MATH3204: Assignment 03

Due: 05-Oct-2020 @11:59pm

1. Krylov Sub-space: Prove the following properties of the Krylov sub-space $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$: **[3 marks each]**

(a) **Scaling:** $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) = \mathcal{K}_k(\alpha \boldsymbol{A}, \beta \boldsymbol{v}), \ \alpha \neq 0, \ \beta \neq 0$

(b) **Translation:** $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) = \mathcal{K}_k(\boldsymbol{A} - \mu \boldsymbol{I}, \boldsymbol{v}), \ \forall \mu \in \mathbb{R}$

(c) **Similarity:** $\mathcal{K}_k(\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{B}, \boldsymbol{B}^{-1}\boldsymbol{v}) = \boldsymbol{B}^{-1}\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$ for any invertible matrix $\boldsymbol{B}$

(d) $\mathcal{K}_{k+1}(\boldsymbol{A}, \boldsymbol{v}) = \text{Span}(\boldsymbol{v}) + \boldsymbol{A}\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}), \quad \forall k \geq 1$

(e) If $\boldsymbol{v} \notin \text{Range}(\boldsymbol{A})$, then $\boldsymbol{A}\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) \subsetneq \mathcal{K}_{k+1}(\boldsymbol{A}, \boldsymbol{v}), \quad \forall k \geq 1$.

2. Projection Framework: Consider solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a non-singular matrix and $\boldsymbol{b} \in \mathbb{R}^n$, using the oblique projection framework, i.e., $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$ s.t. $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{r}_k \perp \mathcal{W}_k$, where $\mathcal{W}_k = \boldsymbol{A}\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$. **[5 marks each]**

(a) Show that the matrix $\boldsymbol{W}_k^{\intercal}\boldsymbol{A}\boldsymbol{K}_k$ is non-singular where $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0) = \text{Range}(\boldsymbol{K}_k)$, and $\mathcal{W}_k = \text{Range}(\boldsymbol{W}_k)$.

(b) Show that $\boldsymbol{x}_k$ obtained under this framework can also be equivalently written as

$$\boldsymbol{x}_k = \underset{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)}{\arg\min} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 .$$

(c) Suppose Arnoldi process breaks down at step t, i.e., $h_{t+1,t} = 0$. Show that $\boldsymbol{x}_t$ from this method would be the exact solution, i.e., $\boldsymbol{A}\boldsymbol{x}_t = \boldsymbol{b}$.

3. GMRES: Consider the generalized minimum residual (GMRES) method. Let $\boldsymbol{H}_{k+1,k} = \boldsymbol{U}_{k+1,k}\boldsymbol{R}_k$ be the reduced QR factorization of $\boldsymbol{H}_{k+1,k}$.

(a) Show that the least-squares sub-problems of GMRES, i.e., **[5 marks]**

$$\min_{\boldsymbol{y}} \left\| \boldsymbol{H}_{k+1,k}\boldsymbol{y} - \|\boldsymbol{r}_0\| \, \boldsymbol{e}_1 \right\|,$$

is solved using

$$\boldsymbol{y} = \boldsymbol{R}_k^{-1}\boldsymbol{U}_{k+1,k}^{\mathsf{T}} \|\boldsymbol{r}_0\| \, \boldsymbol{e}_1,$$

where $\boldsymbol{r}_0$ is the initial residual $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ and $\boldsymbol{e}_1 = [1, 0, 0, \ldots, 0]^{\mathsf{T}} \in \mathbb{R}^{k+1}$.

(b) Show that the residual at the k^{th} iteration can be computed as **[5 marks]**

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\| = \|\boldsymbol{r}_0\| \sqrt{1 - \left\|\boldsymbol{U}_{k+1,k}^{\mathsf{T}}\boldsymbol{e}_1\right\|^2}.$$

(c) There are many situations in which mathematical models lead to linear systems where the matrix is non-symmetric. One such example is the numerical discretization of the steady-state convection-diffusion equation. This equation describes physical phenomena that involve interaction among particles of a certain material (diffusion), as well as a form of motion (convection). An example here would be the manner in which a pollutant, say, spreads as it moves in a stream of water. It moves along with the water and other objects, and at the same time it changes its shape and chemical structure and spreads as its molecules interact with each other in a certain manner that can be described mathematically. In its simplest form, the convection-diffusion equation is defined on the open unit square, $0 < x, y < 1$, and reads

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \sigma\frac{\partial u}{\partial x} + \tau\frac{\partial u}{\partial y} = g(x, y).$$

The parameters σ and τ are associated with the convection. When they both vanish, we get the celebrated Poisson equation. One also imposes a certain boundary condition, e.g., homogeneous Dirichlet boundary. By considering a grid of points on the domain, the solution u can be approximated using an appropriate discretization of the partial derivatives on that grid, which gives rise to a linear system of the form $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{g}$. Here, $\boldsymbol{A}$ is a discretization of the differential operator including the boundary conditions, $\boldsymbol{u}$ is a vector whose components are the approximation of u on the discretization grid, and $\boldsymbol{g}$ is a vector whose components are values of g on the grid. Obviously, to get more accurate solutions, one needs to consider a finer grid, which in turn gives rise to a larger-scale linear system.

The function `lcd`($\beta, \gamma,$ `N`) from Assignment 02, Q5-(g), generates the matrix $\boldsymbol{A}$ using a discretization mesh of size $N \times N$, and centered finite difference schemes for both the first and the second partial derivatives, assuming homogeneous Dirichlet boundary. The values of β and γ are related to σ and τ, respectively (as well as the grid width, which is related to the fineness of the grid).

Set $N = 100, \beta = \gamma = 0.1$, and construct the convection-diffusion matrix $\boldsymbol{A}$. To set up the problem, find the right-hand vector $\boldsymbol{g}$ such that the solution $\boldsymbol{u}$ is a vector of all 1's. Now, forget that we actually know $\boldsymbol{u}$ and let's use GMRES to solve the non-symmetric linear system $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{g}$.

[**5 marks for each of the following parts**]

i. Write your own program for implementing GMRES. Write your own Householder reflections for computing the QR factorization of $\boldsymbol{H}_{k+1,k}$. There is no need to worry too much about memory or computational efficiency for this assignment.

ii. Run your code on the above linear system, starting at $\boldsymbol{x}_0 = \boldsymbol{0}$, and stop the iterations when either $\|\boldsymbol{r}_k\| / \|\boldsymbol{b}\| \leq 10^{-6}$ or the iteration count exceeds 1000. Plot the residuals across iterations.

iii. Compare you result with that obtained using `MATLAB`'s `gmres` function or `Python`'s `scipy.sparse.linalg.gmres`. Hopefully, it is identical, otherwise something is wrong in your code.

For non-symmetric matrices, where Arnoldi cannot be replaced by Lanczos, there is a significant price to pay when using GMRES. As we proceed with iterations, we accumulate more and more basis vectors of the Krylov subspace. We need to have those around since the solution is given as their linear combination. As a result, the storage requirements keep creeping up as we iterate. This may become a burden if the matrix is very large and if convergence is not very fast, which often occurs in realistic scenarios. One way of dealing with this difficulty is by using restarted GMRES. We run GMRES as described above, but once a certain maximum number of vectors, say, $m \ll n$, is reached, we take the current iterate, $\boldsymbol{x}_m$, as our initial guess, i.e., $\boldsymbol{x}_0 = \boldsymbol{x}_m$, and start again. Thus at any given time , the maximum number of vectors that are stored is no more than m.

Of course, restarted GMRES is almost certain to converge more slowly than the full GMRES. Moreover, we do not really have an optimality criterion anymore, although we still have monotonicity in residual norm reduction. But despite these drawbacks of restarted GMRES, in practice it is usually preferred over full GMRES, since the memory requirement issue is critical in large scale settings.

iv. Write your own program for implementing restarted GMRES. As above, use your own Householder reflections for computing the QR factorization of $\boldsymbol{H}_{k+1,k}$.

v. Run your code on the same example and the same parameters as what we used above for full GMRES. Set the restart parameter to be $m = 20$. Plot the residuals across iterations on the same plot as full GMRES so you can compare both full and restarted variants all at once across iterations.

vi. Compare you result with that obtained using `scipy.sparse.linalg.gmres` or `MATLAB`'s `gmres` functions using their respective restart parameter. Hopefully, it is identical, otherwise something is wrong in your code.

4. Let $\mathcal{C} \subseteq \mathbb{R}^d$ be a nonempty, closed and convex set. For any $\boldsymbol{y} \in \mathbb{R}^d$, consider the problem of

$$\min_{\boldsymbol{x} \in \mathcal{C}} \|\boldsymbol{y} - \boldsymbol{x}\| .$$

Show the following:

(a) The problem has a solution. **[10 marks]**

(b) This solution is the unique global minimizer (The unique minimizer is called the projection of $\boldsymbol{y}$ onto $\mathcal{C}$, denoted by $[\boldsymbol{y}]$). **[5 marks]**

(c) $[\boldsymbol{y}]$ is the projection of $\boldsymbol{y}$ onto $\mathcal{C}$ if and only if **[5 marks]**

$$\langle \boldsymbol{y} - [\boldsymbol{y}], \boldsymbol{x} - [\boldsymbol{y}] \rangle \leq 0, \ \forall \boldsymbol{x} \in \mathcal{C}.$$

5. Consider a convex function $f : \mathcal{C} \to \mathbb{R}$. **[5 marks each]**

(a) Show that sub-level sets of f are convex.

(b) Show that the set of global optima of f over $\mathcal{C}$ is convex.

Bonus Question 1

6. Consider the same setting as in Question 4. Show the following: **[5 marks each]**

(a) The mapping $f : \mathbb{R}^d \to \mathbb{R}^d$ defined by $f(\boldsymbol{y}) = [\boldsymbol{y}]$ is 1-Lipschitz continuous. In other words, show that

$$\|[\boldsymbol{x}] - [\boldsymbol{y}]\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d.$$

(b) If $\mathcal{C}$ is a sub-space, then $[\boldsymbol{y}]$ is the projection of $\boldsymbol{y}$ onto $\mathcal{C}$ if and only if

$$\langle \boldsymbol{y} - [\boldsymbol{y}], \boldsymbol{x} \rangle = 0, \ \forall \boldsymbol{x} \in \mathcal{C}.$$

Bonus Question 2

7. Let $\boldsymbol{A}$ be a large and sparse matrix. Among modern classes of preconditioners for such matrices, a very popular family is that of incomplete factorizations, e.g., incomplete LU and incomplete Cholesky for, respectively, non-symmetric and PD matrices. The straightforward yet powerful idea behind them is as follows. Recall that a LU or Cholesky factorizations produce factors that are much denser than $\boldsymbol{A}$ in general. The simplest incomplete LU or incomplete Cholesky factorizations, denoted respectively as ILU(0) and IC(0), construct LU or Cholesky decompositions that follow precisely the same steps as the usual decomposition algorithms, except a nonzero entry of a factor is generated only if the matching entry of $\boldsymbol{A}$ is nonzero! So these factorizations completely avoid fill-in. Obviously, the product of these incomplete factors will not give us exactly our $\boldsymbol{A}$ back, but hopefully can serve as a good preconditioner. Incomplete LU and incomplete Cholesky are typically used as preconditioners for GMRES (and other similar methods) and conjugate gradient, respectively.

Use `MATLAB`'s `gmres` and `ilu` functions or `Python`'s `scipy.sparse.linalg.gmres` and `scipy.sparse.linalg.spilu`, to compare the performance of non-preconditioned and ILU(0)-preconditioned GMRES on the example we used in Question 3-(c) (**Note:** As it stands, I am not sure how one can get zero fill-in ILU with `Python`. The closest is to set `drop_tol` = `fill_factor` = `1`. If you could figure out how to get ILU(0) with native `Python` libraries, then please let me know.) Use restart parameter of $m = 20$, relative residual of 10^{-8}, maximum iteration count of 1000. Plot the residuals across iterations for both techniques. What do you observe? **[10 marks]**

100 marks in total + 20 extra marks for the bonus questions

Note:

- This assignment counts for 15% of the total mark for the course.
- You don't have to do the bonus questions, but they may help bring up your mark if you miss some points on other questions.

• Your mark for this assignment will be calculated as

$$\min\{\text{Total marks obtained for all the questions}, 100\} \times 0.15.$$

So with or without bonus, you can only get a maximum of 15%.

• Although not mandatory, if you could type up your work, e.g., `LaTex`, it would be greatly appreciated.

• Show all your work and attach your code and all the plots (if there is a programming question).

• Combine your solutions, all the additional files such as your code and numerical results, along with your coversheet, **all in one single PDF file**.

• Please submit your single PDF file on Blackboard.