R. Bergmann, E. Çokaj, M. Ludvigsen

Submission Deadline:
Wednesday 14. September, 12:00 (noon)

Exercise #2

07. September 2022

Note. The exercises can be submitted in groups of three. All solutions have to be written into *one* Jupyter notebook (use Markdown cells for the mathematical answers). This notebook has to state the exercise number and *all group members* (again, up to 3) in the very beginning. One of the group members then submits this Jupyter notebook on Blackboard. **Problem 1.**

a) Let $A \in \mathbb{R}^{n\times n}$ denote an invertible n-by-n dimensional matrix. Compute the gradients ∇f or Jacobi matrices J_F (whichever is suitable) of the following functions:

1) $f\colon \mathbb{R}^n \to \mathbb{R}$ with $f(\mathbf{x}) = \|\mathbf{x}\|$
2) $F\colon \mathbb{R}^n \to \mathbb{R}^n$ with $F(\mathbf{x}) = A\mathbf{x}$.
3) $f\colon \mathbb{R}^n \to \mathbb{R}$ with $f(\mathbf{x}) = \|A\mathbf{x}\|^2$
4) $f\colon \mathbb{R}^n \to \mathbb{R}$ with $f(\mathbf{x}) = (A\mathbf{x}, \mathbf{x}) = \mathbf{x}^T A^T \mathbf{x}$, where $(\cdot,\cdot)$ denotes the inner product.

b) Does the gradient in a) 4. simplify when A is symmetric, that is, $A = A^T$?

c) Further compute the hessian $\nabla^2 f$ of the third and fourth function in the case where A is symmetric.

d) Now let $A \in \mathbb{R}^{m\times n}$ be a rectangular matrix and $y \in \mathbb{R}^m$ a vector. Compute the gradient and hessian of the following function

$$f(x) = \frac{1}{2}\|Ax - y\|_2^2 + \mu\|x\|_2^2,$$

where $\mu > 0$. This function is minimized when doing so-called Tikhonov regularization and/or Ridge regression, commonly used in statistics.

Problem 2.

We want to solve a linear system

$$Ax = b$$

using the iterative Jacobi and Gauss-Seidel methods.

Remember, the Jacobi iteration can be written as

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i}^{n} a_{ij}x_j^{(k+1)}\right),$$

or on the more compact matrix form

$$x^{(k+1)} = D^{-1}(b - (U + L)x^{(k)}),$$

where D is the diagonal of A and U,L are the upper and lower triangular parts of A excluding the diagonal, that is $A = D + U + L$.

We denote

$$B_J = D^{-1}(U + L)$$

the iteration matrix, and we denote the spectral radius of B_J, that is, the largest eigenvalue in absolute value $\rho(B_J)$.

Finally, we remember that Jacobi converges when $\rho(B_J) < 1$.

Consider now the $2n \times 2n$ block matrix

$$A = \begin{bmatrix} C & I \\ I & C \end{bmatrix}$$

where I is the $n \times n$ identity matrix and C is a $n \times n$ diagonal matrix with

$$C = \begin{bmatrix} c_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \end{bmatrix}$$

a) What is B_J in this case?

b) What are the maximal and minimal values allowed for the diagonal elements c_i for the methods to converge? (Hint: Start simple, consider when $n = 1$, so A is a 2×2 matrix. What are the eigenvalues in this case?)

c) A sufficient (but not necessary) condition for Jacobi and Gauss-Seidel to converge is that the matrix A is strictly diagonally dominant, that is

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|.$$

Show that this yields the same conditions on C. Which method of determining convergence was the simplest to apply in your opinion?

Problem 3.

Consider the matrix $A \in \mathbb{R}^{n \times n}$ of the form $a_{ij} = 3^{-|i-j|} + 2^{-i-j} + 10^{-6} \cdot$ `uniform()`, where `uniform()` refers to a random number per entry (independently) uniformly distributed between 0 and 1, cf. `numpy.random.uniform`. Further let $\mathbf{b} = (1, \dots, 1)^{\mathrm{T}} \in \mathbb{R}^n$.

We want to compare a few iterative algorithms. For each algorithm, create a function with identical interface, that is, a function that takes in the same arguments and spits out the same output. Note that these same algorithms will be used in the first project in this subject, so the effort you put into this problem will be reused later!

Richardson iteration (`richardson()`)

Repeat for $k = 0, \dots$ until the stopping criterion is fulfilled

$$\begin{aligned}
\mathbf{r}^{(k)} &= \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)},\\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \omega\mathbf{r}^{(\mathbf{k})}.
\end{aligned}$$

Where ω is a sixth (optional) parameter to the function `richardson()`.

Gauß-Seidel Method (`gaussseidel`)

Repeat for $k = 0, \ldots$ until the stopping criterion is fulfilled

$$x_j^{(k+1)} = \frac{1}{a_{jj}}\left(b_j - \sum_{i=1}^{j-1} a_{ji}x_i^{(k+1)} - \sum_{i=j+1}^{n} a_{ji}x_i^{(k)}\right), \quad j = 1, \ldots, n.$$

Gradient Descent with 'perfect' linesearch (`steepestdescent()`)

Initialize

$$\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$$

Repeat for $k = 0, \ldots$ until the stopping criterion is fulfilled

$$\begin{aligned}
\omega_k &= \frac{(\mathbf{r}^{(k)})^{\mathrm{T}}\mathbf{r}^{(k)}}{(\mathbf{r}^{(k)})^{\mathrm{T}}\mathbf{A}\mathbf{r}^{(k)}},\\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \omega_k\mathbf{r}^{(k)},\\
\mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \omega_k\mathbf{A}\mathbf{r}^{(k)}.
\end{aligned}$$

Conjugate Gradient Iteration (`cg()`) (*Bonus Algorithm, only do this if you have the time.*)

Note: CG only works for Positive (semi-)Definite matrices!

Initialize

$$\mathbf{v}^{(0)} = \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$$

Repeat for $k = 0, \ldots$ until the stopping criterion is fulfilled

$$\begin{aligned}
\omega_k &= \frac{(\mathbf{r}^{(k)})^{\mathrm{T}}\mathbf{r}^{(k)}}{(\mathbf{v}^{(k)})^{\mathrm{T}}\mathbf{A}\mathbf{v}^{(k)}},\\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \omega_k \mathbf{v}^{(k)},\\
\mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \omega_k \mathbf{A}\mathbf{v}^{(k)},\\
\beta_k &= \frac{(\mathbf{r}^{(k+1)})^{\mathrm{T}}\mathbf{r}^{(k+1)}}{(\mathbf{r}^{(k)})^{\mathrm{T}}\mathbf{r}^{(k)}},\\
\mathbf{v}^{(k+1)} &= \mathbf{r}^{(k+1)} + \beta_k \mathbf{v}^{(k)}.
\end{aligned}$$

All the algorithms should stop if either the relative error $\frac{\lVert\mathbf{r}^{(k)}\rVert}{\lVert\mathbf{r}^{(0)}\rVert} < \varepsilon$, where ε is our `tolerance` or a maximum number of iterattions `maxiter` is reached; in your experiments, use

- Matrix size `n=2000`,
- `tol` $= 10^{-7}$,
- `maxiter` $= 5000$.

We always start with $\mathbf{x}^{(0)} = \mathbf{b}$.

We are interested in a comparison of runtime, number of iterations and final error for each of the four methods. Either provide this automatically with your code or provide the results in a Markdown table (copied from your measurements in the cells above.