PROJETO COMPUTACIONAL #2 – MS993/MT404 – 2S2016 – IMECC/UNICAMP
Matemática Aplicada

Atenção: O projeto #2 pode ser feito em grupos de até no máximo 3 (três) estudantes.

Atividade	Temas	Disponibilização	Entrega
Projeto #2 (**P2**) Exercícios do David S. Watkins, Fundamentals of Matrix Computations, John Wiley & Sons (3 ed., 2010).	SPD matrices, iterative methods, stationary and nonstationary methods. SOR, SSOR, Conjugate Gradient (CG) method and preconditioned Conjugate Gradient (PCG) method. Basics of preconditioner, Accuracy, Finite precision arithmetic. Residual, Convergence, Computational time, Number of operations.	16/Set	30/Set

The primary learning goal of the project. The goal of this task is explore the application of stationary (fixed point) and nonstationary (Krylov subspace) methods applied to the 2D finite difference solver for approximation of a Poisson model problem linked to a partial differential equation by a system of difference equations defined on the grid. Here we have considered only the most basic technique (finite difference) to motivate the very large linear system of equations in the form Ax=b. The focus is on the use of stationary and nonstationary methods for its numerical solution (i.e., its approximation). The student will in turn discuss the use of the following methods: 1) the symmetric successive overrelaxation (SSOR) method, 2) the successive overrelaxation (SOR) method and 3) the conjugate gradient method. In addition, it is expected you learn more about the follwoing issues: Iterative methods, Accuracy, Finite precision arithmetic (see e.g., ref [6,11]), Residuals, Convergence and Basics of Preconditioner (see Lecture 40 ref [11] and Chapter 9 ref[16]); see also the references pointed in this end page of this project. Feel free to try **OTHER** matrices. You can also find many sample matrices at the Matrix Market http://math.nist.gov/MatrixMarket/ or with the gallery command in Matlab.

Additional comments. Which is better: Direct or iterative solver for solving systems of linear equations Ax=b, with matrix A SPD and a nonzero vector b? Why use a preconditioner ?

A broad class of preconditioners is based on incomplete factorizations of the coefficient matrix (see, e.g., [3,5,16]). We call a factorization incomplete if during the factorization process certain fill elements, nonzero elements in the factorization in positions where the original matrix had a zero, have been ignored. Incomplete factorizations are the first preconditioners we have encountered so far for which there is a non-trivial creation stage. Incomplete factorizations may break down (attempted division by zero pivot) or result in indefinite matrices (negative pivots) even if the full factorization of the same matrix is guaranteed to exist and yield a positive definite matrix.

Jacobi Preconditioning (see, e.g., [9,14,15]). The simplest preconditioner consists of just the diagonal of the matrix A. It is possible to use this preconditioner without using any extra storage beyond that of the matrix itself. However, division operations are usually quite costly, so in practice storage is allocated for the reciprocals of the matrix diagonal. The SSOR preconditioner like the Jacobi preconditioner, can be derived from the coefficient matrix A without any work! There is an optimal parameter value (like the parameter in the SOR method), in which reduces the number of iterations to a lower order. Specifically, such optimal parameter is well-known for second order elliptic problems. In practice, however, the spectral information needed to calculate the optimal is prohibitively expensive to compute (see, e.g., [5,9,14]).

Hints. You may do the programming assignments (SOR, SSOR, CG and PCG) for this task in Matlab, C or Fortran. Indeed, you may use the Matlab “pcg” code in ordr to verify your computations. In Matlab “pcg”, do not forget to set the “maxiter” flag to something large enough. You can get the number of iterations used and a vector with the approximate errors after each iteration from the “pcg” routine in the “iter” and “relres” return values; see the “pcg” help.

PRELIMINARIES

Given *Ax=b* (A is SPD and b is a nonzero rhs vector), it follows from the $r_0, \dots, r_{k-1}$ residuals vectors defined in the CG method that form an orthogonal basis of $K^n(A; r_0) = I\!R^n$. In theory the CG method is a finite method. After *n* iterations the Krylov subspace is identical to $K^n(A; r_0) = I\!R^n$. Remember that the A-inner product (see, e.g., [3,5,12,16]) is defined by,

$$(y, z)_A = y^T A z\,,$$

and and the A-norm (see, e.g., [3,5,12,16]) is defined by,

$$\|y\|_A = \sqrt{(y,y)_A} = \sqrt{y^T A y}\cdot$$

Now, since

$$\|x - x_k\|_A = \min_{y \in K^k(A; r_0)} \|x - y\|_A,$$

is minimized over $K^n(A; r_0) = I\!R^n$ the norm is equal to zero and $x_n = x.$ However in practice this property is never utilized for two reasons: firstly in many applications n is very large so that it is not feasible to do n iterations, secondly even if n is small, rounding errors can spoil the results such that the properties given in CG Theorem do not hold for the computed vectors $r_0, \dots, r_{k-1}$. (See, e.g., ref [5] Section 10.2 and ref [16] Chapters 6 and 7). You can find more details about SOR, SSOR and CG, PCG in [8,9,11,15].

The sequence $\|x - x_k\|_A$ is monotone decreasing, but finite arithmetic it may occur that $\|r_{k+1}\|_2$ is larger than $\|r_k\|_2$. This does not mean that the CG process becomes divergent. Indeed, it can be shown however that $\|r_k\|_2$ is less than the monotone decreasing sequence given by $\sqrt{\|A\|_2}\|x - x_k\|_A$, so after some iterations the norm of the residual decreases again. As CG is a stable method, the influence of rounding errors is only recognizable when the norm residuals is close to machine precision (see also Figure 40.1, page 315, Lecture 40 ref [11]; ref [9] and see Figure 8.3, page 123, ref[8]; see Figure 6.7, page 315).

I) STATIONARY METHODS or FIXED POINT METHODS

- **FAZER.** Exercise 8.2.23. (**Hint**: READ the content from Exercise 8.2.23 to Exercise 8.2.28. See also Table 8.5 that appears in Exercise 8.3.12, page 573). (whenever possible use the stopping criteria described in III) below.)
- **FAZER.** For each method, use the three stopping criteria and plot the following figures: (a) “residuals” versus “number of iterations”, (b) Discrepancies of the “two successive iterates” versus “number of iterations”, and (c) the “time-consuming computations” versus “number of iterations”.
- **FAZER.** Consider to use mesh sizes h = 1/8000 and h = 1/16000. What happens ?
- **FAZER.** Construct a Table like the Table 8.3 as appears in page 564. You might use the sama stopping criteria as appears in Example 8.2.21.
- **Hint** a) use a stopping criterion based on the iterations, as such with a message "Stopping criterion: maximum number of iterations satisfied or achieved!"; see III) in what follows.
- **Hint** b) See Exercise 8.2.24 (page 566, D. Watkins, 3rd) for a Matrix representation of SOR; it is useful for preconditioning purposes.
- **Hint** c) See Exercise 8.2.26 (page 567, D. Watkins, 3rd) for a Matrix representation of SSOR; it is useful for preconditioning purposes.

II) NON-STATIONARY METHOD - or KRYLOV SUBSPACE METHODS

- **FAZER.** All the exercises 8.7.9, 8.7.10, 8.7.11 and 8.7.12; consider the stopping criteria in III)

- **Hint:** See Exercise 8.2.24 and Exercise 8.2.26 David S. Watkins. Read also Preconditioned Congugate Gradient method (PCG) at Section 8.7 The Conjugate-Gradient Method, page 602 in David S. Watkins and Conjugate-Gradient Algorithm, eq. (8.7.1), page 602.
- This is far from the most efficient way to use this matrix or to execute the CG algorithm, but it is convenient and works reasonably well on moderate-sized problems.

III) STOPPING CRITERIA

- See Section 8.5 ON STOPPING CRITERIA in David S. Watkins, page 594

Whenever we use an iterative method, we need a criterion for deciding when to stop. In most cases it does not much matter which stopping criterion we use, but there are some situations where it really does matter. Next, consider the three classical stopping criteria as follows to be used in this current project 2.

A) In the first stopping criteria, the termination iteration will cease when two successive iterates (or iterations) are sufficiently close together, i.e. where ϵ is some small (always positive) number like 10^{-8}.

$$\| x^{(k+1)} - x^{(k)} \|_2 < \epsilon \| x^{(k+1)} \|_2 \qquad \textbf{(\#1)}$$

We notice that the iterate $x^{(k+1)}$ of the gradient method is optimal with respect to $r^{(k)}$ since, due to the choice of, we have α_k and $r^{(k+1)} \perp r^{(k)}$ but this property no longer holds for the successive iterate $x^{(k+2)}$ (First, this means that, unfortunately, accumulation of round-off errors due to finite precision arithmetic causes a loss of orthogonality between these vectors. Second, it is then natural to ask whether there exist descent directions that maintain the optimality of the iterations !)

B) Another criterion that is in widespread use is to stop when the residual is small enough is as follows:

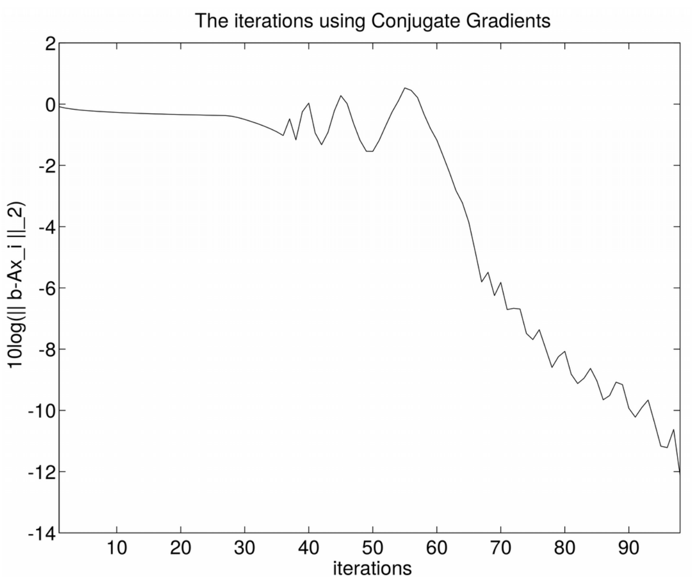
$$\| r^{(k+1)} \|_2 < \epsilon \| b \|_2 \qquad (\#2)$$

It turns out that the above stopping condition is sufficient for backward stability at level ϵ. Thus we have a good justification (check this!) to support the use the above stopping criterium.

C) The third and less stringent criterion is as follows

$$\| r^{(k+1)} \|_2 < \epsilon \left(\| b \|_2 + \| A \|_2 \| x^{(k+1)} \|_2 \right) \qquad (\#3)$$

Indeed, although it is a less stringent criterion, it is necessary and sufficient for backward stability (see Exercise 8.5.3 D. Watkin, 3rd Edition). Thus we also have a very good mathematical justification for its use. (see figure bellow for a ilustration of the typical behavior of the convergence of Conjugate Gradient.)



More additional comments on stopping criteria. In short, we use criterion (#1) because it is simple, traditional, and convenient (less time consuming). but, it does not maintain the optimality condition (orthogonality) along the iterations. On the other hand, stopping criteria (#2) and (#3) have ver nice properties (e.g., optimality condition upon orthogonality of the residuals as well as backward stability). However, a closer look at (#2) and (#3) reveals us the demand for basic operations throughout the iterations. A natural question is: what is the time consuming for each one of the stopping criterium ?

Good additional references on stopping criteria:

1. Tobin A. Driscoll, Kim-Chuan Toh, and Lloyd N. Trefethen, From Potential Theory to Matrix Iterations in Six Steps. SIAM Rev., 40(3) (2006) 547-578
2. M. Arioli, A stopping criterion for the conjugate gradient algorithm in a finite element method framework, Numer. Math. (2004) 97 1-24
3. Strakoš, Zdeněk; Tichý, Petr, On error estimation in the conjugate gradient method and why it works in finite precision computations. ETNA. Electronic Transactions on Numerical Analysis 13 (2002) 56-80.
4. D. Calvetti, S. Morigi, L. Reichel, F. Sgallari, Computable error bounds and estimates for the conjugate gradient method, Numerical Algorithms, 25(1) (2000) 75-88.
5. E. F. Kaasschieter, A practical termination criterion for the conjugate gradient method, BIT Numerical Mathematics, 28(2) (1988) 308-322.

REFERÊNCIAIS (disponíveis na biblioteca do IMECC e com reserva de curso)

[1] Carl D. Meyer, Matrix Analysis and Applied Linear Algebra, Philadelphia, PA, SIAM (2000).
[2] Charles L. Lawson and Richard J. Hanson. Solving least squares problems, Philadelphia, PA, SIAM (1995).
[3] David S. Watkins, Fundamentals of Matrix Computations, New Jersey: John Wiley & Sons (3 ed., 2010).
[4] Erwin Kreyszig. Introductory functional analysis with applications, New York, NY, John Wiley & Sons (1978).
[5] Gene H. Golub and Charles F. Van Loan. Matrix computations, 3rd ed., Johns Hopkins University Press (1996).
[6] Gérard Meurant. The Lanczos and conjugate gradient algorithms: from theory to finite precision computations, Philadelphia, PA, SIAM (2006).
[7] Gilbert Strang. Linear algebra and its applications, 3rd ed., Brooks/Cole, Thomson Learning,(1988).
[8] Henk A. van der Vorst. [recurso eletrônico, digital, PDF file] Iterative Krylov Methods for Large Linear Systems, Cambridge, UK, Cambridge University Press - Monographs on applied and computational mathematics; n. 13, (2003).
[9] James W. Demmel. Applied numerical linear algebra, Philadelphia, PA, SIAM (1997).
[10] Kenneth Hoffman and Ray Kunze. Linear algebra, 2nd ed, Englewood Cliffs, NJ, Prentice-Hall (1971).
[11] Lloyd N. Trefethen, David Bau III. Numerical linear algebra, Philadelphia, PA, SIAM (1997).
[12] Roger A. Horn and Charles R. Johnson. Matrix analysis, Cambridge, MA, Cambridge University Press (1985).
[13] Timothy A. Davis, Direct methods for sparse linear systems (Fundamentals of algorithms Series), SIAM (2006).
[14] William L. Briggs, Van Emden Henson and Steve F. McCormick. A multigrid tutorial, 2nd ed, PA, SIAM (2000).
[15] Wolfgang Hackbusch. Iterative solution of large sparse systems of equations, New York, NY, Springer (1994).
[16] Yousef Saad. Iterative methods for sparse linear systems, 2nd ed. Philadelphia, PA, SIAM (2003).