Numerical Methods (Period 1, 2023/2024)

Assignment

Deadline Friday 13 October 17:00

Upload instructions

The assignment consists of four parts. You will need to implement some code in Python and write a report.

- Work out the required Python code in Spyder. You are required to use the template in this document. Make sure your submission includes the Python files with extension `py`. Also make sure to include all docstrings and include comments whenever necessary to make your code as readable as possible. Part of your grade is based on this – no points will be awarded to hard to read code.

- Next to your Python files, you need to hand in a report. This report contains your results, i.e., the answers to all questions of the assignment, the figures, the mathematical derivations, etc. You do not need to include implementation details of your Python code (your code should be readable instantaneously, after all). Preferably you work out the report in LaTeX, otherwise in Microsoft Word or a similar text editor. Either way, make sure to convert your work to a single `pdf` file.

  Your report must have the name `ReportTeamX.pdf`, where you replace X by your team number.

- In the end, you upload exactly one `zip` file on Canvas that contains your Python files, your report, and possibly some other files that you used.

  The single file that you upload on Canvas must be named `Assignment1TeamX.zip`, where you replace X by your team number.

Important: The last submission before the deadline that adheres to the submission guidelines above will be graded, other submissions will be ignored. Make sure you do not deviate from the instructions above in any way to avoid receiving a 0.

Table of contents

Assignment Numerical Methods, Period 1 2023–2024

Let X be a random variable on $(0, \infty)$ with probability density function (PDF)

$$f(x \mid \lambda, \mu) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left(-\frac{\lambda}{2\mu^2 x}(x-\mu)^2\right), \quad x > 0\,, \tag{1}$$

where $\lambda > 0$ is the scale parameter, and $\mu > 0$ the location parameter. This is the Inverse Gaussian distribution $IG(\lambda, \mu)$, with mean $\mathbb{E}[X] = \mu$ and variance $\mathbb{V}ar(X) = \mu^3/\lambda$.

Part A: Plotting

Let us first get a feeling of what we are dealing with.

- Write a Python script in which you plot the graphs of the PDF on $[0, 5]$, for the combinations
$$(\lambda = 0.5, \mu = 1),\ (\lambda = 1, \mu = 3),\ (\lambda = 5, \mu = 1),\ (\lambda = 10, \mu = 3)\,.$$
Plot these four graphs in one figure with different colors, adding a legend telling what color is associated with what combination.
- Copy the figure to your report.

Part B: Integrating

Let us check numerically that the function defined in (1) is indeed a probability distribution function. Clearly $f(x) > 0$ is satisfied. Thus you have to check $\int_0^\infty f(x)\,\mathrm{d}x = 1$. Do this in Python using the following three methods.

- First, implement the Romberg Algorithm for numerical integration (see Section 5.2, pages 217–219 in the seventh edition of the textbook; or Section 5.3, pages 204–206 in the sixth edition). For this, you may choose any valid combination of the parameters μ and λ. Furthermore, observe that you need to compute $\int_0^K f(x)\,\mathrm{d}x$ for some finite K, but large enough. Experiment with this value.
- Second, approximate the integral $\int_0^K f(x)\,\mathrm{d}x$ by a Riemann sum (see your Analysis or Calculus courses) to obtain a numerical value. Experiment with the widths of the subintervals.
- Third, continue by applying an integration function of `scipy` (for instance `quad`, `quadrature` or `trapz`).
- Implement the three methods above in Python, and report shortly on the idea and techniques behind the algorithms in your report (no implementation details). Also report on the differences between these three methods. Is $f(x)$ indeed a probability distribution?

Part C: Estimating

In Canvas there is a file `xdata.txt` with a sample of 1000 observations $x_1, x_2, \ldots$ from an inverse Gaussian distribution with unknown parameters λ, μ. Our goal for this and the next part is to try to estimate the values of μ and λ that were used to generate this data.

As you know, we can find the most likely values $\hat{\lambda}$ and $\hat{\mu}$ by MLE (maximum likelihood estimation). The most likely location parameter $\hat{\mu}$ is the sample average, i.e.,

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i\,,$$

with $n = 1000$ the number of data.

- Compute the value of $\hat{\mu}$ using Python and copy its value to your report. The easiest is to read the file using the function `genfromtxt` available in `numpy`.

You substitute this computed value in the PDF in (1). Then, for MLE of the scale parameter λ, you need to first do the following by pen and paper.

- Find the expression of the log likelihood function $\ell(\lambda \mid x)$, i.e., as function of the parameter λ, given data $x_1, \ldots, x_n$.
- Find the expression of the derivative $\ell'(\lambda \mid x)$.
- Include your derivations in your report.

You know that your best estimate $\hat{\lambda}$ for the value of λ is the maximum of $\ell(\lambda \mid x)$. We can find that using the root of the derivative $\ell'(\lambda \mid x)$.

Again, let us first take a look at what these functions look like.

- Write a Python script that implements these two expressions, where you use the data of the file `xdata.txt`.
- Then, plot the graphs of $\ell(\lambda \mid x)$ and $\ell'(\lambda \mid x)$. Use two subplots (one subplot per graph), and a domain of λ that shows clearly the maximum of $\ell(\lambda \mid x)$, and the root of $\ell'(\lambda \mid x)$. Add gridlines to the plots so that you can see that the maximum of $\ell(\lambda \mid x)$ and the root of $\ell'(\lambda \mid x)$ are the same (as it should be).
- Copy these plots to your report and comment on the results.

Part D: Root finding

It remains to compute the root of the derivative $\ell'(\lambda \mid x)$ and then the MLE for the parameter λ is complete. We will do this using four different methods.

- First, write a Python script that implements the bisection method for finding the root $\hat{\lambda}$ of the function $\ell'(\lambda \mid x)$. Determine a valid initial interval based on your plot in question c. Experiment with stopping criteria. Output the (approximated) root, the function value, the number of iterations and the final interval.
- Second, implement the False Position Method, see page 121 of the text book, and apply it to find the root of the function $\ell'(\lambda \mid x)$. Use the same initial interval and stopping criteria again, and again output the (approximated) root, the function value, the number of iterations and the final interval.
- Third, implement the Modified False Position Method, page 121-122. Apply again to the function $\ell'(\lambda \mid x)$ using the same initial interval and the same stopping criteria, and output the same.

  Please find an important remark and a hint at the end of this document.
- Fourth, as a verification, apply a root finding function of `scipy`, for instance `root_scalar`, `brentq`, `ridder` or others.
- After implementing this in Python, it's time to report on it. Shortly describe the idea behind the methods you implemented in your report, and properly discuss their numerical accuracy and running time. Also discuss the approximated root values, the function values, the number of iterations and the final intervals. What differences do you see?
- And finally: What is your maximum likelihood estimate $\hat{\lambda}$ for the parameter λ used in the data of `xdata.txt`?

Rubrics

You can score 100 points for this assignment and your grade will be your point total divided by 10. You can obtain at most 50 points for the code (C) and at most 50 points are for the report (R) as follows:

Part A (5 points)

- R: 5 points for the quality of the plot in the report.

Part B (29 points)

- C: 9 points for a correct and readable implementation of the Romberg algorithm.
- C: 9 points for a correct and readable implementation of the Riemann sums.
- C: 1 point for the use of `scipy`.
- R: 10 points for the report of this part.

Part C (18 points)

- C: 3 points for a correct and readable implementation of the computation of $\hat{\mu}$.
- R: 10 points for the report on the derivation of the expressions of $\ell(\lambda \mid x)$ and $\ell'(\lambda \mid x)$.
- R: 5 points for the quality of the plots in the report.

Part D (37 points)

- C: 9 points for a correct and readable implementation of the bisection method.
- C: 9 points for a correct and readable implementation of the False Position method.
- C: 9 points for a correct and readable implementation of the Modified False Position method.
- C: 1 point for the use of `scipy`
- R: 10 points for the report of this part.

Quality of report (10 points)

- R: The points for the report above are purely for its contents. You can get at most 10 points for the quality of your report in terms of writing, language use, structure, layout, the use of images and tables, and the general editorial quality.

The teacher retains the right to award or subtract points based on other criteria and to award different grades to different team members. Examples of situations in which this can occur include the detection of plagiarism, fraud, or the use of AI tools (such as ChatGPT). In these cases, the Board of Examiners will be notified, which can have serious consequences.

Required Python template

You need to use the template file below, indicating part of the Python functions in this assignment. The appropriate input parameters or arguments of the functions are to be provided. You can add other functions yourself, but the functions below need to be part of your program for testing purposes.

```
# FUNCTION f(x|lambda,mu)

def funf(<input args>):
        '''
        docstring with information
        '''

# PART A

def plotfunf(<input args>):
        '''
        docstring with information
        '''

# PART B

def romberg(<input args>):
        '''
        docstring with information
        '''

def riemann(<input args>):
        '''
        docstring with information
        '''

def integratescipy(<input args>):
        '''
        docstring with information
        '''

# PART C

def funl(<input args>):
        '''
        docstring with information
        '''

def funderivativel(<input args>):
        '''
        docstring with information
        '''

def plotfunl(<input args>):
        '''
        docstring with information
```

```
        '''

def plotfunderivativel(<input args>):
        '''
        docstring with information
        '''

# PART D

def bisection(<input args>):
        '''
        docstring with information
        '''

def falsepos(<input args>):
        '''
        docstring with information
        '''

def modfalsepos(<input args>):
        '''
        docstring with information
        '''

def rootfindscipy(<input args>):
        '''
        docstring with information
        '''

# MAIN

def main():
        '''
        magic numbers
        '''

if __name__ == '__main__':
        main()
```

Important remark regarding the Modified False Position method

There is a typo in the display of $c_k^{(m)}$ on page 122. In case that the same endpoint is retained twice, use the top formula if $f(c_k)f(b_k) < 0$, and the bottom formula if $f(c_k)f(b_k) > 0$. The c_k is the original intercept point (see page 121). You see this in Figure 3.3: the left endpoint is retained twice ($a_{k-1} = a_k$), and $f(c_k)f(b_k) > 0$; thus the modified intercept uses half $f(a_k)$, which is the bottom formula. Clearly, next you check the signs of $f(a_k)$ and $f(c_k^{(m)})$ to do the next iteration either on $(a_{k+1} = a_k, b_{k+1} = c_k^{(m)})$, or $(a_{k+1} = c_k^{(m)}, b_{k+1} = b_k)$.

Hint for Modified False Position method

To understand what is going on, consider the following numerical example. Consider $f(x) = 1/x - \log(x) + \log(2)$ and the modified false position is applied on the interval $[0, 20]$. Here are the values of the first iterations (plot the function to visualize it):

- Initialization: $a_0 = 0, b_0 = 20, f(a_0) > 0, f(b_0) < 0$.
- Iteration 1:

$$c = 19, f(c) < 0 : \text{move } b \text{ remain } a$$
$$a_1 = a_0 = 0, b_1 = c = 19$$

- Iteration 2:

$$c = 1, f(c) > 0 : \text{move } a \text{ remain } b$$
$$a_2 = c = 1, b_2 = b_1 = 19$$

- Iteration 3:

$$c = 18, f(c) < 0 : \text{move } b \text{ remain } a$$
$$a_3 = a_2 = 1, b_3 = c = 18$$

- Iteration 4 moves the same endpoint as the previous iteration, so modify:

$$c = 17, f(c) < 0 : \text{move } b \text{ remain } a \text{ (again)}$$
$$c^{mod} = 15, f(c^{mod}) < 0 : \text{move } b \text{ remain } a \text{ first}$$
$$a_4 = a_3 = 1, b_4 = c^{mod} = 15$$

- Iteration 5 moves the same endpoint as the previous iteration, so modify:

$$c = 14, f(c) < 0 : \text{move } b \text{ remain } a \text{ (again)}$$
$$c^{mod} = 2, f(c^{mod}) > 0 : \text{move } a \text{ remain } b \text{ first}$$
$$a_5 = c^{mod} = 2, b_5 = b_4 = 15$$

- Iteration 6:

$$c = 14, f(c) < 0 : \text{move } b \text{ remain } a$$
$$a_6 = a_5 = 2, b_6 = c = 14$$

- Iteration 7:

$$c = 3, f(c) > 0 : \text{move } a \text{ remain } b$$
$$a_7 = c = 3, b_7 = b_6 = 14$$

- Iteration 8 moves the same endpoint as the previous iteration, so modify:

$$c = 4, f(c) > 0 : \text{move } a \text{ remain } b \text{ (again)}$$
$$c^{mod} = 6, f(c^{mod}) > 0 : \text{move } a \text{ remain } b \text{ first}$$
$$a_8 = c^{mod} = 6, b_8 = b_7 = 14$$

- Etc.