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 2: Generating functions

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, factorial
import sympy as sympy             # package for symbolic computation
from sympy import *
```

Basics of generating functions

Let us first explain how we will handle generating functions with `SymPy`. We deal with the example of

$$f(x) = \frac{1}{1-2x} = 1 + 2x + 4x^2 + 8x^3 + 16x^4 + \dots$$

We first introduce variable x and function f as follows:

```
x=var('x')
f=(1/(1-2*x))

print('f = '+str(f))
print('series expansion of f at 0 and of order 10 is: '+str(f.series(x,0,10)))
```

One can extract coefficient n as follows:

- f has to be truncated at order k (for some $k > n$) with `f.series(x,0,k)`
- the n-th coefficient is then extracted by `f.coeff(x**n)`

```
f_truncated = f.series(x,0,8)
print('Truncation of f is '+str(f_truncated))
n=6
nthcoefficient=f_truncated.coeff(x**n)
print(str(n)+'th coefficient is: '+str(nthcoefficient))
```

Exercise 1. Fibonacci generating function

The generating function of the Fibonacci sequence is given by:

$$\mathrm{Fib}(x) = \frac{1}{1 - x - x^2}.$$

Do it yourself.

1. Write a recursive function `Fibonacci(n)` which returns the n-th Fibonacci number.
2. Write another function `FibonacciGF(n)` which also returns the n-th Fibonacci number by extracting the n-th coefficient in $\mathrm{Fib}(x)$.

```
def Fibonacci(n):
    # Computes recursively the n-th Fibonacci number


def FibonacciGF(n):
    # Computes the n-th Fibonacci number using GF's


# Test
print('Recursively, F_12 = ',Fibonacci(12))
print('With GF, F_12 = ',FibonacciGF(12))
```

Do it yourself.

1. Run the following script (which uses `time.process_time()`) to plot the execution times of `Fibonacci(n)` , `FibonacciGF(n)` as a function of n (try with $20 \leq n \leq 35$).
2. What do you observe?

```
import time

RunningTimeFibonacci=[]
RunningTimeFibonacciGF=[]
a=20
b=35

for n in range(a,b):
    time1=time.process_time()
    Fibonacci(n)
    time2=time.process_time()
    RunningTimeFibonacci.append(time2-time1)

for n in range(a,b):
    time1=time.process_time()
    FibonacciGF(n)
    time2=time.process_time()
    RunningTimeFibonacciGF.append(time2-time1)

N=range(a,b)
plt.plot(N,RunningTimeFibonacci,'o',label='recursive')
plt.plot(N,RunningTimeFibonacciGF,'o',label='GF')
plt.legend()
plt.show()
```

Answers.

Exercise 2. Recurrence of order two and asymptotics

Let j_n be defined by

$$\begin{aligned} j_0 &= 0, \\ j_1 &= 1, \\ j_2 &= 2, \\ j_n &= 2j_{n-2} + 5 \qquad (\text{ for every } n \geq 3). \end{aligned} \tag{\#}$$

Do it yourself.
(Theory) Find the expression for the generating function $J(x)$ of the j_n's.
(Remember that you can ask `SymPy` *to solve any equation.)*

Answers.
1.

```
# Here you can ask help to SymPy
```

Answers. We find

$$J(x) = \ldots$$

Do it yourself. 1. Write a function which extracts the n-th coefficient in $J(x)$.
2. Check your results with a recursive function which computes the j_n's.

```
# Question 1

# Question 2
```

Do it yourself.

1. What is the radius of convergence of $J(x)$? (You can ask help to SymPy.)
2. Deduce an upper bound for j_n. (Apply the "exponential growth formula", that we saw in class.)

```
# Here you can ask help to SymPy
```

Answers.

1.
2.

Do it yourself. With a plot, find an approximation of r such that j_n grows like $\mathrm{const} \times r^n$. Compare with the previous question.

```

```

Answers.

Exercise 3. A pair of generating functions

Let a_n, b_n be defined by $a_0 = b_0 = 0$, $a_1 = b_1 = 1$ and, for every $n \geq 1$,

$$\begin{cases} a_{n+1} &= a_n + 2b_n, \\ b_{n+1} &= a_n + b_n. \end{cases} \qquad (\&)$$

Do it yourself.

1. Find a 2×2 system whose solutions are $A(x), B(x)$, where A, B are the generating functions of sequences $(a_n)_{n\geq 0}, (b_n)_{n\geq 0}$. *(Coefficients of this system should depend on x.)*
2. Solve this system with `solve` and use generating function to write a script which returns $a_1, \ldots, a_{20}$.

Answers.

1.

```
#---- Question 2
```

Automatic decomposition of fractions

It is sometimes useful to decompose fractions obtain with GF's like this:

$$\frac{1-x+x^2}{(1-2x)(1-x)^2}=\frac{3}{1-2x}-\frac{1}{1-x}-\frac{1}{(1-x)^2}.$$

Here are examples on how to do that with SymPy.

Exercise 3 (continued)

Do it yourself. The goal of the exercise is to find coefficients α, β, a, b, c such that

$$A(x)=\frac{a}{x-\alpha}+\frac{b}{x-\beta}+c,$$

where

$$A(x)=\frac{-x\,(x+1)}{x^2+2x-1}$$

was defined in the previous exercise.

1. **(Theory)** Compute $\lim_{x\to+\infty} A(x)$ and deduce c.
2. **(Theory + SymPy)** Use `SymPy` to find coefficients α, β.
3. **(Theory + SymPy)** Use `SymPy` again to find coefficients a, b.

Answers. Question 1.

```
# Question 2
```

Answers. Question 2.

```
# Question 3
```

Answers. Question 3.

Do it yourself.

(Theory) Deduce a proof of the formula

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

(Hint: Use the formula

$$\frac{1}{x - \rho} = -\frac{1/\rho}{1 - x/\rho} = -1/\rho \sum_{n \geq 0} x^n (1/\rho)^n. \tag{E}$$

Answers.