Numerical Analysis (CS 450)

Homework Set 4

Due: April 2, 2014 · Out: March 13, 2014

Problem 1: Optimization Theory (20 points)

(a) Prove that any local minimum of a convex function f on a convex set $\mathcal{S} \subseteq \mathbb{R}^n$ is a global minimum of f on $\mathcal{S}$.

Hint: Suppose a local minimum $\boldsymbol{x}$ is not a global minimum. Let $\boldsymbol{y}$ be a point in $\mathcal{S}$ such that $f(\boldsymbol{y}) < f(\boldsymbol{x})$ and obtain a contradiction considering the line segment between $\boldsymbol{x}$ and $\boldsymbol{y}$.

(b) Let $f : \mathbf{R}^n \to \mathbb{R}$ and $g : \mathbf{R}^n \to \mathbf{R}^m$. Consider the optimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}.$$

Show that the Hessian of the *Lagrangian function* $\mathcal{L} : \mathbf{R}^{n+m} \to \mathbb{R}$

$$\mathcal{L}(\boldsymbol{y}) = \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$$

is given by

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{H}_f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i \boldsymbol{H}_{g_i}(\boldsymbol{x}) & \boldsymbol{J}_g^T(\boldsymbol{x}) \\ \boldsymbol{J}_g(\boldsymbol{x}) & \boldsymbol{O} \end{bmatrix}.$$

Hint: Show this component-by-component, i.e. determine each $\frac{\partial^2}{\partial y_i y_j} \mathcal{L}(\boldsymbol{y})$ in turn, for $1 \leq i, j \leq n + m$. You'll find that you will need to distinguish four cases, corresponding to the four quadrants of the matrix above.

(c) Prove that $\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda})$ cannot be positive definite.

Hint: A matrix $\boldsymbol{A}$ is positive definite if and only if $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$.

(d) What does the non-positive-definiteness of $\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda})$ mean for the solution of the problem $\boldsymbol{\nabla} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{0}$?

(continued on next page. . .)

Problem 2: Optimization on quadratic forms (20 points)

Consider the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\, \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b} + c,$$

where $\boldsymbol{A}$ is an $n \times n$ symmetric positive definite matrix, $\boldsymbol{b}$ is an n-vector, and c is a scalar.

(a) Provide an analytic expression for the gradient ∇f and the Hessian H_f.

(b) At what point does f attain a minimum?

(c) Show that Newton's method for finding the minimum converges in one iteration from any starting point, $\boldsymbol{x}_0$, by showing that the error after one step, $\boldsymbol{e}_1 := \boldsymbol{x}_1 - \boldsymbol{x}^*$, is zero.

(d) Show that one step of steepest descent (for this function) can be written as

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \frac{\boldsymbol{r}^T \boldsymbol{r}}{\boldsymbol{r}^T A\, \boldsymbol{r}} \boldsymbol{r},$$

where $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\, \boldsymbol{x}$. What happens to the error after one step if the initial error, $\boldsymbol{e}_0 := \boldsymbol{x}_0 - \boldsymbol{x}^*$, is an eigenvector of $\boldsymbol{A}$?

(continued on next page...)

Problem 3: Optimization and Nonlinear Data Fitting (20 points)

(a) Write a program to find a minimum of Himmelblau's function,

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{1}$$

using each of the following methods.

(i) Steepest descent

(ii) Newton

(iii) Damped Newton (Newton's method with a line search)

For each method, plot the path taken in the plane by the approximate solutions for the following starting vectors in the same plot.

(i) $\begin{bmatrix} 2 & 2 \end{bmatrix}^T$

(ii) $\begin{bmatrix} 2 & -1 \end{bmatrix}^T$

(iii) $\begin{bmatrix} -2 & 2 \end{bmatrix}^T$

(iv) $\begin{bmatrix} -2 & -2 \end{bmatrix}^T$

Your writeup should include three plots, one for each of the methods. Each plot should have four paths, corresponding to each of the starting vectors and the contours of the function $f(x, y)$ in the range $[-4, 4]$.

Which method performs best? Explain.

(b) Using the Gauss-Newton method for nonlinear least squares, fit the model function

$$f(t, \boldsymbol{x}) = x_1 + x_2 t + x_3 t^2 + x_4 e^{x_5 t} \tag{2}$$

to the following data.

t	y
0.00	20.00
0.25	51.58
0.50	68.73
0.75	75.46
1.00	74.36
1.25	67.09
1.50	54.73
1.75	37.98
2.00	17.28

Your function should terminate with an error if the number of iterations exceeds 500. Use a tolerance of 10^{-14} to test for convergence.

For each of the starting vectors given below,

(i) $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$

(continued on next page...)

(ii) $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$

(iii) $\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T$

Print the values of x_i, for $1 \leq i \leq 5$ and the number of iterations. Also, plot the computed function $f(t, \boldsymbol{x})$ and the given data points. If the algorithm does not terminate for some starting vector, state that explicitly. What results do you observe with different starting vectors?

You may use `numpy.linalg.lstsq()` to solve the intermediate linear least squares subproblem at each iteration.

(continued on next page...)

Problem 4: Stability of Interpolation (20 points)

The so-called *Lebesgue constant* describes an upper bound on the maximum of an interpolating polynomial of a function whose absolute value is bounded by 1.

To this end, let S be the set of all such functions:

$$S := \left\{ f : [-1,1] \to \mathbb{R} : f \text{ continuous}, \max_{x \in [-1,1]} |f(x)| \leq 1 \right\}$$

For a continuous function $f : [-1,1] \to \mathbb{R}$, let $\Pi(f)$ be the interpolating polynomial with respect to a given set of interpolation nodes $T = \{x_1, \ldots, x_n\}$. Then the Lebesgue constant is the number

$$\Lambda_T := \max_{f \in S} \max_{x \in [-1,1]} |\Pi(f)(x)|.$$

Large Lebesgue constants are bad, because they imply that the interpolant can become large even if the underlying function being interpolated is small. The Lebesgue constant depends on the interpolation nodes. In this problem, we will determine it for a number of such nodal sets, namely:

- Equispaced nodes

  Use equispaced nodes that include the endpoints -1 and 1.

- Chebyshev nodes

  Hint: Chebyshev nodes in the range $[-1,1]$ are given by

  $$x_i = \cos\left(\frac{2i+1}{2n}\pi\right) \tag{3}$$

  for $i = 0, \ldots, n-1$.

- Gauss-Legendre nodes

  Hint: Use `scipy.special.legendre()` to obtain Gauss nodes in the interval $[-1,1]$ as shown below.

  ```
  import scipy.special as ss
  X = ss.legendre(n).weights[:,0]
  ```

  (These will show up again in the chapter on numerical computation of integrals.)

(continued on next page...)

Given a set $T = \{x_1, \ldots, x_n\}$ of n interpolation points in the interval $[-1, 1]$, we can approximate the Lebesgue constant as follows:

1. Find a Lagrange basis for the interpolating polynomials on your nodal set T:

$$l_j(x) = \prod_{i=1, j \neq i}^{n} \frac{x - x_i}{x_j - x_i}.$$

   Realize that all interpolants will have to be sums of these, and each Lagrange polynomial represents a function that is one at exactly one interpolation node.

2. Take the absolute value of each Lagrange polynomial and compute the sum of all such functions:

$$\lambda_T(x) = \sum_{j=1}^{n} |l_j(x)|.$$

   λ_T provides an upper bound for functions that are one at *all* interpolation nodes.

3. Finally, determine the maximum of λ_T over $[-1, 1]$.

$$\Lambda_{n-1}(T) = \max_{x \in [-1,1]} \lambda_T(x)$$

   Approximate this maximum by using an equispaced grid of 2,000 points and finding the maximum value of λ_T on this grid.

Compute and plot the (approximate) Lebesgue constant for the above nodal sets with node counts $n = 5, 10, 15, 20, 25$.

Your output should give a single plot showing the computed Lebesgue constants for each of the nodal sets.

Relate your results to the Runge phenomenon.

Hint: Use `matplotlib.pyplot.semilogy()` to plot the values with a linear scale in n on the x axis and a logarithmic scale for the Lebesgue constant on the y axis.

(continued on next page...)

Problem 5: Chebyshev polynomials, Vandermonde matrices (20 points)

(a) A function $F_n(t)$ is known to satisfy the Chebyshev three-term recurrence if the following properties hold.

$$\begin{aligned} F_0(t) &= 1 \\ F_1(t) &= t \\ F_{n+1}(t) &= 2tF_n(t) - F_{n-1}(t) \end{aligned}$$

Show that the function

$$F_n(t) = \cos(n \arccos(t))$$

satisfies the Chebyshev three-term recurrence.

Hint: Remember the addition formula for cosines:

$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta).$$

How would this apply to $\cos((n+1)\arccos(t))$?

(b) Deduce that $F_n(t)$ is a polynomial.

(c) The *generalized Vandermonde matrix* $\boldsymbol{V}$ for a set of points $x_1, \ldots, x_n$ with a set of functions $\varphi_1, \ldots, \varphi_n$ is given by

$$V_{ij} = \varphi_j(x_i).$$

Realize that this matrix captures the essential bit of interpolation: It maps coefficients with respect to the functions (φ_j) to point values at the points (x_i):

$$\sum_{j=1} \alpha_j \varphi_j(x_i) = \sum_{j=1} V_{ij}\alpha_j = (\boldsymbol{V}\boldsymbol{\alpha})_i.$$

Obviously, $\boldsymbol{V}^{-1}$ describes the reverse process, and the conditioning of $\boldsymbol{V}$ can tell us quite a bit about how well-behaved of an operation interpolation is with respect to the given sets of functions and nodes.

Perform the following steps:

(i) Construct $n \times n$ (square!) generalized Vandermonde matrices for $n = 5, 10, 15, \ldots, 100$.
(ii) Plot the condition number of the matrix with respect to n.

for each of the cases below:

(a) Equispaced nodes x_i in the interval $[-1, 1]$ (endpoints included), with the monomials $\varphi_j(x) = x^j$.
(b) Chebyshev nodes (3) in the interval $[-1, 1]$ with the monomials.
(c) Equispaced nodes x_i in the interval $[-1, 1]$ (again, endpoints included), with the Chebyshev polynomials F_j.
(d) Chebyshev nodes (3) in the interval $[-1, 1]$ with the Chebyshev polynomials F_j.

Your output should include one (clearly labeled) plot as described above showing (and comparing) the behavior of the condition number for all four cases.

Hint: Use `matplotlib.pyplot.semilogy()` to plot the values with a linear scale in n on the x axis and a logarithmic scale for the condition number on the y axis.

(d) Which of the combinations performs best?