Bachelor of Ecole Polytechnique
Computational Mathematics, year 2, semester 1
Lecturer: Lucas Gerin [(send mail) (mailto:lucas.gerin@polytechnique.edu](mailto:lucas.gerin@polytechnique.edu)

Symbolic computing 1: Proofs with SymPy

Table of contents


```
# execute this part to modify the css style
from IPython.core.display import HTML
def css_styling():
    styles = open("./style/custom2.css").read()
    return HTML(styles)
css_styling()
```

```
## loading python libraries

# necessary to display plots inline:
%matplotlib inline

# load the libraries
import matplotlib.pyplot as plt # 2D plotting library
import numpy as np              # package for scientific computing
from pylab import *

from math import *              # package for mathematics (pi, arctan, sqrt, factori
import sympy as sympy             # package for symbolic computation
from sympy import *
```

Introduction to SymPy

Using python library `SymPy` we can perform *exact* computations. For instance, run and compare the following scripts:

```
print('With Numpy: ')
print('root of two is '+str(np.sqrt(2))+'')
print('the square of (root of two) is '+str(np.sqrt(2)**2)+'')
print('---------')
print('With SymPy: ')
print('root of two is '+str(sympy.sqrt(2))+'')
print('the square of (root of two) is '+str(sympy.sqrt(2)**2)+'')
```

```
With Numpy:
root of two is 1.41421356237
the square of (root of two) is 2.0
---------
With SymPy:
root of two is sqrt(2)
the square of (root of two) is 2
```

One can expand or simplify expressions:

```
print('Simplification of algebraic expressions:')
print('the square root of 40 is '+str(sympy.sqrt(40))+'')
print('(root(3)+root(2))**20 is equal to '+str(expand((sympy.sqrt(3)+sympy.sqrt(2))*
#
print('----------------')
print('Simplification of symbolic expressions:')
var('x') # We declare a 'symbolic' variable
Expression=(x**2 - 2*x + 1)/(x-1)
print(str(Expression) + ' simplifies into '+str(simplify(Expression)))
```

```
Simplification of algebraic expressions:
the square root of 40 is 2*sqrt(10)
(root(3)+root(2))**20 is equal to 4517251249 + 1844160100*sqrt(6)
----------------
Simplification of symbolic expressions:
(x**2 - 2*x + 1)/(x - 1) simplifies into x - 1
```

With `Sympy` one can also obtain Taylor expansions of functions with `series` :

```
# Real variable
var('x')
Expression=cos(x)
print('Expansion of cos(x) at x=0: '+str(Expression.series(x,0)))

# integer variable
var('n',integer=True)
Expression=cos(1/n)
print('Expansion of cos(1/n) when n -> +oo: '+str(Expression.series(n,oo)))   # oo m
```

```
Expansion of cos(x) at x=0: 1 - x**2/2 + x**4/24 + O(x**6)
Expansion of cos(1/n) when n -> +oo: 1/(24*n**4) - 1/(2*n**2) + 1 + O(n
**(-6), (n, oo))
```

`SymPy` can also compute with "big O's". (By default $\mathcal{O}(x)$ is considered for $x \to 0$.)

```
var('x')
simplify((x+O(x**3))*(x+x**2+O(x**3)))
```

```
x**2 + x**3 + O(x**4)
```

> Remark. A nice feature of `Sympy` is that you can export formulas in LateX . For instance:

```
var('x y')
formula=simplify((cos(x+y)-sin(x+y))**2)
print(formula)
print(latex(formula))
```

```
2*cos(x + y + pi/4)**2
2 \cos^{2}{\left (x + y + \frac{\pi}{4} \right )}
```

Warning: Fractions such as $1/4$ must be introduced with `Rational(1,4)` to keep `Sympy` from evaluating the expression. For example:

```
print('(1/4)^3 = '+str((1/4)**3))
print('(1/4)^3 = '+str(Rational(1,4)**3))
```

```
(1/4)^3 = 0.015625
(1/4)^3 = 1/64
```

Let SymPy do the proofs

Exercise 1. A warm-up

> Do it yourself.
> Set $\phi = \frac{1+\sqrt{5}}{2}$. Use `SymPy` to simplify $F = \frac{\phi^4-\phi}{1+\phi^7}$.

```
phi=(1+sqrt(5))/2
formula=(phi**4-phi)/(phi**7+1)
print("F = "+str(formula))
print("simplified F = "+str(simplify(formula)))
```

```
F = (-sqrt(5)/2 - 1/2 + (1/2 + sqrt(5)/2)**4)/(1 + (1/2 + sqrt(5)/2)**7
)
simplified F = -4*sqrt(5)/29 + 14/29
```

Exercise 2. A simple (?) recurrence

We will see how to use SymPy to prove a mathematical statement. Our aim is to make as rigorous proofs as possible, as long as we trust SymPy.

Do it yourself.
Let a, b be two real numbers, we define the sequence $(u_n)_{n\geq 0}$ as follows:
$u_0 = a, u_1 = b$ and for $n \geq 2$

$$u_n = \frac{1 + u_{n-1}}{u_{n-2}}.$$

1. Write a short program which returns the 15 first values of u_n in terms of symbolic variables a, b . The output should be something like:

```
u_0 = a
u_1 = b
u_2 = (b + 1)/a
...
```

2. Use `SymPy` to make and prove a conjecture for the asymptotic behaviour of the sequence (u_n) , for every reals a, b .

```python
def InductionFormula(x,y):
    return (1+x)/y

var('a b')
Sequence=[a,b]

print('u_0 = a')
print('u_1 = b')
for i in range(2,15):
    Sequence.append(simplify(InductionFormula(Sequence[-1],Sequence[-2])))
    print('u_'+str(i)+' = '+str(Sequence[-1]))
```

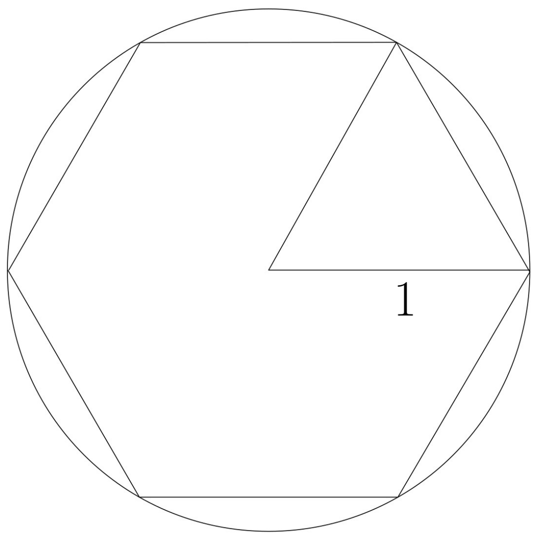
```
u_0 = a
u_1 = b
u_2 = (b + 1)/a
u_3 = (a + b + 1)/(a*b)
u_4 = (a + 1)/b
u_5 = a
u_6 = b
u_7 = (b + 1)/a
u_8 = (a + b + 1)/(a*b)
u_9 = (a + 1)/b
u_10 = a
u_11 = b
u_12 = (b + 1)/a
u_13 = (a + b + 1)/(a*b)
u_14 = (a + 1)/b
```

Answers.

1. See the cell above.
2. If $a, b \neq 0$, the sequence is well defined and we observe that $u_5 = u_0$ and $u_6 = u_1$.
   Since the sequence is defined by a recurrence of order two (*i.e.* u_n is a function of u_{n-1}, u_{n-2} this implies that the sequence is periodic: $u_{n+5} = u_n$ for every n.
   So if we trust Sympy the proof is done.

Exercise 3. What if Archimedes had known `Sympy` ?

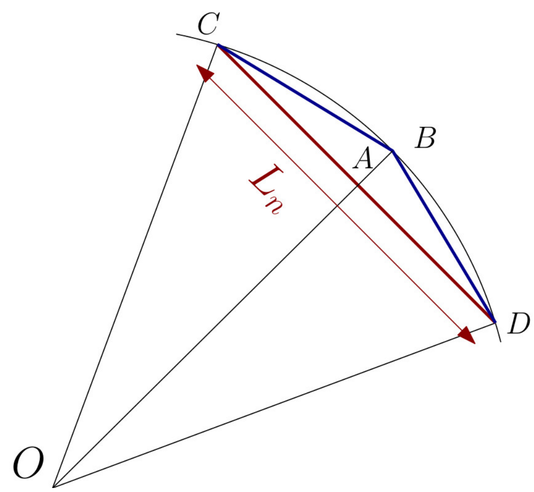
For $n \geq 1$, let $\mathcal{P}_n$ be a regular 3×2^n -gon with radius 1. Here is $\mathcal{P}_1$:



Archimedes (IIIrd century BC) used the fact that $\mathcal{P}_n$ gets closer and closer to the unit circle to obtain good approximations of π.
We will use SymPy to deduce nice formulas for approximations of π.

Do it yourself. Let L_n be the length of any side of $\mathcal{P}_n$. Compute L_1 and use the following picture to write L_{n+1} as a function of L_n:

- O is the center of the circle, $OC = 1$.
- (OB) is the bisector of $\widehat{DOC}$.
- $\widehat{OAC}$ is a right angle.



Answers. As OB is the bisector we have that $CB = BD$, which both are sides of $\mathcal{P}_{n+1}$.

Besides, OAC is rectangle at A. By Pythagora's theorem

$$1^2 = OA^2 + AC^2 = OA^2 + (L_n/2)^2.$$

BAC is also rectangle at A, therefore

$$\begin{aligned} L_{n+1}^2 = BC^2 &= AB^2 + BC^2 \\ &= (1 - OA)^2 + (L_n/2)^2 \\ &= \left(1 - \sqrt{1 - (L_n/2)^2}\right)^2 + (L_n/2)^2 \\ &= 1 + 1 - (L_n/2)^2 - 2\sqrt{1 - (L_n/2)^2} + (L_n/2)^2 \\ &= 2 - 2\sqrt{1 - (L_n/2)^2}. \end{aligned}$$

Finally we obtain

$$L_{n+1} = \sqrt{2 - 2\sqrt{1 - (L_n/2)^2}}.$$

Do it yourself.

1. Write a script which computes exact expressions for the first values $L_1, L_2, \dots, L_n$. (First try for small n's.)
2. Find a sequence a_n such that $a_n L_n$ converges to π (we don't ask for a proof). Deduce some good algebraic approximations of π. Export your results in `Latex` in order to get nice formulas.

(In order to check your formulas, you may compute numerical evaluations. With `SymPy` *, a numerical evaluation is obtained with* `N(expression)` *.)*

```
SuccessiveApproximations=[1]
p=12
for n in range(1,p):
    OldValue=SuccessiveApproximations[-1]
    NewValue=expand(sqrt(2-2*sqrt(1-(OldValue**2)*Rational(1,4))))
    SuccessiveApproximations.append(NewValue)
    print(latex(simplify(3*(2**n)*NewValue)))
    print(N(NewValue*3*2**(n)))
```

```
6 \sqrt{- \sqrt{3} + 2}
3.10582854123025
12 \sqrt{- \sqrt{\sqrt{3} + 2} + 2}
3.13262861328124
24 \sqrt{- \sqrt{\sqrt{\sqrt{3} + 2} + 2} + 2}
3.13935020304687
48 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{3} + 2} + 2} + 2} + 2}
3.14103195089051
96 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{3} + 2} + 2} + 2} + 2} + 2}
3.14145247228546
192 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{3} + 2} + 2} + 2} + 2}
+ 2} + 2}
3.14155760791186
384 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{3} + 2} + 2} + 2}
+ 2} + 2} + 2} + 2}
3.14158389214832
768 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{3} + 2} + 2
} + 2} + 2} + 2} + 2} + 2} + 2}
3.14159046322805
1536 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{3} +
2} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2}
3.14159210599927
3072 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqr
t{3} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2}
3.14159251669216
6144 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqr
t{\sqrt{3} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2} + 2}
3.14159261936538
```

Answers. When n goes large, $\mathcal{P}_n$ gets closer and closer to the unit circle. As the perimeter of $\mathcal{P}_n$ is $3 \times 2^n L_n$, we expect that

$$3 \times 2^n L_n \to 2\pi,$$

therefore we choose $a_n = 3 \times 2^{n-1}$. For $n = 8$ we obtain:

$$\pi \approx 384 \sqrt{- \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{3} + 2} + 2} + 2} + 2} + 2} + 2} + 2} = 3.141583892...$$

Exercise 4. Matrices with SymPy

In Lab 2 we proved that if a_n, b_n are integers defined by

$$a_n + b_n\sqrt{2} = (1+\sqrt{2})^n,$$

then

$$\begin{pmatrix} a_n \\ b_n \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}^{n-1} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Do it yourself.

1. Use `SymPy` to find an explicit formula for a_n.

   (In `SymPy` matrices are defined by `Matrix([[a,b],[c,d]])` .)

2. (Theory) Use the formula obtained at Question 1 to find real numbers c, R such that

$$a_n \overset{n\to+\infty}{\sim} cR^n.$$

```python
#Question 1
A=Matrix([[1,2],[1,1]])
var('n', integer=True)
Power=A**(n-1)
#print(Power)

an=latex(simplify(Power[0,0]+Power[0,1]))
bn=latex(simplify(Power[1,0]+Power[1,1]))
print('a_n ='+str(an))
print('b_n ='+str(bn))
```

```
a_n =\frac{1}{2} \left(1 + \sqrt{2}\right)^{n} + \frac{1}{2} \left(- \s
qrt{2} + 1\right)^{n}
b_n =\frac{\sqrt{2}}{4} \left(\left(1 + \sqrt{2}\right)^{n} - \left(- \
sqrt{2} + 1\right)^{n}\right)
```

Answers.

1. We export the result in LateX:

$$a_n = \frac{1}{2}\left(1+\sqrt{2}\right)^n + \frac{1}{2}\left(-\sqrt{2}+1\right)^n$$

$$b_n = \frac{\sqrt{2}}{4}\left(\left(1+\sqrt{2}\right)^n - \left(-\sqrt{2}+1\right)^n\right)$$

2. As $\left|-\sqrt{2}+1\right| < 1$, we have that $(-\sqrt{2}+1)^n \to 0$. It follows that

$$\begin{aligned} a_n &= \frac{1}{2}\left(1+\sqrt{2}\right)^n + \mathrm{o}(1) \\ &\sim \frac{1}{2}\left(1+\sqrt{2}\right)^n. \end{aligned}$$

Solving equations with SymPy

One can solve equations with Sympy. The following script shows how to solve $x^2 = x + 1$:

```
var('x') # we declare the variable
SetOfSolutions=solve(x**2-x-1,x)
print(SetOfSolutions)
```

```
[1/2 + sqrt(5)/2, -sqrt(5)/2 + 1/2]
```

Exercise 5. Solving equations with Sympy: the easy case

We will use `solve` to handle a more complicated equation.

Let $n \geq 1$ be an integer, we are interested in solving the equation

$$x^3 + nx = 1. \qquad (\star)$$

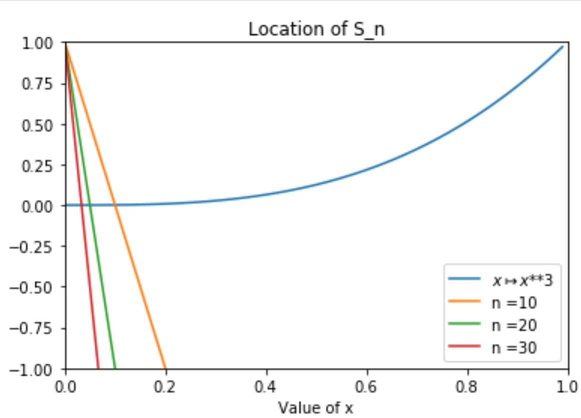
With the above script we plot $x \mapsto x^3$, and $x \mapsto 1 - nx$ for $0 \leq x \leq 1$ and for several (large) values of n. This suggests that Equation $(\star)$ has a unique real solution in the interval $[0, 1]$, that we will denote by S_n .

```
RangeOf_x=np.arange(0,1,0.01)

plt.plot(RangeOf_x,RangeOf_x**3,label='$x\mapsto x$**3')

for n in [10, 20, 30]:
    f=[1-n*x for x in RangeOf_x]
    plt.plot(RangeOf_x,f,label='n ='+str(n)+' ')

plt.xlim(0, 1),plt.ylim(-1, 1)
plt.xlabel('Value of x')
plt.legend()
plt.title('Location of S_n')
plt.show()
```



Do it yourself. (Theory)

1. Prove that indeed for every $n \geq 1$, Equation $(\star)$ has a unique real solution in the interval $[0, 1]$.
2. According to the plot, what can we conjecture for the limit S_n ?

Answers.

1. The map $x \mapsto f(x) = x^3 + nx - 1$ is continuous and increasing on $[0, 1]$, since
$$f'(x) = 3x^2 + n > n > 0.$$
Besides,
$$f(0) = 0^3 - n \times 0 - 1 = -1, \qquad f(1) = 1^3 + n \times 1 - 1 = n > 0.$$
By the intermediate value theorem, this implies that there is a unique $S_n \in (0, 1)$ such that $f(S_n) = 0$, *i.e.*
$$(S_n)^3 + nS_n = 1.$$
2. On the figure above we observe that when $n \to +\infty$, the solution of Equation $(\star)$ seems to get closer and closer to zero.
We therefore conjecture
$$\lim_{n \to +\infty} S_n = 0.$$

Do it yourself.

1. Write a script which computes the exact expression of S_n .
2. Use `SymPy` to get the asymptotic expansion of S_n (up to $\mathcal{O}(1/n^5)$). Check your previous conjecture.

```
var('x')
var('n',integer=True)

# Question 1.
Solutions=solve(x**3+n*x-1,x)
Sn=simplify(Solutions[0])  # The two other solutions are complex numbers
print("Sn = "+str(latex(Sn)))

# Question 2.
Taylor=series(Sn,n,oo,5)
print("Taylor expansion when epsilon -> 0 : "+str(Taylor))
```

```
Sn = \frac{- 2 \sqrt[3]{18} n + \sqrt[3]{12} \left(\sqrt{3} \sqrt{4 n^{
3} + 27} + 9\right)^{\frac{2}{3}}}{6 \sqrt[3]{\sqrt{3} \sqrt{4 n^{3} +
27} + 9}}
Taylor expansion when epsilon -> 0 : -1/n**4 + 1/n + O(n**(-5), (n, oo)
)
```

Answers.

1. According to the above script,

$$\frac{-2\sqrt[3]{18}n + \sqrt[3]{12}\left(\sqrt{3}\sqrt{4n^3+27}+9\right)^{\frac{2}{3}}}{6\sqrt[3]{\sqrt{3}\sqrt{4n^3+27}+9}}.$$

2. SymPy gives

$$S_n = \frac{1}{n} - \frac{1}{n^4} + \mathcal{O}(1/n^5).$$

Indeed, this goes to zero as expected.

(Bonus) Exercise 6. Solving equations: when SymPy needs help

We consider the following equation:

$$X^5 - 3\varepsilon X^4 - 1 = 0, \qquad (\star)$$

where ε is a positive parameter. A quick analysis shows that if $\varepsilon > 0$ is small enough then ($\star$) has a unique real solution, that we denote by S_ε .

The degree of this equation is too high to be solved by `SymPy` :

```
var('x')
var('e')
solve(x**5-3*e*x**4-1,x)
```

```
[]
```

Indeed, `SymPy` needs help to handle this equation.

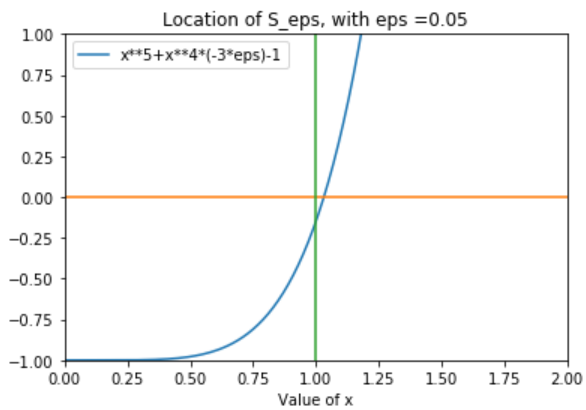
In the above script we plotted the function $f(x) = x^5 - 3\varepsilon x^4 - 1$ for some small ε. This suggests that $\lim_{\varepsilon\to 0} S_\varepsilon = 1$.

```python
RangeOf_x=np.arange(0,2,0.01)

def ToBeZero(x,eps):
    return x**5+x**4*(-3*eps) -1

eps=0.05
plt.plot(RangeOf_x,[ToBeZero(x,eps) for x in RangeOf_x],label='x**5+x**4*(-3*eps)-1'

plt.xlim(0, 2)
plt.ylim(-1, 1)
plt.plot([-2,2],[0,0])
plt.plot([1,1],[-2,2])
plt.xlabel('Value of x')
plt.title('Location of S_eps, with eps ='+str(eps))
plt.legend()
plt.show()
```



> Do it yourself.
> We admit that S_ε can be written as
> $$S_\varepsilon = 1 + r\varepsilon + s\varepsilon^2 + \mathcal{O}(\varepsilon^3),$$
> for some real r, s . Use `SymPy` to find r, s .
>
> *(You can use any* `SymPy` *function already seen in this notebook.)*

```python
var('r')
var('s')
var('eps')
Expression=ToBeZero(1+r*eps+s*eps**2+O(eps**3),eps)

Simple=simplify(Expression)
print(Simple)

solve([-3+5*r,5*s-12*r+10*r**2],[r,s])
```

```
-3*eps + 5*eps*r + 5*eps**2*s - 12*eps**2*r + 10*eps**2*r**2 + O(eps**3
)

[(3/5, 18/25)]
```

Answers. If we plug $1 + r\varepsilon + s\varepsilon^2 + \mathcal{O}(\varepsilon^3)$ into equation (★) we obtain (with the script):

$$0 = -3\varepsilon + 5r\varepsilon + 5s\varepsilon^2 - 12r\varepsilon^2 + 10r^2\varepsilon^2 + \mathcal{O}(\varepsilon^3). \qquad (\mathcal{E})$$

If we divide equation ($\mathcal{E}$) by ε we obtain

$$0 = -3 + 5r + 5s\varepsilon - 12r\varepsilon + 10r^2\varepsilon + \mathcal{O}(\varepsilon^2),$$

which yields $-3 + 5r = 0$ by letting $\varepsilon \to 0$, i.e. $r = 3/5$.

If we plug this into ($\mathcal{E}$) and divide by ε^2 we obtain

$$0 = 5s - 12r + 10r^2 + \mathcal{O}(\varepsilon),$$

which gives $5s - 12r + 10r^2 = 0$, i.e. $s = 18/25$.

Finally,

$$S_\varepsilon = 1 + \tfrac{3}{5}\varepsilon + \tfrac{18}{25}\varepsilon^2 + \mathcal{O}(\varepsilon^2),$$