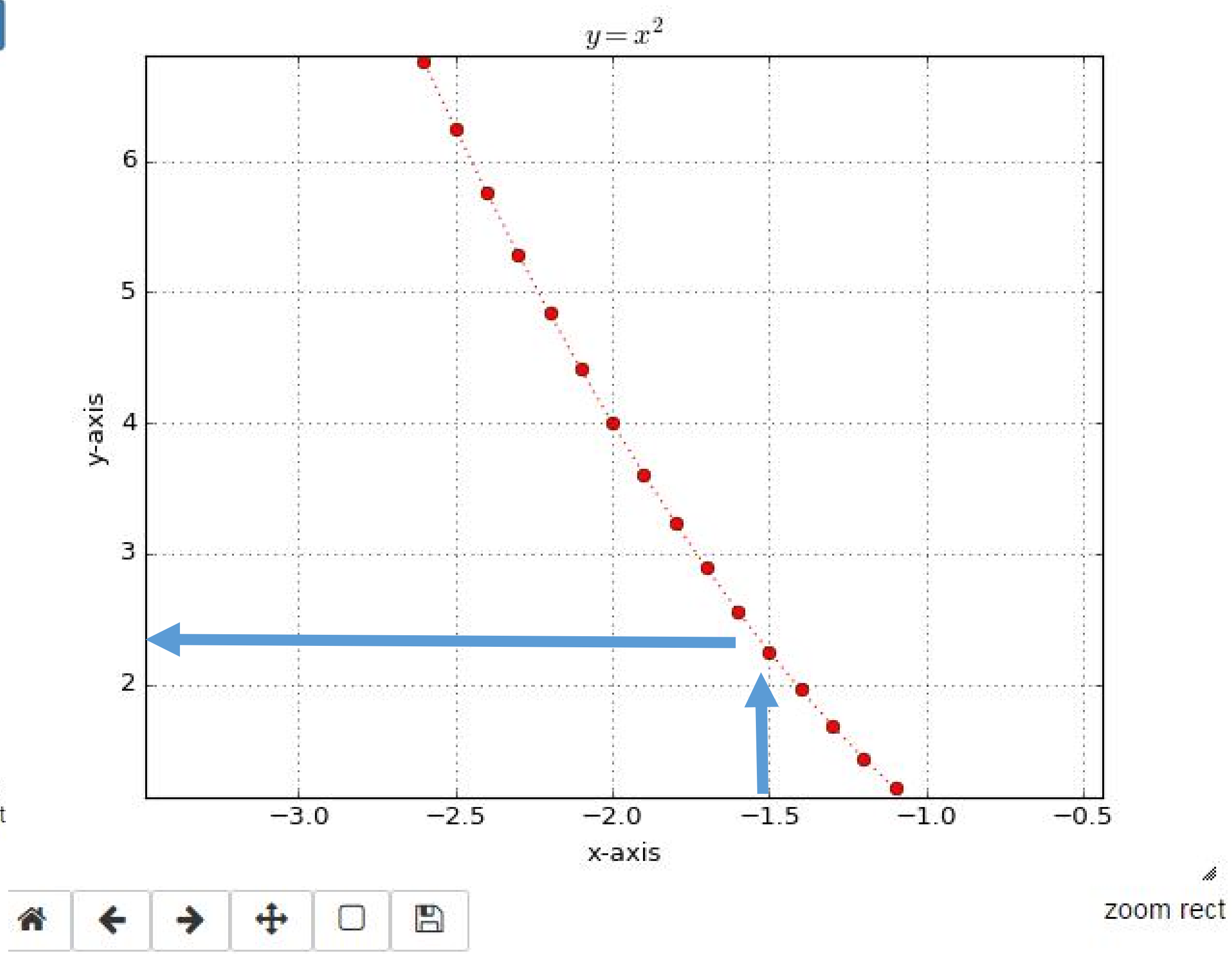
Second graph

```
In [8]: x = arange(-5, 5, 0.1)
        y = x**2
        figure()
        plot(x, y, 'ro:')
        xlabel("x-axis")
        ylabel("y-axis")
        title("$y = x^2$")
        grid()
```



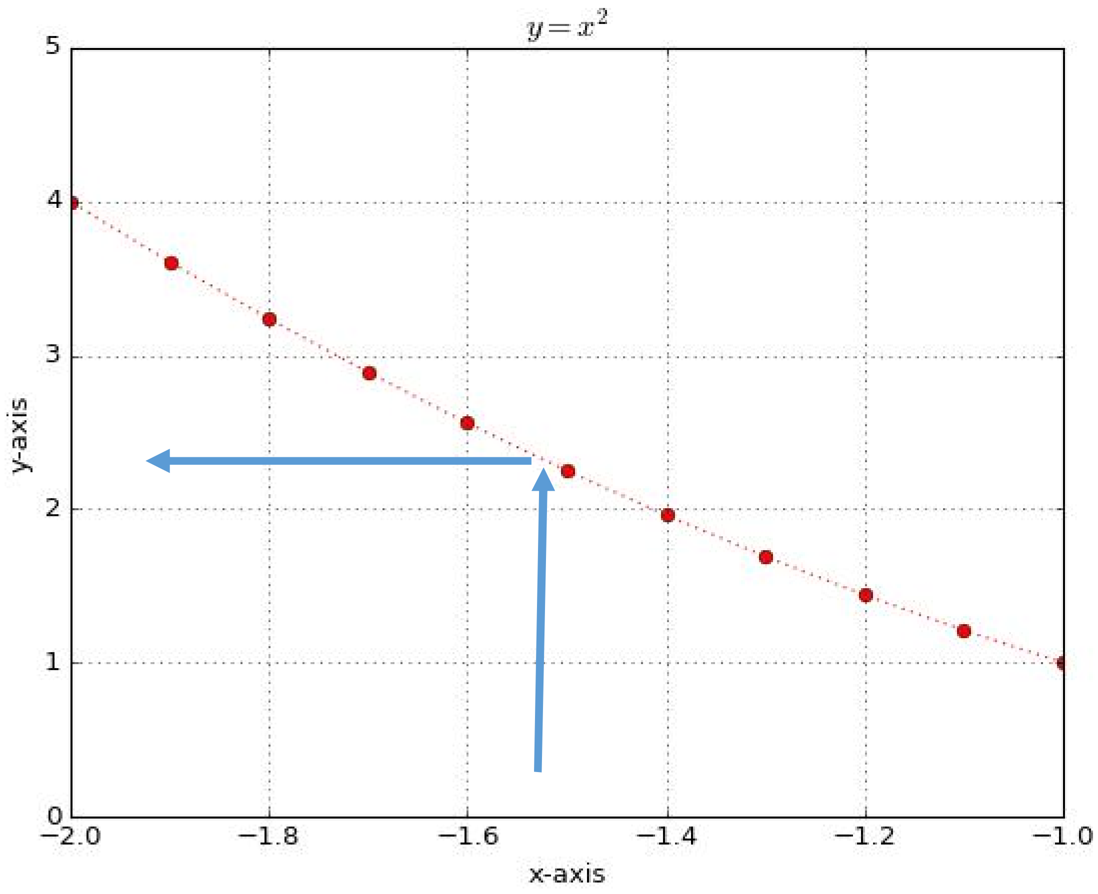
What is the value at x = -1.52?





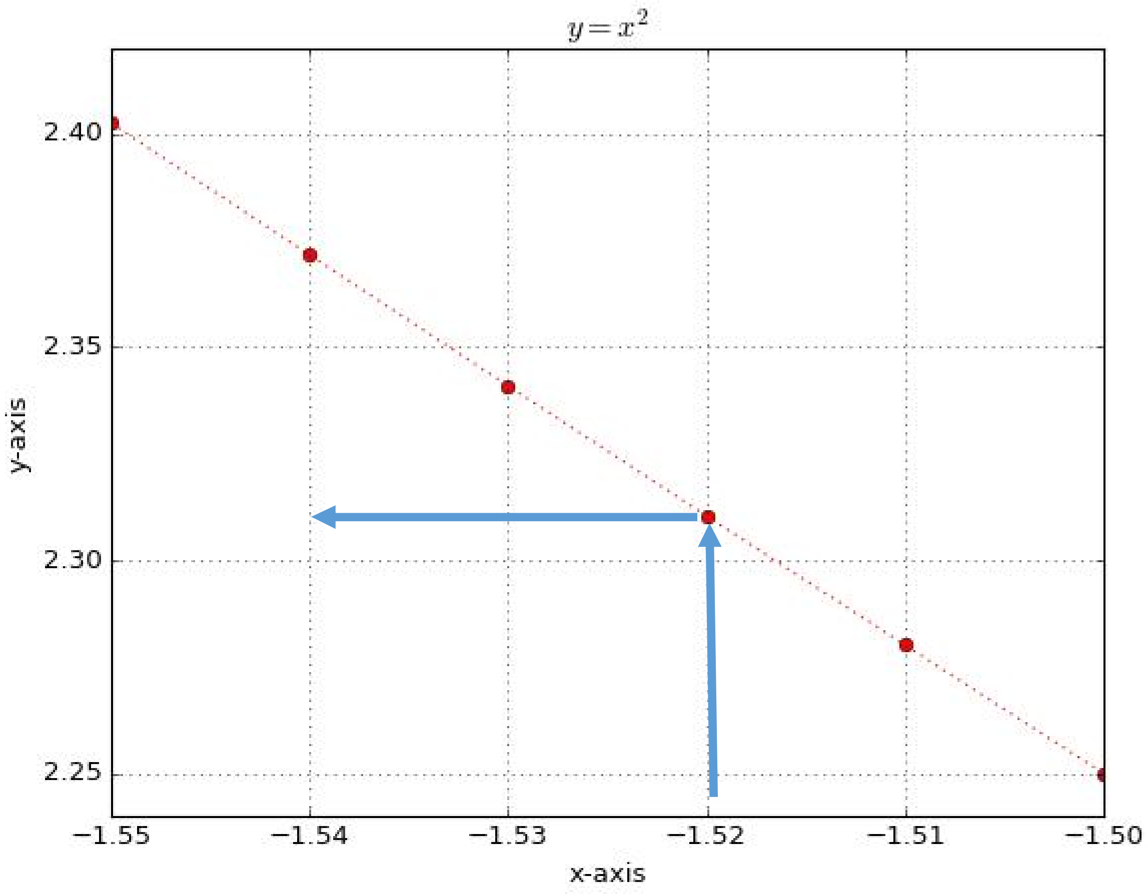
Another way – code way

In [13]:
```
x = arange(-5, 5, 0.1)
y = x**2
figure()
plot(x, y, 'ro:')
xlim([-2.0, -1.0])
ylim([0.0, 5.0])
xlabel("x-axis")
ylabel("y-axis")
title("$y = x^2$")
grid()
```



Better way – limit the values and add more points

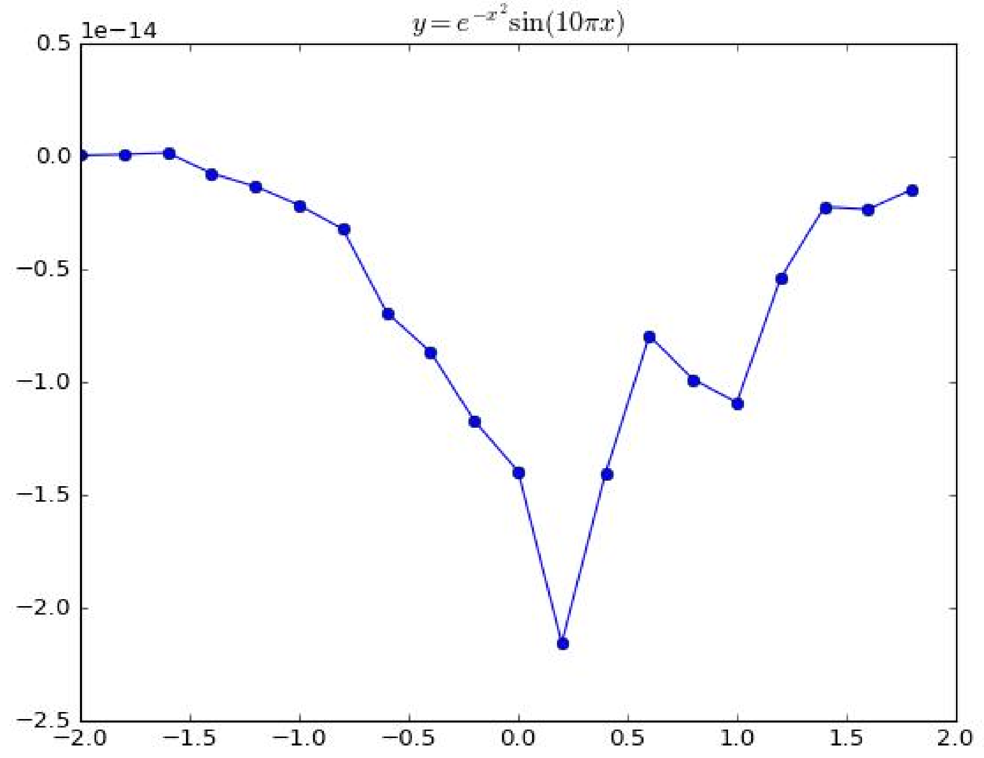
In [16]:
```
x = arange(-1.55, -1.50, 0.01)
y = x**2
figure()
plot(x, y, 'ro:')
xlabel("x-axis")
ylabel("y-axis")
title("$y = x^2$")
grid()
```



How much points do we need? Example

$$y = e^{-x^2} \sin(10\pi x)$$

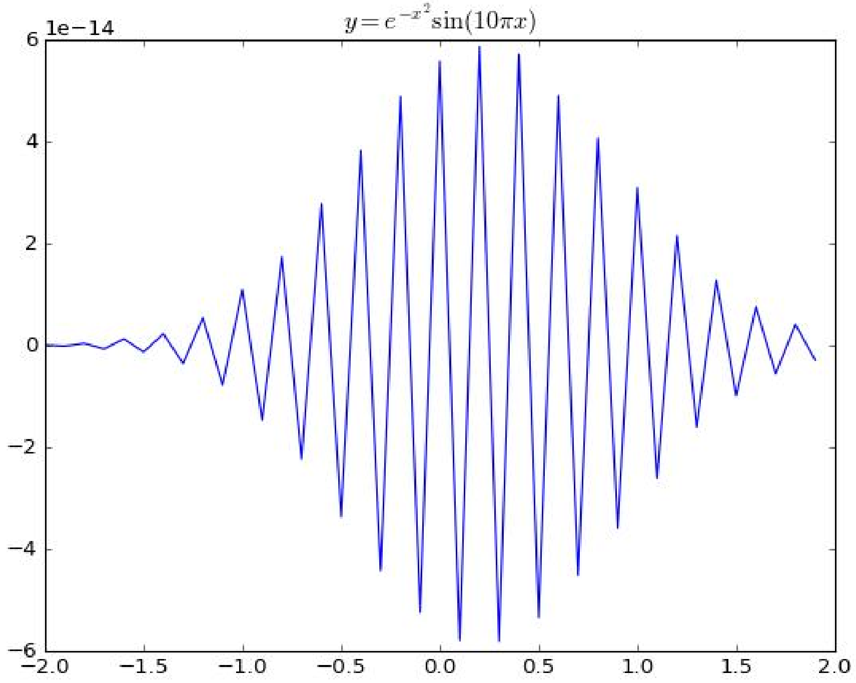
```
In [27]: x = arange(-2.0, 2.0, 0.2)
         y = exp(-x**2)*sin(10.0*pi*x)
         figure()
         plot(x, y, 'o-')
         title('$y = e^{-x^2} \sin(10\pi x)$')
```



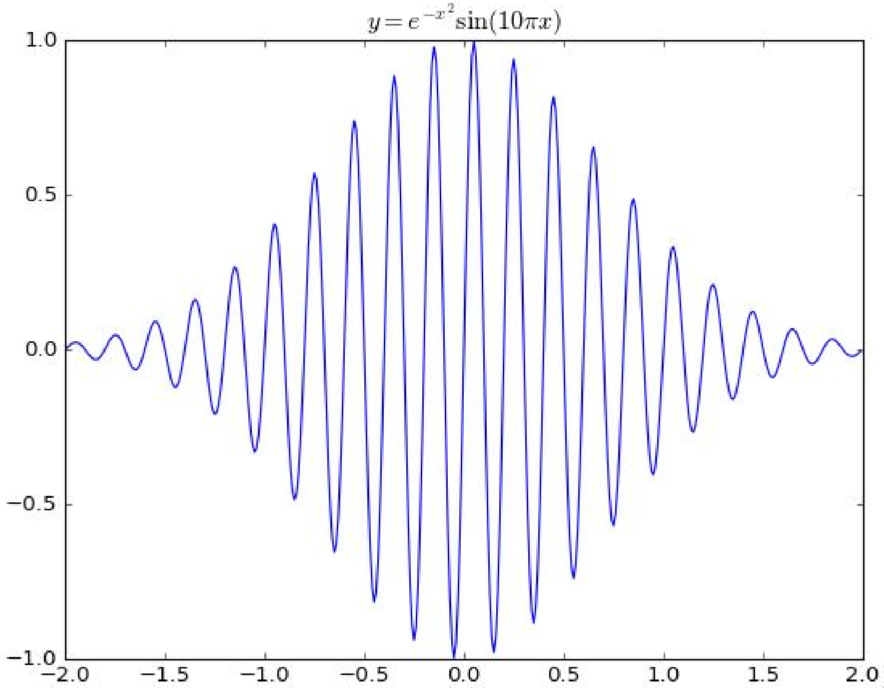
Does it look like an exponentially decaying sinusoidal???

More points – more details and smoother graph

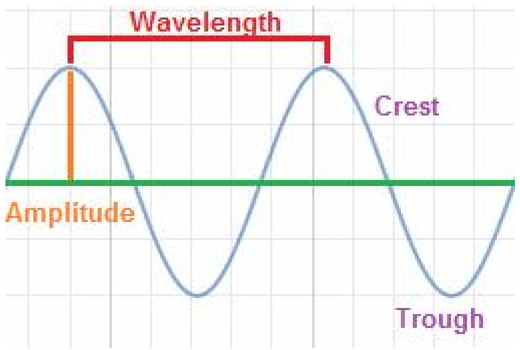
In [25]:
```
x = arange(-2.0, 2.0, 0.1)
y = exp(-x**2)*sin(10.0*pi*x)
figure()
plot(x, y)
title('$y = e^{-x^2} \sin(10\pi x)$')
```



In [26]:
```
x = arange(-2.0, 2.0, 0.01)
y = exp(-x**2)*sin(10.0*pi*x)
figure()
plot(x, y)
title('$y = e^{-x^2} \sin(10\pi x)$')
```



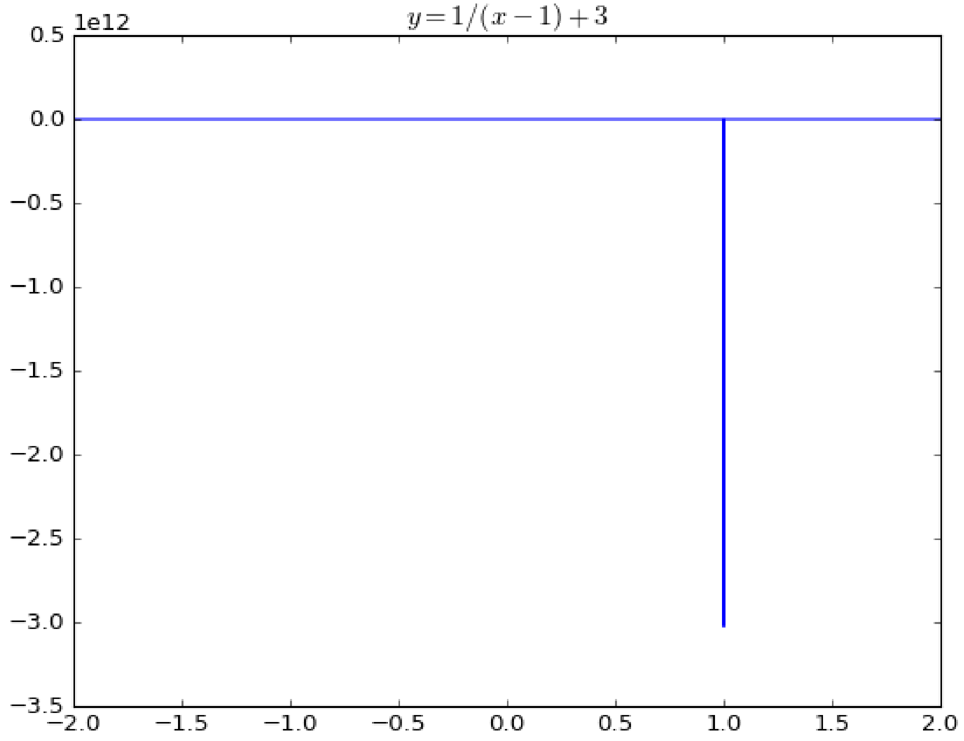
Tips for the details



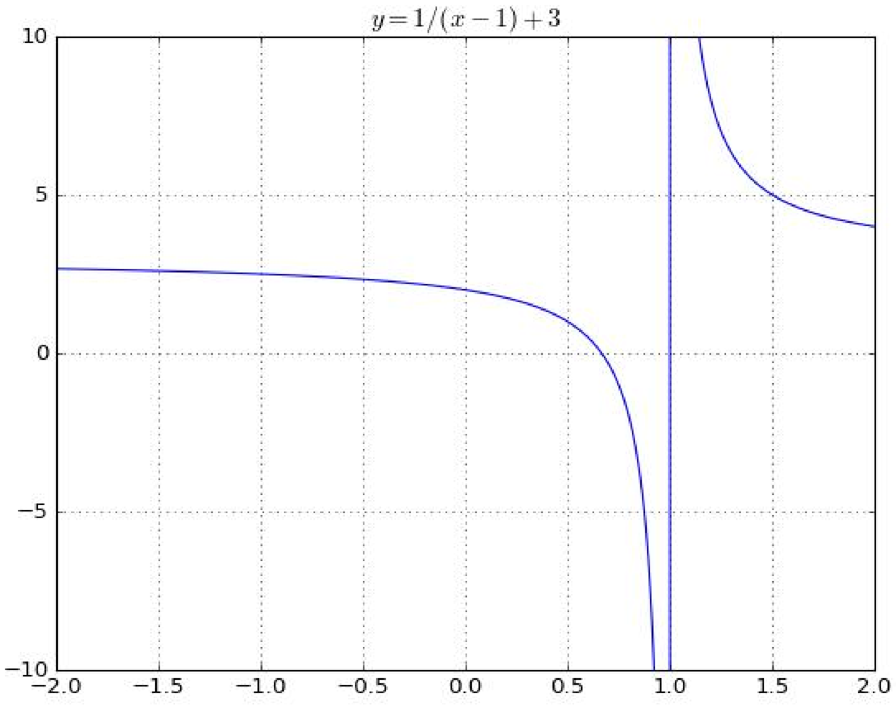
- Straight lines: two points are enough!
- Polynomials: higher polynomial, more details
- Oscillations, like sine-waves: use at least 10 points per wavelength
  - Sin(2*pi*x) è wavelength = 1
  - Sin(10*pi*x) è wavelength = 1/5 = 0.2
- Sudden changes (increase/decrease) in graph: more details
- Smooth curves: less details
- Infinities in function: limit the y-axis (ylim)

Infinities in the graphs

In [35]:
```
x = arange(-2.0, 2.0, 0.001)
y = 1/(x-1.0) + 3.0
figure()
plot(x, y)
title("$y = 1/(x - 1) + 3$")
```

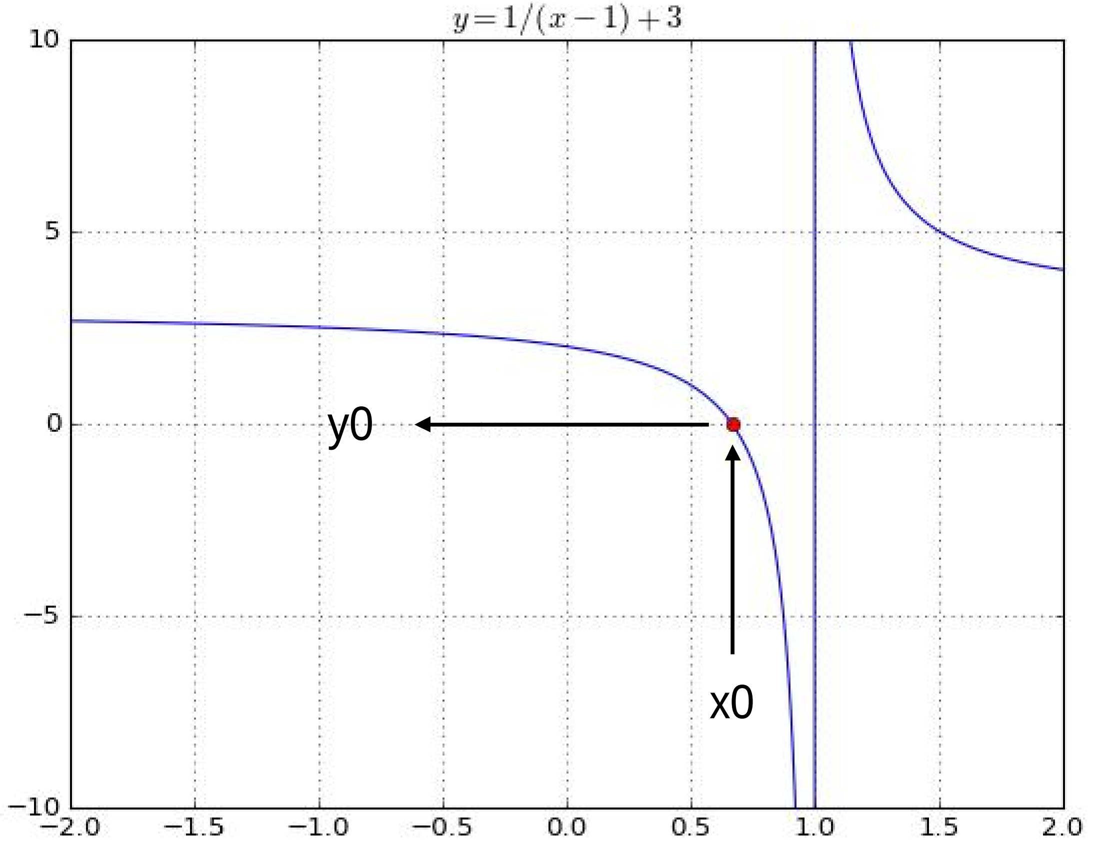


In [38]:
```
x = arange(-2.0, 2.0, 0.001)
y = 1/(x-1.0) + 3.0
figure()
plot(x, y)
ylim([-10, 10])
title("$y = 1/(x - 1) + 3$")
grid()
```



Annotating the graph – Marking the roots

```
In [44]: x = arange(-2.0, 2.0, 0.001)
         y = 1/(x-1.0) + 3.0

         # The root of the function
         x0 = 0.667
         y0 = 1/(x0-1.0) + 3.0

         figure()
         plot(x, y)
         # overlay the root
         plot(x0, y0, 'ro')
         ylim([-10, 10])
         title("$y = 1/(x - 1) + 3$")
         grid()
```



More information

- Pyplot tutorial - http://matplotlib.org/users/pyplot_tutorial.html
- Matplotlib Crash course - https://pythonprogramming.net/matplotlib-python-3-basics-tutorial/
- What is asymptote? https://en.wikipedia.org/wiki/Asymptote
- Maxima and minima - https://en.wikipedia.org/wiki/Maxima_and_minima
- Zero (=root) of function - https://en.wikipedia.org/wiki/Zero_of_a_function