TMA4320 Intro to Scientific Programming
Project 1: Gravity Surveying

Adrian Kirkeby, Anton Evgrafov

January 30, 2018

1 Introduction

Many, if not most, interesting problems in applied mathematics and science are inverse problems. An inverse problem could be defined as follows:

Given an effect (observation), find the cause.

In this project we will look at an inverse problem from the field of geophysics. We will work with a simple model that relates gravity measurements to the mass density distribution in the ground, and explore one way of finding the latter from inexact measurements.

2 Measurement model

Imagine a point mass supported on a spring in such a way, that it can only move in the direction aligned with the spring axis. By measuring the displacement of this point we can measure one component of the gravitational force acting on it.[1]

The gravitational force acting on our spring supported object occurs due to surrounding mass described by the unknown mass distribution. In principle all objects in the universe will contribute to the resulting force. If the point mass M is in the position $\vec{\xi}_0$ and is acted upon by another point mass $\rho(\vec{\xi})\mathrm{d}\vec{\xi}$ located at the position $\vec{\xi}$, the resulting force contribution will be

$$d\vec{F}(\vec{\xi}) = GM\rho(\vec{\xi})\frac{\vec{\xi}-\vec{\xi}_0}{|\vec{\xi}-\vec{\xi}_0|^3}\mathrm{d}\vec{\xi}, \tag{1}$$

where G is the universal gravitational constant. We can see from (1) that the gravitational force decays quadratically with the distance, therefore we will ignore the objects located "far enough" from the device. In fact, we will consider the following simplified one dimensional model[2] of this experiment, shown in Figure 1. In this simplification, we will attempt to find the unknown mass distribution $\rho : I \to \mathbb{R}$ on the interval of interest $I =]a, b[$ by performing measurements of the vertical component of the gravitational force $F : I \to \mathbb{R}$. To simplify the notation we use $M = G^{-1}$, and we assume a constant vertical distance $d > 0$ between the mass distribution and the measurement device. Thus the vertical component of $\vec{\xi} - \vec{\xi}_0$ in (1)

[1] The Hook's law of linearized elasticity postulates that the force $f = k\Delta x$, where Δx is the displacement of the mass, and k is the *stiffness* of the spring, which has to be measured prior to the gravity surveying.

[2] This model fits rather nicely with, but neither proves nor condones the use of Flat Earth model.

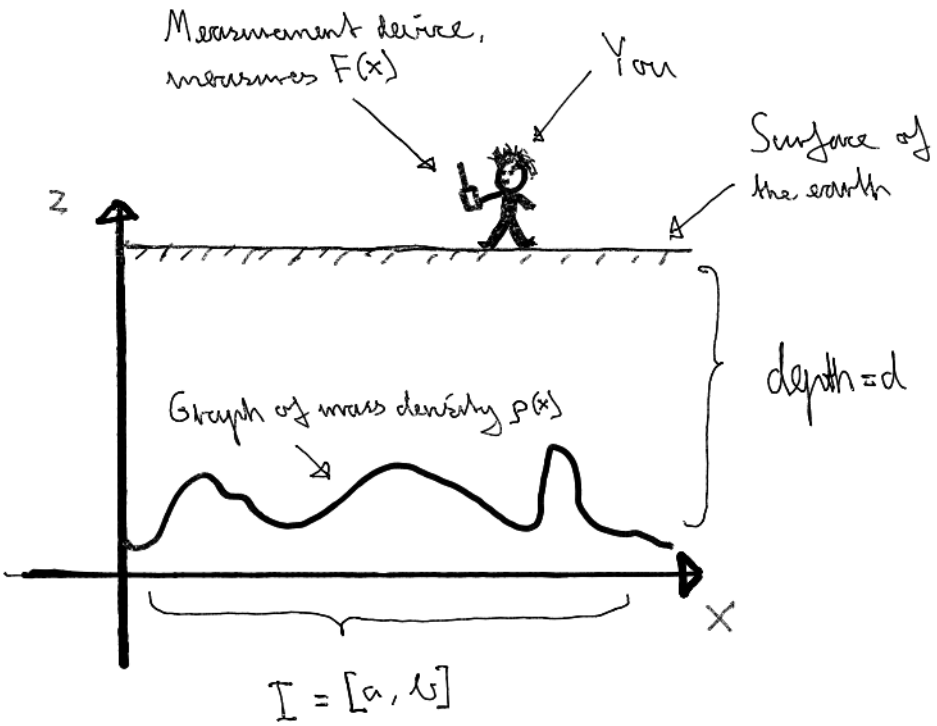

Figure 1: Sketch of geophysical surveying process.

always equals to d[3], and the distance between the measurement device at point $x \in I$ and a mass at point $y \in I$ equals $(d^2 + (x-y)^2)^{1/2}$. Substituting all this into (1) we obtain that the measurement device at point $x \in I$ will measure the total vertical force from the mass distribution $\rho(\cdot)$ which equals to

$$F(x) = \int_a^b \frac{d}{(d^2 + (y-x)^2)^{3/2}} \rho(y) \mathrm{d}y. \tag{2}$$

This equality has to hold at all measurement points, that is, $\forall x \in I$.

Equation (2) relating the known measurements $F(\cdot)$ and the unknown mass distribution $\rho(\cdot)$ is known as *Fredholm integral equation of the first kind* after the Swedish mathematician Ivar Fredholm. The function $K(x,y) = d(d^2 + (y-x)^2)^{-3/2}$ is known as the *kernel* of this equation.

Question 1

To get some insight into the measurement model (2) we will consider the following example. Let the mass density $\rho(\cdot)$ on the interval $I =]0,1[$ be given by the equation

$$\rho(x) = \begin{cases} 1, & \text{if } 1/3 < x < 2/3, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

[3]This is of little importance, but strictly speaking the vertical coordinate axis is pointing down with such a sign convention.

Use the fact that an antiderivative

$$\int K(x,y)\,\mathrm{d}y = \frac{y-x}{d(d^2+(x-y)^2)^{1/2}}, \tag{4}$$

to compute and show on the same plot the corresponding measurement function $F(\cdot)$ for the depths $d = 0.025, 0.25, 2.5$. You may want to use the logarithmic scale on the ordinate axis[4] as the scales of $F(\cdot)$ vary a lot for different d.

3 Spectral collocation discretization of (2)

There are several steps in the discretization of (2).

1. Instead of considering infinitely many equations (2) (one for each $x \in I$), we select some distinct *collocation* points x_i^c, $i = 0, \ldots, N_c - 1$, and consider a system of N_c equations (2).

2. Similarly, instead of considering infinitely many unknowns (one $\rho(x)$ for each $x \in I$) we select some distinct *source* points x_j^s, $j = 0, \ldots, N_s - 1$. We then assume that

$$\rho(x) = \sum_{j=0}^{N_s-1} \hat{\rho}_j L_j(x), \tag{5}$$

   where $L_j(\cdot)$ is the j^{th} Lagrange basis polynomial

$$L_j(x) = \frac{\prod_{m\neq j}(x - x_m^s)}{\prod_{m\neq j}(x_j^s - x_m^s)}. \tag{6}$$

   In this way we have reduced the problem to finding N_s unknown coefficients $\hat{\rho}_j$, $j = 0, \ldots, N_s - 1$.

3. Finally, in order to evaluate the integral in (2) we need to use a numerical quadrature. Such a quadrature is defined by the nodes $x_k^q \in I$ and weights w_k, $k = 0, \ldots, N_q - 1$. Then for an integrable function $\phi : I \to \mathbb{R}$ we have

$$\int_a^b \phi(z)\,\mathrm{d}z \approx \sum_{k=0}^{N_q-1} \phi(x_k^q) w_k. \tag{7}$$

Question 2

Substitute equations (5)–(7) into (2). This should result in a linear algebraic system

$$\boldsymbol{A}\hat{\boldsymbol{\rho}} = \boldsymbol{b}, \tag{8}$$

where $\boldsymbol{A} \in \mathbb{R}^{N_c \times N_s}$, $\hat{\boldsymbol{\rho}} = [\hat{\rho}_0, \ldots, \hat{\rho}_{N_s-1}]^{\mathrm{T}} \in \mathbb{R}^{N_s}$, and finally $\boldsymbol{b} = [F(x_0^c), \ldots, F(x_{N_c-1}^c)]^{\mathrm{T}} \in \mathbb{R}^{N_c}$.

Determine the expressions for the elements of the matrix $\boldsymbol{A}$ in terms of $K(x,y)$, x_i^c, x_j^s, x_k^q, w_k.

[4] That is, the "vertical axis". For plotting with logarithmic scale see `matplotlib.pyplot.semilogy`

Question 3

Implement the functions computing the left- and right-hand sides of the linear algebraic system (8) in Python. For example, you could use the following interface:

```
import numpy as np

def fredholm_rhs(xc,F):
    """
    Set up the RHS of the system
    INPUT:
      xc: defines collocation points
      F : function defining the geological survey measurements
    OUTPUT:
      vector defining the RHS of the system
    """
    Nc = xc.shape[0]
    b  = np.zeros(Nc)
    # FIXIT: implement the function!
    return b

def fredholm_lhs(xc,xs,xq,w,K):
    """
    Set up the LHS of the system
    INPUT:
      xc   :  collocation points
      xs   :  source points
      xq,w:  numerical quadrature
      K    :  integral kernel
    OUTPUT:
      matrix defining the LHS of the system
    """
    Nc = xc.shape[0]
    Ns = xs.shape[0]
    A  = np.zeros((Nc,Ns))
    # FIXIT: implement the function!
    return A
```

Verify your implementation by utilizing the test density $\rho(x) = \sin(\omega x)\exp(\gamma x)$ with $\omega = 3\pi$ and $\gamma = -2$, and the analytical technique for computing the corresponding gravity force described in Appendix A (the computation of $F(\cdot)$ for this $\rho(\cdot)$ is also implemented in `test_example.py` to the absolute accuracy of approximately 10^{-10}). Put $N_c = N_s = 40$, and let $a = 0$, $b = 1$, and $d = 0.025$. Let x_i^c and x_j^s be Chebyshev's interpolation nodes on the interval $]a, b[$ of order N_c and N_s, respectively. Let x_k^q and w_k be given by a mid-point Newton–Côtes quadrature on $]a, b[$ with N_q panels. Plot $\max_{0\le i\le N_c-1} |F(x_i^c) - (\boldsymbol{A}\hat{\boldsymbol{\rho}})_i|$ as a function of N_q, where $\hat{\boldsymbol{\rho}} \in \mathbb{R}^{N_s}$ is the vector of coefficients in the expansion (5), corresponding to $\rho(\cdot)$.[5] Make sure that the graph goes to 0 when you increase the number of panels.

[5] Using `numpy`, the quantity $\max_i |v_i|$ can be easily computed as `numpy.linalg.norm(v,np.Inf)`

Question 4

Repeat the previous exercise, but use Legendre–Gauss quadrature instead. Nodes and weights of this quadrature on the interval $]-1,1[$ can be obtained with

```
import numpy as np
x,w = np.polynomial.legendre.leggauss(Nq)
```

Note that the nodes/weights need to be shifted/scaled to transform this quadrature to an arbitrary interval $]a,b[$. Compare the speed of convergence with that you observe in the previous question.

4 Direct inversion

By now we should have a grasp of the forward problem, that is, the problem of finding the measurement from a specific mass distribution. We will now focus on the inverse problem:

Given gravity measurements collected in a vector $\boldsymbol{b}$,
find the mass distribution $\hat{\boldsymbol{\rho}}$ such that $\boldsymbol{b} \approx \boldsymbol{A}\hat{\boldsymbol{\rho}}$.

The plots you have obtained in Question 1 should provide you with some intuition about this inverse problem, namely about the fact that for increasing values of parameter d the inverse problem may be significantly more difficult to solve than for small values. This is indeed the case, as the exercises in this section demonstrate.

We will begin by trying to simply solve the linear algebraic system (8).[6] In order to have the same number of equations and unknowns, we will set $N_c = N_s$. Furthermore, for simplicity we will use the same source and collocation points (i.e. $x_i^c = x_i^s$, $i = 0, \ldots, N_c - 1$), which coincide with Chebyshev interpolation nodes on $]a,b[$.

Question 5

In this question we will focus on the ideal situation when our measurements are *exact*, i.e. they do not contain any errors[7]. Repeat this numerical experiment for $d = 0.025, 0.25, 2.5$. For $N_c = N_s = 5, 6, \ldots, 30$, compute $\boldsymbol{b} \in \mathbb{R}^{N_c}$ corresponding to the function $F(\cdot)$ used in Questions 3 and 4. Solve the linear system (8) to get the vector $\hat{\boldsymbol{\rho}}$, and compute the errors between the numerical and the true solution, $\max_{0 \le j \le N_s - 1} |\hat{\boldsymbol{\rho}}_j - \rho(x_j^s)|$, where $\rho(\cdot)$ is the same as in Questions 3 and 4. Plot this error as a function of N_c.[8] To obtain maximal possible accuracy it is important to use sufficiently many points in the quadrature (see Questions 3 and 4); we obtained satisfactory results with Legendre–Gauss quadratures and $N_q = N_c^2$.

Question 6

In order to investigate further the behaviour of the inverse problem for various values of the parameter d we will perturb the right hand side of the system (8) with a small random error, which models real-life situations when measurements are not ideal.

[6] Numpy provides a built in linear algebraic solver `numpy.linalg.solve(A,b)`, which is based on Gaussian elimination.

[7] Apart from a tiny 10^{-10} error related to the truncated Taylor series based approximation of the integral, see Appendix A.

[8] See `matplotlib.pyplot.semilogy`

Namely, let $\boldsymbol{b} \in \mathbb{R}^{N_c}$ be the right hand side of the system (8) used in Question 5. In this part of the project, we will use a different right hand side in our computation of the density, namely $\tilde{\boldsymbol{b}} = \boldsymbol{b}(1 + \boldsymbol{\varepsilon})$, where $\boldsymbol{\varepsilon} \in \mathbb{R}^{N_c}$ is a random vector with statistically independent components, each uniformly distributed on the interval $[-\delta, \delta]$[9] with $\delta = 10^{-3}$, i.e. 0.1% perturbation of the measurements. Other than this change, take the same set up as in the previous question, with $N_c = N_s = 30$, and $d = 0.025, 0.25, 2.5$.

Plot $\boldsymbol{b}$ and $\tilde{\boldsymbol{b}}$ as functions of x, and (on a different plot or plots) the analytical solution $\rho(\cdot)$ as well as the solutions to the non-perturbed and perturbed linear systems (8) as functions of x.

What do your results say about the error amplification for the inverse problem, for different parameters d?

5 Tikhonov regularization

The results of the numerical experiments in the previous section may be completely unsatisfactory, and there is a good reason for this. Indeed, the inverse problem (2), or Fredholm integral equation of the first kind is *ill-posed* in the sense that the error amplification factor is infinite! Our discretizations of this problem simply mimic this fact to certain "accuracy".

In order to solve such ill-posed problems in practice, one needs special *regularization* techniques. In this section we will consider one classical technique, called *Tikhonov regularization*.[10] Namely, instead of trying to satisfy the system (8) exactly, we will try to make the difference between the left and the right hand sides as small as possible, while also demanding that the expansion coefficients in (5) are not too large:

$$\underset{\hat{\boldsymbol{\rho}} \in \mathbb{R}^{N_s}}{\text{minimize}} \frac{1}{2} \|\boldsymbol{A}\hat{\boldsymbol{\rho}} - \boldsymbol{b}\|^2 + \frac{\lambda}{2} \|\hat{\boldsymbol{\rho}}\|^2, \tag{9}$$

where parameter $\lambda > 0$, the regularization parameter, quantifies our desire to compromize between the accuracy in (8) and the possibility of obtaining huge (and likely useless) answers $\hat{\boldsymbol{\rho}}$.

Despite looking vastly different from (8), the optimization problem (9) also reduces to solving a system of linear algebraic equations. Indeed, you probably remember that at a point of minimum the derivative (gradient) of the function must be zero. Differentiating the function in (9) and applying this characterization we arrive at the system:

$$(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda \boldsymbol{I})\boldsymbol{\rho} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}, \tag{10}$$

where $\boldsymbol{I}$ is the identity matrix.[11]

There exists no perfect choice of the parameter λ in general, but some choices evidently work better than others. Indeed, if λ is too large, then the answer to the problem above will likely to be small and mostly unrelated to the system (8), whereas if λ is too small, we will be close to the situation explored in the previous section.

[9] Use `numpy.random.uniform(-delta,delta,Nc)` to generate $\boldsymbol{\varepsilon}$

[10] Named after a Soviet mathematician and geophysicist Andrey Nikolayevich Tikhonov.

[11] See `numpy.eye`

Question 7

Consider the setup of Question 6 for the problem with $d = 0.25$, and eventually with $d = 2.5$. Use Tikhonov regularization on the perturbed problem for different values of parameter λ in the range $10^{-14}, \dots 10^{1}$. Solve the regularized problem using (10) and plot the error $\max_j |\rho(x_j^s) - \hat{\boldsymbol{\rho}}_j|$ as a function of λ in the log-log scale.[12] Plot the solution with the smallest error, and compare it with your result from the previous exercise. Note that because the perturbation error is random, you could get different values of λ corresponding to the smallest error for different numerical experiments. Try to choose a value, which works reasonably well for all random perturbations you encounter. Note that the "optimal" value of the perturbation parameter will depend on d. We do not look for a very accurate value of the regularization parameter here, one or two significant digits are sufficient.

6 Reconstruction from given measurements

Question 8

Download the '`*.npz`' files from the course wiki-page. Each file contains five variables: endpoints of the interval $I =]a, b[$, measurement depth d, measurement positions x_i^c, and the corresponding (noisy) measured gravity force $F(x_i^c) + \varepsilon_i$. From this data, try to reconstruct the unknown mass density $\rho(\cdot)$ using Tikhonov regularized spectral collocation method you have implemented. Plot the reconstructed density (for each of the files) in your report.

Note: the collocation points do not have to be the same as source points. You may need to experiment with the regularization parameter λ (as in Question 7) to obtain usable reconstructions. Among the examples we have a harmonic wave, a Gaussian, and a "square window" (as in Question 1).

The measurement can be accessed as follows:

```
import numpy as np

file_name = 'q8_test.npz'
f         = open(file_name, 'rb')
npzfile   = np.load(f)

print(npzfile.files)

print(npzfile['a'], npzfile['b'], npzfile['d'])
print(npzfile['xc'])
print(npzfile['F'])
```

The program above (depending on the file's content) can result in the following output:

```
['a', 'b', 'd', 'xc', 'F']
0 2 2.5
[ 0.    0.25  0.5   0.75  1.  ]
[-0.06017427  0.1067195  -0.60081977  0.86997551  0.73244468]
```

[12] For plotting, use `matplotlib.pyplot.loglog`; for generating points λ, which are spaces uniformly on a logarithmic scale, use `numpy.geomspace(lam_min,lam_max,N_points)`

A Construction of test examples

In terms of special functions, we can express the following antiderivative "analytically":[13]

$$\int \frac{u^n \,\mathrm{d}u}{(d^2+u^2)^{3/2}} = \frac{u^{n+1}\Gamma(n/2+1/2)}{2d^3\Gamma(n/2+3/2)}\,{}_2F_1(3/2, n/2+1/2; n/2+3/2; -(u/d)^2), \tag{11}$$

where $\Gamma(\cdot)$ is Gamma function[14], and ${}_2F_1(\cdot)$ is a Hypergeometric function[15].

We can use this information for evaluating the integrals of the type

$$\int_a^b K(x,y)p(y)\,\mathrm{d}y = d\int_{a-x}^{b-x} \frac{p(u+x)\,\mathrm{d}u}{(d^2+u^2)^{3/2}}, \tag{12}$$

where $p(\cdot)$ is an arbitrary polynomial function; indeed the second integral then reduces to a finite sum of the integrals of the type (11). For example, if $p(x) = c_0 + c_1 x$ then

$$d\int_{a-x}^{b-x} \frac{p(u+x)\,\mathrm{d}u}{(d^2+u^2)^{3/2}} = d(c_0+c_1x)\int_{a-x}^{b-x} \frac{\mathrm{d}u}{(d^2+u^2)^{3/2}} + dc_1\int_{a-x}^{b-x} \frac{u\,\mathrm{d}u}{(d^2+u^2)^{3/2}}. \tag{13}$$

Further, by letting $p(\cdot)$ to be a truncated Taylor series (or another polynomial approximation, such as an interpolation polynomial) of some smooth function $\rho(\cdot)$, we can in theory approximate the integral staying in the right hand side of (2) arbitrarily accurately. Such an approach is implemented in the supplemental file `test_example.py` using the symbolic math package `sympy`.

This is of course an interesting exercise, however numerically it does not work very well. A much more precise and efficient approach is to simply approximate the integral (2) using a numerical quadrature of high enough order N_q, that is

$$\int_a^b K(x,y)\rho(y)\,\mathrm{d}y \approx \sum_{k=0}^{N_q-1} K(x, x_k^q)\rho(x_k^q)w_k, \tag{14}$$

see Section 3.

[13] In the end, some numerical method is needed any ways to evaluate special functions.

[14] `scipy.special.gamma(·)`

[15] `scipy.special.hyp2f1(·,·,·,·)`