Advances in Linear Algebra & Matrix Theory, 2014, 4, 156-171
Published Online September 2014 in SciRes. http://www.scirp.org/journal/alamt
http://dx.doi.org/10.4236/alamt.2014.43014

Scientific Research

On the FOM Algorithm for the Resolution of the Linear Systems $Ax = b$

Mongi Benhamadou

Department of Mathematics, Faculty of Sciences of Sfax, Sfax, Tunisia
Email: mongi.benhamadou@fss.rnu.tn

Received 18 July 2014; revised 18 August 2014; accepted 17 September 2014

Abstract

In this paper, we propose another version of the full orthogonalization method (FOM) for the resolution of linear system $Ax = b$, based on an extended definition of Sturm sequence in the calculation of the determinant of an upper hessenberg matrix in $o(n^2)$. We will also give a new version of Givens method based on using a tensor product and matrix addition. This version can be used in parallel calculation.

Keywords

FOM, Krylov Subspace, Hessenberg Matrix, Sturm Sequence, Givens Method

1. Introduction

The resolution of linear systems $Ax = b$ is in the heart of numerous scientific applications: discretization of partiel differentiel equations, image processing, the linearization of nonlinear problems in a sequence of linear problems [1] [2] [3] etc. There are many kinds of methods that resolve linear systems. Some are direct, others are iterative. Generally, direct methods [4]-[6], such as Gauss, Cholesky, QR, are efficient and proved solvers of small size systems. But for problems of big size, these methods require quite prohibitive memory space and therefore become numerically expensive. More and more, they are replaced by iterative methods. Most iterative methods [2] [6] [7] treat linear systems through vector-matrix products with an appropriate data structure permitting the exploitation of sparsity of the matrix A by storing only nonzero elements which are indispensable for the calculation of these products, which reduces both the memory size and the processing time. We distinguish two kinds of iterative methods for the resolution of linear systems:

- The asymptotic methods [6]-[8] are older and simpler to implement. Among the most known ones, we mention Jacobi's, Gauss-Seidel's and the relaxation methods. Generally, these methods remain less reliable than the direct methods, and more or less efficient with some specific problems. Currently, these methods are no

longer used as linear solvers but remain interesting as preconditioners for other iterative methods. The increase of the size of the systems to be solved leads to the projection methods.

- Krylov methods: They are being used for some time and proved to be very efficient [2]. Their analysis is more complicated than that of the asymptotic methods. They are based on a technique of projection on a Krylov subspace, having the form: $K_m(A, r_0) = span(r_0, Ar_0, \cdots, A^{m-1}r_0)$; $dim(K_m(A, r_0)) = m$, spanned by the successive powers of A applied to the risudual vector $r_0 = b - Ax_0$, where x_0 is a vector of $\mathbb{R}^n$, these Krylov's subspaces constitute an increasing sequence of subspaces, one being included in the other and permitting the construction of a sequence of vectors converging towards the solution. The convergence of these methods is theorically assured for $m = n$ maximum. The Krylov methods resolve linear systems having nonsymmetric matrices. The most used Krylov methods based on Arnoldi algorithm are: the full orthogonalization method (FOM) and its varieties: the method of minimal residue (or shortly GMRES), Lanczos method, and that of conjugate gradient for symmetric and symmetric positive definite matrices. These projection methods are more generalisable, more powerful and currently the most used.

In the first section, we will review the orthogonal projection methods on Krylov's subspaces, focusing on the FOM method for the resolution of linear systems.

In the second section, we will extend the definition of Sturm sequence, which will serve for the calculation of the determinant of upper hessenberg matrices.

In the third section, we will give a new variety of Givens method [4] for the resolution of linear systems having an upper hessenberg matrix.

Finally, in the fourth section, we report the results of some numerical tests.

In what follows, we consider a real regular square matrix A, a vector $b \in \mathbb{R}^n$ and the correspondent linear system:

$$Ax = b \tag{1}$$

If x_0 is a starting approximation of solution (1), then the residual vector associated to x_0 is $r_0 = b - Ax_0$. We call Krylov space of order m, noted k_m, the vectorial space spanned by r_0 and its $(m-1)$ iterated by A:

$$K_m(A, r_0) = vect(r_0, Ar_0, \cdots, A^{m-1}r_0). \tag{2}$$

1.1. The Projection Method

In this part, we will review generallities over iterative methods of projection for the resolution of linear systems (1), by using projections in particular subspaces, namely Krylov's spaces [1] [2].

Most actual iterative techniques used in solving large linear systems use in one way or another a projection procedure. The main idea of projection techniques consist in extracting an approximate solution to system (1) of a subspace of $\mathbb{R}^n$. It leads to a small linear system. This is a basic projection step. If we take $K_m(A, r_0)$ of m dimension (the subspace searched for), then, generally, m constraints must be imposed in order to be able to extract such an approximation. One usual way of describing these constraints is to impose to the residual vectors $r = b - Ax$ to be orthogonal to the linearly independent m vectors, this will define another subspace noted L_m with dimension m, which we call "constraints space". We are then seeking an approximate solution x_m to problem (1) by imposing that $x_m \in K_m$ and that the new residual vector $r_m = b - Ax_m$ must be orthogonal to L_m. If, in addition, we want to use the starting approximation x_0, then the approximate solution x_m must be searched for in the affine space $x_0 + K_m$, then the solution of (1) will be characterized as follows:

$$\begin{cases} x_0 : \text{data} \\ \text{find } x_m \in x_0 + K_m \text{ such that :} \\ r_m \perp L_m. \end{cases} \tag{3}$$

As such, the definition of the projection method is based on two conditions:

- The first fixes the "place" where we should look for the solution at the k^{th} iteration: usually, this place is an affine subvariety of $\mathbb{R}^n$. According to Krylov's methods, this affine subvariety is $x_0 + K_k(A, r_0)$, which gives us the following "space condition":

$$x_k \in x_0 + K_k(A, r_0) \tag{4}$$

- The second condition must fix the x_k that is appropriate to that space (4). We hope that x_k minimizes the vector error $e_k = x - x_k$, or the residual vector $r_k = b - Ax_k$ or one or those two vectors be orthogonal to a space L_k of dimension k. This condition is called "Petrov-Galerkin condition", and is expressed as follows:

$$r_k \perp K_k(A, r_0). \tag{5}$$

As for the projection method, we will study in the next part, Arnoldi's method for the resolution of linear systems. Let's start now by recalling Arnoldi's algorithm.

1.2. Arnoldi's Algorithm

Arnoldi's Algorithm is an iterative process consisting in calculating, simultaneously, an orthonormal basis V_m of K_m, and an upper hessenberg matrix H_m. The matrix H_m is a representation of A with respect to this basis.

In practice, constructing Krylov's spaces leads to determining basis. The natural basis $\{r_0, Ar_0, \cdots, A^{k-1}r_0\}$ can never be used, due to its numerical degeneration, and the Krylov's matrices become increasingly ill-conditioned [2]. To avoid this numerical degeneration of the natural basis of Krylov's space, the solution consists in putting in place an orthonormalization process. Arnoldi's basis is then constructed by applying the modified Gram-Schmidt orthonormalization method to vectors obtained by successive products matrix vector for euclidien inner product. The Arnoldi's algorithm takes the following form:

$$(A_{Arn})\begin{cases}
\text{data}: n,\ A \in \mathbb{R}^{n\times n},\ m \le n,\ V_m \in \mathbb{R}^{n\times m} \\
x_0 \in^n,\ r_0 = b - Ax_0,\ h_{jp1j} = 1.0,\ j = 1 \\
\bullet\ \text{compute}: v_1 = \dfrac{r_0}{\|r_0\|_2},\ \varepsilon = 10^{-9},\ V_1 = \{v_1\}. \\
\quad \text{while}\left(h_{jp1j} \ge \varepsilon\right) \\
\qquad \bullet\ \text{compute}: w = Av_j \\
\qquad \text{for } i = 1 \text{ to } j \\
\qquad\quad \bullet\ \text{compute}: h_{ij} = \langle w, v_i \rangle \\
\qquad\quad \bullet\ \text{compute}: w = w - h_{ij}v_i \\
\qquad \text{end } i \\
\qquad \bullet\ \text{compute}: h_{j+1,j} = \|w\|_2 \\
\qquad \text{if } h_{j+1,j} = 0,\ \text{goto}: \text{"end while"} \\
\qquad \bullet\ \text{compute}: v_{j+1} = \dfrac{w}{h_{j+1,j}},\ V_{j+1} := \{v_1, \cdots, v_{j+1}\}. \\
\qquad j = j + 1 \\
\quad \text{end while} \\
//\ \text{output}: m = j,\ \text{matrices}:\ V_m, H_m.
\end{cases}$$

If we assign h_{ij} as coefficient of orthogonalization of Av_j with respect to v_i, $i = 1, 2, \cdots, j+1$ and $h_{j+1,j}$ as norm of vector w obtained by orthogonalization of vector Av_j with respect to vectors v_i, then, the formula of Arnoldi's basis construction can be written:

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i \tag{6}$$

Let V_m be the matrix having n rows and m columns, in which the m columns are the first vectors of the Arnoldi's basis. The orthonormality of vectors, in terms of matrices can be written:

$$^tV_m V_m = I_m \tag{7}$$

Likewise, Equation (6), which defines Arnoldi's vectors, can be translated matricially by:

$$AV_m = V_{m+1} H_{m+1,m} \tag{8}$$

where the matrix $H_{m+1,m}$ is a matrix having $m+1$ rows and m columns where h_{ij} coefficients are the coefficients of orthonormalization of the Equation (6): where $h_{ij} \neq 0$ for $i < j+1$, and the other coefficients being zero.

$$H_{m+1,m} = \begin{pmatrix} h_{11} & h_{12} & \cdots & \cdots & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & \cdots & \cdots & h_{2m} \\ 0 & h_{32} & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & h_{mm-1} & h_{mm} \\ 0 & \cdots & \cdots & \cdots & 0 & h_{m+1m} \end{pmatrix}. \tag{9}$$

The main diagonal block of $H_{m+1,m}$ is a square matrix of dimension m, noted H_m, which, according the Equations (7) and (8), verifies:

$$H_m = {}^tV_m A V_m = \begin{pmatrix} h_{11} & h_{12} & \cdots & \cdots & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & \cdots & \cdots & h_{2m} \\ 0 & h_{32} & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & h_{mm-1} & h_{mm} \end{pmatrix}. \tag{10}$$

Such a matrix is called "*upper hessenberg*": $h_{i,j} = 0 \ \forall i > j+1$.

Equation (10) shows that H_m is the matrix of the projection in Krylov's space K_m of the linear map associated to the matrix *A*, in the basis $V_m = \{v_1, v_2, \cdots, v_m\}$ of Arnoldi's vectors. This can be summed up as follows:

Proposition 1:

In Arnoldi's algorithm (A_{Arn}), we have at the step k the following relations:

$$\begin{cases} AV_k = V_k H_k + h_{k+1,k} v_{k+1} {}^t e_k^{(k)} \\ AV_k = V_{k+1} H_{k+1,k} \\ {}^tV_k A V_k = H_k \end{cases} \tag{11}$$

where $e_k^{(k)}$ is the vector of the canonical basis of $\mathbb{R}^k$

Proof: [2].

1.3. Arnoldi's Method for Linear Systems (FOM)

In this paragraph, we will remind the important results needed for the resolution of linear systems (1).

For that, we will begin by applying the Arnoldi's algorithm (A_{Arn}) to the matrix A, this allows us to obtain at each step k, a matricial pair $(V_k, H_{k+1,k})$ where V_k is an orthonormal basis of Krylov's space K_{k+1}, and $H_{k+1,k}$ is specific an upper hessenberg matrix given by (9).

The full orthogonalization method (FOM) is variant of problem (3), verifying the space condition (4) and "Petrov-Galerkin condition" (5) by taking $L_m = K_m = K_m(A, r_0)$. This method is defined by:

$$\begin{cases} \text{data} : x_0 \in \mathbb{R}^n \\ \forall k = 1, \cdots, m \leq n \\ \text{find } x_k \in x_0 + K_k \text{ such that :} \\ \qquad r_k \perp K_k \end{cases} \tag{12}$$

Production of the Solution

We are seeking the solution x_m verifying the space condition (4) and "Petrov-Galerkin condition" (5).

(4) $\Leftrightarrow x_m \in x_0 + K_m \Leftrightarrow x_m = x_0 + V_m y$ with $y \in \mathbb{R}^m$.

So, the residual vector $r_m = b - Ax_m$ is written:

$$r_m = r_0 - AV_m y, \tag{13}$$

and "Petrov-Galerkin condition" becomes:

(5) $\Leftrightarrow r_m \perp K_m \Leftrightarrow {}^tV_m r_m = 0$, with (13)

$\Leftrightarrow {}^tV_m \left(r_0 - AV_m y\right) = 0$ but then ${}^tV_m AV_m = H_m$

$\Leftrightarrow \left({}^tV_m v_1\right)\beta_1 = H_m y$. This is equivalent to:

$$H_m y = \beta_1 e_1^{(m)}. \tag{14}$$

So, (14) implies $y = \beta_1 H_m^{-1} e_1^{(m)}$, and the solution x_m can be written:

$$x_m = x_0 + \beta_1 V_m H_m^{-1} e_1^{(m)} \tag{15}$$

and the problem becomes:

$$\begin{cases} \text{Given } x_0 \in \mathbb{R}^n : \\ \bullet \text{ Seek } y \in^m \text{ verifying } (14) : H_m y = \beta_1 e_1^{(m)}. \\ \bullet \text{ Compute the solution } x_m = x_0 + V_m y. \end{cases}$$

A method based on this approach is called a full orthogonalization method (FOM). It is given by Y. Saad in 1981.

This is the corresponding algorithm using the Arnoldi's process $\left(A_{Arn}\right)$.

$$\left(A_{fom}\right): \begin{cases} \text{data} : n,\ A \in \mathbb{R}^{n\times n},\ m \le n,\ V_m \in \mathbb{R}^{n\times m} \\ x_0 \in \mathbb{R}^n,\ r_0 = b - Ax_0,\ h_{jp1\,j} = 1.0,\ j = 1 \\ \bullet \text{ compute} : v_1 = \dfrac{r_0}{\|r_0\|_2},\ \varepsilon = 10^{-9},\ V_1 = \{v_1\}. \\ \quad \text{while } \left(h_{jp1\,j} \ge \varepsilon\right) \\ \quad\quad \bullet \text{ compute} : w = Av_j \\ \quad\quad \text{for } i = 1 \text{ to } j \\ \quad\quad\quad \bullet \text{ compute} : h_{ij} = \langle w, v_i \rangle \\ \quad\quad\quad \bullet \text{ compute} : w = w - h_{ij} v_i \\ \quad\quad \text{end } i \\ \quad\quad \bullet \text{ compute} : h_{j+1,j} = \|w\|_2 \\ \quad\quad \text{if } h_{j+1,j} = 0, \text{ "Lucky breakdown" then } m = j; \text{ goto } ® \\ \quad\quad \bullet \text{ compute} : v_{j+1} = \dfrac{w}{h_{j+1,j}},\ V_{j+1} := \{v_1, \cdots, v_{j+1}\}. \\ \quad\quad h_{jp1\,j} = h_{j+1,j},\ j = j+1 \\ \quad \text{end while} \\ / / \text{ output} : m = j,\ \text{matrices} : V_m, H_m \\ ® : \text{ compute} : H_m y = \beta_1 e_1 \\ \quad\ \text{compute} : x_m = x_0 + V_m y. \end{cases}$$

The algorithm $\left(A_{fom}\right)$ depends on parameter m, which is the dimension of the Krylov's subspace. In practice, it is desirable to select m in a dynamic way. This is possible if the norm of the residue r_m of the

solution x_m is calculated with a numerically low cost. (without having to compute x_m itself). Then the algorithm can be stopped at the appropriate step using this information. The following result gives an answer in this direction.

Proposition 2:

The residue vector r_m of the approximate solution x_m calculated by the algorithm $\left(A_{fom}\right)$ is such that:

$$r_m = b - Ax_m = -h_{m+1,m}\,{}^t e_m^{(m)} y v_{m+1},$$

and so,

$$\left\|b - Ax_m\right\|_2 = h_{m+1,m}\left|{}^t e_m^{(m)} y\right|. \tag{16}$$

Proof: [2].

At this stage, the interesting feature is the possibility to compute the norm of residue without producing the solution.

According to this proposition, at the step k, the residual norm $\|r_k\|_2$ is then given directly by the absolute value of the last component y_k of the vector $y \in \mathbb{R}^k$. This provides the stopping test and

allows to calculate x_k only when the stopping criterion used for the solution is satisfied. This is the main idea of this article. It consists in calculating locally, at each iteration k, uniquely the last component y_k of the vector $y \in \mathbb{R}^k$: solution of the system (14) $H_k y = \beta_1 e_1^{(k)}$. The classical formulae of Cramer's rule directly gives the component y_k in the form of a determinant quotient: $y_k = \dfrac{det\left(D_k\right)}{det\left(H_k\right)}$, the computation of determinants $det\left(D_k\right)$ is given by the following result:

Proposition 3:

Let $H_k \in \mathbb{R}^{k\times k}$ be an invertible upper hessenberg matrix and $\beta_1 e_1^{(k)} \in \mathbb{R}^k$, the second member of the linear system (14) $H_k y = \beta_1 e_1^{(k)}$. Then we have:

$$y_k = \frac{\begin{vmatrix} h_{1,1} & \cdots & \cdots & h_{1,k-1} & \beta_1 \\ h_{2,1} & h_{2,2} & & \vdots & 0 \\ & h_{3,2} & \ddots & \vdots & \vdots \\ & & \ddots & h_{k-1,k-1} & \vdots \\ & & & h_{k,k-1} & 0 \end{vmatrix}}{det\left(H_k\right)} = \frac{det\left(D_k\right)}{det\left(H_k\right)}. \tag{17}$$

with:

$$det\left(D_k\right) = \left(-1\right)^{k+1} \beta_1 \prod_{i=2}^{k} h_{i,i-1}. \tag{18}$$

$$det\left(D_{k+1}\right) = -h_{k+1,k}\, det\left(D_k\right) \quad \forall k \geq 2. \tag{19}$$

What we need now, is to calculate $det\left(H_k\right)$; H_k being an upper hessenberg matrix.

2. Sturm Sequence for the Upper Hessenberg Matrices (U.H)

In this paragraph, we extend the definition of Sturm sequence to the hessenberg matrices, and then we give an algorithm computing their determinants. Let's start by reminding the following principal result:

Proposition 4:

Let $H \in \mathbb{R}^{n\times n}$ be an (U.H) matrix and let $\left(H - \lambda I\right)_k$ be the k^{th} principal submatrix of $\left(H - \lambda I\right)$, *i.e.*,

$$\left(H - \lambda I\right)_k = H_k - \lambda I_k = \begin{pmatrix} h_{1,1} - \lambda & h_{1,2} & \cdots & \cdots & h_{1,k} \\ h_{2,1} & h_{2,2} - \lambda & & & \vdots \\ 0 & h_{3,2} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & h_{k-1,k} \\ 0 & \cdots & 0 & h_{k,k-1} & h_{k,k} - \lambda \end{pmatrix}$$

and let $P_k(\lambda)$ be the charcteristic polynomial of H_k, then: the polynomial $P_k(\lambda)$, $k=0,\cdots,n$ are given by to the following recurrent formula :

$$\begin{cases} P_0(\lambda)=1 \\ P_1(\lambda)=h_{1,1}-\lambda \\ P_k(\lambda)=\sum_{i=1}^{k-1}(-1)^{(k-i)}\, h_{i,k}\prod_{j=i+1}^{k} h_{j,j-1}P_{i-1}(\lambda)+(h_{k,k}-\lambda)P_{k-1}(\lambda) \end{cases} \tag{20}$$

Proof: by recurrence.

Definition 1:

The sequence $\{P_k(\lambda)\}_{k=0}^{n}$ defined by (20) is called Sturm sequence of the upper hessenberg matrix.

Remark 1:

We note that, for the programming, we can win $\frac{(n-1)(n-2)}{2}$ memory words if we use a storage numbering function [8] for upper hessenberg matrices by storing nonzero elements uniquely.

2.1. Computing the Determinant of the Hessenberg Matrix H_n

For calculating the determinant of an upper hessenberg matrix, Proposition (2) gives us a simple method in $o(n^2)$. This can be summed by the following result:

Corollary 1:

Let $H=(h_{i,j})\in\mathbb{R}^{n\times n}$ be an invertible (U.H) matrix. If we set $\delta_k=\det(H_k)$, then we have:

$$\begin{cases} \delta_0=1 \\ \delta_1=h_{11} \\ \delta_k=h_{kk}\delta_{k-1}+\sum_{\ell=2}^{k}(-1)^{(\ell+1)}\, h_{k+1-\ell,k}\delta_{k-\ell}\prod_{j=k+2-\ell}^{k} h_{j,j-1} \quad \forall k=2,\cdots,n. \end{cases} \tag{21}$$

$\delta_n=\det(H)$. In the Formula (21), we calculate the products $\prod_{j=k+2-\ell}^{k} h_{j,j-1}$, without repeting multiplications. So, we obtain a very interesting algorithm $(A_{\det})$ computing $\delta_n=\det(H_n)$ as follows:

$$(A_{\det}):\begin{cases} \text{Data}: n,\ H_n\in\mathbb{R}^{n\times n}(\text{U.H}),\ \delta\in\mathbb{R}^{n+1} \\ \quad \delta_0=1.0,\ \delta_1=h_{1,1} \\ \text{for } k=1 \text{ to } n-1 \text{ do}: \\ \quad \text{begin} \\ \quad\quad S=h_{kk}*\delta_k,\ codiag=1.0,\ sg=-1 \\ \quad\quad \text{for } i=k-1 \text{ step } -1,\ 0 \text{ do}: \\ \quad\quad\quad \text{begin} \\ \quad\quad\quad\quad codiag=codiag*h_{i+1,i} \\ \quad\quad\quad\quad P=h_{i,k}*codiag*\delta_i \\ \quad\quad\quad\quad S=S+sg*P,\ sg=-sg \\ \quad\quad\quad \text{end}\{i\} \\ \quad\quad \delta_{k+1}=S \\ \quad \text{end}\{k\} \\ //\ \text{Output}:\ det(H_n)=\delta_n \end{cases}$$

The number of operations necessary in calculating δ_n is:

- additions: $\sum_{k=2}^{n}(k-1)=\dfrac{n(n-1)}{2}$
- multiplications: $\sum_{k=1}^{n}5k=\dfrac{5n(n+1)}{2}$.

The total number of operations required is $\sim 3n^2$. This is more convenient when n is big.

We will adopt the algorithm $(A_{\det})$ in the full orthogonalization method (FOM). Having at each step k, $(V_{k+1},H_{k+1,k})$, we calculate by the Formulaes (18) and (19): $y_k=\dfrac{\det(D_k)}{\det(H_k)}$. This gives us the stopping test and allows to calculate x_k only when the stopping criterion used for the solution is satisffied, namely:

$$\|r_k\|_2=h_{k+1,k}*|y_k|<\varepsilon$$

where ε, being the required precision. The FOM algorithm (A_{Fom}) takes the following form:

$(A_{Fom}):$

Data : $n,\ A\in^{n\times n},\ m\le n,\ \varepsilon=10^{-8}$ (for example)
$x_0\in^{n},\ \delta\in^{n+1}\ H\in^{m+1\times m},\ V\in^{n\times m},\ sg\in\{-1,1\}:sign$
Initialization : $r_0=b-Ax_0,\ \ \delta_0=1.0,$
compute : $\beta_1=\|r_0\|_2,\quad v_0=\dfrac{r_0}{\beta_1},\quad V_0=\{v_0\}.$
//calculate the column h_0 of the matrix H_0
  compute : $w=Av_0,\ \ h_{00}=\langle w,v_0\rangle.$
  compute : $w=w-h_{00}v_0$
  compute : $h_{1,0}=\|w\|_2$ if $h_{1,0}=0,$"Luckybreakdown", goto ®
  compute : $v_1=\dfrac{w}{h_{1,0}},\quad H=[h_0],V_1=[v_0,v_1]$
  $k=1;\ \delta_1=h_{0,0},\ detDkm1=\beta_1,\ \|r_k\|_2=h_{0,0};$
  while $(\|r_k\|_2>\varepsilon)$
  //calculate the colomn h_k of the matrix H_k
  compute : $w=Av_k$
  for $i=0$ to k
  compute : $h_{ik}=\langle w,v_i\rangle$
  compute : $w=w-h_{ik}v_i$
  end i
  compute : $h_{k+1,k}=\|w\|_2$, if $h_{k+1,k}=0,$"Luckybreakdown", goto ®
  compute : $v_{k+1}=\dfrac{w}{h_{k+1,k}},\quad \begin{cases}V_{k+1}=[v_0,\cdots,v_{k+1}]\\ H_k=[h_0,\cdots,h_k]\end{cases}$
  //compute $\det(H_k)$
  $detHk=h_{k,k}*\delta_k;\ \ codiag=1.0;\ \ sg=-1;$
  for $i=k-1$ to 0 step -1
  $codiag=codiag*h_{i+1,i};$
  $P=h_{ik}*codiag*\delta_i;$
  $detHk=detHk+sg*P;\ \ sg=-sg;$
  end i
  $\delta_{k+1}=detHk;\quad detDk=-detDkm1*h_{k,k-1};$
  $y_k=detDk/detHk;\ \|r_k\|_2=h_{k+1,k}*|y_k|;$
  $detDkm1=detDk,\ \ k:=k+1;$
  end{while}
® Output : $m=k,$ resolve : $H_m y=\beta_1 e_1^{(m)}.$

When the stopping criterion used for the solution is satisffied, we take the value y_k for calculating y_{k-1}, $y_{k-2},\cdots,y_1$ by using the following algorithm:

2.2. Algorithm of Resolution of the System: $H_m y = \beta_1 e_1$

$$
(A_{Res}):\begin{cases}
\text{Data}:\begin{cases} n \geq m \geq 2; \quad \beta_1 \in \mathbb{R} \\ V_m = [v_0, \cdots, v_{m-1}] \in M_{n,m}(\mathbb{R}) \\ H_m = [h_0, \cdots, h_{m-1}] \in M_m(\mathbb{R}) \\ y \in \mathbb{R}^m,\ y_m \in \mathbb{R},\ \text{calculated already} \\ x_0 \in \mathbb{R}^n, \quad x \in \mathbb{R}^n, \quad b = \beta_1 e_1^{(m)} \in \mathbb{R}^m. \end{cases} \\
\bullet\ \text{compute the system solution}: Hy = b \\
\forall\ i = m-1, \cdots, 1, \text{ step } -1 \\
\qquad y_{i-1} = -\dfrac{\sum_{j=i}^{m-1} h_{ij} y_j}{h_{i,i-1}} \\
A_i \\
//\ \text{being } y \in \mathbb{R}^m, \text{ compute the solution } x \in \mathbb{R}^n \\
\bullet\ \text{compute}: x = x_0 + V_m * y;
\end{cases}
$$

The above algorithm (A_{Res}) requires $2nm$ flops.

Several authors solve the linear system $H_m y = \beta_1 e_1^{(m)}$ using Givens method applied to an upper hessenberg matrix.

3. The Givens Method for (U.H) Matrices

In this section, we give a new variant of the Givens method for the resolution of linear systems with an upper hessenberg matrices coming from Arnoldi's algorithm. This method is based on the addition of two matrices and two tensorial products.

The Givens rotations constitute an important tool to selectively eliminate certain coefficients of a matrix. This is the case of our upper hessenberg matrix, where we eliminate the codiagonal in order to obtain an upper triangular matrix.

To justify the next results, we introduce the following notations:

- $\tilde{a}_k^{(r)}$: the k^{th} row of the matrix $A^{(r)}$ at the step r.
- $\tilde{\tilde{A}}^{(r)}$: the matrix $A^{(r)}$ with the r^{th} and $(r+1)^{th}$ row put to zero.
- a_k: k^{th} column vector of A.
- ${}^tX$: the superscript t will denote the matrix or the transposed vector.
- $x \otimes {}^t y$: tensor product of two vectors x, y in $\mathbb{R}^n$.
- $\left[g_{ij}\right]_k$: k^{th} column vector of the rotation matrix $G_{i,j}$ in the plan $\{e_i, e_j\}$.

3.1. Givens Formulation for Upper Hessenberg Matrices

The classical formulation of the Givens method for an upper hessenberg matrix $H \in^{n\times n}$ is given under the following form:

$$
\begin{cases}
H^{(1)} = H \\
H^{(2)} = G_{21} \times H^{(1)} \\
H^{(3)} = G_{32} \times H^{(2)} \\
\quad \vdots \\
H^{(k+1)} = G_{k+1,k} \times H^{(k)} \\
\quad \vdots \\
H^{(n-1)} = G_{n-1,n-2} \times H^{n-2} \\
H^{(n)} = G_{n,n-1} \times H^{(n-1)}
\end{cases} \tag{22}
$$

with $G_{k+1,k}$ is of the form:

$$G_{k+1,k}=\begin{array}{c} \\ k\to \\ k+1\to \end{array}\begin{array}{c} \begin{array}{ccccccccc} & & & k\downarrow & k+1\downarrow & & & & \end{array} \\ \begin{pmatrix} 1 & & & \vdots & \vdots & & & & \\ & \ddots & & \vdots & \vdots & & O & & \\ & & 1 & \vdots & \vdots & & & & \\ \cdots & \cdots & \cdots & c_k & s_k & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & -s_k & c_k & \cdots & \cdots & \cdots & \\ & & & \vdots & \vdots & 1 & & & \\ & O & & \vdots & \vdots & & \ddots & & \\ & & & & & & & 1 & \end{pmatrix} \end{array} \tag{23}$$

with $\det\left(G_{k+1,k}\right)=1$ and where c_k and s_k are calculated at each step k by the following formulas:

$$\begin{cases} d_{k+1,k}^{(k)}=\sqrt{\left(h_{k,k}^{(k)}\right)^2+\left(h_{k+1,k}^{(k)}\right)^2}. \\ c_k=\dfrac{h_{k,k}^{(k)}}{d_{k+1,k}^{(k)}}. \\ s_k=\dfrac{h_{k+1,k}^{(k)}}{d_{k+1,k}^{(k)}}. \end{cases} \tag{24}$$

The relations (22) give us an upper triangular matrix, $H^{(n)}$:

$$H^{(n)}=G_{n,n-1}G_{n-1,n-2}\cdots G_{3,2}G_{2,1}H, \tag{25}$$

and therefore $H=G_{21}^{-1}G_{32}^{-1}\cdots G_{n-1,n-2}^{-1}G_{n,n-1}^{-1}H^{(n)}$.

As the matrices G_{ij} are orthogonal, we have: ${}^tG_{ij}G_{ij}=I_d$, from which

$$H={}^tG_{21}\,{}^tG_{32}\cdots{}^tG_{n-1,n-2}\,{}^tG_{n,n-1}H^{(n)}. \tag{26}$$

Theorem 1:

If $H\in\mathbb{R}^{n,n}$ is an invertible upper hessenberg matrix, then $\forall\ 1\le k\le n-1$

$$H^{(k+1)}=G_{k+1,k}H^{(k)}\Leftrightarrow H^{(k+1)}=\tilde{H}^{(k)}+\left[g_{k+1,k}\right]_k\otimes\tilde{h}_k^{(k)}+\left[g_{k+1,k}\right]_{k+1}\otimes\tilde{h}_{k+1}^{(k)}. \tag{27}$$

where $\tilde{h}_k^{(k)}$ is the k^{th} row vector of the matrx $H^{(k)}$ and $\left[g_{k+1,k}\right]_k$ is the k^{th} column vector of the matrix $G_{k+1,k}$.

The Givens algorithm for the triangularisation of the matrix $H\in\mathbb{R}^{n\times n}$ (U.H), is simplified to be written in its new form:

$$\left(A_{Giv}\right):\begin{cases} \bullet\text{Data}: n,\ H\in^{n\times n+1}\text{ augmented }(\text{U.H})\text{ matrix of }b \\ \text{for } k=1 \text{ to } n-1 \text{ do}: \\ \quad \text{If } h_{k+1,k}^{(k)}=0.0 \quad \text{then}: \text{continue} \\ \quad \bullet \text{ calculation of the rotation matrix } (23)\ G_{k+1,k}^{(k)} \\ \qquad G_{k+1,k}^{(k)}:\begin{cases} \bullet \text{ compute } d_{k+1,k}^{(k)}=\sqrt{\left(h_{k,k}^{(k)}\right)^2+\left(h_{k+1,k}^{(k)}\right)^2} \\ \bullet \text{ compute } c_k=\dfrac{h_{k,k}^{(k)}}{d_{k+1,k}^{(k)}}, s_k=\dfrac{h_{k+1,k}^{(k)}}{d_{k+1,k}^{(k)}} \end{cases} \\ \quad \bullet \text{ compute}: \left[g_{k+1,k}\right]_k=\begin{pmatrix} 0 \\ \vdots \\ 0 \\ c_k \\ -s_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}\leftarrow k; \left[g_{k+1,k}\right]_{k+1}=\begin{pmatrix} 0 \\ \vdots \\ 0 \\ c_k \\ s_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}\leftarrow k \\ \quad \bullet \text{compute}: H^{(k+1)}=\tilde{H}^{(k)}+\left[g_{k+1,k}\right]_k\otimes\tilde{h}_k^{(k)}+\left[g_{k+1,k}\right]_{k+1}\otimes\tilde{h}_{k+1}^{(k)} \\ fin\{k\} \end{cases}$$

3.1.1. Complexity of the Algorithm (A_{Giv})

To eliminate the $(n-1)$ elements situated under the diagonal, we need $(n-1)$ rotations $G_{k+1,k}^{(k)}$. Each rotation requires 5 operations and an extraction of a square root $(\sqrt{})$.

For a fixed k

$\rightarrow$ Computation of tensor product:

- $\left[g_{k+1,k}\right]_k \otimes \tilde{h}_k^{(k)}$: $2(n-k+2)(*)$
- $\left[g_{k+1,k}\right]_{k+1} \otimes \tilde{h}_{k+1}^{(k)}$: $2(n-k+2)(*)$

$\rightarrow$ Computation of matricial addition:

- $\left[g_{k+1,k}\right]_k \otimes \tilde{h}_k^{(k)} + \left[g_{k+1,k}\right]_{k+1} \otimes \tilde{h}_{k+1}^{(k)}$: $2(n-k+2)(+)$

and $\forall k = 1,\cdots,n-1$, we have:

$$6\sum_{k=1}^{n-1}(n-k+2)+\sum_{k=1}^{n-1}5 \simeq 6\int_{k=0}^{n-2}(n-k+2)\mathrm{d}k \simeq 3n^2$$

Conclusion: The complexity of Givens' algorithm (A_{Giv}) for an upper hessenberg matrix requires $3n^2$ flops and $(n-1)(\sqrt{})$.

In consequense, we find the calculating formula of the determinant of the matrix H given by:

Corollary 2 :

Let H be an (U.H) matrix. By applying classical Givens' formulation, given by the relation (22) and the Formula (26), we obtain:

$$\det(H) = \det\left(H^{(n)}\right) = \prod_{i=1}^{n} h_{i,i}^{(i)}. \tag{28}$$

We can decompose H matrix (U.H) of rank n, into an addition of a triangular matrix and $(n-1)$ matrices of rang 1 given by:

$$H = H^{(n+1)} - \sum_{k=1}^{n-1}\left(\left[g_{k+1,k}\right]_k - e_k\right)\otimes \tilde{h}_k^{(k)} + \left(\left[g_{k+1,k}\right]_{k+1} - e_{k+1}\right)\otimes \tilde{h}_{k+1}^{(k)}. \tag{29}$$

3.1.2. Numerical Example

Let $H \in \mathbb{R}^{4\times 4}$ be an invertible matrix (U.H), and $b \in \mathbb{R}^4$.

We want to resolve $Hx = b$.

$$H = \begin{pmatrix} 1 & 0 & -1 & 2 \\ 1 & 2 & -3 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}; \quad b = \begin{pmatrix} 6 \\ -4 \\ -1 \\ 1 \end{pmatrix}$$

We apply the algorithm (A_{Giv}) to the augmented matrix $H \in \mathbb{R}^{4\times 5}$ by the second member b.

step *k* = 1

- Compute the rotation in plane $\{e_2, e_1\}$: $G_{21} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$

Column 1: $[g_{21}]_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{pmatrix}$; column 2: $[g_{21}]_2 = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 \end{pmatrix}}_{\tilde{H}^{(1)}} + \underbrace{\begin{array}{c|ccccc} \tilde{h}_1^{(1)}: & 1 & 0 & -1 & 2 & 6 \\ \hline \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 2\frac{\sqrt{2}}{2} & 6\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & -2\frac{\sqrt{2}}{2} & -6\frac{\sqrt{2}}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}}_{[g_{21}]_1 \quad [g_{21}]_1 \otimes \tilde{h}_1^{(1)}} + \underbrace{\begin{array}{c|ccccc} \tilde{h}_2^{(1)}: & 1 & 2 & -3 & 0 & -4 \\ \hline \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \sqrt{2} & -3\frac{\sqrt{2}}{2} & 0 & -4\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \sqrt{2} & -3\frac{\sqrt{2}}{2} & 0 & -4\frac{\sqrt{2}}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}}_{[g_{21}]_2 \quad [g_{21}]_2 \otimes \tilde{h}_2^{(1)}}$$

$$= \underbrace{\begin{pmatrix} \sqrt{2} & \sqrt{2} & -2\sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & -5\sqrt{2} \\ 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 \end{pmatrix}}_{H^{(2)}}$$

step $k = 2$

- Compute the rotation in plane $\{e_3, e_2\}$: $G_{32} = \begin{pmatrix} \frac{\sqrt{2}}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{-1}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix}$

Column 2: $[g_{32}]_2 = \begin{pmatrix} \frac{\sqrt{2}}{\sqrt{3}} \\ \frac{-1}{\sqrt{3}} \end{pmatrix}$; column 3: $[g_{32}]_3 = \begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix}$

$$\underbrace{\begin{pmatrix} \sqrt{2} & \sqrt{2} & -2\sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \end{pmatrix}}_{\tilde{H}^{(2)}} + \underbrace{\begin{array}{c|ccccc} \tilde{h}_2^{(2)}: & 0 & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & -5\sqrt{2} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{2}{\sqrt{3}} & \frac{-2}{\sqrt{3}} & \frac{-2}{\sqrt{3}} & \frac{-10}{\sqrt{3}} \\ \frac{-1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}} & \frac{5\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}}_{[g_{32}]_2 \quad [g_{32}]_2 \otimes \tilde{h}_2^{(2)}}$$

$$+ \underbrace{\begin{array}{c|ccccc} \tilde{h}_3^{(2)}: & 0 & 1 & -1 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & 0 & \frac{-1}{\sqrt{3}} \\ \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{\sqrt{2}}{\sqrt{3}} & \frac{-\sqrt{2}}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}}_{[g_{32}]_3 \quad [g_{32}]_3 \otimes \tilde{h}_3^{(2)}} = \underbrace{\begin{pmatrix} \sqrt{2} & \sqrt{2} & -2\sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & \frac{3}{\sqrt{3}} & \frac{-3}{\sqrt{3}} & \frac{-2}{\sqrt{3}} & \frac{-11}{\sqrt{3}} \\ 0 & 0 & 0 & \frac{\sqrt{2}}{\sqrt{3}} & \frac{4\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & -1 & 1 & 1 \end{pmatrix}}_{H^{(3)}}$$

step $k = 3$

- Compute the rotation in plane $\{e_4, e_3\}$: $G_{43} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

Column 3: $[g_{43}]_3 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$; column 4: $[g_{43}]_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$$\underbrace{\begin{pmatrix} \sqrt{2} & \sqrt{2} & -2\sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & \frac{3}{\sqrt{3}} & \frac{-3}{\sqrt{3}} & \frac{-2}{\sqrt{3}} & \frac{-11}{\sqrt{3}} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{\tilde{H}^{(3)}} + \begin{array}{c|ccccc} \tilde{h}_3^{(3)}: & 0 & 0 & 0 & \frac{\sqrt{2}}{\sqrt{3}} & \frac{4\sqrt{2}}{\sqrt{3}} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & \frac{-4\sqrt{2}}{\sqrt{3}} \end{array}$$

$[g_{43}]_3 \qquad [g_{43}]_3 \otimes \tilde{h}_3^{(3)}$

$$+ \begin{array}{c|ccccc} \tilde{h}_4^{(3)}: & 0 & 0 & -1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} = \underbrace{\begin{pmatrix} \sqrt{2} & \sqrt{2} & -2\sqrt{2} & \sqrt{2} & \sqrt{2} \\ 0 & \frac{3}{\sqrt{3}} & \frac{-3}{\sqrt{3}} & \frac{-2}{\sqrt{3}} & \frac{-11}{\sqrt{3}} \\ 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -\frac{\sqrt{2}}{\sqrt{3}} & \frac{-4\sqrt{2}}{\sqrt{3}} \end{pmatrix}}_{H^{(4)}}$$

$[g_{43}]_4 \qquad [g_{43}]_4 \otimes \tilde{h}_4^{(3)}$

we obtain a triangular system $H^{(4)}x = b^{(4)}$ whose solution is x:

$$H^{(4)} = \begin{pmatrix} \sqrt{2} & \sqrt{2} & -2\sqrt{2} & \sqrt{2} \\ 0 & \frac{3}{\sqrt{3}} & \frac{-3}{\sqrt{3}} & \frac{-2}{\sqrt{3}} \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & \frac{-\sqrt{2}}{\sqrt{3}} \end{pmatrix}; \quad b^{(4)} = \begin{pmatrix} \sqrt{2} \\ \frac{-11}{\sqrt{3}} \\ 1 \\ \frac{-4\sqrt{2}}{\sqrt{3}} \end{pmatrix}; \quad x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

and we have $\det(H) = \det\left(H^{(4)}\right) = 2$.

4. Numerical Tests

In this section, we present four numerical tests for the resolution of linear systems (1): $AX = b$ for the proposed new version of full orthogonalization method (FOM). These tests are done using a portable DELL (Latitude D505 intel (R) Pentium (R) M, processor 1.70 GHz, 594 MHz) and a program written in C_{++}, with double precision.

Test 1: A_n is band (SPD) matrix $D = 2d+1$, $d = 3$, [9] and $X \in \mathbb{R}^n$ is the exact solution.

$$A_n = \begin{pmatrix} 5 & 2 & 1 & 1 & 0 & \cdots & 0 \\ 2 & 6 & 3 & 1 & 1 & \ddots & \vdots \\ 1 & 3 & 6 & \ddots & \ddots & \ddots & 0 \\ 1 & \ddots & \ddots & \ddots & \ddots & \ddots & 1 \\ 0 & \ddots & \ddots & \ddots & \ddots & 3 & 1 \\ \vdots & \ddots & 1 & 1 & 3 & 6 & 2 \\ 0 & \cdots & 0 & 1 & 1 & 2 & 5 \end{pmatrix}, \quad b = \begin{pmatrix} 16 \\ 2\times 16 \\ \vdots \\ (n-3)16 \\ 15n-23 \\ 13n-18 \\ 9n-7 \end{pmatrix}, \quad X = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n-2 \\ n-1 \\ n \end{pmatrix}$$

From test 1, we deduce the following numerical results in **Table 1**.

Test 2: A_n is a full matrix, X is the exact solution, and A_n^{-1} is the inverse matrix.

$$A_n = \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 2 & \cdots & 2 & 1 \end{pmatrix}, X = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{pmatrix}, A_n^{-1} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \ddots & 1 \\ 2 & 0 & \cdots & 0 & -1 \end{pmatrix}$$

where: $\|A_n\|_\infty = 2n-1, \|A_n^{-1}\|_\infty = 3$ which imply: $cond_\infty(A_n) \simeq 6n$.

$$\begin{cases} b_i = \dfrac{2i(i-1)+(n+i)(n-i+1)}{2} \\ \forall i = 1, \cdots, n \end{cases} \tag{30}$$

From test 2, we deduce the following numerical results in **Table 2**.

Test 3:

In the third test, we focus our attention on the resolution of the partial differential equations with Neuman condition:

$$\begin{cases} -\Delta u + u = f & \text{on } \Omega =]0, \gamma[\times]0, \gamma[\\ \dfrac{\partial u}{\partial \nu} = 0 & \text{on } \Gamma \text{ the frontier} \end{cases} \tag{31}$$

with: $\begin{cases} h_x = h_y = 0.01: \text{step of discretization at } x \text{ and at } y. \\ n \in \mathbb{N}^*: \text{number of nodes on the } x \text{ axis.} \\ \gamma = h_x(n-1). \end{cases}$

Table 1. Test 1's numerical results.

n	m	ε	$\|r_k\|_2$
100	66	$0.9 10^{-3}$	$0.8 10^{-3}$
500	171	$0.9 10^{-3}$	$0.8931 0^{-3}$
1000	215	$0.9 10^{-3}$	$0.8941 0^{-3}$
1500	247	$0.9 10^{-3}$	$0.881 0^{-3}$
2000	270	$0.9 10^{-3}$	$0.8951 0^{-3}$
2500	291	$0.9 10^{-3}$	$0.8961 0^{-3}$
3000	309	$0.9 10^{-3}$	$0.8981 0^{-3}$

Table 2. Test 2's numerical results.

n	m	ε	$\|r_k\|_2$	$cond_\infty(A_n) \simeq 6n$
100	47	$0.9 10^{-3}$	$0.5431 0^{-3}$	600
200	115	$0.9 10^{-3}$	$0.8471 0^{-3}$	3000
1000	169	$0.9 10^{-3}$	$0.7881 0^{-3}$	6000
1500	211	$0.9 10^{-3}$	$0.7851 0^{-3}$	9000
2000	246	$0.9 10^{-3}$	$0.871 0^{-3}$	12,000
2500	278	$0.9 10^{-3}$	$0.841 0^{-3}$	15,000
3000	307	$0.9 10^{-3}$	$0.831 0^{-3}$	18,000

If we consider $u(x,y)=\cos\left(\frac{\pi}{\gamma}x\right)\cos\left(\frac{3\pi}{\gamma}y\right)$ as a solution, we obtain the function:

$$f(x,y)=\left(\frac{10\pi^2+\gamma^2}{\gamma^2}\right)\cos\left(\frac{\pi}{\gamma}x\right)\cos\left(\frac{3\pi}{\gamma}y\right) \tag{32}$$

as a second member.

After doing a regular maillage and numbering the nodes, the calculation of the stiffness matrix leads to a (SPD) "well-conditioned" matrix. For the resolution we use the new version of full orthogonalization method (FOM).

Table 3 contains the numerical results of Test 3:

Test 4: A_n is band, (SPD) matrix $D=2d+1$, $d=2$, [9].

$$A_n=\begin{pmatrix} 5 & -4 & 1 & 0 & \cdots & 0 \\ -4 & 6 & \ddots & \ddots & \ddots & \vdots \\ 1 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 1 \\ \vdots & \ddots & \ddots & \ddots & 6 & -4 \\ 0 & \cdots & 0 & 1 & -4 & 5 \end{pmatrix}, b=\begin{pmatrix} 0 \\ \vdots \\ 0 \\ -(n+1) \\ 2(n+1) \end{pmatrix}, X=\begin{pmatrix} 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{pmatrix}$$

$$\lambda_k=16\sin^4\left(k\pi 2(n+1)\right) \qquad \forall k=1,\cdots,n \tag{33}$$

On the one hand, the numerical results for the matrices A_n are considerably polluted with rounding errors because the matrices A_n are very "ill-conditioned" But on the other hand, we are constructing matrices M_n very "well-conditioned" from matrices A_n with translations:

$$M_n=A_n-\mu_n I_d$$

where $\mu_n=16\sin^4\left(n\pi 2(n+1)\right)$.

From Test 4, we deduce the following numerical results in **Table 4**.

Table 3. Test 3's numerical results.

n	m	ε	$\|r_k\|_2$
100	25	0.910^{-3}	0.52410^{-3}
900	75	0.910^{-3}	0.48110^{-3}
1600	94	0.910^{-3}	0.66210^{-3}
2500	114	0.910^{-3}	0.610^{-3}
3600	131	0.910^{-3}	0.8510^{-3}

Table 4. Test 4's numerical results.

n	m	ε	$\|r_k\|_2$	$cond_2(A_n)=\frac{\lambda_n}{\lambda_1}$	$cond_2\left(\underbrace{A_n-\mu_n I_d}_{M_n}\right)$
100	10	0.910^{-3}	0.8810^{-3}	1.777810^{7}	$cond_2(M_n)\simeq 2$
500	11	0.910^{-3}	0.7610^{-3}	1.038910^{10}	$cond_2(M_n)\simeq 2$
1000	12	0.910^{-3}	0.2610^{-3}	1.649110^{11}	$cond_2(M_n)\simeq 2$
1500	11	0.910^{-3}	0.39110^{-3}	8.337710^{11}	$cond_2(M_n)\simeq 2$
2000	12	0.910^{-3}	0.5210^{-3}	2.633710^{12}	$cond_2(M_n)\simeq 2$
2500	12	0.910^{-3}	0.6510^{-3}	6.666710^{12}	$cond_2(M_n)\simeq 2$
3000	12	0.910^{-3}	0.7810^{-3}	1.333310^{13}	$cond_2(M_n)\simeq 2$
3500	13	0.910^{-3}	0.15610^{-3}	2.468010^{13}	$cond_2(M_n)\simeq 2$

5. Conclusions

Test 1 and 2 show the results of applying the proposed new version of full orthogonalization method (FOM): the dimension m of Krylov's space K_m is acceptable compared to n because the matrices A_n are moderately "ill-conditioned".

For the third test, the stiffness matrix coming from the discretization of the partial differential Equations (31) leads to a (SPD) well-conditioned matrix. We note that dimension m of Krylov's space K_m is weak compared to n, which can give a positive judgment about the new version of (FOM) method.

In the final test, the constructed matrices M_n are very "well-conditioned", and the dimension m is as small as required.

Acknowledgements

The author is grateful to the referees for their valuable comments and suggestions which have helped to improve the presentation of this work.

References

[1] Toumi, A. (2005) Utilisation des filtres de Tchebycheff et construction de préconditionneurs spéciaux pour l'accélération des methods de Krylov. Thèse No. 2296, de l'Institut National Polytechnique de Toulouse, France.

[2] Saad, Y. (2000) Iterative Methods for Sparse Linear Systems. 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia.

[3] Benhamadou, M. (2000) Développement d'outils en Programmation Linéaire et Analyse Numérique matricielle. Thèse No. 1955, de l'Université Paul Sabatier Toulouse 3, Toulouse, France.

[4] Wilkinson, J.H. (1965) The Algebraic Eigenvalue Problem. Clarendon Press, Oxford.

[5] Gastinel, N. (1966) Analyse Numérique Linéaire. Hermann, Paris.

[6] Ciarlet, P.G. (1980) Introduction à l'Analyse Numérique Matricielle et à l'Optimisation. Masson, Paris.

[7] Lascaux, P. and Théodor, R. (1993) Analyse Numérique Matricielle Appliquée à l'Art de l'Ingénieur. Tome 1, Tome 2, Masson, Paris.

[8] Jennings, A. (1980) Matrix Computation for Engineers and Scientists. John Wiley and Sons, Chichester.

[9] Gregory, R.T. and Karney, D.L. (1969) A Collection of Matrices for Testing Computational Algorithms. Wiley-Interscience, John Wiley & Sons, New York, London , Sydney, Toronto.

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.