NumIntro Block 1, 2020

Project (Obligatorisk Opgave)

Responsible: M. Bladt and S. Talebi *Date: October 2, 2020*

1. Practical

The project must be uploaded before Friday 9.10.2020 at 23:59 hrs. The theoretical part must be uploaded as a PDF file to the link provided in the Assignment "Obligatorisk opgave" of Week 6 in Absalon, and the Python programs, which should be named prog1.py, prog2.py, prog3.py and prog4.py, should be uploaded through the same link, preferably zipped together.

2. Background and Purpose

This project deals with the calculation of matrix exponentials. We will focus on taking the exponential of matrices which are either intensity or sub–intensity matrices. These matrices appear as so–called infinitesimal generators in the theory of Markov process, and their matrix exponentials provide the matrix of transition probabilities.

Intensity matrices are matrices which have non–negative off diagonal elements, negative diagonal elements and rows which sum to zero. Sub–intensity matrices have non–negative off diagonal elements, negative diagonal elements and non–positive row sums. More precisely, a square matrix $\boldsymbol{A} = \{a_{ij}\}_{i,j=1,\ldots,n}$ for which

$$a_{ij} \geq 0, \quad i \neq j \quad \text{and} \quad a_{ii} \leq -\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}, \quad i = 1,\ldots,n,$$

is called a sub–intensity matrix. If equality holds in the latter, then the matrix is said to be an intensity matrix. E.g.,

$$\boldsymbol{A} = \begin{pmatrix} -5 & 2 & 3 \\ 2 & -6 & 4 \\ 4 & 5 & -9 \end{pmatrix} \quad \text{and} \quad \boldsymbol{A}_s = \begin{pmatrix} -7 & 2 & 3 \\ 2 & -6 & 4 \\ 4 & 5 & -9 \end{pmatrix}$$

are examples of an intensity and sub–intensity matrix, respectively.

A matrix $\boldsymbol{P} = \{p_{ij}\}_{i,j=1,\ldots,n}$ is said to be sub–stochastic if

$$p_{ij} \geq 0 \quad \text{for all } i,j, \quad \text{and} \quad \sum_{j=1}^{n} p_{ij} \leq 1, \quad i = 1,\ldots,n.$$

If equality holds, the matrix is said to be stochastic.

The numerical evaluation of matrix exponentials for (sub-)intensity matrices may be troublesome, but can be solved by some nice numerical tricks. This exercise will take you through this little journey. We hope that you enjoy it!

3. **Problems**

The exponential function of a matrix $\boldsymbol{A}$ is defined by

$$\exp(\boldsymbol{A}x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} \boldsymbol{A}^n, \tag{1}$$

where $\boldsymbol{A}^0 = \boldsymbol{I}$ and where $\boldsymbol{I}$ denotes the identity matrix. We assume throughout that $x \geq 0$.

1. Implement in Python a function which calculates $\exp(\boldsymbol{A}x)$ for a given matrix $\boldsymbol{A}$ and scalar x using (1) directly. Hence you must write a function which takes as input a matrix $\boldsymbol{A}$ and a scalar x, and returns the matrix exponential $\exp(\boldsymbol{A}x)$. For $x = 1$, test your implementation on the intensity matrix

$$\boldsymbol{A} = \begin{pmatrix} -5 & 2 & 3 \\ 2 & -6 & 4 \\ 4 & 5 & -9 \end{pmatrix} \tag{2}$$

as well as on $\boldsymbol{A}/20$ (the matrix $\boldsymbol{A}$ scaled by $1/20$). Please report on the stopping criteria you used regarding possible convergence to a given precision. What can you conclude? You may check your result up against Pythons matrix exponential function (*expm* which can be called by using "from scipy.linalg import expm" in the header of the program). Name the implementation prog1.py when uploading.

□

2. Let $\boldsymbol{A} = \{a_{ij}\}_{i,j=1,\ldots,n}$ be a sub–intensity matrix, and let $\eta = \max_i(-a_{ii}) > 0$.

2.1. Show, that

$$\boldsymbol{P} = \boldsymbol{I} + \eta^{-1}\boldsymbol{A}$$

is then sub–stochastic, and show that $\boldsymbol{P}$ is stochastic if $\boldsymbol{A}$ is an intensity matrix. Do the opposite implications hold as well?

2.2. Show that

$$\exp(\boldsymbol{A}x) = e^{-\eta x} \sum_{n=0}^{\infty} \frac{(\eta x)^n}{n!} \boldsymbol{P}^n. \tag{3}$$

2.3. Let $\|\cdot\|$ be the matrix norm induced by the maximum norm for $\mathbb{R}^n$. Show that

$$\| \exp(\boldsymbol{A}x) - e^{-\eta x} \sum_{n=0}^{\ell} \frac{(\eta x)^n}{n!} \boldsymbol{P}^n \| < \varepsilon \tag{4}$$

if

$$\sum_{n=0}^{\ell} \frac{(\eta x)^n}{n!} e^{-\eta x} > 1 - \varepsilon.$$

2.4. Implement a method which uses (3) for the calculation of the matrix exponential as an alternative to (1). Further, use (4) to ensure that the error of the matrix exponential (entrywise) is at most ε. You must write a function taking as input a matrix $\boldsymbol{A}$, a scalar x, and a tolerance ε, and as ouput produces the matrix exponential $\exp(\boldsymbol{A}x)$ up to the required precision ε. This method is referred to as *uniformisation*. Again, apply your implementation to the matrix (2) with $x = 1$ and comment on the result. Name this program prog2.py .

2.5. Investigate if there is any relation between large x and the solution ℓ to (4). One way of reducing a large x is by so–called *scaling-and-squaring*, where it is used that

$$\exp(\boldsymbol{A}x) = [\exp(\boldsymbol{A}x/m)]^m .$$

Show that it is clever to use m on the form $m = 2^N$ to calculate the mth power of $\exp(\boldsymbol{A}/m)$. Code a matrix exponential function which uses uniformisation together with scaling-and-squaring; input should be for x, which is scaled down to a $x/2^N \in [1,2]$, or the smallest N for which $x/2^N \in [0,1]$. Again, apply your implementation to the matrix (2) now with $x = 1, 10, 100$ and comment on the result. Name this program prog3.py

2.6. Here you may write your general conclusion on how to calculate the matrix exponential of an intensity matrix: which method can you recommend and which methods will likely fail (and possibly why).

3. We present a third method for calculating the matrix exponential. Assume that $\boldsymbol{A}$ is an arbitrary diagonisable matrix, i.e.$\boldsymbol{A}$ has n distinct eigenvalues λ_1,...,λ_n with corresponding eigenvectors $\boldsymbol{v}_1, ..., \boldsymbol{v}_n$ such that

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{-1},$$

where $\boldsymbol{V}$ is the matrix with columns $\boldsymbol{v}_1, ..., \boldsymbol{v}_n$, and $\boldsymbol{D}$ is the diagonal matrix with $\lambda_1, ..., \lambda_n$ in the diagonal, namely $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$.

3.1. Show that

$$\exp(\boldsymbol{D}x) = \text{diag}(e^{\lambda_1 x}, ..., e^{\lambda_n x}).$$

3.2. Show that

$$\exp(\boldsymbol{A}x) = \boldsymbol{V}\,\text{diag}(e^{\lambda_1 x}, ..., e^{\lambda_n x})\boldsymbol{V}^{-1}.$$

3.3. By using 3.1 and 3.2, write a function which calculates the matrix exponential for a diagonisable matrix. Compare the results to the uniformisation method (prog3.py) for the matrix (2) (with $x = 1, 10, 100$) in terms of execution (CPU) times and precision (error). Regarding eigenvalues and eigenvectors you may use Python routines which provide these (from e.g. the library "scipy.linalg") and execution times can e.g. be found by using the library "time". Name the resulting program prog4.py .

3.4. The comparison in 3.3 is performed for 3×3 matrices. What do you think will happen to each of the methods as the size of the matrix increases?