Norwegian University of Science and Technology

Project 1, TMA4320: Examining local mass-density distributions by gravity surveying

Stu.nr: 478679   Stu.nr: 478694   Stu.nr: 478704

Jan/Feb 2018

1 Introduction

This document aims to answer the questions posed in project 1 from the course TMA4320, spring 2018. The goal of the project is to approximate the density-distribution in an one-dimensional region based on a given set of measurements of the gravitational force in the region.

2 Answers to questions

Question 1

We are given the anti-derivative of the kernel of the Fredholm equation, and $F(x)$ can analytically be determined (equation 2 in the worksheet). The fundamental theorem of calculus implies that

$$F(x) = \int_{a_0}^{b_0} K(x,y)dy = \left[\frac{y-x}{d(d^2+(x-y)^2)^{\frac{1}{2}}} \right]_{b_0}^{a_0}, \tag{1}$$

where $[a_0, b_0] = [\frac{1}{3}, \frac{2}{3}]$ is the interval where $\rho(x)$ evaluates to 1; otherwise zero. This can be evaluated as a function in python like shown below. Here also follows an example of plotting with the matplotlib.pyplot library.

```python
import numpy as np; import matplotlib.pyplot as plt
a0=1/3; b0=2/3; acc=400

def F(x,a,b,d):
  return (b-x)/(d*(d**2+(x-b)**2)**(1/2)) - (a-x)/(d*(d**2+(x-a)**2)**(1/2))

# Plotting (example)
xvalues = np.linspace(a,b,acc); yvalues = np.zeros(acc)
for i in range(acc):
    yvalues[i] = F(xvalues[i], a0, b0, d)
plt.semilogy(xvalues, yvalues, label=r'$d$', lw=3)
plt.legend(loc="best"); plt.xlabel(r"$x$"); plt.ylabel(r"$F(x)$")
plt.grid()
plt.show()
```

This yields the plot shown in figure 1.

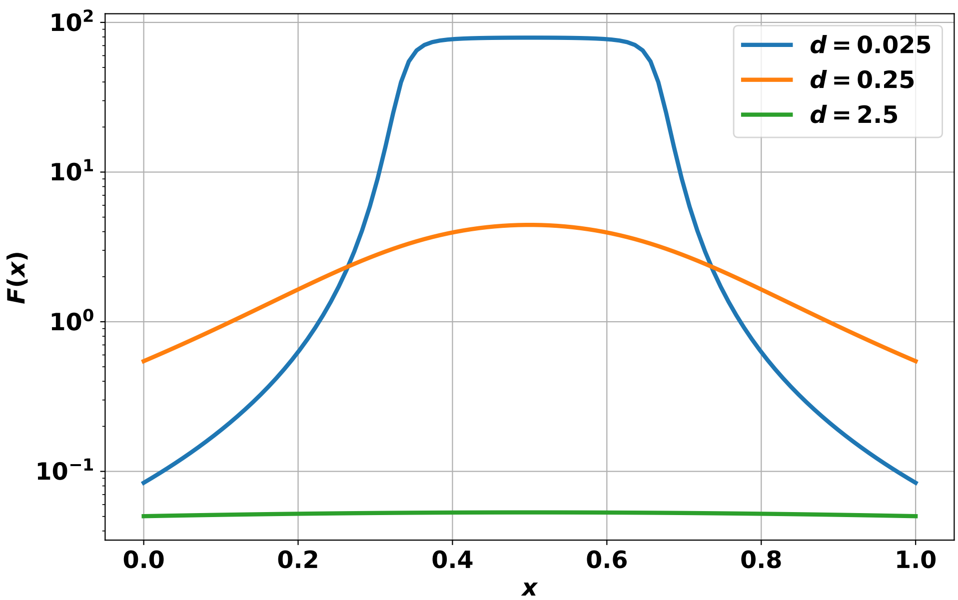

Figure 1: Question 1. Plot of the analytic solutions for F given different values of d.

Question 2

Inserting the appropriate functions into the theoretical approximations given, we get

$$F(x) = \int K(x,y)\rho(y)dy \approx \int K(x,y) \sum_{j=0}^{Ns-1} \hat{\rho}_j L_j(y)dy = \sum_{j=0}^{Ns-1} \int K(x,y)\hat{\rho}_j L_j(y)dy$$

$$\Rightarrow F(x) \approx \sum_{j=0}^{Ns-1} \sum_{k=0}^{Nq-1} w_k K(x, x_k^q)\hat{\rho}_j L_j(x_k^q), \tag{2}$$

where

$$L_j(x) = \frac{\prod_{m\neq j}(x - x_m^s)}{\prod_{m\neq j}(x_j^s - x_m^s)}$$

is the j-th Lagrange interpolation basis polynomial.

To construct the system of linear equations from equation 2 satisfying

$$A\vec{\hat{\rho}} = \vec{F}, \tag{3}$$

where

$$\vec{F} = \begin{bmatrix} F(x_0^c) \\ F(x_1^c) \\ \vdots \\ F(x_{Ns-1}^c) \end{bmatrix} \qquad \vec{\hat{\rho}} = \begin{bmatrix} \hat{\rho}_0 \\ \hat{\rho}_1 \\ \vdots \\ \hat{\rho}_{Ns-1} \end{bmatrix},$$

the element at row i and column j in matrix A from equation 3 has to satisfy

$$A_{ij} = \sum_{k=0}^{Nq-1} w_k K(x_i^c, x_k^q) L_j(x_k^q) = \sum_{k=0}^{Nq-1} w_k K(x_i^c, x_k^q) \frac{\prod_{m\neq j}(x_k^q - x_m^s)}{\prod_{m\neq j}(x_j^s - x_m^s)}, \tag{4}$$

to give the correct expression for $F(x)$ (cf. (2)) in a given collocation point.

Question 3

We implement the appropriate function for solving the left hand side of the equation (3) according to the guidelines given in the worksheet. The components of $\vec{F}$ are calculated from the test function $\rho(x) = e^{\gamma x}\sin(\omega x)$. To verify that the implementation is correct, we evaluate the error between $\vec{F}$ and $A\vec{\rho}$. In the test, we let the number of collocation points and source points (N_c and N_s respectively) be fixed at 40. The number of quadrature points N_q, given by applying the midpoint Newton-Côtes quadrature, varied.

The result in figure 2 show that for sufficiently large values of N_q, the implementation of $A\vec{\rho}$ yields the desired values of $F(x_i^c)$.

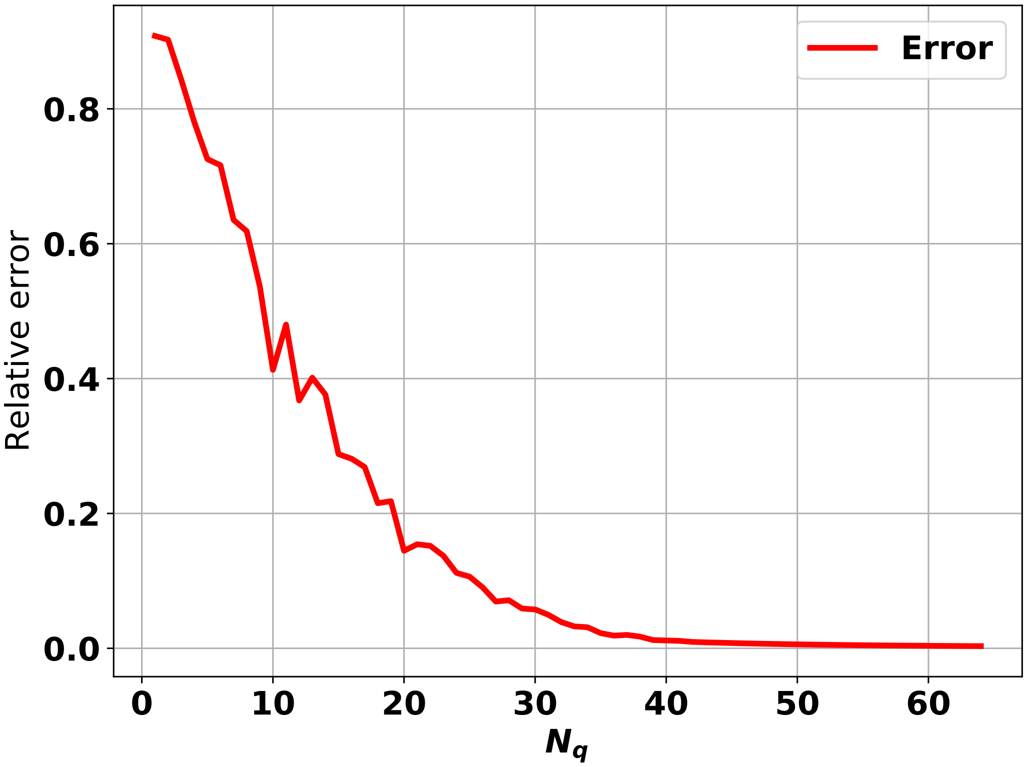

Figure 2: Question 3. Plot of the relative error $\max\limits_{0\leq i\leq N_c-1} \frac{|F(x_i^c)-(A\vec{\rho})_i|}{|(A\vec{\rho})_i|}$ using Newton-Côtes midpoint quadrature.

To successfully implement the error function $\max\limits_{0\leq i\leq N_c-1}|F(x_i^c) - (A\vec{\rho})_i|$, x_q and $\{w_i\}$ had to be re-calculated for every value of N_q. Thus, also A had to be recalculated for each N_q. Similarly to question 1, each value of N_q maps the corresponding value of $F(x_i^c)$ to an element in an array, and the array is plotted. In accordance with (4), the implementation of the left-hand side of the Fredholm equation set in python is:

```
def fredholm_lhs(xc, xs, xq, w, K, Nq, d):
```

```
    Nc = xc.shape[0]; Ns = xs.shape[0]
    A = np.zeros((Nc, Ns)); a = 0
    for i in range(Nc):
        for j in range(Ns):
            a = 0
            for k in range(Nq):
                a += w[k] * K(xc[i], xq[k], d) * lagrangePolyJ(j, xq[k], xs)
            A[(i, j)] = a
    return A
```

Question 4

By repeating the method from question 3 but replacing the Newton-Côtes midpoint quadrature with a Legendre-Gauss quadrature, we acquire an equivalent plot of the error. The Legendre-Gauss quadrature algorithm used by the pythonmodule *Numpy* returns a quadrature-partitioning of the interval [-1,1]. To correct this, we construct a transformation to map the interval [-1,1] to [0,1] after the quadrature-execution:

```
a = 0; b = 1
xq, w = np.polynomial.legendre.leggauss(nq[i]) ##Numpy's algorithm
xq = xq * (b-a)/2 + (a+b)/2 ## Mapping the points from [-1,1] to [0,1]
w = w * (b-a)/2 ## Reducing the weights to fit the new interval
```

Using this definition of w and X_q in the implementation of question 3, we get the error plot given in fig 3.

By comparing figure 2 and figure 3, the convergence-pattern is nearly indistinguishable from one another.

Question 5

We are now interested in solving (2) for $\vec{\rho}$ given some $\vec{F}$. This is achieved by the linalg.solve function in the numpy library:

```
A = plib.fredholm_lhs(xc, xs, xq, w, K, Nq, d)
b = plib.fredholm_rhs(xc, F)
rhoHat = np.linalg.solve(A,b)
```

We find the relative error in a similar manner as before, still using the Legendre-Gauss quadrature. This time, we vary the amount of collocation points N_c. We let $N_s = N_c$,

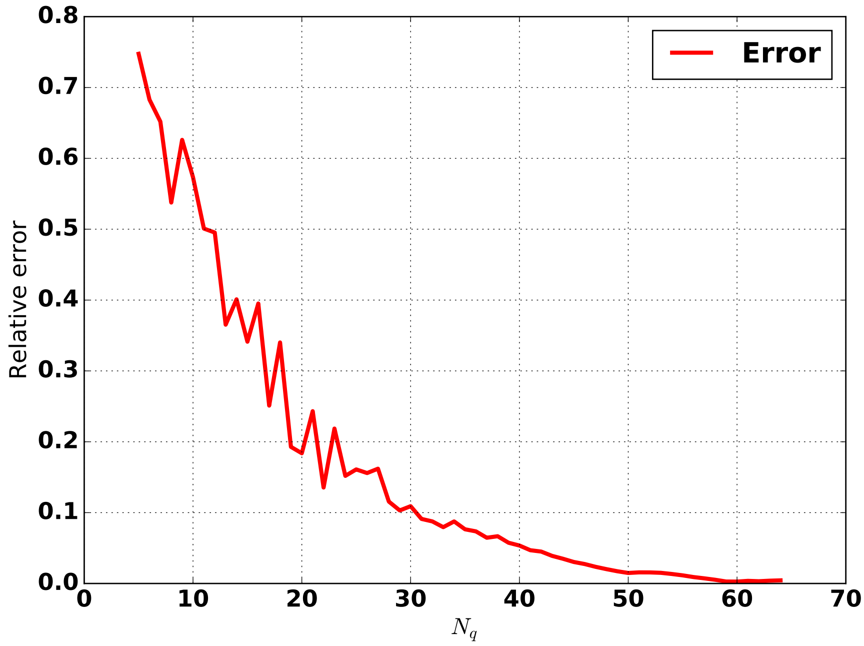

Figure 3: Question 4. Plot of the relative error $\max\limits_{0\leq i\leq N_c-1} \frac{|F(x_i^c)-(A\vec{\rho})_i|}{|(A\vec{\rho})_i|}$ using Legendre-Gauss quadrature.

and choose N_q to be sufficiently high; in our case $N_q = N_c^2$. Then, we plotted the relative error of $\vec{\tilde{\rho}}$ for different values of d, as shown in fig 4

Question 6

Using python's standard random library, we generated random errors on the $\vec{b}$ from the right-hand side of (2). The implementation is:

The value of δ was given as 10^{-3}, i.e. a 0.1% perturbation. We plotted the values of $\vec{b}$ and the perturbed $\vec{\tilde{b}}$ when varying x (i.e. in the collocation points). We also plotted the random error's effect on the solution of (2). These are shown in fig 5.

```
def getRandomPerturbation(b, delta):
    Nc = len(b)
    bTilda = np.zeros(Nc)
    for i in range(Nc):
        randNum = random.uniform( delta, delta)
        bTilda[i] = b[i] + randNum
    return bTilda
```

As we can see from the figure, the random error is negligible compared to the error caused by our computations. The "wild" behaviour of the plot for d_3, is somewhat

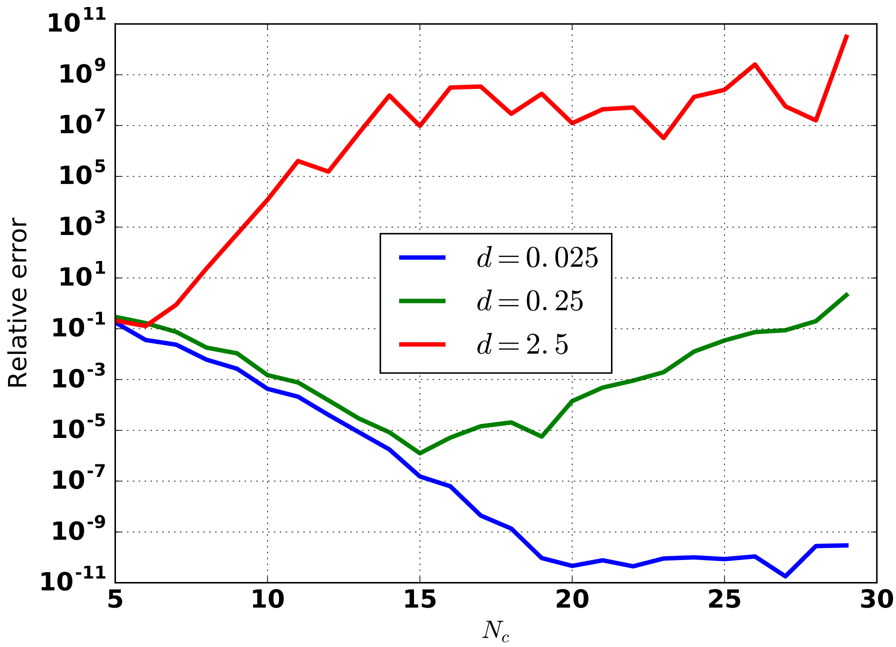

Figure 4: Question 5. Plot of the relative error $\max\limits_{0 \leq j \leq N_s - 1} \frac{|\vec{\tilde{\rho}}_j - \rho(x_j^s)|}{|\vec{\rho}|}$ for different values of d.

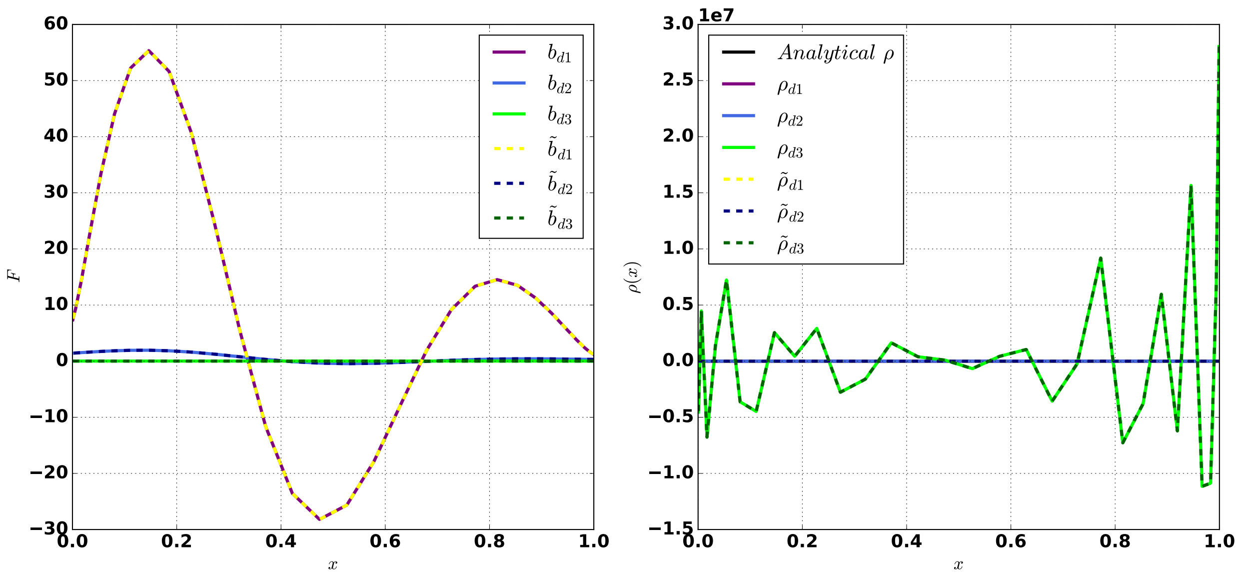

Figure 5: Question 6. Left: Plot of $\vec{b}$ and $\vec{\tilde{b}}$ for different values for d. Right: Plot of $\vec{\rho}$ and $\vec{\tilde{\rho}}$ for different values for d.

expected if you consider the large error for $\vec{\rho}$ shown in fig 4. The error is not really escalated by random errors.

Question 7

We wish to minimize the computational error of ρ as seen in fig 5. We used the Tikhonov regularization technique as described. The compromise between accuracy and efficiency is given by minimizing

$$\frac{1}{2}||A\vec{\hat{\rho}} - \vec{b}||^2 + \frac{\lambda}{2}||\vec{\hat{\rho}}||^2 \tag{5}$$

for some parameter λ. Minimizing this, gives the possibility to minimize the error in ρ. The plot of the error for different λ is shown in fig 6. The system of equations resulting from the Tikhonov regularization was solved by a program like this:

```
def getDiffQ7(N, lambdaList, ind, A, vecBTilde, rhovec):
    Ns = Nc = N
    rhoHatTilde = np.zeros(Nc)
    rhoHatTilde = plib.tikhonovSystem(A,vecBTilde,lambdaList[ind])
    return np.linalg.norm((rhovec - rhoHatTilde), np.Inf) / np.linalg.
norm(rhovec, np.Inf)
```

This gives that we should choose λ to be approximately 10^{-4} for $d = 0.25$ and 10^{-8} for $d = 2.5$.

Question 8

We now import the sample measurement. This includes an interval $[a, b]$, the depth d and a set of measurement points (respectively, the collocation points and the corresponding vertical force measurements).

The file **q8_1** gave $d = 0.25$, so we started looking for well-suited λ around there. By experimenting with the parameter λ regarding error-minimization, we chose $\lambda = 5 \cdot 10^{-4}$. The resulting plot of ρ is plotted in figure 7.

The file **q8_2** gave $d = 1.0$, so we started looking for a suitable λ in the interval $[10^{-6}, 10^{-3}]$ as recommended by figure 6. Despite the recommended values of λ from figure 6, the final value chosen was $\lambda = 2$, and the resulting plot is shown in fig 8.

The file **q8_3** gave $d = 0.25$. With respect to figure 6, and similar experimentation, the final value selected was $\lambda = 10^{-3}$, and the resulting plot is shown in fig 9.

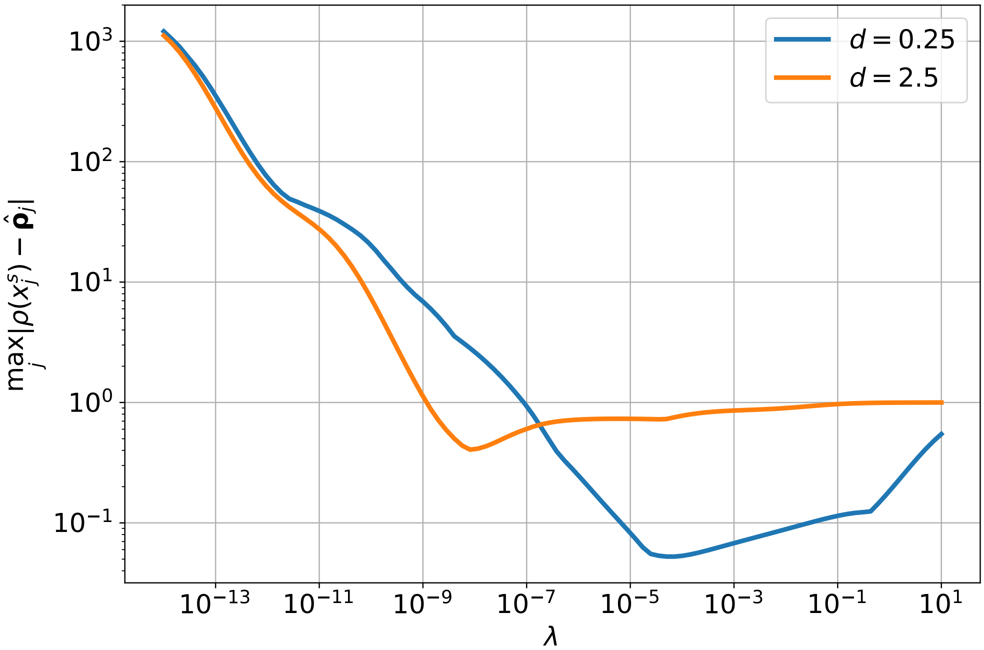

Figure 6: Question 7. Plot of the error in ρ for different values of the regularization parameter λ.

The polynomials used in this section were interpolated using Newton's divided differences method on source points defined by Chebyshev interpolation (see [Sau11] chapter 3, section 3 for details about this interpolation method). The system of equations were constructed with Thikonov regularization, and solved using numerical linear algebra from the Numpy-library in Python.

Figure 7,8 and 9 with corresponding files **q8_1**, **q8_2** and **q8_3** respectively, reveals shapes of ρ that is in agreement with the expectations motivated in question 8 from the worksheet.

3 Conclusion

The main goal of this project was to reconstruct a mass-density distribution using interpolation and integral-quadratures, given a set of gravitational force measurements over an one-dimensional space. By answering the questions from the worksheet; mainly keeping track of the numerical errors throughout the computations, we reconstructed

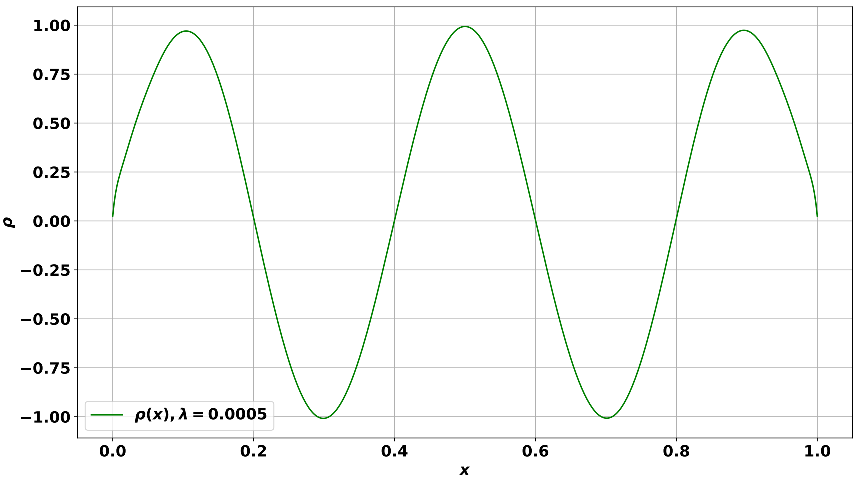

Figure 7: Reconstruction of ρ using Tikhonov regularization and Newton's divided differences, with $\lambda = 5 \cdot 10^{-4}$. The plot reveals a harmonic behaviour of ρ.

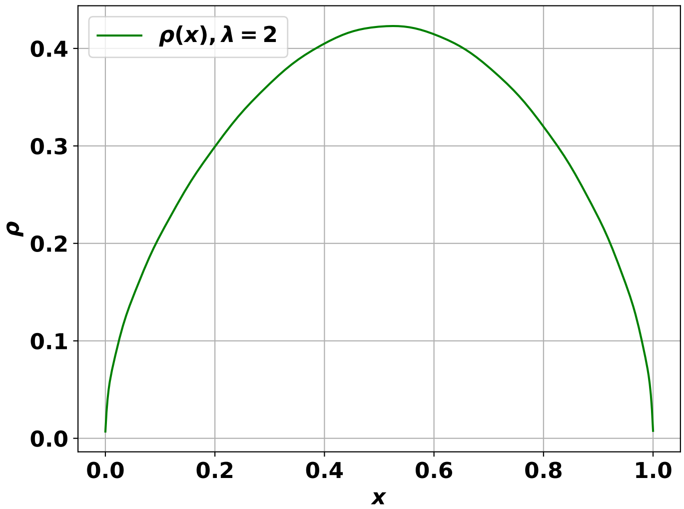

Figure 8: Reconstruction of ρ using Tikhonov regularization and Newton's divided differences, with $\lambda = 2$. The plot reveals a parabolic/square shape of ρ.

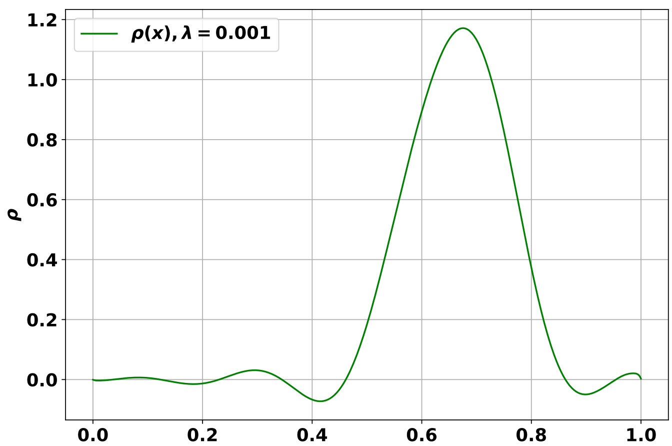

Figure 9: Reconstruction of ρ using Tikhonov regularization and Newton's divided differences, with $\lambda = 10^{-3}$. The plot reveals a Gaussian shape of ρ.

the mass-density in three different areas with sufficiently good accuracy to recognize a harmonic, a parabolic/square and a Gaussian- distributed mass-density.

References

[Sau11] T. Sauer. "Numerical Analysis". In: 2nd ed. Addison-Wesley Publishing Company, 2011.