MAT3110 FALL 2023 Oblig 2

Hand in deadline

Thursday October 26, 2023, 14:30, uploaded to Canvas.

Instructions

You can choose between scanning handwritten notes or typing the solution directly on a computer (for instance with LaTeX). **The assignment must be submitted as a single PDF file.** Scanned pages must be clearly legible. The submission must contain your name, course and assignment number.

It is expected that you give a clear presentation with all the necessary explanations. Remember to include all relevant plots and figures. All aids, including collaboration, are allowed, but the submission must be written by you and reflect your understanding of the subject. In exercises where you are asked to write a computer program, you need to hand in the code along with the rest of the assignment. (Add the code to the single pdf.) You can use your programming language of choice.

There is only one attempt to pass the assignment and you must have a score of at least 60% to pass it.

Application for postponement

If you need to apply for a postponement of the submission deadline due to illness or other reasons, you have to contact the Student Administration at the Department of Mathematics (e-mail:`studieinfo@math.uio.no`) well before the deadline.

Both mandatory assignments in this course must be approved in the same semester before you are allowed to take the final examination.

Complete guidelines on compulsory assignments

For further details on the hand in of compulsory assignments, see:
`https://www.uio.no/english/studies/examinations/compulsory-activities/mn-math-mandatory.html`

Problems

Problem 1.

Consider the matrix

$$A = \begin{bmatrix} 5 & 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 5/2 & 7/2 \\ -1/\sqrt{2} & 7/2 & 5/2 \end{bmatrix}$$

a) Find a Householder transformation H such that $B = HAH^T$ is a tridiagonal matrix. This involves some bookkeeping, so it may help to use the aid of a computer for calculating the linear algebra in steps that you describe.

b) Use Gershgorin's second theorem in combination with a similarity transformation to estimate the spectrum of A. Show that all eigenvalues are distinct and describe the location of the eigenvalues as accurately as you can.

Hint: The matrix

$$T_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

combined with B for a suitable $\alpha > 0$ may be helpful.

c) Computer exercise: Approximate the largest eigenvalue of A using power iteration and approximate the other two eigenvalues with inverse iteration (a shift is needed for the middle eigenvalue). Draw a random vector $x^{(0)}$ as start vector (for instance, with the components being independent, standard normal distributed). Include your implementation code and the output of 10 iterations for each approximation.

Problem 2

Problem introduction (1 page)

Curve-fitting least squares problems are either linear or nonlinear, depending on the class of functions one seeks to fit over. To motivate a numerical method for solving such nonlinear problems, let us first describe a linear least squares problem for curve fitting.

Let $f(x;\beta) = \beta_1 + \beta_2 x + \beta_2 x^2$ where $\beta \in \mathbb{R}^3$ are unknown coefficients that we seek to fit to $m \geq 3$ measurements $b_1, \ldots, b_m$ so that the following holds as least squares sense

$$f(x_i;\beta) = b_i \quad i = 1, 2, \ldots m. \tag{1}$$

The least squares problem is in other words to find a $\beta \in \mathbb{R}^3$ that minimizes

$$S(\beta) = \sum_{i=1}^{m} (r_i(\beta))^2, \tag{2}$$

where $r_i(\beta) = f(x_i;\beta) - b_i$ denotes the residual for the i-th equation. This is a linear least squares problem because $f(x;\beta)$ is a linear function in β_1, β_2 and β_3; it is indeed equivalent to the least squares problem

$$A\beta = y$$

where

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 \end{bmatrix}.$$

When on the other hand the function $f(x;\beta)$ is nonlinear in at least one of the β_i parameters, then the curve-fitting least squares problem also becomes a nonlinear. Closed form solutions are not generally available for these nonlinear problems, but the Gauss–Newton method is often a suitable numerical method. The method proceeds as follows: Suppose now that $\beta \in \mathbb{R}^n$ for some $n \geq 1$ (i.e., that the curve-fitting has n degrees of freedom) and let

$$R(\beta) := \begin{bmatrix} r_1(\beta) \\ \vdots \\ r_m(\beta) \end{bmatrix} = \begin{bmatrix} f(x_1;\beta) - b_1 \\ \vdots \\ f(x_m;\beta) - b_m \end{bmatrix}$$

and with Jacobian

$$J_R(\beta) \in \mathbb{R}^{m\times n}, \quad \text{with components} \quad (J_R(\beta))_{ij} = \frac{\partial r_i(\beta)}{\partial \beta_j}.$$

Given start guess $\beta^{(0)} \in \mathbb{R}^n$ let

$$\beta^{(k+1)} = \beta^{(k)} - \left(((J_R(\beta^{(k)}))^T J_R(\beta^{(k)})\right)^{-1} (J_R(\beta^{(k)}))^T R(\beta^{(k)}) \quad \text{for} \quad k = 0, 1, \ldots \tag{3}$$

The motivation for this iteration algorithm is that a local minimizer β^* of (2) satisfies that $\nabla S(\beta^*) = 0$, and, under some assumptions, this is indeed the equation that the Gauss–Newton method solves. Noting that

$$\nabla S(\beta) = 2(J_R(\beta))^T R(\beta),$$

and applying Newton's method to solve $F(\beta) = 2(J_R(\beta))^T R(\beta) = 0$ yields the iterations

$$\beta^{(k+1)} = \beta^{(k)} - (J_F(\beta^{(k)}))^{-1} F(\beta^{(k)}). \tag{4}$$

And if $R(\beta)$ is a sufficiently smooth function and β is sufficiently close to β^*, it often holds that the Jacobian of F, $J_F(\beta) \in \mathbb{R}^{n\times n}$, is well-approximated by

$$J_F(\beta) \approx 2J_R(\beta))^T J_R(\beta).$$

Plugging this approximation of J_F into the Newton method (4) produces the Gauss–Newton method (3). Under sufficient regularity and provided $\beta^{(0)}$ is sufficiently close to β^*, one can show that the Gauss–Newton sequence converges to β^*. Such convergence holds for the problem we will study below.

The problem:

a) Consider the function $f(x, y; \beta) = (x - \beta_1)^2 + (y - \beta_2)^2 - \beta_3^2$ with arguments $(x, y) \in \mathbb{R}^2$ and parameters $\beta \in \mathbb{R}^3$. Given β, the level set of all points (x, y) such that $f = 0$ describes the circle centered at (β_1, β_2) and with radius $|\beta_3|$.

We are given the following list of 20 noisy measurements of points on a circle (and the measurements be downloaded as a .mat file from this link)

```
   x          y
-2.3073   -3.5569
-1.6627   -4.5479
 4.4413   -0.1823
-2.1021   -0.7938
-2.0460   -0.3176
 5.1864   -3.9465
-2.6359   -4.3716
-1.0931    0.5035
 1.9392    0.9895
 4.7061   -3.2146
 4.9005   -2.3397
 0.5666   -5.9482
 3.3504   -4.7778
 1.7892    1.4214
 1.2574    0.7869
-1.1229   -4.4437
 0.3167   -6.1661
 0.6024    0.7662
 4.7246   -1.2589
 4.9007   -3.6789
```

and we seek to find the circle β such that

$$f(x_i, y_i; \beta) = 0 \qquad i = 1, \ldots, 20 \tag{5}$$

holds in least squares sense. Describe the function $R(\beta)$ for the nonlinear least squares problem for determining this circle and compute its Jacobian $J_R(\beta) \in \mathbb{R}^{20\times 3}$.

Extracting data points from file: For part b) and c) of this exercise. If you prefer to download the "circle-measurements.mat" from the aforementioned link rather than copying the above measurements, then, assuming you store the downloaded file in the same folder as you run Matlab or Pyton from, you can open it and extract its information by the command **load('circle-measurements.mat')** in Matlab, and by the following commands in Python:

```
import scipy.io
measurementData = scipy.io.loadmat('circle-measurements.mat')
x = measurementData['x']
y = measurementData['y']
```

b) Find a reasonable guess for $\beta^{(0)}$ and compute the circle β^* that best describes the measurements in least squares sense using the Gauss–Newton method (3) on a computer. Include one plot of containing the circle β^* you obtain and the measurement points.

Hint: The intervals $(\min_i x_i, \max_i x_i)$ and $(\min_i y_i, \max_i y_i)$ may point you to a reasonable start guess. A circle β may for instance be plotted using polar coordinates as follows in Matlab (and probably similarly in Python):

```
phi = linspace(0,1);%100 points \theta_i = i/99 on interval [0,1]
s1 = beta(1) + beta(3) * cos(2*pi*phi);
s2 = beta(2) + beta(3)*sin(2*pi*phi);

%plotting measurements points(x,y) as dots
%'*' and the circle (s_1,s_2) as a curve
plot(x,y, '*', s1, s2)
```

c) Some nonlinear least squares problems, such as the one considered here, can be reformulated as linear least squares problem. Before doing so, let us first assign more intuitive labels to our parameters; let $(x_c, y_c) = (\beta_1, \beta_2)$ denote the center of the circle and let $r = \beta_3$ denote its radius. With this notation, the system of equations (5) can be written

$$(x_i - x_c)^2 + (y_i - y_c)^2 = r^2 \qquad i = 1, \ldots, 20$$

Expanding the squares and introducing the new unknown $w = (r^2 - x_c^2 - y_c^2)/2$, the equations can also be written

$$x_i x_c + y_i y_c + w = \frac{x_i^2 + y_i^2}{2} \qquad i = 1, \ldots, 20.$$

Solve the linear least squares problem with unknowns (x_c, y_c, w) on a computer and use the equation for w to determine the circle's radius, r. How does the solution of this linear least squares problem compare to what you obtained in part b)?

Problem 3.

Interpolation and numerical integration in 2D.

a) Let $\mathcal{P}_{n,n} = \{\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x^i y^j \mid c_{ij} \in \mathbb{R} \quad 0 \le i, j \le n\}$ denote the set of 2-variate polynomials in x and y that are of degree $\le n$ in x and of degree $\le n$ in y.

For a given 2D square $[0,1]^2$ with a square mesh $(x_i, y_j) = (ih, jh)$ for $0 \le i, j \le n$ and $h = 1/n$, and $f \in C([0,1]^2)$, consider the interpolation problem: find $p \in \mathcal{P}_{n,n}$ such that

$$p(x_i, y_j) = f(x_i, y_j) \qquad \text{for all} \quad 0 \le i, j \le n. \tag{6}$$

We recall from the lectures that this problem has a solution

$$p(x, y) = \sum_{k=0}^{n}\sum_{\ell=0}^{n} f(x_k, y_\ell) L_k(x) \widehat{L}_\ell(y), \tag{7}$$

where $L_k(x)$ is a univariate polynomial in x of degree n and $\widehat{L}_\ell(y)$ is a univariate polynomial of degree n in y.

Describe $L_k(x)$ and $\widehat{L}_\ell(y)$ and show that $p(x, y)$ in (7) with the functions L_k and $\widehat{L}_\ell$ that you define indeed is a solution of (6).

b) Leaning on the fact that any univariate polynomial p of degree $\le n$ that has $n+1$ or more zeros is equal to the 0 function, meaning $p \equiv 0$, show that the solution to the 2 dimensional interpolation problem (6) is unique.

Hint: Let $q \in \mathcal{P}_{n,n}$ be another solution and consider the polynomial $r = p - q$. Then $r \in \mathcal{P}_{n,n}$, so it can be written

$$r(x, y) = \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x^i y^j$$

for some $c_{ij} \in \mathbb{R}$. For any fixed y_ℓ, for $0 \le \ell \le n$, the function $r(x, y_\ell)$ is a univariate polynomial in x of degree $\le n$, namely,

$$r(x, y_\ell) = \sum_{i=0}^{n} \underbrace{\Big(\sum_{j=0}^{n} c_{ij} y_\ell^j \Big)}_{=:a_i(y_\ell)} x^i = \sum_{i=0}^{n} a_i(y_\ell) x^i.$$

How many zeros does $r(x, y_\ell)$ have, what does this imply about the coefficients $a_i(y_\ell)$ for all $0 \le i \le n$, and how can this be used to show that $c_{ij} = 0$ for all $0 \le i, j \le n$?

c) The composite trapezoidal rule in 2D on the square mesh presented above can be described as follows:

$$\int_0^1 \int_0^1 f(x, y) dx\, dy \approx \sum_{i=1}^{n} \sum_{j=1}^{n} \int_{x_{i-1}}^{x_i} \int_{y_{j-1}}^{y_j} p_{i,j}(x, y)\, dx\, dy =: T(n),$$

where $p_{i,j} \in \mathcal{P}_{1,1}$ denotes the unique polynomial that goes through f in the four points (x_{i-1}, y_{j-1}), (x_i, y_{j-1}), (x_{i-1}, y_j) and (x_i, y_j).

Determine $p_{i,j}(x, y)$ and verify that the 2D square-mesh trapezoidal rule is given by

$$T(n) = \frac{h^2}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} \Big(f(x_{i-1}, y_{j-1}) + f(x_i, y_{j-1}) + f(x_{i-1}, y_j) + f(x_i, y_j) \Big).$$

d) For $n_s = 2^{1+s}$, $\quad s = 1, 2, \ldots, 7$, compute $T(n_s)$ to estimate the integral

$$I = \int_0^1 \int_0^1 \exp(-(x - \sin(y^2))^3)\, dx\, dy.$$

Since we do not know the value of I, we approximate the error $E(s) = |\tilde{I} - T(n_s)|$, using the pseudo-reference solution $\tilde{I} := T(2^{10})$. Estimate numerically the order of convergence r in

$$E(n) = cn^{-r} + \mathcal{O}(n^{-(r+1)})$$

for example, through estimating the slope of the curve $(\log(n_s), \log(E(n_s))$ in a plot, or by studying the ratio

$$r \approx -\frac{\log(E(n_s)) - \log(E(n_{s-1}))}{\log(n_s) - \log(n_{s-1})},$$

which typically becomes more accurate for larger values of s. (See this link for more on using loglog plots for convergence estimates.)