Example 96. Python The following code measures how well a function f is approximated by the polynomial interpolating f at the given points. It returns an approximation of the maximal error on the interval $[a,b]$.

```
>>> from numpy import linspace, pi, cos, sin

>>> from scipy import interpolate

>>> def max_interpolation_error(f, a, b, xpoints, nr_sample_points):
        ypoints = [f(x) for x in xpoints]
        poly = interpolate.lagrange(xpoints, ypoints)
        max_error = max([abs(f(x)-poly(x)) for x in linspace(a,b,nr_sample_points)])
        return max_error
```

Let us verify that this works using an example we have discussed before:

```
>>> max_interpolation_error(sin, 0, pi, [0,pi/2,pi], 100)

    0.0560067197786
```

This agrees with the maximal error that we observed at the end of Example 93. Let us look how the error develops as we add more points:

```
>>> [max_interpolation_error(sin, 0, pi, linspace(0,pi,n), 100) for n in range(2,9)]

    [0.9998741276738749 6, 0.056006719778558423, 0.043613266903306247,
    0.0018097268033398783, 0.0013114413108160916, 3.385907546618605e-05,
    2.4246231325325551e-05]
```

It is pleasing to see that the error decreases. However, as we will see in the next example, this does not have to be the case.

Comment. Note that the error seems to really decrease every second step (i.e. after adding two more points). Can you offer an explanation for what might be the cause of this?

Example 97. Python However, this is not the end of the story. It turns out that the interpolation error does not always go down if we add additional points.

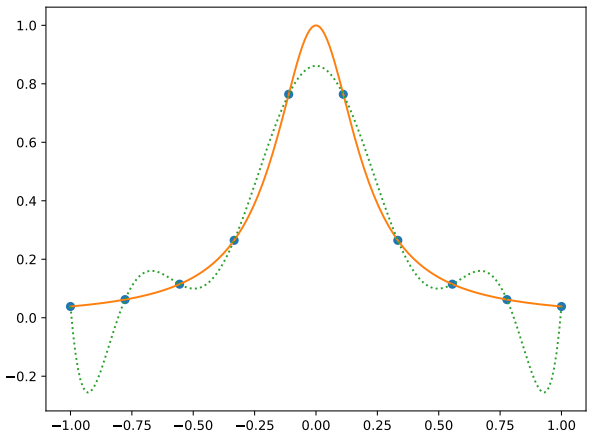
```
>>> def f(x):
        return 1/(1+25*x**2)

>>> [max_interpolation_error(f, -1, 1, linspace(-1,1,n), 100) for n in range(2,18)]

    [0.9589941912351845, 0.6459699748665507, 0.7044952736346626, 0.4382728746134098,
    0.43032461596244886, 0.6164015686420344, 0.24528527039305037, 1.0450782163781276,
    0.297971540151836, 1.9154342696798625, 0.5538529081557272, 3.6117015978042333,
    1.064460371610917, 7.189298472061041, 2.0967229089912, 14.013534491466531]
```

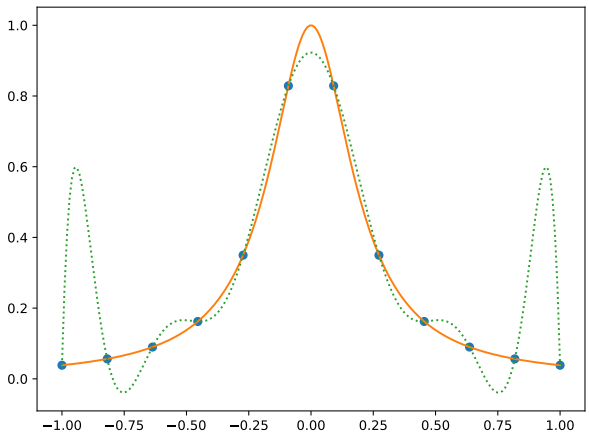
The function $f(x)=\frac{1}{1+25x^2}$ in this example is known as the **Runge function** and one can show that, by adding more points, the error grows without bound.

$$\lim_{n\to\infty} \max_{x\in[-,1,1]} |f(x)-P_n(x)| = \infty$$

https://en.wikipedia.org/wiki/Runge%27s_phenomenon

The following plots show the situation using 10 and 12 interpolation nodes.





While the approximation becomes better towards the center of the interval $[-1,1]$, the oscillations towards the ends of the interval become more violant (resulting in an increasing worst-case error).

Next, we will see that we can avoid this issue if we don't choose equally spaced points but carefully chosen ones called **Chebyshev nodes**.