Some software tools to control the accuracy in scientific computing

Jean-François Carpraux

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Some software tools to control the accuracy in scientific computing

J.F. Carpraux

N° 2382

1994

PROGRAMME 6

ISSN 0249-6399

Some software tools to control the accuracy in scientific computing

J.F. Carpraux

Programme 6 — Calcul scientifique, modélisation et logiciel numérique
Projet Aladin

Rapport de recherche n°2382 — 1994 — 123 pages

Abstract: A main concern of scientific computing is the validation of numerical simulations. Indeed, several factors contribute to the degradation of accuracy in the final result. This thesis deals with new tools to control the accuracy for the following eigenproblem :

Problem (P) : *Given* $A \in \mathbb{C}^{n\times n}$, *find some* $\lambda \in \mathbb{C}$ *and/or* $x \in \mathbb{C}^n$ *such that :* λ *is an eigenvalue of* A *and* x *is the associated eigenvector.*

We present in the first part of this thesis an expert system called SESAME, which can either select the sequence of LAPACK routines solving the given problem (P) or validate a user choice of routines. It can eventually give an estimation of the accuracy of the result. The spectral portrait of a matrix provides useful informations about it. Usually, the spectral portrait is computed using a Singular Value Decomposition, but this approach is not suitable for large sparse matrices. The second part of the thesis is devoted to the computation of the spectral portrait for large sparse matrices. Krylov subspaces have an important place in sparse linear algebra, since numerous iterative methods in linear algebra dealing with large sparse matrices require these subspaces. In the third part of the thesis, we study theoretically the condition number of Krylov bases and subspaces.

Key-words: Eigenproblem, Knowledge-base system, Sparse matrix, Spectral portrait, Condition number, Krylov basis and subspace.

(Résumé : tsvp)

Quelques outils d'aide au contrôle de précision en calcul scientifique

Résumé : Une des principales préoccupations des scientifiques est de déterminer le niveau de confiance de leurs simulations numériques. En effet, plusieurs facteurs concourent à la dégradation de la précision du résultat obtenu. Cette thèse propose de nouveaux outils pour contrôler la précision du problème aux valeurs propres suivant :

Problème (P) : *Etant donnée* $A \in \mathbb{C}^{n \times n}$*, trouver des éléments* $\lambda \in \mathbb{C}$ *et/ou* $x \in \mathbb{C}^n$ *tels que :*
λ *soit une valeur propre de* A *et* x *soit un vecteur propre de* A *associé à* λ.

On présente dans la première partie un système expert, appelé SESAME, qui détermine la séquence de procédures LAPACK à appeler pour résoudre un problème (P) donné. Le système peut également valider un choix de procédures fait par l'utilisateur et donner une estimation de l'erreur commise sur le résultat. Le portrait spectral d'une matrice donne des renseignements très utiles. La méthode de calcul du portrait spectral fondée sur la décomposition en valeurs singulières de matrices n'est pas utilisable pour une grande matrice creuse. La seconde partie est consacrée au calcul du portrait spectral d'une grande matrice creuse. Les sous-espaces de Krylov jouent un rôle très important pour les méthodes itératives d'algèbre linéaire traitant des grandes matrices creuses. La troisième partie est une étude du conditionnement des bases et sous-espaces de Krylov.

Mots-clé : Problème aux valeurs propres, Système à base de connaissance, Matrice creuse, Portrait spectral, Conditionnement, Base et sous-espace de Krylov.

Contents

Part I

Introduction

A main concern of scientific computing is the validation of numerical simulations. Indeed, several factors contribute to the degradation of accuracy in the final result. Firstly, the modelisation error (translation from the physical problem to mathematical equations) and the method error (discretisation of equations). These errors are due to the approximation of the initial problem (physical problem). Next, the error due to the choice of an iterative method, since the computed solution satisfies the problem only up to a tolerance threshold. Finally, the finite precision of computation generates rounding errors increasing highly when the initial problem is unstable or ill-conditioned (a problem is said ill-conditioned when its result is very sensitive to a data perturbation [49]). Therefore, we have to be concerned with the quality of scientific software and in particular with the numerical quality of the results. The toolbox AQUARELS [18] gathers into a consistent and user-friendly structure some solutions to control or improve the accuracy of scientific computations. This thesis deals with new tools that can be included in AQUARELS.

We have chosen to study the following problem (P) :

Given a square complex matrix $A \in \mathbb{C}^{n\times n}$, find some $\lambda \in \mathbb{C}$ and/or $x \in \mathbb{C}^n$ such that :

λ is an eigenvalue of A (i.e. $\det(A - \lambda I) = 0$)
and x is the associated eigenvector (i.e. $Ax = \lambda x$).

This problem arises in particular in the computation of the eigenmodes in structural mechanics as well as in the study of the stability of dynamic systems which needs, for instance, to find precisely at which side of the imaginary axis the eigenvalues are located. The condition number of the problem (P) (measure of the sensitivity of the problem (P) to a data perturbation) is now well known [10, 45], but the computation of error bounds is complicated.

As far as small matrices are concerned, the methods of resolution for the problem (P) [13, 21, 25, 38, 50] are quite efficient. Indeed, a theoretical study of these methods shows that they are numerically stable (the effect of the rounding errors can be bounded) [50]. Moreover, it is possible to estimate condition numbers [3]. The spectral portrait [31, 46] (picture of the eigenvalues of the nearest matrices) provides useful information about the result. For instance, some dynamic systems are stable only when all the eigenvalues have a negative real part. The spectral portrait is useful in this case, since it enables on one hand to locate the eigenvalues and on another hand to measure the distance between the matrix and the nearest matrix which does not satisfy this condition. Similarly, the spectrum dichotomy [35] enables to find precisely which part of the spectrum is located on one side of the imaginary axis.

For large sparse matrices, we are faced to the problem of memory size and CPU time. The methods used to solve the problem (P) with small matrices require some matrix transformations which destroy the sparsity. Therefore, these methods cannot be used efficiently for large sparse matrices so that new methods have been designed. For instance, methods that require the matrix only in matrix-vector products, as in the power method, Lanczos method [33], or Arnoldi method [2]. Moreover, the control of accuracy for small matrices, such as the study of stability (sensitivity to rounding errors) and convergence of the methods, the condition number estimate, the computation of spectral portrait and spectrum dichotomy, are not suitable for large sparse matrices, because of memory size and computation time.

In order to ensure a good numerical quality when solving the problem (P), we advise to use the LAPACK library [1] for small matrices. Since the LAPACK routines are stable, the accuracy of the result depends only upon the condition number. The condition number is well known in the symmetric (or hermitian) case and is estimated with a few LAPACK routines in the nonsymmetric (or nonhermitian) case.

However, the use of a library requires help by means of artificial intelligence, for instance, as the NAG society is doing for its various libraries [5, 20, 28]. In the first part of this thesis, we present an expert system called SESAME [6], which can either select the LAPACK routines sequence to solve the given problem (P) or validate a user choice of routines. Finally, it can give an estimate of the accuracy of the result.

Two new tools are described in this thesis in order to analyze methods of resolution for the problem (P) in the case of large sparse matrices.

The spectral portrait of a matrix [31, 46] provides useful information. For instance, the measure of its distance to singularity or an estimate of the eigenvalue condition numbers. Usually spectral portraits are computed using a Singular Value Decomposition, but this approach is not suitable for large sparse matrices. The second part of the thesis is devoted to the computation of a spectral portrait for large sparse matrices [7]. It is based on a modification of the Davidson's algorithm [15, 39].

Krylov subspaces [32] have an important place in sparse linear algebra, since numerous iterative methods in linear algebra dealing with large sparse matrices require these subspaces. They are used in projection methods such as the Lanczos method [33], or the Arnoldi method [2], for the eigenproblem. The third part of the thesis is devoted to a theoretical study of the condition number of Krylov bases and subspaces [8].

Part II

SESAME : A Knowledge-Based System for Eigenvalue Problems

Chapter 1

Introduction

Because scientific libraries grow rapidly, it becomes difficult for the user to select the best routine to solve a given problem. Therefore, it is necessary to provide intelligent software for guiding the user. For instance, the NAG society is developing expert systems to help the user for selecting a routine from its various libraries [5, 20, 28]. A prototype for solving linear equations has been realized at the CNES [44]. The calling sequence for the selected routines can also be generated as, for instance, in the expert system for the MODULEF library [34].

Conversely, it might happen that the user wants to check whether a selected routine can solve the given problem, for instance whether the data are in the domain of the chosen numerical method. An expert system will have to work in the opposite sense, i.e. validate the correctness of the routine selection for the solution of a given problem.

The accuracy of a result in scientific computing depends upon the error in the data, the numerical stability of the algorithm, and the condition number of the problem. A few tools can now give an estimate of the error in the result. For instance, the expert system in [30] selects and executes some routines that use interval arithmetic to provide a result interval with guaranteed accuracy. The toolbox AQUARELS [18] collects some solutions to control or improve the accuracy of scientific computations into a consistent and user-friendly structure. An expert system might suggest the use of such tools for the estimate of the final error.

We have developed a first prototype at IRISA, called SESAME (Expert System for the Selection and vAlidation of numerical MEthods) [6]. Our expert system deals with the three following tasks :

- selection of routine,

- validation of routine,
- validation of result.

We have chosen to deal with the LAPACK library [1] and more precisely with a subset devoted to eigenproblems and containing about hundred routines. We claim however that this domain is sufficiently large and complex to show the capabilities of our system.

The choice of the LAPACK library allows us to simplify and conclude the validation of results. Indeed, since the algorithms used in LAPACK are stable, we do not worry about the rounding errors. Moreover, the library provides routines to estimate the condition number for the computation of the eigenvalues and eigenvectors, which allow to estimate the error in the result, given the error in the initial matrix.

For several reasons, the system is based on artificial intelligence techniques. Firstly, an advantage of an expert system is to explain how it finds the wanted result. Further, an interface facilitates its use. Moreover, the knowledge modelisation is easier and can relatively grow easily.

The numerical framework is modeled with a knowledge base of objects describing the mathematical entities to be manipulated. These objects are organized into hierarchies which can be complex due to the multiple inheritance. The basic inference mechanism is the classification process which is very efficient in the context of scientific computing. During the classification, some attribute values can be inferred from default values or attached procedures. Thus, the functioning of the selection of routine is nearly the same as for a decision tree. We only need to infer the name of the routine to use in the leaf which has been classified as "sure". But the same knowledge base allows also to check the validity of a routine, thanks to the attribute classes. In this case, the classification takes into account the name of the routine and the routine is valid if and only if a leaf is classified "sure".

Our system is based on the development shell SHIRKA [41], written in Le-Lisp, which provides means to describe knowledge bases and to apply a classification mechanism upon them.

The second chapter describes the numerical problem and formalizes it to obtain a modelisation by objects. The third chapter is devoted to the software SHIRKA and to the way it is used.

Chapter 2

Numerical expertise

The user's problem is the following : Given a square complex matrix A, find some $\lambda \in \mathbb{C}$ and/or $x \in \mathbb{C}^n$ such that :
λ is an eigenvalue of A (i.e. $\det(A - \lambda I) = 0$)
and x is the associated eigenvector (i.e. $Ax = \lambda x$).

For the moment we only consider matrices that can be treated with LAPACK, i.e. relatively small order matrices (say of order less than 5000), and with dense, packed or band storage. The case of large sparse matrices is not treated in LAPACK. We do not discuss here the generalized eigenvalue problem. Of course, it could be an extension of the knowledge base.

Our expert system executes the following different tasks :

- Find the sequence of routines to use for solving a given eigenproblem.
- Check the validity of a sequence of routines for solving a given eigenvalue problem.
- Provide the numerical quality of the result.

2.1 Selection and validation of routine

The goal of this section is to give the different methods of resolution of this problem and the corresponding LAPACK routines. These methods depend upon some properties of the data. Since the first letter of the routine names depends only on the type of computation (single or double precision) and on the storage of the matrix elements (real or complex storage), we change it to an "x". Indeed, this first letter is a "S" for single real precision computation, a "D" for double real precision computation, a "C" for single complex precision computation and a "Z" for double complex precision computation.

2.1.1 Eigenvalue computation

When computing eigenvalues, we have first to reduce the matrix A to a matrix which has the same eigenvalues but with more null elements, in order to reduce the time of computation. This is usually achieved by applying the Householder method [13, 50], which constructs a unitary matrix S ($S^* = S^{-1}$) such that $H = S^*AS$ is an upper Hessenberg matrix, i.e. a matrix with null elements below the subdiagonal. The Householder method respects the symmetry, i.e. if A is a symmetric matrix (resp. hermitian) then the reduced matrix will be a tridiagonal symmetric matrix (resp. tridiagonal hermitian). In the hermitian case, we can transform the tridiagonal hermitian matrix into a real tridiagonal symmetric matrix.

If the matrix is nonsymmetric then the LAPACK routine for this reduction is xGEHRD, and in the real symmetric case (resp. hermitian), it is xSYTRD (resp. xHETRD) for a dense storage, xSPTRD (resp. xHPTRD) for a packed storage and xSBTRD (resp. xHBTRD) for a band matrix.

We have now constructed a matrix $H = S^*AS$ with more null elements than A and which has the same eigenvalues. Therefore, we can restrict our discussion to one of the two following types of matrices :

1. Upper Hessenberg
2. Real tridiagonal symmetric

There does not exist an efficient method to compute only a part of the spectrum of a Hessenberg matrix (real or complex), therefore, we always compute all the eigenvalues. We use the double-shift QR method [21, 25] (routine xHSEQR). This method is an iterative method which consists in the construction of a matrices sequence H_k unitary similar (i.e. for all k it exists a unitary matrix Q_k such that $H_k = Q_k^* H Q_k$). This sequence converges to an upper block triangular matrix called Schur form of H. Each of the diagonal submatrices correspond to eigenvalues with the same modulus. The shifts enable to accelerate the convergence to the Schur

form.
Practically, the size of the diagonal blocks is 1 (single eigenvalue) or 2 (complex conjugate eigenvalues).

If the matrix is real tridiagonal symmetric, we can compute the eigenvalues one by one by using the method of bisection (routine xSTEBZ), which enables to approach as close as desired the wanted eigenvalue. If all the eigenvalues are desired, we use the QL method (routine xSTEQR if we want to compute some eigenvectors and routine xSTERF which is more efficient if we only want to compute the eigenvalues). This method is the QR method in which L is a lower triangular matrix. It is prefered for its advantages of programming.

We remark heuristically that it is more efficient to use the method QL as soon as we want to compute more than 25% of the eigenvalues.

When using the QR (or QL) method, we construct a unitary matrix Q (($Q^* = Q^{-1}$), such that $T = Q^*HQ$ is in Schur form.

Remark 2.1 *The Schur form of a symmetric matrix is a diagonal matrix.*

2.1.2 Eigenvector computation

After computing the eigenvalues of A by the QR (or QL) method, we can compute all its eigenvectors by the backward substitution method (routine xTREVC in the nonsymmetric case and xSTEQR in the symmetric case) which we describe now : let T in Schur form constructed with the QR method, we compute its eigenvectors (easy and efficient) and multiply then by SQ : these new vectors are the eigenvectors of A. S and Q have been defined in subsection 2.1.1. Therefore, after computation of these two matrices we have to store them.

If we are only interested by computing an eigenvector associated to an eigenvalue for which we know a good approximation λ', or if the eigenvalues have not been computed by the QR (or QL) method, we may use the method of inverse iterations [38] (routine xSTEIN in the symmetric case and xHSEIN in the nonsymmetric case). This iterative method is defined in the following way (Power method for $(T - \lambda' I)^{-1}$) :

- Let u_0 be a non-null arbitrary vector
- For $k \geq 0$, we define u_{k+1} by $(T - \lambda' I)u_{k+1} = u_k$

It is proved that u_k converges toward the eigenvector associated to the nearest eigenvalue of λ', except if u_0 is orthogonal to this eigenvector.

We decide heuristically to use the method of inverse iterations if we want to compute less than 25% of the eigenvectors.

2.1.3 Formalization

We just have seen that the method of choice depends upon certain properties of the matrix (symmetric or not, real or complex, ...) and the number of eigenvalues and vectors wanted.

Here are the different steps to solve the eigenproblem along with the names of LAPACK's routines implementing those steps.

1. **Symmetric matrix** :
   (a) **Matrix transformation** : (Householder method)
      i. Dense storage : xSYTRD (real), xHETRD (complex).
      ii. Packed storage : xSPTRD (real), xHPTRD (complex).
      iii. Band storage : xSBTRD (real), xHBTRD (complex).
   (b) **Eigenvalue computation** :
      i. Less than 25% : xSTEBZ (Bisection method)
      ii. More than 25% (QR method)
         A. More than 25% of the eigenvectors are desired : xSTEQR
         B. Less than 25% of the eigenvectors are desired : xSTERF
   (c) **Eigenvector computation** :
      i. Less than 25% : xSTEIN (Inverse Iterations)
      ii. More than 25%
         A. Eigenvalues computed with xSTEBZ : xSTEIN (Inverse Iterations)
         B. Eigenvalues computed with xSTEQR : xSTEQR (Backward Substitution)
2. **Nonsymmetric matrix** :
   (a) **Matrix transformation** : xGEHRD (Householder method)

(b) **Eigenvalue computation** : xHSEQR (QR method)

(c) **Eigenvector computation** :

i. Less than 25% : xHSEIN (Inverse Iterations)

ii. More than 25% : xTREVC (Backward Substitution)

In the case 1(c)iiA we consider that a part of the eigenvalues is given by the user.

2.1.4 The driver and expert routines

For standard eigenproblems, we can also use driver or expert routines calling directly a sequence of these previous routines. In the symmetric case, we have 28 routines (14 to compute a few eigenvalues and, if required, its associated eigenvectors and 14 routines to compute all of them) and 16 routines in the nonsymmetric case (8 with condition number computation and 8 without). We present here these standard eigenproblems and the names of the routines solving them.

1. **Symmetric matrix** :

   (a) **Computation of all the eigenvalues and, if required, all the eigenvectors** :

   i. Real dense storage : xSYEV

   ii. Complex dense storage : xHEEV

   iii. Real packed storage : xSPEV

   iv. Complex packed storage : xHPEV

   v. Real band storage : xSBEV

   vi. Complex band storage : xHBEV

   vii. Real tridiagonal storage : xSTEV

   (b) **Computation of a few eigenvalues and, if required, of the associated eigenvectors** :

   i. Real dense storage : xSYEVX

   ii. Complex dense storage : xHEEVX

   iii. Real packed storage : xSPEVX

   iv. Complex packed storage : xHPEVX

   v. Real band storage : xSBEVX

   vi. Complex band storage : xHBEVX
   vii. Real tridiagonal storage : xSTEVX

2. **Nonsymmetric matrix** :

   (a) **Computation of all the eigenvalues and, if required, all the eigenvectors** :
      i. Without computation of condition number : xGEEV.
      ii. With computation of condition number : xGEEVX.

   (b) **Computation of all the eigenvalues and, if required, all the Schur vectors** :
      i. Without computation of condition number : xGEES.
      ii. With computation of condition number : xGEESX.

This presentation which looks like a decision tree allows to select a sequence of routines for a given problem, but it does not enable to determine whether a sequence of routines can solve a given problem (validation of routine). Indeed, these trees advise the best routine to solve a given problem but do not provide a list of all the available routines. However, we can't exclude the use of a routine for the reason that it is not the best. Therefore, for the validation of routines, we have to create a knowledge base in which we give all the available routines to solve a given problem.

2.1.5 Validation of routine

We want to remark that most of these routines have been implemented with respect to a particular matrix storage (symmetric, band, packed, ...).
Therefore, we wish, for instance, to prevent the use of a routine treating nonsymmetric matrices if the matrix of the problem is symmetric, although it is mathematically feasable.

Therefore, we will have to take into account that we can not exclude the use of a routine computing all the eigenvalues (or all the eigenvectors), even if only a few of them are required, or to compute all the eigenelements of a matrix one by one ...

Here are the different available routines **(except the best one)** to solve a given problem :

1. **Symmetric matrix** :
   (a) **Eigenvalue computation** :
      i. More than 25% : xSTEBZ.
      ii. Less than 25% : xSTERF, xSTEQR.

   (b) **Eigenvector computation** :
      i. More than 25% : xSTEIN.
      ii. Less than 25% : xSTEQR (if the eigenvalues have been computed by this same routine).
2. **Nonsymmetric matrix** :
   (a) **Eigenvalue computation** : None.

   (b) **Eigenvector computation** :
      i. More than 25% : xHSEIN.
      ii. Less than 25% : xTREVC if the matrix is on Schur form.

Here is the list of all the driver and expert routines available for a given problem **(except the best one)** :

1. **Symmetric matrix** :
   (a) **Computation of all the eigenvalues (and eigenvectors)**
      i. Real dense storage : xSYEVX.
      ii. Complex dense storage : xHEEVX.
      iii. Real packed storage : xSPEVX.
      iv. Complex packed storage : xHPEVX.
      v. Real band storage : xSBEVX.
      vi. Complex band storage : xHBEVX.
      vii. Real tridiagonal storage : xSTEVX.

   (b) **Computation of a few eigenvalues (and associated eigenvectors)**
      i. Real dense storage : xSYEV.
      ii. Complex dense storage : xHEEV.
      iii. Real packed storage : xSPEV.

   iv. Complex packed storage : xHPEV.
   v. Real band storage : xSBEV.
   vi. Complex band storage : xHBEV.
   vii. Real tridiagonal storage : xSTEV.

2. **Nonsymmetric matrix** : None.

2.2 Validation of the result

The solution of a given problem is more or less sensitive to data perturbations. The condition number of the problem is a measure for this variation.
We will assume here that the computations themselves are done without rounding errors. Thus, the computed results correspond exactly to the data.
A problem is said well-conditioned when its sensitivity to the data perturbation is not high. On the same way, it is said ill-conditioned when it is very sensitive to the perturbations.

The condition number of an eigenproblem consists of a measure of the variation of the eigenvalues and vectors due to a matrix perturbation ΔA on A. This has been studied in detail in [10]. We recall here the results used in LAPACK and in the expert system.

2.2.1 Condition number in the symmetric case

The eigenvalues are well conditioned with a condition number equal to 1. The condition number of the eigenvector or invariant subspace depends only upon the distance between the cluster of eigenvalues associated to the computed invariant subspace and the rest of the spectrum : let

- λ be an eigenvalue and x its associated eigenvector,
- σ be a cluster of eigenvalues and M its associated invariant subspace.

then

$$\begin{aligned} cond(\lambda) &= 1 \\ (cond(x))^{-1} &= dist\Big(\lambda, sp(A) - \{\lambda\}\Big) \\ (cond(M))^{-1} &= dist\Big(\sigma, sp(A) - \{\sigma\}\Big) \end{aligned}$$

2.2.2 Condition number in the nonsymmetric case

The formulation of the condition number is complicated [10] and is related to the Jordan form of the matrices. The computation of this Jordan form is very difficult [26]. Therefore, we present here a method to estimate the condition numbers which is more efficient than computing them exactly. This method [3] is for nonsymmetric matrices in Schur form (upper block triangular matrix given by the method QR).

Let T (n by n matrix) on Schur form :

$$T = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

where T_{11} (m by m matrix) is associated with m eigenvalues of the same modulus for which we want to compute the condition number.

Remarks :

- *If the eigenvalues are complex conjugate then the matrix* T_{11} *will be* 2 *by* 2.
- *In practice, if the multiplicity of the eigenvalue is* $m > 1$ *then* T_{11} *will have* m *distinct close eigenvalues (rounding errors). Therefore, we prefer to consider clusters of eigenvalues rather than multiple eigenvalues.*

We assume henceforth that ΔT is a matrix perturbation of T, $\epsilon_2 = ||\Delta T||_2$ and $\epsilon_F = ||\Delta T||_F$ where $||T||_F = \sqrt{\sum_{i,j=1}^{n} T_{ij}^2}$.

1. **Single eigenvalue and associated eigenvector**

   (a) **Single eigenvalue** :

   Let λ be an eigenvalue of T, and λ' the eigenvalue of $T + \Delta T$ closest to λ, we can prove [50] :
   $\boxed{|\lambda - \lambda'| \leq \epsilon_2 \, ||P||_2 + O(\epsilon_2^2)}$ with $||P||_2 = \dfrac{|x^* y|}{||x||_2 \, ||y||_2}$ where x and y are respectively the right and left eigenvectors of T associated to λ ($Tx =$

$\lambda x, \;\; T^* y = \overline{\lambda} y)$.

Remark : *In this case, the computation of $||P||_2$ requires only the right and left eigenvectors associated with the considered eigenvalue.*

(b) **Associated eigenvector** :

Let x be the right eigenvector of T associated to λ and x' the correspon-ding perturbed eigenvector of $T + \Delta T$, we prove [16] :

$$\boxed{\theta(x, x') \leq \frac{2\, \epsilon_F}{sep\,(T_{11}, T_{22})} + O(\epsilon_F^2)}$$

where $sep\,(T_{11}, T_{22})$ is defined by :

i. if T_{11} is 1 by 1 (real eigenvalue), we have $T_{11} = \lambda$ and then [48] :

$$sep\,(T_{11}, T_{22}) = \min_{||x||_2=1} \;||(\lambda\, I - T_{22})\; x||_2$$

ii. if T_{11} is a 2 by 2 block (complex eigenvalue), then we use a unitary rotation to triangularize this block to get :

$$T' = \begin{pmatrix} \lambda & t_{12} \\ 0 & T'_{22} \end{pmatrix}$$

$$\text{and} \qquad sep\,(T_{11}, T_{22}) = \min_{||x||_2=1} \;||(\lambda\, I - T'_{22})\; x||_2$$

In both cases, $\boxed{sep\,(T_{11}, T_{22}) = \min_{||x||_2=1} \;||(\lambda\, I - T'')\; x||_2}$ where $T'' = T_{22}$ if the eigenvalue is real and $T'' = T'_{22}$ if the eigenvalue is complex.

Remark : *We can give an estimate of sep (T_{11}, T_{22}) by computing $||K^{-1}||_1$ où $K = (T'' - \lambda\, I)$ because sep $(T_{11}, T_{22}) = ||K^{-1}||_2^{-1}$ and*

$$\frac{1}{\sqrt{n-1}}\; ||K^{-1}||_1 \;\leq\; ||K^{-1}||_2 \;\leq\; \sqrt{n-1}\; ||K^{-1}||_1$$

2. **Cluster of eigenvalues and associated invariant subspace**

   (a) **Cluster of eigenvalues** :

   The computation of the condition number of an eigenvalue of multiplicity $m > 1$ requires its index [10] (size of its largest Jordan block associated). But practically, it is very difficult to compute the index. Therefore, we consider the following quantities :

   - $\overline{\lambda} = \frac{trace(T_{11})}{m}$ the mean of the block of eigenvalues of T_{11},
   - $\overline{\lambda}'$ the mean of the block of eigenvalues of $T + \Delta T$ corresponding to $\overline{\lambda}$,

   and we prove [29] that $\boxed{|\overline{\lambda} - \overline{\lambda}'| \leq \epsilon_2 \, \|P\|_2 + O(\epsilon_2^2)}$ with the spectral projector P defined by :

   $$P = \begin{pmatrix} I_m & R \\ 0 & 0 \end{pmatrix}$$

   where R is the solution of the Sylvester equation $T_{11}R - RT_{22} = T_{12}$.

   Remark : *We have* $\|P\|_2 = \sqrt{1 + \|R\|_2^2}$, *but its computation is quite expensive. Therefore, we prefer to compute* $\|P\|' = \sqrt{1 + \|R\|_F^2}$. *Indeed,*

   $$\|R\|_2^2 \leq \|R\|_F^2 \leq n \, \|R\|_2^2 \Longrightarrow \frac{1}{\sqrt{1 + \|R\|_F^2}} \leq \frac{1}{\|P\|_2} \leq \frac{\sqrt{n}}{\sqrt{1 + \|R\|_F^2}}$$

   (b) **Associated invariant subspace** :

   Let :

   - M be the right invariant subspace of T_{11} and M' the corresponding perturbed subspace,
   - $\theta_{max}(M, M')$ the angle between the subspaces M and M' defined by :

   $$\theta_{max}(X, Y) = \max_{x \in X, x \neq 0} \; \min_{y \in Y, y \neq 0} \; \theta(x, y) = \max_{y \in Y, y \neq 0} \; \min_{x \in X, x \neq 0} \; \theta(x, y)$$

We prove [16] that $\boxed{\theta_{max}(M, M') \leq \frac{2\ \epsilon_F}{sep\ (T_{11}, T_{22})} + O(\epsilon_F^2)}$

$$\text{where} \quad sep\ (T_{11}, T_{22}) = \min_{X \neq 0} \frac{\|T_{11}X - XT_{22}\|_F}{\|X\|_F} = \sigma_{min}(K)$$

$$\text{with} \quad K = I_{n-m} \otimes T_{11} - T_{22}^T \otimes I_m.$$

σ_{min} designs the smallest singular value and $A \otimes B$ is the tensorial product (or Kronecker product) of the matrices A and B.

Remark : *An estimate of sep* (T_{11}, T_{22}), *is given by* $\|K^{-1}\|_1^{-1}$.

$$\textit{Indeed,} \quad sep\,(T_{11}, T_{22}) \ = \ \sigma_{min}(K) \ = \ \|K^{-1}\|_2^{-1}$$

$$\textit{and} \ \frac{1}{\sqrt{m\,(n-m)}} \ \|K^{-1}\|_1 \ \leq \ \|K^{-1}\|_2 \ \leq \ \sqrt{m\,(n-m)} \ \|K^{-1}\|_1$$

2.2.3 Estimation of the condition number

Let T (n by n nonsymmetric matrix) in Schur form :

$$T = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

where the eigenvalues of T_{11} (m by m matrix) define the considered cluster. For instance, we group the eigenvalues which are equal at the machine precision.

Given the previous upper bounds of subsection 2.2.2, we only need to compute $\|P\|_2$ and $\frac{1}{sep\ (T_{11}, T_{22})}$ to estimate respectively the condition numbers of the eigenvalues and eigenvectors.
In practice, we prefer to estimate the reciprocals of the condition numbers ($\|P\|_2^{-1}$ and $sep\ (T_{11}, T_{22})$), because these two quantities lie between 0 and 1. The values close to 0 correspond to large condition numbers.

The estimate of a single eigenvalue costs $O(n)$ operations, and about $O(n^2)$ operations for the associated eigenvector. The LAPACK routine corresponding to these estimates is xTRSNA. Thanks to a parameter of this routine, we can chose to estimate only one of the two condition numbers (eigenvalue or eigenvector) or both.

The computation of the condition number of a cluster of eigenvalues or of the associated invariant subspace cost $O(n^3)$ operations or $O(n^2)$ if $m \ll n$. A LAPACK routine estimates these condition numbers : it is xTRSEN.

Remark : *Since* $\|P\|_2 = 1$ *and* $sep\,(T_{11}, T_{22}) = dist\,(sp\,(T_{11}), sp\,(T_{22}))$ *when the matrix is symmetric, we can compute in this case these quantities by hand.*

2.2.4 The stability of the algorithms

The stability of an algorithm allows us to study the impact of rounding errors on the result [19] : let us consider the following problem :

$$\text{Compute } x \text{ such that } F(x, d) = 0 \tag{2.1}$$

The idea of the backward error [49] is to prove that the approximate solution $\overline{x}$ is the exact solution of an approximate problem, i.e. $F(\overline{x}, \overline{d}) = 0$ with $\|\overline{d} - d\|$ as small as possible. We suppose that the floating-point system is consistent, i.e. for any arithmetic operation or elementary function T and any operation or function g, there exists constant K such that :

$$\forall x, y \in \mathcal{F} \qquad |fl\,(xTy) - xTy| \leq K\epsilon$$

$$\forall x \in \mathcal{F} \qquad |fl\,(g(x)) - g(x)| \leq K\epsilon$$

where $\mathcal{F}$ is a set of floating-point numbers and where ϵ is defined by :

$$\forall x \in \mathbb{R} \text{ such that } fl(x) \text{ exists } |fl(x) - x| \leq |x|\epsilon$$

Definition 2.1 *[11] An algorithm is numerically stable in* $\mathcal{V}$ *if for any* $d \in \mathcal{V}$ *such that* $F(x, d) = 0$, *the set* $\overline{D}(d) = \{\overline{d}$ *such that* $F(\overline{x}, \overline{d}) = 0\}$ *is non-empty and if there exists a constant* K *such that*

$$\forall d \in \mathcal{V} \qquad \inf_{\overline{d} \in \overline{D}(d)} \|\overline{d} - d\| \leq K\epsilon$$

The backward error of a stable algorithm in $\mathcal{V}$ *is given by :*

$$B\epsilon = \sup_{d \in \mathcal{V}} \inf_{\overline{d} \in \overline{D}(d)} \|\overline{d} - d\|$$

Remark 2.2 *Let Δd be a data perturbation. Taking into account the rounding errors, we obtain the following error estimation :*

$$\|\Delta\,\overline{x}\| \leq C\,(B\epsilon + \|\Delta d\|)$$

where $\Delta\,\overline{x} = fl\,(x + \Delta x) - x$, and C is the condition number of the problem 2.1.

Stability of the LAPACK routines

We can prove the following results [25, 50] :

1. **Householder reduction of A to Hessenberg form**
   The computed matrix H satisfies $H = Q^*\,(A+E)\,Q$, with $\|E\|_F \leq c\,n^2\,\epsilon\,\|A\|_F$ where c is a little constant and n is the dimension of the matrix A.

2. **Tridiagonalization of Householder**
   Let A be the n by n symmetric matrix to tridiagonalize and $\lambda_i,\ i = 1, n$ its eigenvalues. Let T the tridiagonal matrix computed with the Householder method and $\mu_i,\ i = 1, n$ its eigenvalues. We have then :

$$\sqrt{\frac{\sum(\mu_i - \lambda_i)^2}{\sum \lambda_i^2}} \leq 40\,n\,\epsilon\,(1 + 20\,\epsilon)^{2n}$$

3. **QR method**
   The Schur matrix T computed by the QR method satisfies $T = Q^*\,(A + E)\,Q$ with $\|E\|_2 \simeq \epsilon\,\|A\|_2$

4. **Method of bisection**
   Let T be a n by n symmetric tridiagonal matrix with diagonal elements $\alpha_i,\ i = 1, n$ and subdiagonal $\beta_i,\ i = 1, n$.
   This method consists in approximating an eigenvalue of T by successives valuation of the Sturm sequences in values μ closer and closer to λ.
   It can be shown that for any μ the computed values of the Sturm sequences :

$p_0(\mu), ..., p_n(\mu)$ are the exact ones of a tridiagonal matrix with diagonal elements $\alpha_i + \delta\alpha_i,\ i = 1, n$ and subdiagonal $\beta_i + \delta\beta_i,\ i = 1, n$, where $\delta\alpha_i$ and $\delta\beta_i$ satisfy for $i = 1, n$:

$$|\delta\alpha_i| \leq (3.01)\,\epsilon\,(|\alpha_i| + |\mu|)$$

$$|\delta\beta_i| \leq (1.51)\,\epsilon\,(|\beta_i|)$$

5. **Inverse iterations method**
   The computed eigenvector x is such that (λ, x) is an eigenelement of a matrix $A + E$ with $||E||_\infty \leq c\,\epsilon\,||A||_\infty$ where c is a constant of order unity.

Therefore, we may further assume that all the LAPACK routines are numerically stable.

2.2.5 Error estimation

We want to validate a result of computation of eigenvalues and/or vectors, i.e. we use a sequence of routines to solve the eigenproblem and we need to have an **estimate** of the error done on the result.
We just saw that the quality of the result depends upon the condition number of the computed eigenvalue or vector and the stability of the algorithm. We have also to take into account that the result depends upon the data precision.

In order to estimate the error in the result, we will theoretically have to :

1. Evaluate the error in the matrix (ΔA)
2. Compute the arithmetic stability of the used algorithms and estimate the backward error (B)
3. Estimate the condition number of the computed eigenvalue or vector if necessary (C)

The system will use the following error formula : $C * (B\epsilon + \Delta A)$, where ϵ represents the computer precision.

Taking into account that the used routines are stable and that it is very difficult to estimate B, we have chosen the arbitrary value $B = 1$ for all the computations,

that means we assume that major part of the error comes from errors on A.
Strictly speaking, we would have to estimate the error ΔH in the matrix reduction and use this bound subsequently. Since this step is well conditioned, we assume for simplification that $\Delta H = \Delta A$.
We obtain then the following simplified errors :

$$|\Delta\lambda| \leq C_\lambda(\epsilon + ||\Delta A||)$$

$$|\theta_{max}(M, M')| \leq C_\theta(\epsilon + ||\Delta A||)$$

where C_λ and C_θ are the estimates of the condition numbers of the eigenvalues and vectors respectively, and where $\Delta\lambda$ and $\theta_{max}(M, M')$ represent the errors when computing an eigenvalue and an invariant subspace, respectively.

The estimate for the condition numbers C_λ and C_θ depends upon the matrix :

- if the matrix is symmetric then the condition number of an eigenvalue is always equal to 1 and the condition number of an invariant subspace of dimension m is equal to the distance between the block of associated eigenvalues and the rest of the spectrum. The choice of m depends upon the results ; in practice we will group close eigenvalues.
- if the matrix is nonsymmetric, the condition numbers C_λ and C_θ can be estimated by an expert routine of LAPACK when computing the eigenvalues and vectors, or after computation with a LAPACK routine. The choice of the size m still depends upon the results.

Chapter 3

Choices for the modelisation

We will discuss in the first section of this chapter our motivations for the selection of the development shell SHIRKA [41] in order to model the numerical expertise presented in Chapter 2. In the following section, we describe how SHIRKA is used to design our expert system SESAME. And, in the last section, we describe some problems encountered when developing SESAME.

3.1 Why did we choose SHIRKA ?

The prototype deals only with eigenproblems for dense matrices but we must keep in mind that a true expert system should cover all the LAPACK library and even more. Therefore we want a development shell allowing an easy extension and evolution.

In particular, only dense matrices are treated in the LAPACK library but the field of sparse linear algebra is very active nowadays. Hence an expert system including large sparse eigenproblems should evolve easily and rapidly take into account new algorithms and new results on convergence or accuracy.

The expert system should also provide an efficient interface for scientific users and allow a natural modeling of the numerical expertise.

3.1.1 Presentation of SHIRKA

The development shell SHIRKA is written in Le-Lisp, and provides means to describe knowledge bases and to apply a classification mechanism upon them. All the knowledge is contained into the objects which are organized into hierarchies.

We will now give a brief definition of some notions of artificial intelligence used in SHIRKA.

An **object**, static entity, is a structured assembly of all the characteristics related to a concept. The specification of this assembly is done with the **attributes**. These objects are organized into **hierarchies** in order to create a lattice of **classes** which pass on some informations by the **inheritance** mechanism. The **classification** mechanism provides the dynamic of the objects. A class is a mould from which we create some specimen called **instance** of this class.

The inheritance mechanism enables the classes to pass on, during the classification process, some information (the values of the attributes) to its subclasses or to its instances. Inheritance can be single or multiple. With multiple inheritance, a subclass inherits from several classes, it means that it inherits the union of the information contained in these classes.

The classification process allows to locate the instance in the hierarchy in function of the values of its different attributes and to determine others by inference.

3.1.2 Motivations of this choice

We have chosen the artificial intelligence approach for its standard advantages. The evolution of the base will be easier. Thanks to AI, the system can explain its answer because it does not forget the way which led to its conclusion. Moreover, the classification applied on the hierarchies provides a dynamic of them, contrary to C++ for instance.

The modeling with hierarchies of objects and the multiple inheritance allow to describe complex objects and are well suited to describe mathematical entities. Each object represents a mathematical context, and the properties of this context are the attributes. The inference rules are implicitly embedded into the knowledge base because of the organization in hierarchies and to default values or attached procedures. No explicit rule such as "if-then-else" has to be written : the basic inference mechanism is the classification process.

For all these reasons, it is easier to design and maintain a knowledge-base with modeling by objects rather than by rules. Finally, we can both select and validate a sequence of routines with the same hierarchy by the attributes classes, contrary to a modeling by rules.

As we have seen, there are several advantages to prefer artificial intelligence over a usual program, and to do a modeling by objects.

3.2 SHIRKA for SESAME

We present here how we will use SHIRKA to create SESAME ; it means that we indicate how we use the notions presented in subsection 3.1.1.

The mathematical contexts will be represented by the **objects**.

The **attributes** will represent the mathematical properties : information about the matrix, about the number of eigenelements to compute, about the accuracy wanted ...

We will create a **hierarchy of classes** with the different values that can be taken by these attributes.

In some classes, we will have enough information about one single step of the eigenproblem in order to find the name of the appropriate routine. This name will be contained in an attribute.

In some cases, it is interesting to group some attributes. This is possible with the **attribute classes** defined in SHIRKA. For instance, we will group all the attributes dealing with the names of the routines in an attribute class. Indeed, to determine routines names we will be able to classify our instance without taking into account this attribute class, since we want here to infer the values of these attributes. On the other hand, for the validation of routines we will have to take into account this attribute class, because the names of the routines are here some data of the problem. These attribute classes enable us to define only one base for both selecting and validating a sequence of routines.

The general idea of a base is to create a hierarchy composed by a first class : a concept (for instance the name of a problem). It contains some attributes for its characterization. Next, we define a hierarchy of classes by defining some subclasses to this first class, and so on. When defining a subclass, we specify the value of some attributes. Therefore, the deeper we go in the hierarchy, the more the problem is defined. It means that the number of valuate attributes increases, when following the tree away from the first class. When the problem is enough well defined, we can conclude the name of the routine. As we will see next, this conclusion may appear in the leafs of the hierarchy or just before for some modelings.

3.3 The story of SESAME

The main objective is to prove the feasibility of an expert system in respect with the following tasks :

- Find the sequence of routines to be executed for solving a given eigenproblem
- Check the validity of a sequence of routines for solving a given eigenproblem
- Provide the numerical quality of the result
- Can evolve easily

Remark : *We do not intend here to generate code, since in scientific computing, the computations are not always done on working stations but often on supercomputers.*

We will now formalize all the expertise described in Chapter 2.

3.3.1 A unique base for SESAME

The first idea is to create only one base with all the information. This idea is very interesting for the simplicity of its use : we create only one instance, we classify it and we get the results. However, this solution was rejected quickly, because we were faced with several difficulties :

1. *Size of the base :* the base became so big that it was very difficult to add new problems.
2. *Validation of a sequence of routines :* to validate a sequence of routines, we begin with a classification and then we have to interpret the results of the classification. In fact, we have to count the number of classes where a result is given (classes where a name of routine is determined) and which are classified "sure". But with a unique base, it was difficult to find these classes with a result, and to make sure they would stay there after an evolution of the base.
3. *Duplication of knowledge :* the solution paths of some different eigenproblems coincide after a few steps of resolution.

   For instance, whenever we consider the eigenvalue computation of a hermitian matrix on one hand and of a real symmetric matrix on the other hand, we first reduce it to a real tridiagonal symmetric matrix. In this example, the routine of matrix reduction is not the same since the storage of the matrix elements is different. But after the matrix reduction, we have the same problem in both cases : computation of eigenvalues of a real tridiagonal symmetric matrix. Therefore, the same routines to solve it are detected.

We are faced with the following problem : to select the routine of matrix reduction, we have to distinguish the two eigenproblems in the hierarchies and SHIRKA does not allow to group together these two problems (hermitian case on one hand, and real symmetric on the other hand) after the determination of the routine for the matrix reduction, in order to select the routines for eigenvalues and eigenvector computation.

For the solution of the last problem we have thought of several solutions :

- Duplicate the knowledge to select the routines for the eigenvalue and eigenvector computation. But this was quickly rejected : inappropriate and too difficult to add some new objects if the base grows. Moreover, the first two problems were not solved.
- Begin by selecting the routine of eigenvalues and eigenvectors computation in the symmetric case and finally the routine of matrix reduction. This is not realizable because some eigenproblems require a special method of matrix reduction in order to select the method of eigenvector computation.
- Create several hierarchies : one for each step of the solution process. This is the solution that we have adopted. It is more flexible but it requires an interface for communication between each step.

3.3.2 Several bases

First, we create two hierarchies for the data : one to describe the matrix (symmetric or not, real or complex ...) and another for the problem (number of eigenvalues wanted : none, a few or many ; similarly, for the eigenvectors).

Next, we create one hierarchy for each of the three steps : matrix reduction, eigenvalue computation and eigenvector computation.
This modeling is of interest since it explains the numerical resolution of the eigenproblem. Moreover, the different bases are small enough to allow the addition of new

problems with little effort. The validation of routines will be straightforward because the classes in which the results are given are always the leafs of the hierarchies. Therefore, to validate a routine, we will only have to check if a leaf has been classified "sure" by the classification. But as we mentioned in the previous subsection, the system sometimes has to know the name of the routine for the matrix reduction in order to find the name of the routine for the eigenvalue computation. But, as we have here one hierarchy for each of the three steps, we must have a link between the different hierarchies. We choose to create a new attribute to follow the form of the matrix through the different steps of the resolution. This attribute will be helpful for particular cases, such as eigenvalue computation of a reduced matrix. Indeed, if the matrix is already reduced then the first step of the eigenproblem resolution (matrix reduction) has to be omitted. Due to this attribute the system detects whether the matrix reduction has to be done or not and only gives the names of the routines to execute.

As we said at the beginning of section 2.1, the first letter of each routine name only depends upon the type of arithmetic : real or complex, single or double precision. Therefore, in order to avoid the unnecessary duplication of the knowledge, we have to determine and treat the first letter of the routine names independently. This will be done by a new hierarchy which can be considered as data. But this letter causes some problems. For instance, for the validation of selected routines, we have to be sure that the user has not created nonexistent names, by combining the first letter with forbidden remaining parts of names. Therefore, we have decided to create a new hierarchy (Nom-routines) which provides the names of the existent routines of LAPACK and for the validation of routines, the user has to select its name in the proposed list of routine names.

Chapter 4

Description of the knowledge base

The user provides some information about a matrix (real or complex storage, symmetric or not ...), the number of eigenvalues and vectors wanted and the computer precision. Then, he can choose between :

- Selecting the sequence of LAPACK routines to solve this problem.
- Validating the selected sequence of LAPACK routines.
- Validating the result of a routine (executed outside the system) by computing the condition number if necessary, and choosing another routine if the result is not acceptable.

The different concepts of the knowledge base are subdivided into three groups, described in the three following sections :

- **Description of the problem** : the matrix (object *Matrice*), the requirements (number of eigenvalues and eigenvectors) (object *Pb*), the environment (precision) (object *Type*).
- **Routines** : the driver routine (object *Driver-routine*), the routine for the matrix reduction (object *Transfo-matrice*), the routine for the eigenvalue computation (object *Calc-val-prop*), the routine for the eigenvector computation (object *Calc-vect-prop*), the routines names to check if the user's routines belong to LAPACK (object *Nom-routines*).

- **Validation of the result** : the routine to estimate the condition number (object *Calc-cond*), the formula to estimate the error (object *Precision*), the error analysis (object *Qualite-resultat*).

4.1 The concepts *Matrice, Pb, Type*

They enable to describe the problem (the data) :

- **Matrice** : defines the matrix thanks to three attributes : **real, symmetric, storage** ;
  - **Real**, boolean, is true if the matrix elements are real, false if they are complex.
  - **Symmetric**, boolean, is true if the matrix is symmetric, false if not. We assume here that symmetric means hermitian in the complex case.
  - **Storage**, string of characters which indicates the storage of the matrix : Dense, Packed, Band, Tridiagonal.
- **Pb** : defines the number of eigenvalues and vectors wanted thanks to two attributes **nb-eigva** et **nb-eigve**. These are two strings of characters which valuable domain is : None, A few (less than 25%), Many (more than 25%).
- **Type** : defines the environment of arithmetic : real or complex, single or double precision. The two first attributes are **mat** which represents the matrix described previously and **precision**, a string of characters, which value domain is single or double. The third attribute **firstlet**, string of characters (attribute class : **results**), is inferred when classifying and it gives the first letter of the LAPACK routines to execute.

4.2 The concepts *Driver-routine, Transfo-matrice, Calc-val-prop, Calc-vect-prop, Nom-routines*

They enable to infer or validate the sequence of routines to execute for the solution of the problem (attribute **problem** (concept *Pb*)) which uses the matrix defined by the attribute **mat** (concept *Matrice*) :

For any of the first four concepts, we affect an attribute string of characters for the name of the routine (respectively *driver, met-reduc, met-eigva, met-eigve*).

This attribute (attribute class : **results**) is either inferred when classifying (routine selection), or provided by the user (routine validation).

- **Driver-routine** : its attributes are **mat, problem, driver**(name of the driver-routine). Moreover, in the case of a nonsymmetric matrix, we indicate whether we want to estimate the condition numbers (attribute **comp-condition-number**, boolean), and whether we want to keep the Schur vectors (attribute **Schur-vect**, boolean).
- **Transfo-matrice** : attributes **mat, met-reduc**(name of the routine for the matrix reduction). A last attribute **mat-state** (attribute class : **info**) enables to infer after classification the state of the reduced matrix (Hessenberg matrix or real tridiagonal symmetric matrix).
- **Calc-val-prop :** attributes **mat, problem, mat-state** and **met-eigva**(name of the routine for the eigenvalue computation).
- **Calc-vect-prop** : attributes **mat, problem, mat-state, met-eigva** and **met-eigve**(name of the routine for the eigenvector computation).
- **Nom-routines** : attributes **typ, tree** and **proc**(attribute class : **results**)(name of all the LAPACK routines for the selected tree and the type of computation).

4.3 The concepts *Calc-cond, Precision, Qualite-resultat*

- **Calc-cond** : determines whether the computation of the condition number of the eigenvalues and/or vectors is required and by which routine this has to be done. The inputs are the attributes **mat** (concept matrice), **problem** (concept pb) and **known-cond**, boolean. Moreover, if the matrix is nonsymmetric, we indicate the value of the attribute **mat-state**. We indicate, if required, the multiplicity of the eigenvalue via the attribute **type-eigva**, a string of characters. We then know the appropriate computations by inspecting, after classification with inference, the attribute **computation** (attribute class : **results**) and the name of the routines to execute for these computations via the attribute **routine** (attribute class : **results**)
- **Precision** : estimates the error in the result from the values of the attribute of real type **cond, stab, delta** and **epsilon**, by using the formula : $cond * (stab * epsilon + delta)$. The values of *cond, delta, epsilon* have to be specified

by the user, with the possibility of excluding *delta*, but *stab* will be set to 1 in the system (if not specified). The estimated error is found in the attribute **prec** of real type.

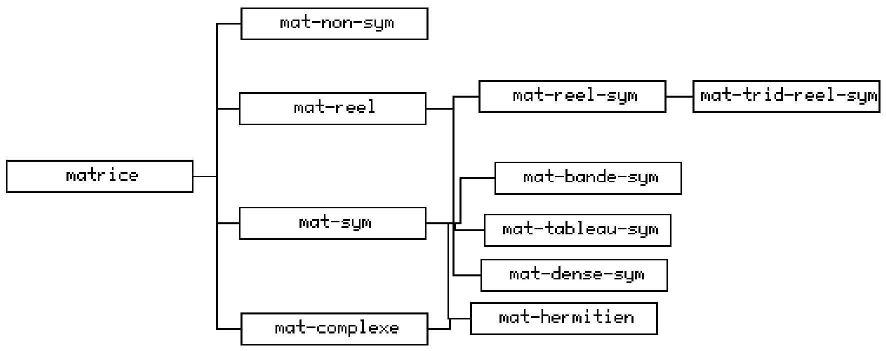
- **Qualite-resultat** : provides a diagnosis (attribute **diagnosis**, string of characters, attribute class : **results**) about the result. The attributes to know are **satisfied** (boolean) and possibly **firstlet** (string of characters).

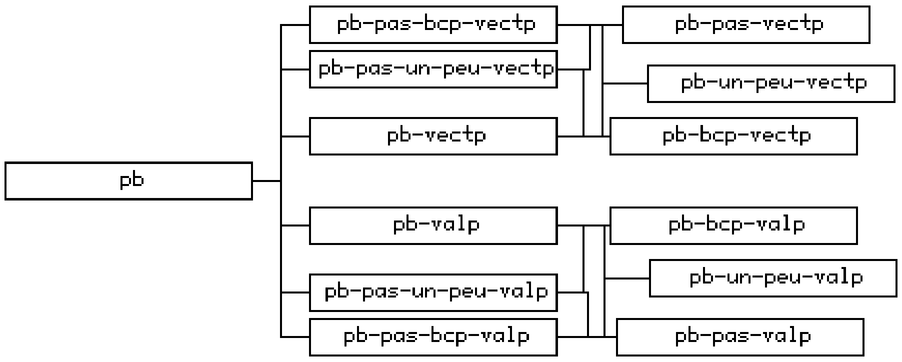
4.4 The knowledge base

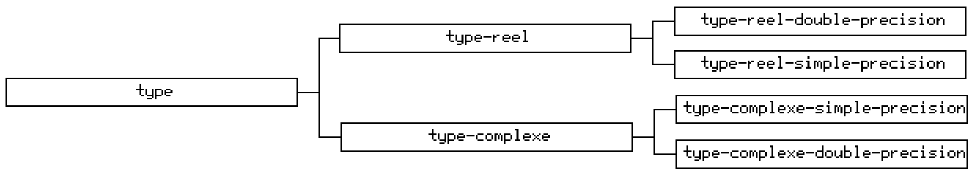
All the objects contained in the knowledge base are pictured below ; we recall that they are subdivided into three groups, corresponding to their use :

- **Description of the problem** : the matrix (object *Matrice*), the requirements (number of eigenvalues and eigenvectors) (object *Pb*), the environment (precision) (object *Type*).
- **Routines** : the driver routine (object *Driver-routine*), the routine for the matrix reduction (object *Transfo-matrice*), the routine for the eigenvalue computation (object *Calc-val-prop*), the routine for the eigenvector computation (object *Calc-vect-prop*), the routine names to check if the user's routines belong to LAPACK (object *Nom-routines*).
- **Validation of the result** : the routine to estimate the condition number (object *Calc-cond*), the formula to estimate the error (object *Precision*), the error analysis (object *Qualite-resultat*).

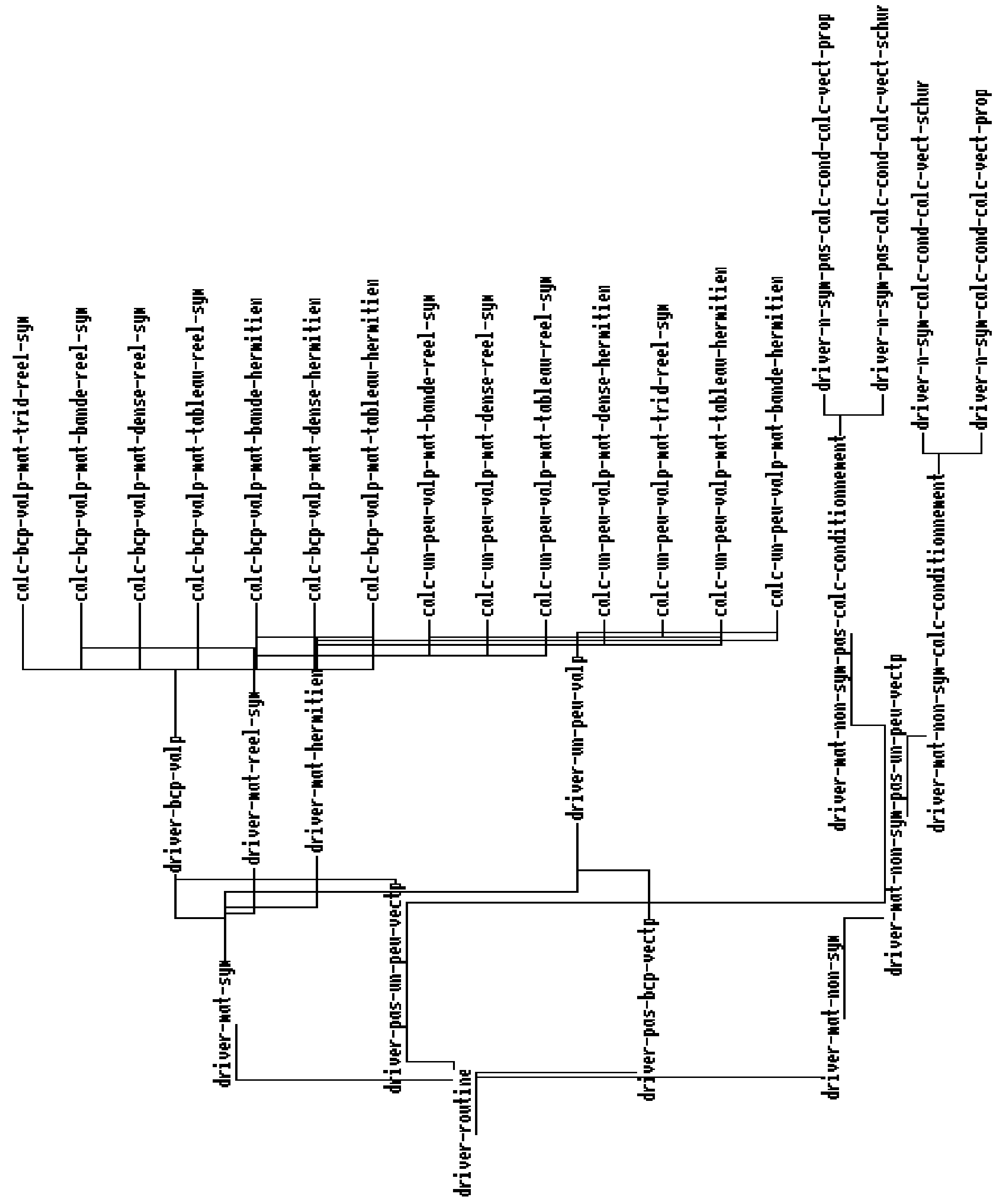
Objects *Matrice, Pb, Type*



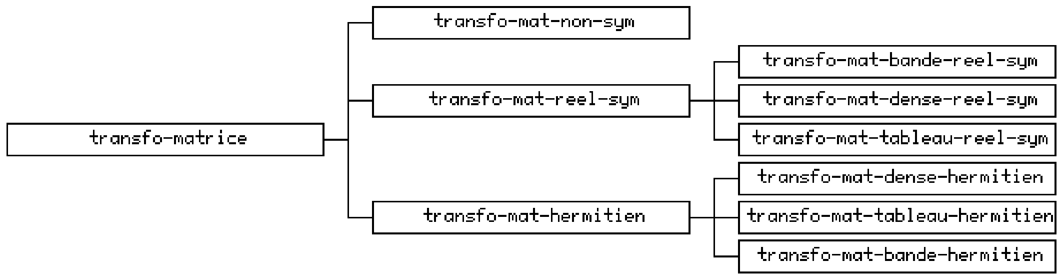


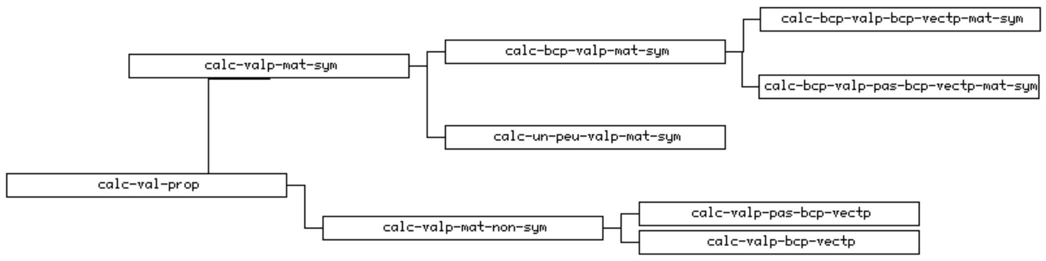

type
type-reel
type-complexe
type-reel-double-precision
type-reel-simple-precision
type-complexe-simple-precision
type-complexe-double-precision

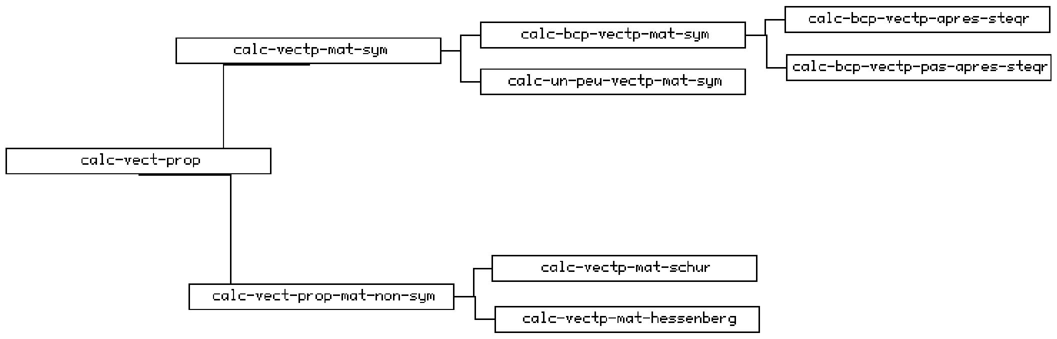
Object *Driver-routine*


driver-routine
driver-mat-sym
driver-pas-un-peu-vectp
driver-pas-bcp-vectp
driver-mat-non-sym
driver-bcp-valp
driver-mat-reel-sym
driver-mat-hermitien
driver-un-peu-valp
driver-mat-non-sym-pas-un-peu-vectp
driver-mat-non-sym-pas-calc-conditionnement
driver-mat-non-sym-calc-conditionnement
calc-bcp-valp-mat-trid-reel-sym
calc-bcp-valp-mat-bande-reel-sym
calc-bcp-valp-mat-dense-reel-sym
calc-bcp-valp-mat-tableau-reel-sym
calc-bcp-valp-mat-bande-hermitien
calc-bcp-valp-mat-dense-hermitien
calc-bcp-valp-mat-tableau-hermitien
calc-un-peu-valp-mat-bande-reel-sym
calc-un-peu-valp-mat-dense-reel-sym
calc-un-peu-valp-mat-tableau-reel-sym
calc-un-peu-valp-mat-dense-hermitien
calc-un-peu-valp-mat-trid-reel-sym
calc-un-peu-valp-mat-tableau-hermitien
calc-un-peu-valp-mat-bande-hermitien
driver-n-sym-pas-calc-cond-calc-vect-prop
driver-n-sym-pas-calc-cond-calc-vect-schur
driver-n-sym-calc-cond-calc-vect-schur
driver-n-sym-calc-cond-calc-vect-prop

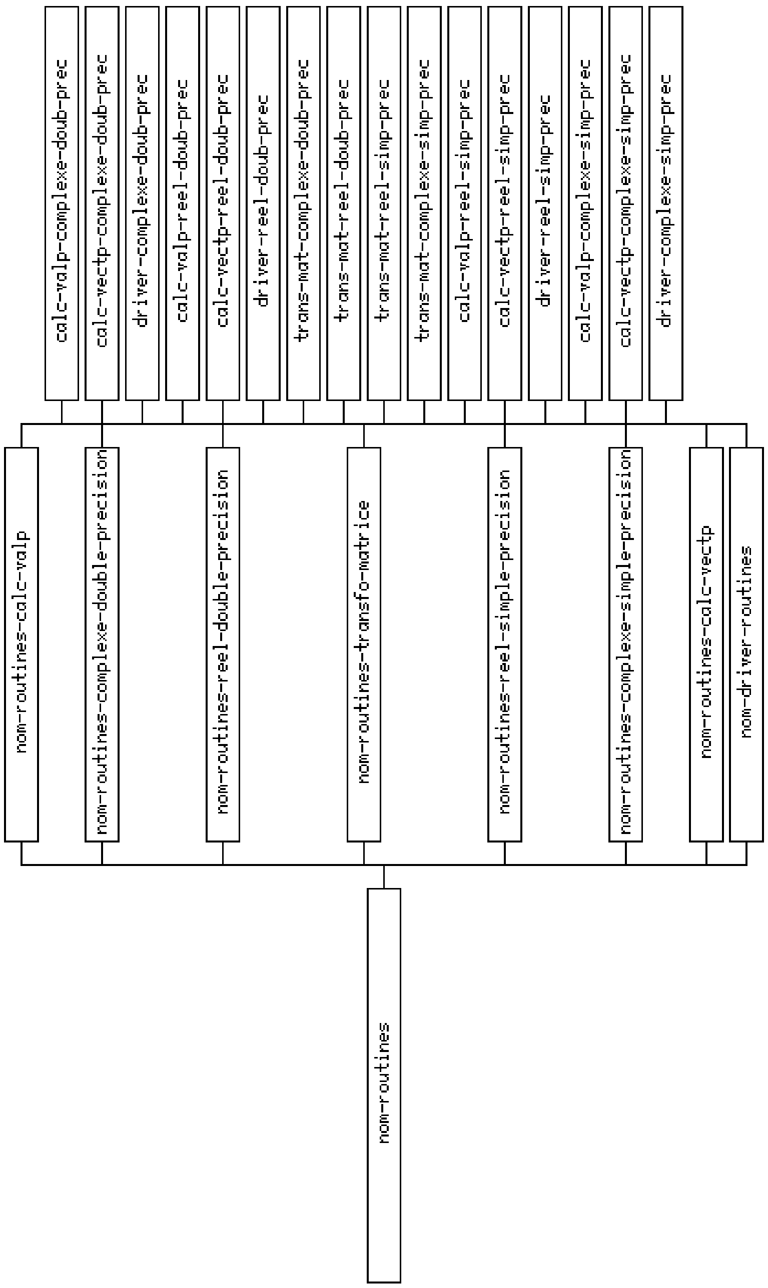
Objects *Transfo-matrice, Calc-val-prop, Calc-vect-prop*


transfo-matrice
transfo-mat-non-sym
transfo-mat-reel-sym
transfo-mat-hermitien
transfo-mat-bande-reel-sym
transfo-mat-dense-reel-sym
transfo-mat-tableau-reel-sym
transfo-mat-dense-hermitien
transfo-mat-tableau-hermitien
transfo-mat-bande-hermitien

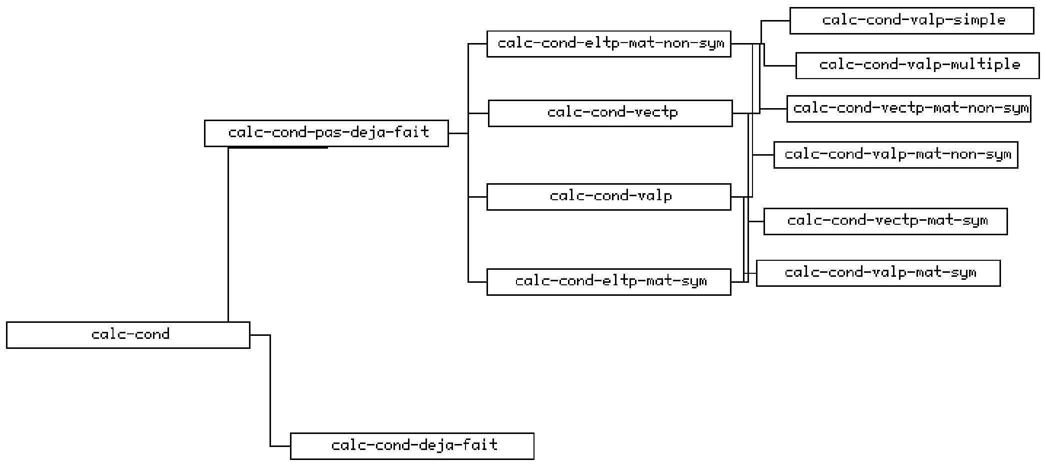


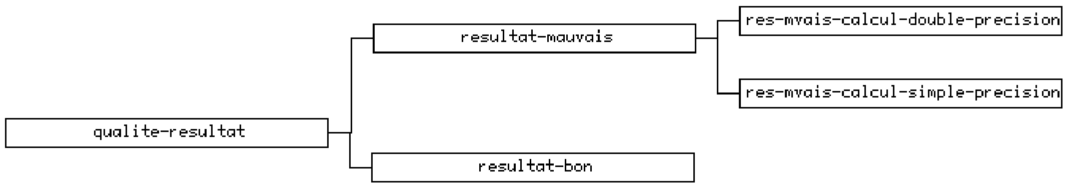

calc-vect-prop
calc-vectp-mat-sym
calc-bcp-vectp-mat-sym
calc-bcp-vectp-apres-steqr
calc-bcp-vectp-pas-apres-steqr
calc-un-peu-vectp-mat-sym
calc-vect-prop-mat-non-sym
calc-vectp-mat-schur
calc-vectp-mat-hessenberg

Object *Nom-routines*


calc-valp-complexe-doub-prec
calc-vectp-complexe-doub-prec
driver-complexe-doub-prec
calc-valp-reel-doub-prec
calc-vectp-reel-doub-prec
driver-reel-doub-prec
trans-mat-complexe-doub-prec
trans-mat-reel-doub-prec
trans-mat-reel-simp-prec
trans-mat-complexe-simp-prec
calc-valp-reel-simp-prec
calc-vectp-reel-simp-prec
driver-reel-simp-prec
calc-valp-complexe-simp-prec
calc-vectp-complexe-simp-prec
driver-complexe-simp-prec
nom-routines-calc-valp
nom-routines-complexe-double-precision
nom-routines-reel-double-precision
nom-routines-transfo-matrice
nom-routines-reel-simple-precision
nom-routines-complexe-simple-precision
nom-routines-calc-vectp
nom-driver-routines
nom-routines

Objects *Calc-cond, Precision,Qualite-resultat*




qualite-resultat
resultat-mauvais
resultat-bon
res-mvais-calcul-double-precision
res-mvais-calcul-simple-precision

Chapter 5

Use of SESAME

5.1 Selection of routine

The user's problem is to compute some eigenvalues and/or vectors of a given matrix.

With the concepts that we have described, we will be able to identify and select the sequence of routines that has to be executed for the resolution of his problem.

We identify here the different steps to find the sequence of routines.
All the classification will have to be done **with inference** and without taking into account the attribute class *results*.

1. Create an instance of the concept **matrice**, in order to get the properties of the matrix which will be necessary for the following.
2. Create an instance of the concept **pb**, in order to know the number of desired eigenvalues and/or vectors.
3. Create an instance of the concept **type** ; it defines the environment of computation. After classification of this instance, we know the attribute *firstlet*.
4. Create and classify an instance of the concept **driver-routine**, where the attributes *mat* and *problem* are two instances of the concepts matrice, and pb, respectively.

Two cases might appear :

- **A leaf of the tree is classified "sure"** : that means that the attribute *driver* has been inferred. Therefore, there exists a driver-routine which solves the given problem. The system only needs then to retrieve the value of the attribute *driver*, which is the wanted result.

- **No leaf has been classified "sure"** : there is no driver-routine that solves the problem. We have to create and classify successively :

  1. An instance of the concept **transfo-matrice**. Along with the attribute class *results*, the attribute class *info* is ignored when classifying. When the instance will be classified, the value of the attribute *mat-state* will be inferred in order to use it as attribute of the following instance.
  2. An instance of the concept **calc-val-prop**. We retrieve the value of the attribute *mat-state* too which may have been changed.
  3. An instance of the concept **calc-vect-prop**.

  For each of these three instances, at most one leaf has been classified "sure". We have to visualize the attributes of these leafs, in order to get the possible values of the attributes *met-mat, met-eigva* and *met-eigve*, and we have to display these attributes.

According to the particular case, we may have to display either only one routine name (attribute *driver*) or more (attributes *met-mat, met-eigva* and *met-eigve*). However, these attributes only contain the end of the routines name, since the first letter of the name was detained by the attribute *firstlet.*
Therefore, for each routine name, we will have to display its first letter with the attribute *firstlet*, then the rest of the name which will be contained, according to the case, in the attribute *driver, met-mat, met-eigva* or *met-eigve.*

However, we would like to mention here a problem with our approach. After reducing a symmetric or hermitian matrix, in order to compute eigenvalues and/or eigenvectors, we obtain in both cases a real symmetric tridiagonal matrix. The library LAPACK takes this fact into account and there is no specific routine for the computation of the eigenvalues and/or eigenvectors in the complex hermitian case, since the routines for the real case are available, and after reduction, the hermitian matrix is stored in real form.

For instance, if we want to compute a few eigenvalues of a dense stored hermitian matrix in single precision, we will have to use the following sequence of routines : CHETRD + SSTEBZ. But the system will give the answer CHETRD + CSTEBZ, since the first letter is determined from the storage of the input matrix numbers (here a complex number). In the same way the system will advise ZSTEBZ, CSTERF and ZSTERF instead of DSTEBZ, SSTERF and DSTERF respectively. This problem must be solved by an interface between the system and the user.

5.2 Validation of routine

The goal here is to establish whether a routine sequence, given by the user, really solves the given problem. Therefore, the user specifies a routine sequence besides the matrix and the problem.

In order to realize this we proceed as follows :

1. Create the instances of the concepts **matrice** and **pb**.
2. Create and classify (**with inference** and without taking into account the attribute class *results*) an instance of the concept **type**, where the attributes *mat* and *problem* will be given as input.
3. Create and classify (**with inference** and without taking into account the attribute class *results*) four instances of the concept **nom-routines**, where *typ* (concept type) and the attribute *tree* are given as input. We get back the list of existent routines for the selected tree (attribute *tree*) and the required type of computations (attribute *typ*) as value of the attribute*proc* (attribute class *results*). We create four instances in order to determine :
   (a) the driver-routines (*tree* = "driver-routine")
   (b) the routines of reduction of matrix (*tree* = "transfo-matrice")
   (c) the routines for the computation of the eigenvalues (*tree* = "eigva-comp")
   (d) the routines for the computation of the eigenvectors (*tree* = "eigve-comp")

   In contrast to the routines selection, all the following classifications have to be done **without inference** and with taking into account the attribute class *results*.

4. If the user has a driver-routine : create and classify an instance of the concept **driver-routine**, where we will also value the attribute *driver*.

   If a leaf is classified "sure", we can solve the given problem with the proposed driver-routine : the choice of the routine is correct.
   Otherwise, the user cannot solve the problem with this driver-routine : his choice is false.
   In both cases, it is useless to do the following step, because we can already answer the user.

5. If the user has a routine of matrix reduction (otherwise go to the next step ; indeed, the matrix may be already in reduced form, therefore he can directly compute the eigenvalues and/or vectors) : create and classify an instance of the concept **transfo-matrice**, where we have to valuate the attribute *met-mat*. We will not valuate *mat-state* which will be inferred during the classification.

   If a leaf is classified "sure", we can reduce the matrix with the proposed routine, and we proceed with the next step.
   Otherwise, this routine does not solve the given problem. It is the useless to continue, since this choice of routine is false.

6. If the user has a routine for eigenvalue computation (otherwise, we go to the next step : indeed, the user may desire only the computation of the eigenvectors if he already knows the eigenvalues) : create and classify an instance of the concept **calc-val-prop**, where we have to valuate the attribute *met-eigva*.

   If a leaf has been classified "sure", then we may compute the eigenvalues with the proposed routine : proceed with the next step.
   Otherwise, the choice of the routine has been wrong : it is useless to continue.

7. If the user has a routine for eigenvector computation : create and classify an instance of the concept **calc-vect-prop**, where we have to valuate the attribute *met-eigve*.

   If a leaf is classified "sure", we may compute the eigenvectors with the proposed routine. The choice of routines has been good.
   Otherwise, this routine is not acceptable for the problem.

5.3 Result validation

Here we deal with the following problem : a computation of eigenvalues and/or vectors has been done with a sequence of LAPACK routines and the user wants to know the accuracy in the result (see subsection 2.2.5).

We will accomplish this as follows :

1. In order to know whether an estimate of the condition numbers has to be done or not and, if yes, by which routine :
   - Create an instance of the concept **calc-cond** without valuating the attribute of the class *results*.
   - Classify this instance (classification **with inference** and without taking into account the attributes of the class *results*).
   - Return the attributes *computation* and *routine*
2. In order to know whether the solution is acceptable :
   - Create an instance of the concept **precision** without valuating the attribute *prec*.
   - Classify this instance (classification **with inference**).
   - Return the attribute *prec*.
   - Create an instance of the concept **qualite-resultat** with no value for the attribute of the class *results*.
   - Classify this instance (classification **with inference** and without taking into account the attribute of the class *results*).
   - Return the attribute *diagnosis*.

Chapter 6

Conclusion

We have proposed in this part an expert system, SESAME, based on artificial intelligence techniques and dealing with the following tasks :

- Find the sequence of routines for the solution of a given eigenproblem
- Check the validity of a sequence of routines for solving a given eigenproblem
- Provide a measure for the numerical quality of the result

The main advantage of such a system is in its flexibility to extend the knowledge base without changing the existing base. Indeed, the work done already corresponds to existing problems. If, in the future, new situations arise then, we only will have to add them to the existing base. These additions might consist in an enlargement of the value domain of some attributes and in creating some subclasses corresponding to these news values. For instance, the system does not include the case of large sparse matrices because this is not yet covered by LAPACK. However, it will be covered in the future, and then we will have to insert it in the expert system. To realize this, we only will have to enlarge the value domain of the attribute *storage* of the concept *matrice*, by adding the value *sparse* in order to create a subclass *sparse-matrix* for the classification of sparse matrices, and in the concepts of routine selection, we will add some subclasses corresponding to sparse matrices.

This modeling by objects enables us to use only one base for both selection and validation of routines. Thus, the knowledge is not unnecessarily duplicated. Moreover, this modeling by objects is natural because each object represents a mathematical context.

However, in its present state, the use of the system is close to artificial intelligence and it requires some knowledge on the domain. Therefore, it would be interesting to have an interface between the system and the user in order to improve the use by a scientist. Moreover, when a routine is declared improper, the artificial intelligence enables us to explain the reason for rejection. However, the explanations are in terms of objects and classes and they are quite difficult to understand by a scientist who ignores the system itself. It would be helpful to mask these artificial intelligence aspects for the user.

We have decided not to generate the calling sequences for the selected routines, because it would increase the complexity of the base enormously. Indeed, we have, for instance, to treat the particular cases where the user only wants to compute eigenvalues or vectors. Moreover, we have to take into account that the routine sequence contains some routines which are uninteresting from the mathematical point of view. These routines are at the moment advised in comments.

Our accuracy analysis is based on the numerical stability of the used algorithms and on the estimates of the condition numbers provided by LAPACK. The case of multiple eigenvalues is treated by the choice of a block size specified by the user. It would be interesting to analyze this case more precisely, particularly for defective eigenvalues, but our numerical knowledge in this area is not advanced enough yet to include it into an expert system.

Some new tools such as spectral portraits are very promising and might help a lot to assess the numerical quality of a result.
So the next chapters of the thesis will be devoted to the analysis and development of new methods which might be integrated in the toolbox Aquarels and advised by the expert system SESAME.

Part III

Spectral Portrait for Non Hermitian Large Sparse Matrices

Chapter 7

Introduction

In actual situations where we have to compute some eigenvalues of a large square sparse matrix $A \in \mathbb{C}^{n \times n}$, the matrix is quite often the result of a previous computation. Moreover, the backward analysis [49] of algorithms for computing eigenvalues aims at characterizing these computed eigenvalues as the exact ones of a nearby matrix. So, we have to consider that we compute some exact eigenvalues of a matrix $A + \Delta$, $\Delta \in \mathbb{C}^{n \times n}$ such that $\|\Delta\|_2$ is small, and we wish that these eigenvalues are not too far from the eigenvalues of A. In other words, we want to bound the error on an eigenvalue by some constant times the perturbation $\|\Delta\|_2$. This perturbation analysis leads to define and estimate, if it exists, the constant above, which is called the condition number of the eigenvalue [10].

In the hermitian case, it is well-known that the condition number of an eigenvalue is equal to one, so that the error in the computed eigenvalues is only of order $\|\Delta\|_2$. In contrast, condition numbers can be very large in the non hermitian case. In particular, the condition number of defective eigenvalues is infinite. Several condition number estimators have been designed, see [3] for example. Another approach to study the eigenvalues of perturbed matrices is to create the spectral portrait of the matrix. It amounts to estimate all the eigenvalues of all perturbed matrices $A + \Delta$, with $\|\Delta\|_2$ varying in a prescribed range. This spectral portrait provides such information and can be used in various problems [47], for example:

- To measure the distance to a singular matrix :

$$\min\{\|E\|_2 \text{ such that } A + E \text{ is singular}\}.$$

- For the stability of some problems it is necessary that the eigenvalues have negative real parts. With a spectral portrait, we can measure the quantity

$$\min\{||E||_2 \text{ such that the problem with } A+E \text{ is not stable}\}.$$

- Study the convergence of linear iterative solvers.

Some methods for the estimate of a spectral portrait already exist, but only for small matrices [31, 47]. These methods are based on the Singular Value Decomposition algorithm, see for example [25], but this algorithm cannot be applied to large matrices because of the expense of storage requirements, and computational complexity.

This part of the thesis has been organized as follows : in Chapter 8 we first recall the link between the condition number and the spectral portrait ; then we propose an algorithm for computing the spectral portrait of large matrices [7] based upon the computation of the smallest singular value by a modification of Davidson's method [39], and in Chapter 9 we give some numerical examples.

Chapter 8

Definitions and Algorithms

8.1 Condition number and spectral portrait

8.1.1 Condition number of the eigenvalue problem

The condition number of an eigenvalue consists in a measure of the variation of this eigenvalue due to a matrix perturbation Δ [10].
If the eigenvalue λ of A is not defective, then the error $|\Delta\lambda| = |\lambda' - \lambda|$, where λ' is the nearest by an eigenvalue of $A + \Delta$, can be bounded by $|\Delta\lambda| \leq C_\lambda \, ||\Delta||_2$, where C_λ is the condition number of λ.

8.1.2 Spectral portrait in the complex plane

The spectral portrait of a matrix is the collection of its ϵ-spectra for $\epsilon \in [\epsilon_1, \epsilon_2]$, where the ϵ-spectrum of A, denoted by $\Lambda_\epsilon(A)$, is, for fixed $\epsilon \geq 0$, the union of all the eigenvalues of all the matrices $A + \Delta$ with $||\Delta||_2 \leq \epsilon \, ||A||_2$.

Definition 8.1 *Let $\mu \in \mathbb{C}$, then $\mu \in \Lambda_\epsilon(A)$ if there exists a matrix Δ ($||\Delta||_2 \leq \epsilon \, ||A||_2$) such that $det(A + \Delta - \mu I) = 0$.*

This definition is equivalent to the following :

Definition 8.2 *Let $\mu \in \mathbb{C}$, then $\mu \in \Lambda_\epsilon(A)$ if $\sigma_{min}(A - \mu I) \leq \epsilon \, ||A||_2$ where $\sigma_{min}(A - \mu I)$ is the smallest singular value of $A - \mu I$.*

8.1.3 Relation between condition number and spectral portrait

The spectral portrait can be used for the construction of an estimate for the condition number of the eigenvalue problem. Indeed, for fixed ϵ, $\Lambda_\epsilon(A)$ is the union of patches around clusters of eigenvalues of A. For $\epsilon = 0$, $\Lambda_0(A) = \Lambda(A)$: the patches are reduced to points (the different eigenvalues of A), and there is a value of ϵ for which $\Lambda_\epsilon(A)$ contains the convex hull of the eigenvalues of A ($\epsilon = 2\,\|A\|_2$). So, let λ be a non defective eigenvalue of A, then we can consider some $\epsilon \in \,]\,0, \epsilon_\lambda]$ for which the cluster of eigenvalues of the patch Γ_ϵ around λ is restricted to λ. For these values of ϵ, we can estimate the condition number by :

$$C_\lambda \approx \frac{diam(\Gamma_\epsilon)}{\epsilon\,\|A\|_2} \qquad \text{with} \quad diam(\Gamma_\epsilon) = \max_{\mu_1,\mu_2 \in \Gamma_\epsilon} |\mu_1 - \mu_2|$$

The computation of the spectral portrait consists of the computation of $\epsilon(\mu, A) = \sigma_{min}(A - \mu I)$ for μ taking discrete values in describing a grid of the complex plane.

Remark: $\mu \in \Lambda_\epsilon(A)$ for all $\epsilon \geq \dfrac{\epsilon(\mu, A)}{\|A\|_2}$.

8.2 Computation of $\sigma_{min}(A_\mu)$ where $A_\mu = (A - \mu I)$

The following algorithm is an adaptation of the modified Davidson's algorithm which computes the smallest singular value ν and the associated right singular vector x of a large sparse matrix [39]. In fact, it computes the smallest eigenvalue and the associated eigenvector of the hermitian matrix $A_\mu^H A_\mu$ (where A_μ^H is the conjugate transpose of A_μ).

C_k stands for a $n \times n$ preconditioning matrix whose choice is discussed in [39]. Here, C_k is an approximation of $(A_\mu^H A_\mu)^{-1}$ (in fact, we realize an incomplete *LU* decomposition of A_μ). *MGS* stands for Modified Gram Schmidt Procedure, and m is an integer which limits the size of the basis V_k.

Algorithm 1 :

```
• Choose $m$ and $tol$
• Choose an initial vector $V_1 \in \mathbb{C}^{n\times 1}$, such that $||V_1||_2 = 1$
• for $k = 1, \ldots$ do
    1. Compute the matrix $U_k := A_\mu V_k$
    2. Compute the matrix $W_k := A_\mu^H U_k$
    3. Compute the Rayleigh matrix $H_k := V_k^H W_k$
    4. Compute the smallest eigenpair $(\nu_k^2, y_k)$ of $H_k$
    5. Compute the Ritz vector $x_k := V_k y_k$
    6. Compute the residual $r_k := W_k y_k - \nu_k^2 x_k$
    7. if $||r_k|| \le tol$ then exit
    8. Compute the new direction $t_k := C_k r_k$
    9. if $\dim(V_k) \le m - 1$
          then $V_{k+1} := [V_k, \frac{t_k}{||t_k||_2}]$ where $t_k := (I - V_k V_k^H) t_k$
          else $V_{k+1} := MGS(x_k, t_k)$
       end if
  end for
```

The idea behind this algorithm is to build gradually a dense hermitian matrix $H_k := V_k^H A_\mu^H A_\mu V_k$ (steps 1-3) using projection techniques, then we compute the smallest eigenpair of the projected matrix H_k (step 4). If this eigenpair is a good approximation of the smallest eigenpair of $A_\mu^H A_\mu$ then we stop (step 6). Otherwise, we increase the basis V_k (step 9) by incorporating a new direction (step 8) to the previous subspace. This algorithm is an algorithm with restart, this means that if the size of the basis V_k is greater than a fixed size m, we restart the algorithm with the last Ritz vector x_k and the corresponding direction t_k (step 9).

The starting vector V_1 is chosen randomly and it is unlikely that it contains any singular vector of $A - \mu I$. At convergence, ν_k and x_k approximate the smallest singular value and singular vector.
An important characteristic of Algorithm 1 is that the matrix $A_\mu^H\, A_\mu$ is not required explicitly. We only need two subroutines that compute $A_\mu\, u$ and $A_\mu^H\, v$ for given u and v. At step k, the basis V_{k+1} is obtained from V_k by incorporating the vector $t_k := C_k\, r_k$ after orthonormalization. The subspace spanned by V_k is not a Krylov subspace, and since the matrix C_k is not diagonal, a linear system must be solved at each iteration. The hope is to reach the convergence very quickly with a small value of m, thus rewarding the extra cost involved by this system resolution. A detailed convergence analysis of Algorithm 1 can be found in [14], and a simplified convergence result for the smallest singular value in [39].

8.3 Computation of the spectral portrait

We cannot compute the spectral portrait of a matrix A "everywhere" (more precisely in the disk centered at 0 of radius $||A||_2$), because this computation would be too expensive. So, we assume that we only want to know the spectral portrait in the neighbourhood of a complex value, to check for example whether this value is a good approximation for an eigenvalue of A. So, we define a grid on the complex plane over which we want to obtain the spectral portrait. For this, we give two points of the complex plane, (x_1, y_1) (bottom left point of the grid) and (x_2, y_2) (upper right one), and the number of points in the two directions, nx and ny.

The spectral portrait computation consists merely in the computation of $\epsilon(\mu, A) = \sigma_{min}(A - \mu I)$ for each $\mu = x + iy$ of the grid.

Algorithm 2 :

```
• $\mu = x_1 + i\, y_1, \quad step_x = \dfrac{x_2 - x_1}{nx - 1}, \quad step_y = \dfrac{y_2 - y_1}{ny - 1}$
• for $j = 1, nx$ do
```

```
    1. for $k = 1, ny$ do
       (a) Compute $\sigma_{min}(A - \mu I)$                    (by Algorithm 1)
       (b) $\mu = \mu + i\, step_y$                 (next $\mu$ in the current column)
       end for
    2. $\mu = \mu + step_x$                                     (next column)
    3. $step_y = -step_y$               (we change the direction of going through
       the column)
end for
```

We see that we have to compute $\sigma_{min}(A - \mu I)$ for nearby μ. For that reason, it is interesting that, in Algorithm 1, we compute not only the smallest singular value but also the associated vector. Indeed, in the first step of Algorithm 1, we have to choose an initial basis V which is a vector. Now, during the computation of $\sigma_{min}(A - \mu I)$ for one μ, we increase the basis V by adding at each iteration a new direction, and the last x_k provides convergence. So, if we take it as an initial vector for the computation of $\sigma_{min}(A - \mu' I)$ where μ' is near from μ, then it should improve the convergence since closed matrices have closed singular values and often closed singular vectors.

For this reason, we go through the grid as described in Algorithm 2 (steps 1b, 2, 3). We make sweeps over the grid column by column and alternatively from bottom to top and from top to bottom. This means that we always deal with closed values μ.

Now we define a color map by subdividing the range $[\min\{\epsilon(\mu, A)\}, \max\{\epsilon(\mu, A)\}]$ into intervals of equal length and by assigning a color to each interval. Therefore each point μ of the grid will be assigned to a color according to the value $\epsilon(\mu, A)$. Points of the same color correspond to an ϵ-spectrum.

In fact, it would be very interesting to find an effective method, to compute only the ϵ-spectrum for a given ϵ. Indeed, we can imagine that the user knows the error in the computation of the matrix A, and with this error he wants only to know the corresponding ϵ-spectrum. To realize this, we might follow the level lines [31], but the implementation of this does not seem to be efficient as far as we know.

Chapter 9

Examples of spectral portraits

9.1 Comparison between the algorithm based on SVD and Algorithm 2

We would like to show here the validity of Algorithm 2. We choose as an example a matrix with two ill-conditioned eigenvalues, hence with a large ϵ-spectrum even for small ϵ. The spectral portraits computed on one hand with the SVD algorithm and on the other hand with Algorithm 2, are depicted in Figures 9.1, 9.2 for the following triangular matrix A [22]:

$$A = \begin{pmatrix} -2 & 25 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 10 & 3 & 3 & 3 & 0 \\ 0 & 0 & 2 & 15 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 15 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 25 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3 \end{pmatrix}$$

Clearly, the eigenvalues are $\{-3,-2,0,2,3\}$ where -3 and -2 are eigenvalues of multiplicity 2, which are ill-conditioned as can be seen on the spectral portrait. Even in this difficult case where very small singular values are attained at points of the grid, Figures 9.1 and 9.2 are nearly the same.

The grid used in Figures 9.1, 9.2 and 9.3 is defined by:

$$(x_1, y_1) = (-4, -1), \ \ (x_2, y_2) = (4, 1), \ \ nx = ny = 100$$

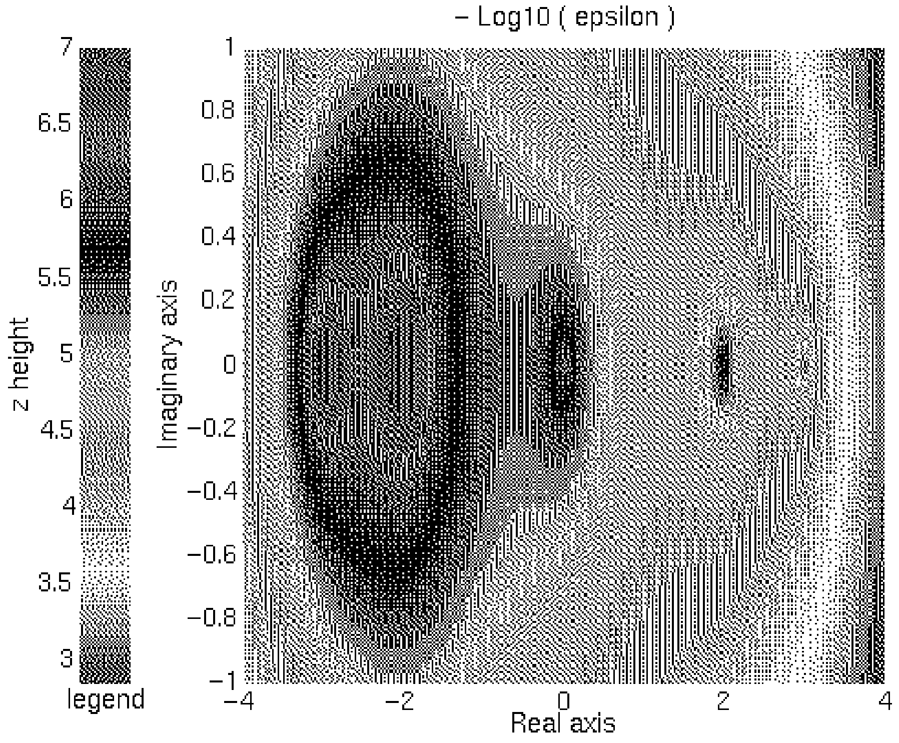


Figure 9.1: using Singular Value Decomposition

In order to estimate the condition number C_λ of an eigenvalue λ of a matrix A with a spectral portrait (see section 8.1.3), we consider the diameter of a patch Γ_ϵ

around λ, and the value of ϵ corresponding to Γ_ϵ. To estimate ϵ, we look at the value corresponding to the color of Γ_ϵ in the legend : this is the value of $-Log10(\epsilon)$ from which we infer the value of ϵ. An estimate of C_λ is then given by the ratio of the diameter of Γ_ϵ and ϵ.

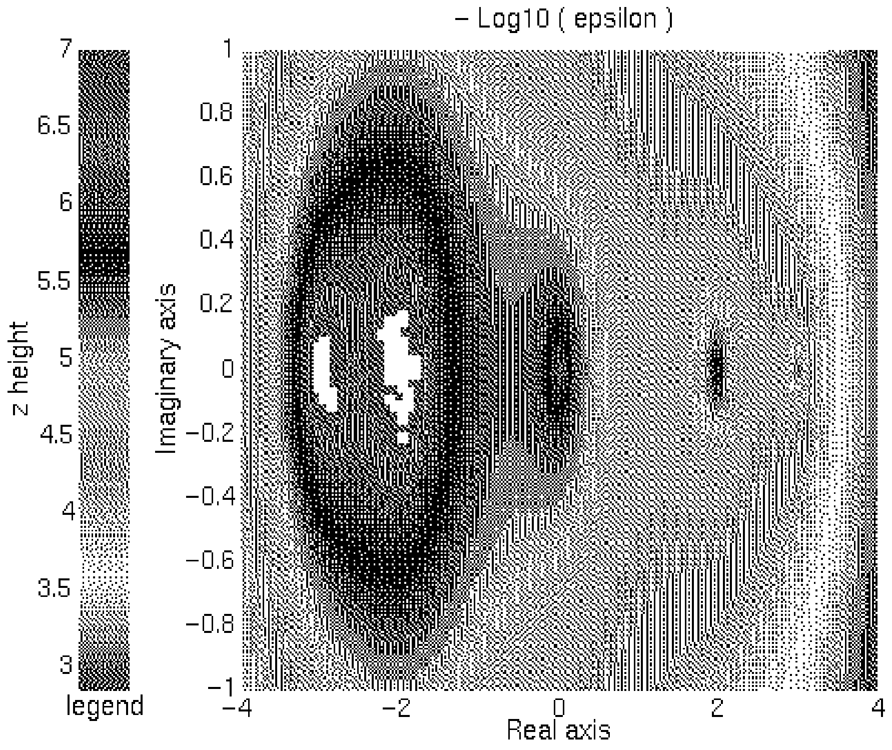


Figure 9.2: using Algorithm 2 with *tol* $= 10^{-3}$ and $m = 2$

As we can see in Figure 9.2, computed with Algorithm 2, some values are missing. These values correspond to very small singular values of the order $10^{-7}, 10^{-8}$, that is to say to small eigenvalues of $A_\mu^H A_\mu$ of the order $10^{-14}, 10^{-16}$. It is well-known that $||\nu_k^2 - \nu^2|| \leq ||r_k||^2$ (see [37] for example). Hence the convergence threshold *tol* must be very small to estimate small eigenvalues.

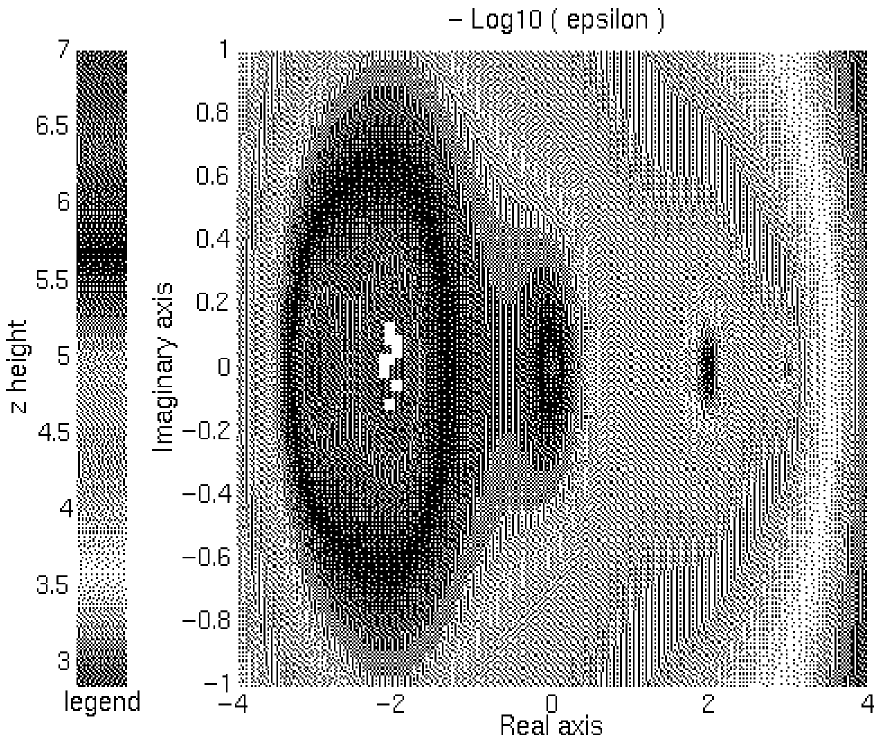


Figure 9.3: using Algorithm 2 with $tol = 10^{-14}$ and $m = 2$

We plot in Figure 9.3 the same spectral portrait as in Figure 9.2 with $tol = 10^{-14}$ instead of 10^{-3} and observe now only a few missing values, as expected. But the computation time is very high. For example, in the previous example, with $tol = 10^{-14}$ the computation time is at least five times greater than for $tol = 10^{-3}$. So, we prefer to keep tol not too small, because we have enough information with this value and the time of computation is reduced.

9.2 Spectral portrait of a few large matrices

We give here examples of spectral portraits for three non hermitian large sparse matrices coming from the Harwell-Boeing set of test matrices [17]. We can deduce from the three pictures that the eigenvalues are well conditioned, since the values of $-Log10(\epsilon)$ are not too large (see legend).

The following picture is the spectral portrait of the matrix HOR131. It arises in a flow network problem. It is a square matrix of order 434 with 4710 nonzero elements. The grid used is defined by:

$$(x_1, y_1) = (-0.5, -0.2),\ (x_2, y_2) = (1, 0.2),\ nx = 1000,\ ny = 10$$

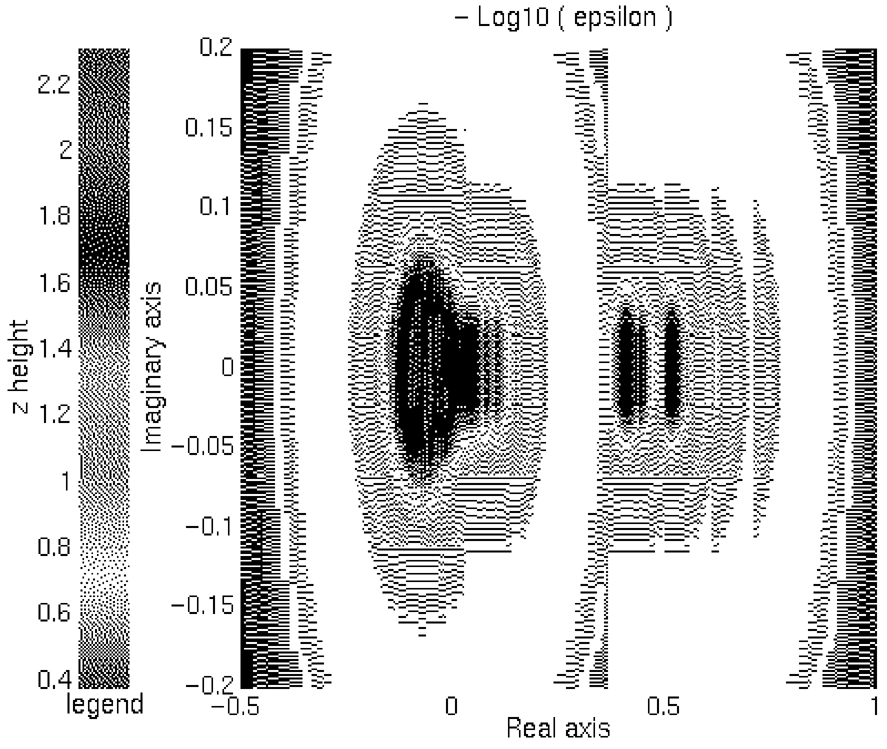


Figure 9.4: Matrix HOR131, $tol = 10^{-3}$, $m = 40$

In Figure 9.4, we see that the eigenvalues are close to each other, but still well conditioned.

The next picture was realized with the matrix PORES3. It arises from reservoir simulation. It is a square matrix of order 532 and has 3474 nonzero elements. The grid used is defined by:

$$(x_1, y_1) = (-6150, -3.5), \ (x_2, y_2) = (-6100, 35), \ nx = 50, \ ny = 100$$

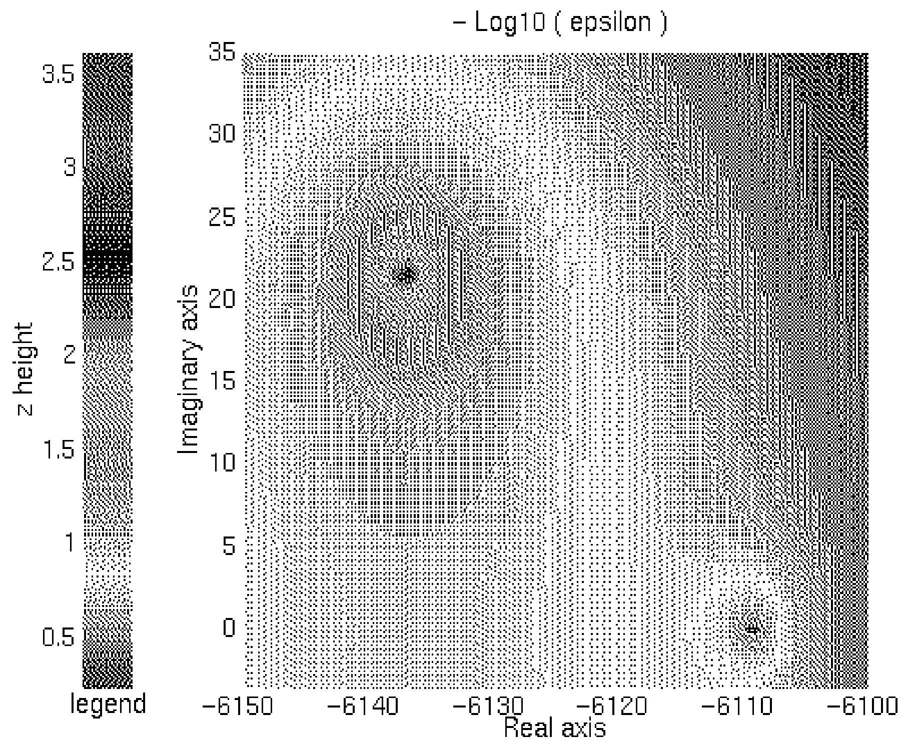


Figure 9.5: Matrix PORES3, $tol = 10^{-4}$, $m = 40$

In Figure 9.5, two eigenvalues are plotted. We clearly see that the eigenvalue in the top left corner is not as well-conditioned as the other one, because the patches around it are bigger.

Chapter 10

Conclusion

We have proposed an algorithm for estimating the spectral portrait of non hermitian large sparse matrices. The algorithm retains the advantage of Davidson's procedure in that the matrix A (resp. A^H) is accessed only in the form of matrix vector products. We would like to conclude with the following remarks :

- The algorithm we proposed can easily be parallelized. It involves sparse matrix-vector products and BLAS primitives.
- The choice of the preconditioner (step 8 of Algorithm 1) is crucial for the success of the method and it can be improved.
- We cannot easily reduce the number of grid points, because we must be sure to capture all the eigenvalues.

Part IV

Stability of the Krylov Bases and Subspaces

Chapter 11

Introduction

When using a computer for the solution of a numerical problem, we have to check the accuracy of the result by computing the condition number of the problem, i.e. to give a measure of the sensitivity of the result to a data perturbation. We deal, in this part, with a method for the study of the condition number of a Krylov basis and a subspace. The Krylov subspaces [32], built with a matrix $A \in \mathbb{R}^{n \times n}$ and an arbitrary vector $f \in \mathbb{R}^n$, are often used in scientific computing with large sparse matrices [2, 43]. In these problems, we need to project the large matrix onto a subspace in order to obtain a smaller matrix which is used to solve the initial problem. We propose a method to measure the sensitivity of the Krylov basis and subspace to a matrix or vector perturbation. This work was done in collaboration with S.K. Godunov and S.V. Kuznetsov [8].

First, in chapter 12, we give the definitions of the distance between two bases and two subspaces, the definition of the Krylov basis and subspace and the condition number of them. At the end of this chapter, we show that we can restrict ourselves to the case of a matrix perturbation on a Hessenberg matrix with the vector $f = (1, 0, \ldots, 0)^T$. In chapter 13, we give the method for the computation of the condition numbers : we study the sensitivity, in first order (of small perturbation), of the Krylov basis F of dimension k constructed from A and f, to a matrix perturbation Δ, i.e. we search X such that $(I + X)F$ is a Krylov basis of $A + \Delta$. Then, we show that X is solution of a Sylvester equation. More precisely, X can be found from the solution of a linear system involving a large triangular matrix $B^{(A,k)}$ constructed from the elements of A. We prove then that the condition number for the computation of the Krylov basis and subspace are deduced from the 2-norm of the inverse of $B^{(A,k)}$. Chapter 15 is devoted to the stability of the algorithm for the computation of the condition number and give some bounds to ensure the quality of the result. Finally, in chapter 16, we illustrate this method by computing condition numbers for several matrices and vectors.

Chapter 12

Definitions and preliminaries

12.1 Distance between two bases and subspaces

Let $\mathcal{F}$ and $\mathcal{G}$ be two subspaces of $\mathbb{R}^n$ of dimension k, and let F and G be two orthonormal bases of $\mathcal{F}$ and $\mathcal{G}$, respectively. Then, there exists some matrix $W \in \mathbb{R}^{n \times n}$, such that $W^*W = I$ and $G = WF$. Let $\mathcal{W}$ be the set of such matrices, then $\forall W \in \mathcal{W}$, there exists some unitary matrices $U \in \mathbb{R}^{n \times n}$ such that

$$W = U^* \begin{pmatrix} \cos\omega_1 & -\sin\omega_1 & & & & & & & \\ \sin\omega_1 & \cos\omega_1 & & & & & & & \\ & & \ddots & & & \mathbf{0} & & & \\ & & & \cos\omega_j & -\sin\omega_j & & & & \\ & & & \sin\omega_j & \cos\omega_j & & & & \\ & & & & & 1 & & & \\ & & \mathbf{0} & & & & \ddots & & \\ & & & & & & & 1 & \\ & & & & & & & & \pm 1 \end{pmatrix} U$$

Definition 12.1

The distance between F and G is given by $d(F,G) = \min_{W \in \mathcal{W}} \sqrt{\sum_{i=1}^{j} \omega_i^2}$ where $\mathcal{W} = \{W \in \mathbb{R}^{n \times n}$ such that $G = WF$ and $W^*W = I\}$.

The distance between $\mathcal{F}$ and $\mathcal{G}$ is given by $d(\mathcal{F},\mathcal{G}) = \min_{F,G} d(F,G)$ where F and G are respectively two orthonormal bases of $\mathcal{F}$ and $\mathcal{G}$.

If F and G are close to each other, then $G = WF$ with $W = I + X + O(\|X\|_F)^2$ where $\|X\|_F \ll 1$ and $X^* = -X$. Let $\mathcal{X}$ be the set of all these matrices X. Then $\forall X \in \mathcal{X}$,

$$X = U^* \begin{pmatrix} 0 & -\omega_1 & & & & \\ \omega_1 & \ddots & \ddots & & \mathbf{0} & \\ & \ddots & \ddots & -\omega_j & & \\ & & \omega_j & 0 & & \\ & \mathbf{0} & & & \ddots & \\ & & & & & 0 \end{pmatrix} U.$$

Since $\|X\|_F = \sqrt{2\sum_i \omega_i^2}$, we get the following:

Lemma 12.1 *$d(F,G) = \min_{X\in\mathcal{X}} \frac{1}{\sqrt{2}}\|X\|_F + O(\|X\|_F)^2$, where*
$\mathcal{X} = \{X$ such that $\|X\|_F \ll 1, X^* = -X$ and $G = (I + X + O(\|X\|_F)^2)\, F\}$

12.2 Krylov subspace and basis

Let $A \in \mathbb{R}^{n\times n}$ and $f \in \mathbb{R}^n$, $\|f\|_2 = 1$. The Krylov subspaces are, for $1 \le k \le n$, the subspaces $\mathcal{K}_k(A,f) = span[f, Af, A^2 f, \ldots, A^{k-1}f]$ of dimension $\le k$.

Let l be the dimension of $\mathcal{K}_n(A,f) = span[f, Af, \ldots, A^{n-1}f]$: in fact, we have $\mathcal{K}_n(A,f) = span[f, Af, \ldots, A^{l-1}f] = \mathcal{K}_l(A,f)$.

Definition 12.2 *For $1 \le k \le l$, the natural orthonormal Krylov basis of $\mathcal{K}_k(A,f)$ is an orthonormal basis $F_k = \{f_1, f_2, \ldots, f_k\}$ such that for $1 \le j \le k$, F_j is the Krylov basis of $\mathcal{K}_j(A,f)$. F_k is unique except for the sign. In particular $f_1 = \pm f$.*

Remark *F_k can be constructed by the Arnoldi process for example [2].*

Let V be an orthonormal basis of $\mathbb{R}^n$ such that $V = (F_l, F')$, where F' is an orthonormal basis of $\mathcal{K}_l^{\perp}(A,f)$, and such that, in this basis, $f = (1,0,\ldots,0)^T$ then A is a Hessenberg matrix, where $a_{l+1,l} = 0$ and $a_{i+1,i} \neq 0$ for $1 \leq i < l$.

Remark *If $l < n-1$ then V is not unique.*

12.3 Condition number of Krylov subspace and basis

We give here the definitions for the condition numbers of the Krylov subspace $\mathcal{K}_k(A,f)$ for $1 \leq k \leq l$ and of its natural orthonormal Krylov basis through a matrix perturbation Δ. These condition numbers are denoted by $\mu\{\mathcal{K}_k(A,f)\}$ and $\mu_b\{\mathcal{K}_k(A,f)\}$, respectively.

Let

- $\mathcal{K} = \mathcal{K}_k(A,f)$, and F its natural orthonormal Krylov basis. Since $k \leq l$, F is of dimension k.
- $\tilde{\mathcal{K}} = \mathcal{K}_k(A+\Delta,f)$, and $\tilde{F}$ its natural orthonormal Krylov basis.

We assume that $||\Delta||$ is small enough to ensure that $\tilde{F}$ is also of dimension k. We apply the usual definition of condition number [49], where the metric on the set of subspaces is defined by Definition 12.1 and we choose the Frobenius norm on the space of matrices.

Definition 12.3

$$\mu\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon>0}\left\{\sup_{||\Delta||_F\leq\epsilon}\left(\frac{d(\mathcal{K},\tilde{\mathcal{K}})}{||\Delta||_F}||A||_F\right)\right\}$$

and

$$\mu_b\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon>0}\left\{\sup_{||\Delta||_F\leq\epsilon}\left(\frac{d(F,\tilde{F})}{||\Delta||_F}||A||_F\right)\right\}$$

Remarks

- *It is trivial to see that $\mu\{\mathcal{K}_1(A,f)\} = \mu_b\{\mathcal{K}_1(A,f)\} = 0$, since $\mathcal{K}_1(A,f) = \{f\}$.*
- *If $l = n$, then $\mu\{\mathcal{K}_n(A,f)\} = 0$ and $\mu_b\{\mathcal{K}_n(A,f)\} = \mu_b\{\mathcal{K}_{n-1}(A,f)\}$ because in this case, the last vector is uniquely defined up to the sign.*

12.4 Simplification of the problem

We saw in section 12.2 that there exists some orthonormal bases V of $\mathbb{R}^n$, such that $H = V^*AV$ is a Hessenberg matrix, and $V^*f = (1, 0, \ldots, 0)^T = e_1$. The following theorem shows that the condition number does not depend on the orthogonal basis in which A and f are expressed.

Theorem 12.1 *For $1 \leq k \leq l$,*

$$\mu\{\mathcal{K}_k(H, e_1)\} = \mu\{\mathcal{K}_k(A, f)\} \quad \textit{and} \quad \mu_b\{\mathcal{K}_k(H, e_1)\} = \mu_b\{\mathcal{K}_k(A, f)\}$$

Lemma 12.2 *Let U be an orthonormal basis of $\mathbb{R}^n$,*

- *Let $X \in \mathbb{R}^{n \times n}$, then $\|U^*XU\|_F = \|X\|_F$.*
- *Let G be an orthonormal basis of $\mathcal{K}_k(A, f)$, then UG is an orthonormal basis of $U\mathcal{K}_k(A, f)$.*

Proof of the theorem

$$\begin{aligned}
\mathcal{K}_k(H, e_1) &= \mathcal{K}_k(V^*AV, V^*f) \\
&= span[V^*f, V^*Af, V^*A^2f, \ldots, V^*A^{k-1}f] \\
&= V^*\mathcal{K}_k(A, f)
\end{aligned}$$

Let F and $\tilde{F}$ be the natural orthonormal basis of $\mathcal{K}_k(A, f)$ and $\mathcal{K}_k(A + \Delta, f)$, then V^*F and $V^*\tilde{F}$ are the natural orthonormal basis of $V^*\mathcal{K}_k(A, f)$ and $V^*\mathcal{K}_k(A + \Delta, f) = \mathcal{K}_k(H + V^*\Delta V, e_1) = \mathcal{K}_k(H + \Delta', e_1)$ where $\|\Delta'\|_F = \|\Delta\|_F$. By construction, we have

$$d(V^*F, V^*\tilde{F}) = \min_{Y \in \mathcal{Y}} \frac{1}{\sqrt{2}}\|Y\|_F, \text{ where}$$

$$\begin{aligned}
\mathcal{Y} &= \left\{Y \text{ s.t. } \|Y\|_F \ll 1, Y^* = -Y \text{ and } V^*\tilde{F} = \left(I + Y + O(\|Y\|_F^2)\right) V^*F\right\} \\
&= \left\{Y \text{ s.t. } \|Y\|_F \ll 1, Y^* = -Y \text{ and } \tilde{F} = \left(I + VYV^* + O(\|Y\|_F^2)\right) F\right\} \\
&= \left\{Y \text{ s.t. } \|Y\|_F \ll 1, Y^* = -Y \text{ and } \tilde{F} = \left(I + Y + O(\|Y\|_F^2)\right) F\right\}
\end{aligned}$$

Hence from Lemma 12.1 $\quad d(V^*F, V^*\tilde{F}) = d(F, \tilde{F}) \quad$ and

$$\mu_b\{\mathcal{K}_k(H, e_1)\} = \mu_b\{\mathcal{K}_k(A, f)\}.$$

$$\begin{aligned}
d(\mathcal{K}_k(H,e_1),\mathcal{K}_k(H+\Delta',e_1)) &= d\left(V^*\mathcal{K}_k(A,f),V^*\mathcal{K}_k(A+\Delta,f)\right) \\
&= \min_{F,\tilde{F}} d(V^*F,V^*\tilde{F}) \\
&= \min_{F,\tilde{F}} d(F,\tilde{F}) \\
&= d(\mathcal{K}_k(A,f),\mathcal{K}_k(A+\Delta,f))
\end{aligned}$$

Therefore $\mu\{\mathcal{K}_k(H,e_1)\} = \mu\{\mathcal{K}_k(A,f)\}$.

□

12.5 Condition number through a vector perturbation

We can relate the condition number for the computation of the Krylov subspace ($\nu\{\mathcal{K}_k(A,f)\}$) and basis ($\nu_b\{\mathcal{K}_k(A,f)\}$) through a vector perturbation to the condition number for the computation of the Krylov subspace ($\mu\{\mathcal{K}_k(A,f)\}$) and basis ($\mu_b\{\mathcal{K}_k(A,f)\}$) through a matrix perturbation.

Theorem 12.2 *Let*

- $\mathcal{K} = \mathcal{K}_k(A,f)$, *and* F *its natural orthonormal Krylov basis.*
- $\tilde{\mathcal{K}} = \mathcal{K}_k(A,f+\delta)$, *and* $\tilde{F}$ *its natural orthonormal Krylov basis.*

Then,

$$\nu_b\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon>0}\left\{\sup_{\|\delta\|_2\le\epsilon}\left(\frac{d(F,\tilde{F})}{\|\delta\|_2}\right)\right\} \le 1+2\sqrt{2}\,\mu_b\{\mathcal{K}_k(A,f)\}$$

and

$$\nu\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon>0}\left\{\sup_{\|\delta\|_2\le\epsilon}\left(\frac{d(\mathcal{K},\tilde{\mathcal{K}})}{\|\delta\|_2}\right)\right\} \le 1+2\sqrt{2}\,\mu\{\mathcal{K}_k(A,f)\}$$

Proof We saw in the previous section that the condition number does not depend on the basis in which A and f are expressed. Therefore, we assume here that $f = (1,0,\ldots,0)^T$. Let δ be a small vector perturbation of f such that $\|f+\delta\|_2 = 1$. We would like to measure $d(F,\tilde{F})$ and $d(\mathcal{K},\tilde{\mathcal{K}})$. There exists some rotation matrices R

such that $R^*R = I$ and $Rf = (f+\delta)$. Let θ be the angle between f and $f+\delta$, then $\|\delta\|_2 = 2\sin\frac{\theta}{2}$. Moreover, there exists some orthonormal bases in which $R = I + S$ where

$$S = \begin{pmatrix} \cos\theta - 1 & -\sin\theta & \\ \sin\theta & \cos\theta - 1 & \\ & & \mathbf{0} \end{pmatrix} = -2\sin\frac{\theta}{2}\begin{pmatrix} \sin\frac{\theta}{2} & -\cos\frac{\theta}{2} & \\ \cos\frac{\theta}{2} & \sin\frac{\theta}{2} & \\ & & \mathbf{0} \end{pmatrix}$$

Then $\|S\|_F = 2\sqrt{2}\sin\frac{\theta}{2} = \sqrt{2}\,\|\delta\|_2 \ll 1$. Hence, $R^* = I - S$ at first order. Moreover,

$$\begin{aligned} \tilde{\mathcal{K}} = \mathcal{K}_k(A, f+\delta) &= span[f+\delta, A(f+\delta), \ldots, A^{k-1}(f+\delta)] \\ &= span[Rf, R(R^*AR)f, \ldots, R(R^*AR)^{k-1}f] \\ &= R\mathcal{K}_k(R^*AR, f) = R\mathcal{K}_k(A+\Delta, f) \end{aligned}$$

where $\Delta = R^*AR - A = AS - SA$ at first order in $\|S\|_F$.

Then $d(\mathcal{K}, \tilde{\mathcal{K}}) = d\left(\mathcal{K}_k(A,f), R\mathcal{K}_k(A+\Delta, f)\right)$. Let G be the natural orthonormal basis of $\mathcal{K}_k(A+\Delta, f)$, then we know that, at first order, $d(F,G) = \dfrac{1}{\sqrt{2}}\|X\|_F$, where X is such that $\|X\| \ll 1, X^* = -X$ and $G = (I+X)F$.

At first order, $\tilde{F} = (I+S)G = (I+S)(I+X)F = (I+S+X)F$ then

$$\begin{aligned} d(F,\tilde{F}) = \frac{1}{\sqrt{2}}\|S+X\|_F &\le \frac{\|S\|_F}{\sqrt{2}} + \frac{\|X\|_F}{\sqrt{2}} = \|\delta\|_2 + d(F,G) \\ &\le \|\delta\|_2 + \mu_b\{\mathcal{K}_k(A,f)\}\frac{\|\Delta\|_F}{\|A\|_F} \end{aligned}$$

We remark that $\|\Delta\|_F \le 2\,\|A\|_F\,\|S\|_F = 2\sqrt{2}\,\|A\|_F\,\|\delta\|_2$, hence

$$d(F,\tilde{F}) \le \left(1 + 2\sqrt{2}\,\mu_b\{\mathcal{K}_k(A,f)\}\right)\ \|\delta\|_2$$

Then, $\nu_b\{\mathcal{K}_k(A,f)\} \le 1 + 2\sqrt{2}\,\mu_b\{\mathcal{K}_k(A,f)\}$.

The same technique applies for $\nu\{\mathcal{K}_k(A,f)\}$. □

Chapter 13

Method to compute the condition numbers of Krylov subspace and basis

We want here to give a method to compute the condition number of the Krylov subspace $\mathcal{K}_k(A,f)$ and of its natural orthonormal basis.
We saw in section 12.4 that we can assume that A is an Hessenberg matrix and that $f=(1,0,\ldots,0)^T$. Let us assume now that $\boxed{2 \le k \le \min(l, n-1)}$

Remark *l is such that $a_{l+1,l}$ is the first zero of the subdiagonal of A, therefore l is the dimension of $\mathcal{K}_n(A,f)$.*

Let $F=[f_1,\ldots,f_k]$ be the natural orthonormal basis of $\mathcal{K}_k(A,f)$ where $f_1 = f = (1,0,\ldots,0)^T$, and let $\tilde{F}=[\tilde{f}_1,\tilde{f}_2,\ldots,\tilde{f}_k]$ be the natural orthonormal basis of $\mathcal{K}_k(\tilde{A},f)$ where $\tilde{A} = A + \Delta + O(\|\Delta\|^2)$ ($\|\Delta\| \ll 1$) and $\tilde{f}_1 = f_1 = (1,0,\ldots,0)^T$. Then, in a basis obtained by completion of F, the matrices of F and $\tilde{F}$ are given by

$$F = \begin{pmatrix} I_k \\ 0 \end{pmatrix} \qquad \tilde{F} = \left(\begin{array}{c|c} 1 & \\ 0 & \\ \vdots & \tilde{f}_2,\ldots,\tilde{f}_k \\ 0 & \end{array}\right)$$

If we find all matrices $X \in \mathbb{R}^{n\times n}$ ($\|X\| \ll 1$) such that $\tilde{F} = (I + X + O(\|\Delta\|^2))F$ with $X^* = -X$, then, due to Lemma 12.1, we will be able to compute $d(F,\tilde{F})$, and then, the condition number of the Krylov subspace and basis (see Definition 12.3).

13.1 Structure of the matrix X

Definition 13.1 *(Definition of the operator $\mathcal{L}_k$)*
Let $M \in \mathbb{R}^{n \times n}$ then $\mathcal{L}_k\{M\}$ denotes the first $k-1$ columns below the subdiagonal of M, i.e.

$$\mathcal{L}_k\{M\} = \begin{pmatrix} 0 & \cdots\cdots\cdots & & & 0 \\ 0 & & & & \\ m_{3,1} & \ddots & & \mathbf{0} & \\ \vdots & \ddots & \ddots & & \\ \vdots & & m_{k+1,k-1} & 0 \ \ldots & 0 \\ \vdots & & \vdots & \vdots & \vdots \\ m_{n,1} & \ldots & m_{n,k-1} & 0 \ \ldots & 0 \end{pmatrix}$$

Remark *$\mathcal{L}_k$ is linear and its kernel is the subspace of Hessenberg matrices for the first $k-1$ columns.*

We have to find X such that $X^* = -X$, $\tilde{f}_1 = f_1$, . Therefore X has the following structure :

$$X = \begin{pmatrix} 0 & 0 & \cdots\cdots & & 0 \\ 0 & 0 & -x_{3,2} & \cdots\cdots & -x_{n,2} \\ \vdots & x_{3,2} & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -x_{n,n-1} \\ 0 & x_{n,2} & \cdots\cdots & x_{n,n-1} & 0 \end{pmatrix}$$

Moreover, X has to be such that $\tilde{F} = (I + X + O(\|\Delta\|^2))F$ is an orthonormal basis of $\mathcal{K}_k(\tilde{A}, f)$, i.e.

$$\left(I - X + O(\|\Delta\|^2)\right)\left(A + \Delta + O(\|\Delta\|^2)\right)\left(I + X + O(\|\Delta\|^2)\right) = \hat{A} + O(\|\Delta\|^2)$$

where $\hat{A}$ is a Hessenberg matrix for the first $k-1$ columns. This previous equation is equivalent to :

$$\mathcal{L}_k\left\{\left(I - X + O(\|\Delta\|^2)\right)\left(A + \Delta + O(\|\Delta\|^2)\right)\left(I + X + O(\|\Delta\|^2)\right)\right\} = O(\|\Delta\|^2)$$

$$\text{that is} \quad \mathcal{L}_k\left\{A + \Delta + AX - XA + O(\|\Delta\|^2)\right\} = O(\|\Delta\|^2)$$

$$\boxed{\iff \quad \mathcal{L}_k\left\{XA - AX\right\} = \mathcal{L}_k\left\{\Delta\right\} + O(\|\Delta\|^2) \qquad (13.1)}$$

In particular, the part under the diagonal of the $n-k$ last columns of X can be arbitrarily chosen. We take $x_{i,j} = 0$ for $j > k$ and $i > k$, yielding :

$$\boxed{X = \begin{pmatrix} 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & -x_{3,2} & \cdots & \cdots & \cdots & -x_{n,2} \\ \vdots & x_{3,2} & \ddots & \ddots & & & \vdots \\ \vdots & \vdots & \ddots & 0 & -x_{k+1,k} & \cdots & -x_{n,k} \\ \vdots & \vdots & & x_{k+1,k} & & & \\ \vdots & \vdots & & \vdots & & \mathbf{0} & \\ 0 & x_{n,2} & \cdots & x_{n,k} & & & \end{pmatrix} \qquad (ST_X)}$$

13.2 Computation of X

For Δ given, we are searching X with the structure (ST_X) and solution of (13.1). Therefore, in first order of $\|\Delta\|$, we are searching X, the solution of :

$$\boxed{\left\{\begin{array}{l} X \text{ has the structure } (ST_X) \text{ defined in section 13.1} \\ \mathcal{L}_k\left\{XA - AX\right\} = \mathcal{L}_k\{\Delta\} \end{array}\right. \qquad (13.2)}$$

Let m be the number of unknown components of X, given by

$$m = (k-1)n + 1 - k(k+1)/2,$$

and introduce the vectors

$$x^{(k)} = (x_{3,2}, \ldots, x_{n,2} \,|\, x_{4,3}, \ldots \,|\, \ldots \,|\, x_{k+1,k}, \ldots, x_{n,k})^T \in \mathbb{R}^m$$

$$\text{and} \quad \delta^{(k)} = (\delta_{3,1}, \ldots, \delta_{n,1} \,|\, \ldots \,|\, \delta_{k+1,k-1}, \ldots, \delta_{n,k-1})^T \in \mathbb{R}^m$$

Then, we can prove (see Chapter 14) that

$$\boxed{X \text{ solution of } (13.2) \quad \text{if and only if } \; x^{(k)} \text{ solution of } \; B^{(A,k)}x^{(k)} = \delta^{(k)}}$$

where $B^{(A,k)} \in \mathbb{R}^{m\times m}$ is a triangular matrix built with the elements of A.

Lemma 13.1 *$B^{(A,k)}$ is non singular because its diagonal elements $(a_{j+1,j},\ j = 1,\dots,k-1)$ are non zero.*

13.3 Condition number of Krylov subspace and basis

Let us write $X = X_k^{(1)} + X_k^{(2)} + X_k^{(3)}$ where

$$X_k^{(1)} = \left(\begin{array}{ccccc|c} 0 & 0 & \dots\dots\dots\dots & & 0 & \\ 0 & 0 & -x_{3,2} & \dots\dots & -x_{k,2} & \\ \vdots & x_{3,2} & 0 & \ddots & \vdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & -x_{k,k-1} & \\ 0 & x_{k,2} & \dots\dots & x_{k,k-1} & 0 & \\ \hline & & \mathbf{0} & & & \mathbf{0} \end{array}\right) \quad \begin{array}{c}\text{is such that } I + X_k^{(1)} \\ \text{is a rotation} \\ \text{in } \mathcal{K}_k(A,f)\end{array}$$

$$X_k^{(2)} = \left(\begin{array}{cccc|ccc} & & & & 0 & \dots & 0 \\ & & & & -x_{k+1,2} & \dots & -x_{n,2} \\ & \mathbf{0} & & & \vdots & & \vdots \\ & & & & -x_{k+1,k} & \dots & -x_{n,k} \\ \hline 0 & x_{k+1,2} & \dots & x_{k+1,k} & & & \\ \vdots & \vdots & & \vdots & & \mathbf{0} & \\ 0 & x_{n,2} & \dots & x_{n,k} & & & \end{array}\right) \quad \text{moves } \mathcal{K}_k(A,f)$$

$X_k^{(3)} = X - X_k^{(1)} - X_k^{(2)}$ is such that $I + X_k^{(3)}$ is a rotation in $\mathcal{K}_k^{\perp}(A,f)$

Remark *In section 13.1, we decided to take $X_k^{(3)} = 0$. Indeed, $I + X_k^{(3)}$ is a rotation in $\mathcal{K}_k^{\perp}(A,f)$, so it does not perturb the computation of $\mathcal{K}_k(A,f)$ and of its*

Krylov basis.

We give here the condition numbers of the Krylov subspace and of its natural orthonormal basis defined in Definition 12.3.

Theorem 13.1 *Let l be the dimension of $\mathcal{K}_n(A,f)$, then for $k \in [2, \min(l, n-1)]$, the matrix $C^{(A,k)} = \left(B^{(A,k)}\right)^{-1}$ exists (see Lemma 13.1), and then :*

The condition number of the natural orthonormal basis of $\mathcal{K}_k(A,f)$ is

$$\mu_b\{\mathcal{K}_k(A,f)\} = \|C^{(A,k)}\|_2 \|A\|_F$$

The condition number of $\mathcal{K}_k(A,f)$ is

$$\mu\{\mathcal{K}_k(A,f)\} \le \|\hat{C}^{(A,k)}\|_2 \|A\|_F$$

where $\hat{C}^{(A,k)}$ is the matrix composed by the rows of $C^{(A,k)}$ such that

$$\hat{x}^{(k)} = (x_{k+1,2}, \ldots, x_{n,2} \mid \ldots \mid x_{k+1,k}, \ldots, x_{n,k})^T = \hat{C}^{(A,k)} \delta^{(k)}$$

with $x^{(k)} = C^{(A,k)} \delta^{(k)}$.

Proof Definition 12.3 gives us :

$$\mu_b\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon > 0} \left\{ \sup_{\|\Delta\|_F \le \epsilon} \left(\frac{d(F,\tilde{F})}{\|\Delta\|_F} \|A\|_F \right) \right\}.$$

We have to remark that if Δ is a Hessenberg matrix for the $k-1$ first columns (i.e. $\mathcal{L}_k\{\Delta\} = 0$) then $d(F,\tilde{F}) = 0$. Therefore

$$\mu_b\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon > 0} \left\{ \sup_{\|\mathcal{L}_k\{\Delta\}\|_F \le \epsilon} \left(\frac{d(F,\tilde{F})}{\|\mathcal{L}_k\{\Delta\}\|_F} \|A\|_F \right) \right\}.$$

Then, using Lemma 12.1 and the equality $\|\delta^{(k)}\|_2 = \|\mathcal{L}_k\{\Delta\}\|_F$, we find

$$\begin{aligned} \mu_b\{\mathcal{K}_k(A,f)\} &= \inf_{\epsilon > 0} \left\{ \sup_{\|\delta^{(k)}\|_2 \le \epsilon} \left(\frac{\frac{1}{\sqrt{2}} \|X_k^{(1)} + X_k^{(2)}\|_F}{\|\delta^{(k)}\|_2} \|A\|_F \right) \right\} \\ &= \inf_{\epsilon > 0} \left\{ \sup_{\|\delta^{(k)}\|_2 \le \epsilon} \frac{\|x^{(k)}\|_2}{\|\delta^{(k)}\|_2} \right\} \|A\|_F = \|C^{(A,k)}\|_2 \|A\|_F. \end{aligned}$$

Thus, we find $\mu\{\mathcal{K}_k(A,f)\} = \inf_{\epsilon>0}\left\{\sup_{\|\Delta\|_F\le\epsilon}\left(\frac{d(\mathcal{K},\tilde{\mathcal{K}})}{\|\Delta\|_F}\|A\|_F\right)\right\}$

$$= \inf_{\epsilon>0}\left\{\sup_{\|\mathcal{L}_k\{\Delta\}\|_F\le\epsilon}\left(\frac{d(\mathcal{K},\tilde{\mathcal{K}})}{\|\mathcal{L}_k\{\Delta\}\|_F}\|A\|_F\right)\right\}.$$

But $d(\mathcal{K},\tilde{\mathcal{K}}) = \min_{F,\tilde{F}} d(F,\tilde{F}) = \min_{X} \frac{1}{\sqrt{2}}\|X_k^{(1)} + X_k^{(2)}\|_F$, then

$$\begin{aligned}\mu\{\mathcal{K}_k(A,f)\} &\le \inf_{\epsilon>0}\left\{\sup_{\|\delta^{(k)}\|_2\le\epsilon}\left(\frac{\frac{1}{\sqrt{2}}\|X_k^{(2)}\|_F}{\|\delta^{(k)}\|_2}\|A\|_F\right)\right\}\\ &= \inf_{\epsilon>0}\left\{\sup_{\|\delta^{(k)}\|_2\le\epsilon}\frac{\|\hat{x}^{(k)}\|_2}{\|\delta^{(k)}\|_2}\right\}\|A\|_F = \|\hat{C}^{(A,k)}\|_2\|A\|_F\end{aligned}$$

where $\hat{x}^{(k)}$ and $\hat{C}^{(A,k)}$ are defined in the theorem. □

Chapter 14

The matrix $B^{(A,k)}$

Let $A, \Delta, \delta^{(k)}$ and the operator $\mathcal{L}_k$ be defined as in chapter 13. We are going to prove that X solution of (13.2) is equivalent to $x^{(k)}$ solution of $B^{(A,k)}x^{(k)} = \delta^{(k)}$ where $B^{(A,k)} \in \mathbb{R}^{m\times m}$ is a non singular triangular matrix built with the elements of A.

X has the structure (ST_X) defined in section 13.1, therefore we can write $X = X_L + X_U$ where

$$X_L = \begin{pmatrix} 0 & & & & \\ 0 & 0 & & & \\ \vdots & x_{3,2} & 0 & \mathbf{0} & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & x_{n,2} & .. & x_{n,n-1} & 0 \end{pmatrix}, X_U = \begin{pmatrix} 0 & 0 & \dots\dots\dots & 0 \\ & 0 & -x_{3,2} \quad .. & -x_{n,2} \\ & & 0 \quad \ddots & \vdots \\ & \mathbf{0} & \ddots & -x_{n,n-1} \\ & & & 0 \end{pmatrix}.$$

As A is a Hessenberg matrix, we see that $X_U A - AX_U$ is an upper triangular matrix, then $\mathcal{L}_k\{X_U A - AX_U\} = 0$, and therefore

$$\boxed{\mathcal{L}_k\{XA - AX\} = \mathcal{L}_k\{\Delta\} \quad \text{is equivalent to} \quad \mathcal{L}_k\{X_L A - AX_L\} = \mathcal{L}_k\{\Delta\}}$$

which writes

$$\sum_{l=1}^{j+1} x_{i,l}a_{l,j} - \sum_{l=i-1}^{n} a_{i,l}x_{l,j} = \delta_{i,j} \qquad \forall\ 1 \le j \le k-1 \text{ and } j+2 \le i \le n$$

But $\forall i,\ x_{i,1} = 0$, so $\quad \forall\, 1 \leq j \leq k-1$ and $j+2 \leq i \leq n, \quad$ we have :

$$\left(\sum_{l=2}^{j-1} a_{l,j} x_{i,l}\right) + \left(a_{j,j} x_{i,j} - \sum_{l=i-1}^{n} a_{i,l} x_{l,j}\right) + (a_{j+1,j} x_{i,j+1}) = \delta_{i,j}$$

equivalent to $\forall j \in [1, k-1], \qquad \sum_{l=2}^{j-1} a_{l,j} \left(\mathbf{0}_{j-l+1} \,\middle|\, \mathbf{I}_{n-j-1} \right) \begin{pmatrix} x_{l+1,l} \\ \vdots \\ x_{n,l} \end{pmatrix}$

$$+ \left(a_{j,j} \left(\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \,\middle|\, \mathbf{I}_{n-j-1} \right) - \begin{pmatrix} a_{j+2,j+1} & \cdots\cdots\cdots & a_{j+2,n} \\ & \ddots & \vdots \\ \mathbf{0} & a_{n,n-1} & a_{n,n} \end{pmatrix} \right) \begin{pmatrix} x_{j+1,j} \\ \vdots \\ x_{n,j} \end{pmatrix}$$

$$+ a_{j+1,j} \mathbf{I}_{n-j-1} \begin{pmatrix} x_{j+2,j+1} \\ \vdots \\ x_{n,j+1} \end{pmatrix} = \begin{pmatrix} \delta_{j+2,j} \\ \vdots \\ \delta_{n,j} \end{pmatrix}$$

This is equivalent to

$$\sum_{l=2}^{j-1} a_{l,j} J_{n-j-1}^{n-l} \begin{pmatrix} x_{l+1,l} \\ \vdots \\ x_{n,l} \end{pmatrix} + \left(a_{j,j} J_{n-j-1}^{n-j} - A^{(j+2)} \right) \begin{pmatrix} x_{j+1,j} \\ \vdots \\ x_{n,j} \end{pmatrix}$$

$$+ a_{j+1,j} \mathbf{I}_{n-j-1} \begin{pmatrix} x_{j+2,j+1} \\ \vdots \\ x_{n,j+1} \end{pmatrix} = \begin{pmatrix} \delta_{j+2,j} \\ \vdots \\ \delta_{n,j} \end{pmatrix}, \qquad \forall j \in [1, k-1]$$

where $J_i^j = \left(\mathbf{0}_{j-i} \,\middle|\, \mathbf{I}_i \right) \in \mathbb{R}^{i \times j}$

$$\text{and } A^{(i)} = \begin{pmatrix} a_{i,i-1} & \cdots\cdots\cdots & a_{i,n} \\ & \ddots & \vdots \\ \mathbf{0} & a_{n,n-1} & a_{n,n} \end{pmatrix} \in \mathbb{R}^{(n-i+1) \times (n-i+2)}$$

This last system of equations is equivalent to $B^{(A,k)}x^{(k)} = \delta^{(k)}$ where $B^{(A,k)}$ is the following triangular matrix built with the elements of A :

$$\begin{pmatrix}
a_{2,1}I_{\text{n-2}} & & & & \\
a_{2,2}J_{\text{n-3}}^{\text{n-2}} - A^{(4)} & a_{3,2}I_{\text{n-3}} & & & \\
a_{2,3}J_{\text{n-4}}^{\text{n-2}} & \ddots & \ddots & \mathbf{0} & \\
\vdots & \ddots & \ddots & \ddots & \\
a_{2,\text{k-1}}J_{\text{n-k}}^{\text{n-2}} & \dots & a_{\text{k-2,k-1}}J_{\text{n-k}}^{\text{n-k+2}} & a_{\text{k-1,k-1}}J_{\text{n-k}}^{\text{n-k+1}} - A^{(\text{k+1})} & a_{\text{k,k-1}}I_{\text{n-k}}
\end{pmatrix}$$

Chapter 15

The algorithm and its stability

Let $A \in \mathbb{R}^{n \times n}$ and $f \in \mathbb{R}^n$. We propose here an algorithm to compute the condition numbers of the Krylov subspaces $\mathcal{K}_k(A, f)$ and of their natural orthonormal bases, for k from 2 to $n-1$.

This algorithm consists in the construction of a large triangular matrix $B^{(A,n-1)}$, for which we have to estimate the norm of its inverse. We want to compute the norm precisely in order to bound the result. Therefore, we compute the inverse by using a special technique for scalar products. This technique enables us in most cases to avoid underflow and overflow.

In the second part, we study the stability of the algorithm in order to provide lower and upper bounds for the computed condition numbers.

15.1 Algorithms

First, we give the algorithm for computation of the condition numbers (Algorithm 1) and then we give in detail the algorithm for computation of the inverse matrix (Algorithm 2).

15.1.1 Algorithm 1 : computation of the condition numbers

1. Compute the Householder reflections $P_0, P_1, \ldots, P_{n-2}$ such that $P_0 f = (1, 0, \ldots, 0)^T$, and $P^T AP$ is a Hessenberg matrix, where $P = P_0 P_1 \ldots P_{n-2}$.

   Let $A := P^T AP$ *and* $f := (1, 0, \ldots, 0)^T$.

2. Compute the lower triangular matrix $B^{(A,n-1)}$ (Chapter 14).

3. Compute the matrix $C^{(A,l)} = \left(B^{(A,l)}\right)^{-1}$ by Algorithm 2.

4. For $k = 2 \ldots l$, give the condition numbers (Section 13.3) :

   - For the basis : $\|C^{(A,k)}\|_2$,
   - For the subspace $\mathcal{K}_k(A, f)$: $\|\hat{C}^{(A,k)}\|_2$, where $\hat{C}^{(A,k)}$ is defined in section 13.3.

15.1.2 Computation of the inverse

Let $B = (b_{i,j})_{i,j=1}^m = B^{(A,n-1)}$, and $C = (c_{i,j})_{i,j=1}^m = C^{(A,n-1)}$. Classically, the $c_{i,j}$ are computed as follows :

```
for  $i = 1, m$ do
    1. $c_{i,i} = \frac{1}{b_{i,i}}$
    2. for  $j = 1, i-1$ do
           $c_{i,j} = \frac{-1}{b_{i,i}} \left( \sum_{k=j}^{i-1} b_{i,k} c_{k,j} \right)$
       end for
end for
```

We have to take into account that the diagonal of the matrix B may contain elements which are very close to 0 : if there exists an i_0 such that $b_{i_0,i_0} \approx 0$ then the computation of $c_{i_0,j}$ may fail if $\sum_{k=j}^{i_0-1} b_{i_0,k} c_{k,j}$ is very small too (underflow). On the other hand, if $\sum_{k=j}^{i_0-1} b_{i_0,k}c_{k,j}$ is not small, then $c_{i_0,j}$ is very big ($b_{i_0,i_0} \approx 0$). Therefore, for $i_1 > i_0$, $\sum_{k=j}^{i_1-1} b_{i_1,k}c_{k,j}$ can be very big (overflow). The following alternative method for the computation of the $c_{i,j}$ has been designed to avoid such problems.

Algorithm 2

```
for  $i = 1, m$  do

  1. $c_{i,i} = 2^{-r} \dfrac{1}{\frac{b_{i,i}}{2^r}}$

  2. for  $j = 1, i-1$  do
```

$$c_{i,j} = -2^{(p+q-r)} \frac{\sum_{k=j}^{i-1} \frac{b_{i,k}}{2^p} \frac{c_{k,j}}{2^q}}{\frac{b_{i,i}}{2^r}} \tag{15.1}$$

```
         where  $r = 1 + \lfloor \log_2 |b_{i,i}| \rfloor$,  $p = 1 + \max_{j \le k \le i-1} \lfloor \log_2 |b_{i,k}| \rfloor$
         and  $q = 1 + \max_{j \le k \le i-1} \lfloor \log_2 |c_{k,j}| \rfloor$.
     end for

end for
```

15.2 Stability

Since floating-point computation is not exact, we have to bound the error in the computation of the matrix $C^{(A,k)} = (B^{(A,k)})^{-1}$ (see the following theorem) in order to ensure the validity of the result (norm of the inverse matrix). Because of Theorem 15.1, we can bound (see Remark 15.1) the exact norm of the inverse matrix by using the norm of the computed inverse.

Let $B = (b_{i,j})_{i,j=1}^m = B^{(A,k)}$, $C = (c_{i,j})_{i,j=1}^m = C^{(A,k)}$ and C_{mach} the inverse computed with Algorithm 2.

Theorem 15.1 *Suppose that the computed operations satisfy :*

$$\left\{\begin{array}{ccccc} (a+b)_{mach} & = & a \oplus b & = & (a+b)(1+\psi)+\xi \\ (a*b)_{mach} & = & a \otimes b & = & (a*b)(1+\eta)+\zeta \\ (a/b)_{mach} & = & a \oslash b & = & (a/b)(1+\phi)+\chi \end{array}\right.$$

where $|\psi|, |\eta|, |\phi| \leq \epsilon_1$ *(computer precision) and* $|\xi|, |\zeta|, |\chi| \leq \epsilon_0$ *(smallest number in the machine).*
If $m\epsilon_1 < 1$ *and* $\delta = \dfrac{(m+1)\epsilon_1}{1-m\epsilon_1}$ *then*

$$BC_{mach} = I + \Omega \quad where \quad ||\Omega||_F \leq 3\delta ||B||_F ||C_{mach}||_F.$$

Remark 15.1 *If* $3\delta ||B||_F ||C_{mach}||_F < 1$ *then*

$$||C_{mach}||_2 \frac{1-2||\Omega||_F}{1-||\Omega||_F} \leq ||B^{-1}||_2 \leq ||C_{mach}||_2 \frac{1}{1-||\Omega||_F}$$

Proof of the remark
We assume that $3\delta||B||_F||C_{mach}||_F < 1$ then $||\Omega||_2 \leq ||\Omega||_F < 1$. We deduce that $(I+\Omega)^{-1}$ exists and then $B^{-1} = C_{mach}(I+\Omega)^{-1}$.

So, $||B^{-1} - C_{mach}||_2 \quad \leq \quad ||C_{mach}||_2 \dfrac{||\Omega||_2}{1-||\Omega||_2} \quad \leq \quad ||C_{mach}||_2 \dfrac{||\Omega||_F}{1-||\Omega||_F}$

but $||C_{mach}||_2 - ||B^{-1} - C_{mach}||_2 \quad \leq \quad ||B^{-1}||_2 \quad \leq \quad ||B^{-1} - C_{mach}||_2 + ||C_{mach}||_2.$

$$\text{Finally,} \quad ||C_{mach}||_2 \frac{1-2||\Omega||_F}{1-||\Omega||_F} \leq ||B^{-1}||_2 \leq ||C_{mach}||_2 \frac{1}{1-||\Omega||_F}$$

□

Proof of the theorem
The proof consists in two parts.

1. First, we will prove that the $c_{i,j}$ computed with Algorithm 2 are such that

$$(c_{i,i})_{mach} = 2^{-r}\left(\frac{1+\phi}{\tilde{b}_{i,i}} + \chi\right)$$

and, assuming that $m\epsilon_1 < 1$,

$$(c_{i,j})_{mach} = -2^{(p+q-r)} \left(\sum_{\substack{k=j \\ k \neq k_0}}^{i-1} \frac{\tilde{b}_{i,k}\tilde{c}_{k,j}}{\tilde{b}_{i,i}} (1+\delta_k) \quad + \quad \frac{\tilde{c}_{k_0,j}}{\tilde{b}_{i,i}} (\tilde{b}_{i,k_0} + \delta_{k_0}) \right)$$

where $\forall j \leq k < i, \quad |\delta_k| \leq \delta = \dfrac{(m+1)\epsilon_1}{1-m\epsilon_1}$

2. Finally, we will prove that these computed $c_{i,j}$ are the solution of the equation $BC_{mach} = I + \Omega$ where $\|\Omega\|_F \leq 3\delta \|B\|_F \|C_{mach}\|_F$.

Part I : In fact, we compute $\quad (c_{i,i})_{mach} = 2^{-r} \left(\dfrac{1}{\tilde{b}_{i,i}} \right)_{mach}$ and

$$(c_{i,j})_{mach} = -2^{(p+q-r)} \left(\frac{\left(\sum_{k=j}^{i-1} \tilde{b}_{i,k}\tilde{c}_{k,j} \right)_{mach}}{\tilde{b}_{i,i}} \right)_{mach} \tag{15.2}$$

where $\quad \tilde{b}_{i,k} = \dfrac{b_{i,k}}{2^p}, \quad \tilde{c}_{k,j} = \dfrac{(c_{k,j})_{mach}}{2^q}$ and $\tilde{b}_{i,i} = \dfrac{b_{i,i}}{2^r}, \quad (p,q,r)$ defined as in (15.1).

It is easy to show that $\boxed{(c_{i,i})_{mach} = 2^{-r}(1 \oslash \tilde{b}_{i,i}) = 2^{-r} \left(\dfrac{1+\phi}{\tilde{b}_{i,i}} + \chi \right)}$

We consider now $(c_{i,j})_{mach}$. First, we assume that

$$\begin{cases} \tilde{b}_{i,k} \otimes \tilde{c}_{k,j} = \tilde{b}_{i,k}\tilde{c}_{k,j}(1+\eta_k) + \zeta_k \\ \tilde{b}_{i,k} \otimes \tilde{c}_{k,j} \; \oplus \; \tilde{b}_{i,k+1} \otimes \tilde{c}_{k+1,j} = \left(\tilde{b}_{i,k} \otimes \tilde{c}_{k,j} \; + \; \tilde{b}_{i,k+1} \otimes \tilde{c}_{k+1,j} \right) (1+\psi_k) + \xi_k \end{cases}$$

where $\quad |\eta_k|, |\psi_k| \leq \epsilon_1 \quad$ and $\quad |\zeta_k|, |\xi_k| \leq \epsilon_0. \quad$ Then

$$\begin{aligned} \left(\sum_{k=j}^{i-1} \tilde{b}_{i,k}\tilde{c}_{k,j} \right)_{mach} &= \tilde{b}_{i,j} \otimes \tilde{c}_{j,j} \; \oplus \; \tilde{b}_{i,j+1} \otimes \tilde{c}_{j+1,j} \; \oplus \ldots \oplus \; \tilde{b}_{i,i-1} \otimes \tilde{c}_{i-1,j} \\ &= \sum_{k=j}^{i-1} \left\{ \left(\tilde{b}_{i,k}\tilde{c}_{k,j}(1+\eta_k) + \zeta_k \right) \prod_{l=k-1}^{i-2} (1+\psi_l) + \xi_k \prod_{l=k+1}^{i-2} (1+\psi_l) \right\} \end{aligned}$$

where we assume that $\xi_{i-1} = \psi_{j-1} = 0$.

Let us assume also that

$$\left(\sum_{k=j}^{i-1} \tilde{b}_{i,k}\tilde{c}_{k,j}\right)_{mach} \oslash \tilde{b}_{i,i} = \left(\sum_{k=j}^{i-1} \tilde{b}_{i,k}\tilde{c}_{k,j}\right)_{mach} \left(\frac{1+\phi_i}{\tilde{b}_{i,i}}\right) + \chi_i,$$

Equation (15.2) gives :

$$(c_{i,j})_{mach} = -2^{(p+q-r)} \left\{ \sum_{k=j}^{i-1} \tilde{b}_{i,k}\tilde{c}_{k,j}(1+\eta_k) \prod_{l=k-1}^{i-2} (1+\psi_l) \left(\frac{1+\phi_i}{\tilde{b}_{i,i}}\right) \right.$$
$$\left. +\chi_i + \sum_{k=j}^{i-1} \left(\zeta_k \prod_{l=k-1}^{i-2} (1+\psi_l) + \xi_k \prod_{l=k+1}^{i-2} (1+\psi_l) \right) \left(\frac{1+\phi_i}{\tilde{b}_{i,i}}\right) \right\}.$$

Let k_0 be such that $q = 1 + \lfloor \log_2 |c_{k_0,j}| \rfloor$, then $c_{k_0,j} = \tilde{c}_{k_0,j} 2^q$ where $0.5 \leq |\tilde{c}_{k_0,j}| < 1$ and we find that :

$$\boxed{(c_{i,j})_{mach} = -2^{(p+q-r)} \left(\sum_{\substack{k=j \\ k\neq k_0}}^{i-1} \frac{\tilde{b}_{i,k}\tilde{c}_{k,j}}{\tilde{b}_{i,i}} (1+\delta_k) \quad + \quad \frac{\tilde{c}_{k_0,j}}{\tilde{b}_{i,i}} (\tilde{b}_{i,k_0} + \delta_{k_0}) \right),}$$

where $\delta_k = (1+\phi_i)(1+\eta_k) \prod_{l=k-1}^{i-2} (1+\psi_l) - 1$,

and $\delta_{k_0} = \tilde{b}_{i,k_0} \left((1+\eta_{k_0})(1+\phi_i) \prod_{l=k_0-1}^{i-2} (1+\psi_l) - 1 \right)$

$$+\frac{\chi_i \tilde{b}_{i,i}}{\tilde{c}_{k_0,j}} + \frac{1+\phi_i}{\tilde{c}_{k_0,j}} \sum_{k=j}^{i-1} \left(\zeta_k \prod_{l=k-1}^{i-2} (1+\psi_l) + \xi_k \prod_{l=k+1}^{i-2} (1+\psi_l) \right)$$

with $\quad 0.5 \le |\tilde{c}_{k_0,j}|, |\tilde{b}_{i,i}| < 1, \quad \xi_{i-1} = 0 \quad$ and $\quad \psi_{j-1} = 0.$

We will now prove that if $m\epsilon_1 < 1$, then $\forall j \le k < i \;:\; |\delta_k| \le \delta = \dfrac{(m+1)\epsilon_1}{1 - m\epsilon_1}$

First, we note that if $N\rho < 1$ and $|\rho_i| \le \rho$, $\forall i \in [1, N]$, then

$$\left|\prod_{i=1}^{N}(1+\rho_i) - 1\right| \le \frac{N\rho}{1-N\rho} \quad \text{and} \quad \left|\prod_{i=1}^{N}(1+\rho_i)\right| \le \frac{1}{1-N\rho}$$

1. $k \ne k_0$, then

   $|\phi_i|, |\eta_k|, |\psi_l| \le \epsilon_1$ implies that $\quad |\delta_k| \le \dfrac{(i-j+1)\epsilon_1}{1-(i-j+1)\epsilon_1} \le \dfrac{m\epsilon_1}{1-m\epsilon_1}$

2. $k = k_0$, then $\quad |\tilde{b}_{i,i}|, |\tilde{b}_{i,k_0}| < 1, \quad \left|\dfrac{1}{\tilde{c}_{k_0,j}}\right| \le 2, \quad |\chi_i|, |\zeta_k|, |\xi_k| \le \epsilon_0,$

   implies that $\quad |\delta_{k_0}| \le \dfrac{(i-j+1)\epsilon_1}{1-(i-j+1)\epsilon_1} + 2\epsilon_0 + 2(1+\epsilon_1)\displaystyle\sum_{k=j}^{i-1} \dfrac{2\epsilon_0}{1-(i-k)\epsilon_1}$

   implies that $\quad |\delta_{k_0}| \le \dfrac{m\epsilon_1 + 2\epsilon_0(1+2m+m\epsilon_1)}{1-m\epsilon_1} \le \delta$

Part II : Let $\Omega = (\omega_{i.j})_{i,j=1}^{m}$, then

$$\begin{cases} |\omega_{i,i}| = |b_{i,i}(c_{i,i})_{mach} - 1| < \epsilon_1 + \epsilon_0 \le \delta \\[2ex] |\omega_{i,j}| = \left|\displaystyle\sum_{k=j}^{i} b_{i,k}(c_{k,j})_{mach}\right| \le 3\delta \|b_{i,*}\|_2 \|(c_{*,j})_{mach}\|_2, \qquad \text{for } j < i \end{cases}$$

Indeed, $(c_{i,i})_{mach} = \dfrac{2^{-r}}{\tilde{b}_{i,i}}\left(1 + \phi + \tilde{b}_{i,i}\chi\right)$ is equivalent to $b_{i,i}(c_{i,i})_{mach} - 1 = \phi + \tilde{b}_{i,i}\chi$

but $|\phi| \le \epsilon_1$, $|\chi| \le \epsilon_0$ and $|\tilde{b}_{i,i}| < 1$ (by construction) so $|\omega_{i,i}| = |\phi + \tilde{b}_{i,i}\chi| < \epsilon_1 + \epsilon_0$

For $j < i$, $\quad (c_{i,j})_{mach} = -2^{(p+q-r)} \left(\sum_{\substack{k=j \\ k \neq k_0}}^{i-1} \frac{\tilde{b}_{i,k}\tilde{c}_{k,j}}{\tilde{b}_{i,i}} (1+\delta_k) + \frac{\tilde{c}_{k_0,j}}{\tilde{b}_{i,i}} (\tilde{b}_{i,k_0} + \delta_{k_0}) \right)$

is equivalent to

$$b_{i,i}(c_{i,j})_{mach} = - \left(\sum_{\substack{k=j \\ k \neq k_0}}^{i-1} b_{i,k}(c_{k,j})_{mach}(1+\delta_k) + (c_{k_0,j})_{mach}(b_{i,k_0} + 2^p \delta_{k_0}) \right)$$

equivalent to $\quad \sum_{k=j}^{i} b_{i,k}(c_{k,j})_{mach} = - \left(\sum_{\substack{k=j \\ k \neq k_0}}^{i-1} b_{i,k}(c_{k,j})_{mach}\delta_k + (c_{k_0,j})_{mach} 2^p \delta_{k_0} \right).$

Therefore, $\quad |\omega_{i,j}| \leq \delta \left(\sum_{\substack{k=j \\ k \neq k_0}}^{i-1} |b_{i,k}||(c_{k,j})_{mach}| + |(c_{k_0,j})_{mach}| 2 \max_{j \leq k < i} |b_{i,k}| \right)$

then $\quad |\omega_{i,j}| \leq 3\delta \sqrt{\sum_{k=1}^{m} b_{i,k}^2} \sqrt{\sum_{k=1}^{m} (c_{k,j})_{mach}^2} = 3\delta ||b_{i,*}||_2 ||(c_{*,j})_{mach}||_2.$

We conclude that

$$||\Omega||_F = \sqrt{\sum_{i,j=1}^{m} \omega_{i,j}^2} = \sqrt{\sum_{i=1}^{m} \omega_{i,i}^2 + \sum_{i=1}^{m} \sum_{j=1}^{i-1} \omega_{i,j}^2} \leq \sqrt{\sum_{i,j=1}^{m} 9\delta^2 ||b_{i,*}||_2^2 ||(c_{*,j})_{mach}||_2^2}$$

then $\quad ||\Omega||_F \leq 3\delta \sqrt{\sum_{i=1}^{m} ||b_{i,*}||_2^2 \sum_{j=1}^{m} ||(c_{*,j})_{mach}||_2^2} = 3\delta ||B||_F ||C_{mach}||_F.$

□

Chapter 16

Examples

We give here some examples of the computation of the condition numbers of Krylov bases and subspaces. For each example, we made a picture showing the values of the condition numbers for increasing sizes of the Krylov bases and subspaces. We also give a table in which we list the lower and upper bounds for the condition numbers of the basis (M is an upper bound of $||\Omega||_F$ computed with result in Theorem 15.1) :

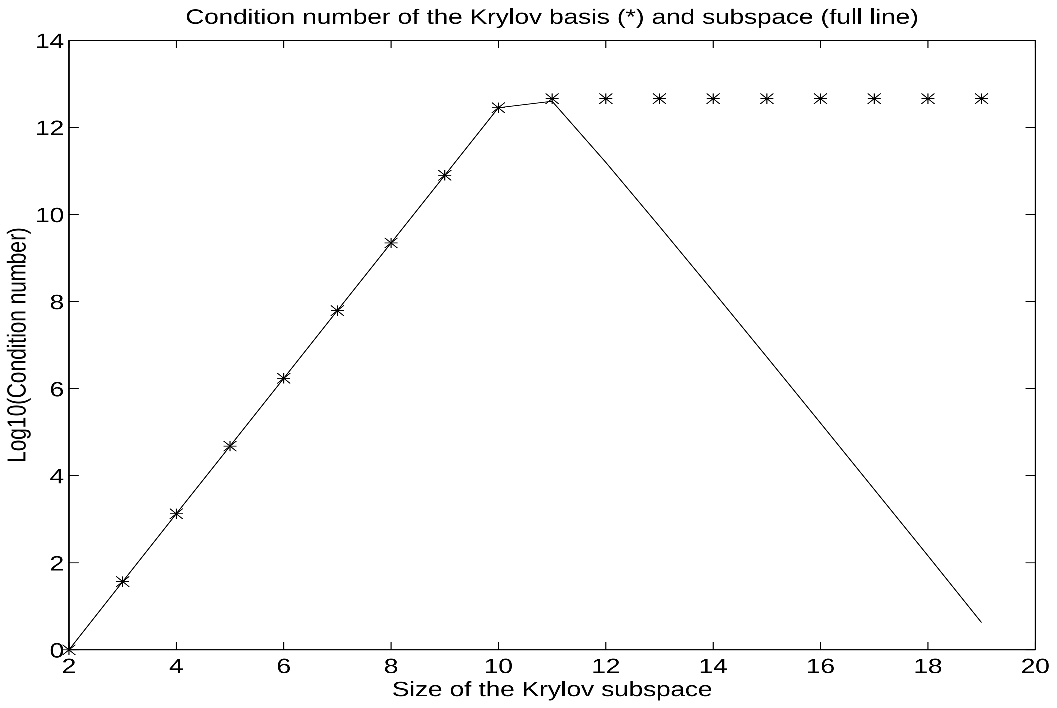
$$\text{Lower bound} = ||C_{mach}||_2 \frac{1-2M}{1-M} ||A||_F$$
$$\text{and Upper bound} = ||C_{mach}||_2 \frac{1}{1-M} ||A||_F$$

Indeed, let $B = B^{(A,k)}$ and $C = C^{(A,k)}$, then we have to be sure that the computed matrix C_{mach} is not too far from B^{-1}. Therefore, we computed these bounds which allow us to validate the result when they are enough close from each other. These bounds can only be safely computed if $M < 0.5$. If not, we put "??" in the table. Moreover, we compute the quantity $||\Omega||_2 = ||BC_{mach} - I||_2$ in order to give an idea of the reliability in the result.

16.1 Example 1

We consider the following matrix $A \in \mathbb{R}^{20\times 20}$ and $f \in \mathbb{R}^{20}$:

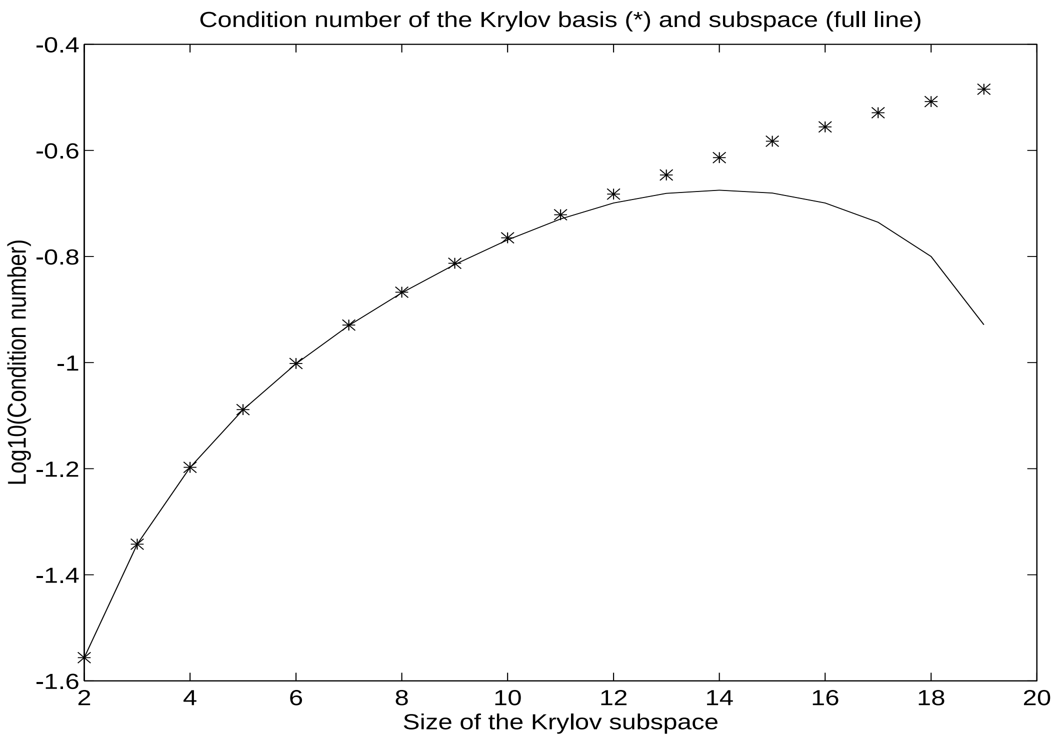
$$A = \begin{pmatrix} -7 & 36 & & \\ -1 & 0 & \ddots & \mathbf{0} \\ & \ddots & \ddots & 36 \\ \mathbf{0} & & -1 & 0 \end{pmatrix} \qquad f = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$



size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	1.571e+02	1.571e+02	1.571e+02	1.571e+02	0
3	5.809e+03	5.809e+03	5.809e+03	5.809e+03	0
4	2.092e+05	2.092e+05	2.092e+05	2.092e+05	0
5	7.525e+06	7.525e+06	7.525e+06	7.525e+06	0
6	2.706e+08	2.706e+08	2.707e+08	2.706e+08	0
7	9.685e+09	9.728e+09	9.770e+09	9.728e+09	0
8	2.793e+11	3.492e+11	4.192e+11	3.492e+11	0
9	??	1.250e+13	??	1.250e+13	0
10	??	4.440e+14	??	4.440e+14	0
11	??	7.180e+14	??	6.276e+14	0
12	??	7.180e+14	??	2.466e+13	0
13	??	7.180e+14	??	8.392e+11	0
14	??	7.180e+14	??	2.693e+10	0
15	??	7.180e+14	??	8.366e+08	0
16	??	7.180e+14	??	2.547e+07	0
17	??	7.180e+14	??	7.643e+05	0
18	??	7.180e+14	??	2.271e+04	0
19	??	7.180e+14	??	6.667e+02	0

We now take the transposed previous matrix and the same vector :

$$A = \begin{pmatrix} -7 & -1 & & \\ 36 & 0 & \ddots & \mathbf{0} \\ & \ddots & \ddots & -1 \\ \mathbf{0} & & 36 & 0 \end{pmatrix} \in \mathbb{R}^{20\times 20} \quad \text{and} \quad f = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{20}$$



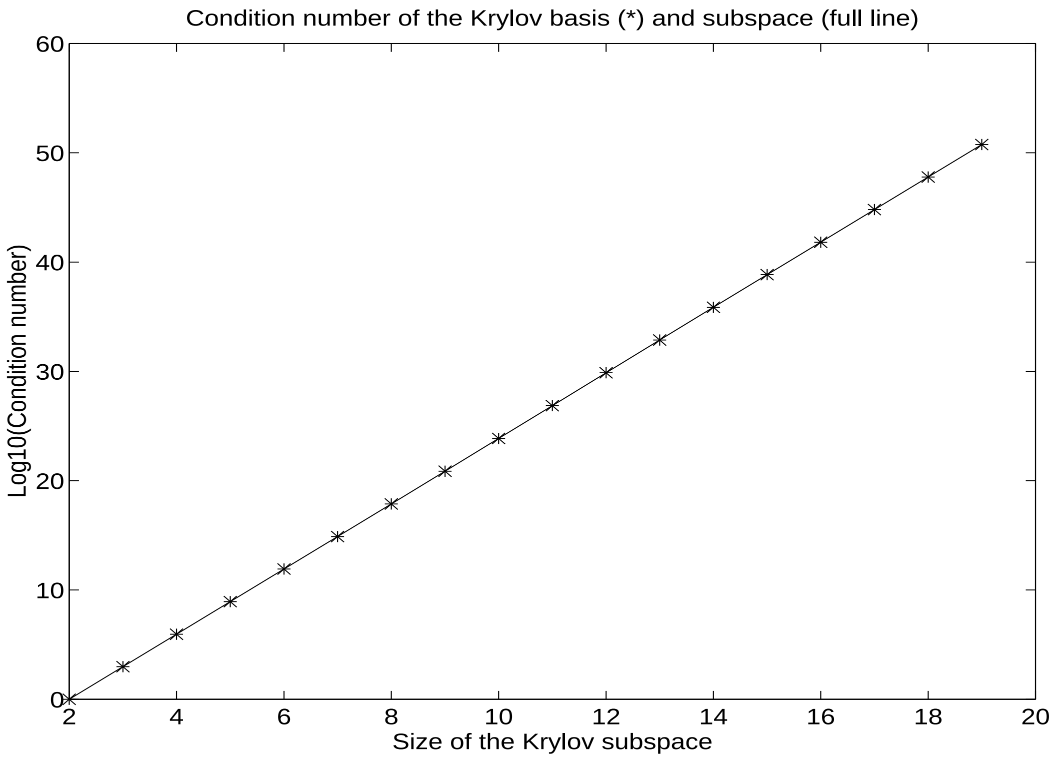
size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	4.365e+00	4.365e+00	4.365e+00	4.365e+00	0
3	7.146e+00	7.146e+00	7.146e+00	7.145e+00	0
4	9.971e+00	9.971e+00	9.971e+00	9.969e+00	0
5	1.281e+01	1.281e+01	1.281e+01	1.280e+01	3.4e-21
6	1.565e+01	1.565e+01	1.565e+01	1.564e+01	5.6e-21
7	1.849e+01	1.849e+01	1.849e+01	1.846e+01	7.7e-21
8	2.133e+01	2.133e+01	2.133e+01	2.127e+01	9.8e-21
9	2.418e+01	2.418e+01	2.418e+01	2.406e+01	1.1e-20
10	2.700e+01	2.700e+01	2.700e+01	2.677e+01	1.2e-20
11	2.985e+01	2.985e+01	2.985e+01	2.932e+01	1.2e-20
12	3.265e+01	3.265e+01	3.265e+01	3.142e+01	1.2e-20
13	3.549e+01	3.549e+01	3.549e+01	3.275e+01	1.3e-20
14	3.824e+01	3.824e+01	3.824e+01	3.322e+01	1.3e-20
15	4.106e+01	4.106e+01	4.106e+01	3.280e+01	1.3e-20
16	4.370e+01	4.370e+01	4.370e+01	3.141e+01	1.3e-20
17	4.649e+01	4.649e+01	4.649e+01	2.889e+01	1.3e-20
18	4.879e+01	4.879e+01	4.879e+01	2.490e+01	1.3e-20
19	5.149e+01	5.149e+01	5.149e+01	1.852e+01	1.3e-20

16.2 Example 2

We now take the symmetric tridiagonal matrix $A \in \mathbb{R}^{20\times 20}$ such that :

- Subdiagonal = Superdiagonal = $(1, \ldots, 1)$
- Diagonal = $(1000, \underbrace{50, \ldots, 50}_{6}, \underbrace{0, \ldots, 0}_{6}, \underbrace{50, \ldots, 50}_{6}, 1000)$

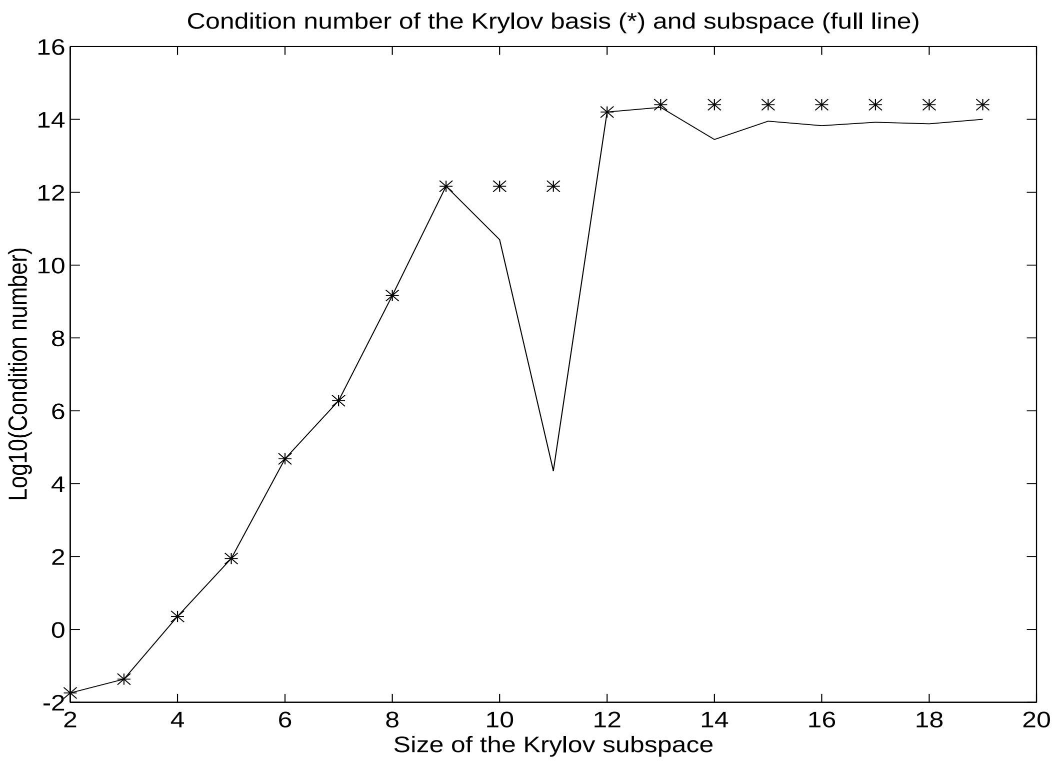
With $f = (1, 0, \ldots, 0)^T \in \mathbb{R}^{20}$, we find :



size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	1.425e+03	1.425e+03	1.425e+03	1.425e+03	0
3	1.354e+06	1.354e+06	1.354e+06	1.354e+06	0
4	1.286e+09	1.286e+09	1.286e+09	1.286e+09	0
5	1.139e+12	1.222e+12	1.304e+12	1.222e+12	0
6	??	1.161e+15	??	1.161e+15	0
7	??	1.102e+18	??	1.102e+18	0
8	??	1.047e+21	??	1.047e+21	0
9	??	1.047e+24	??	1.047e+24	0
10	??	1.047e+27	??	1.047e+27	0
11	??	1.047e+30	??	1.047e+30	0
12	??	1.047e+33	??	1.047e+33	0
13	??	1.047e+36	??	1.047e+36	0
14	??	1.047e+39	??	1.047e+39	0
15	??	9.950e+41	??	9.950e+41	0
16	??	9.452e+44	??	9.452e+44	0
17	??	8.980e+47	??	8.980e+47	0
18	??	8.531e+50	??	8.531e+50	0
19	??	8.104e+53	??	8.104e+53	0

We keep the same matrix but we take the vector

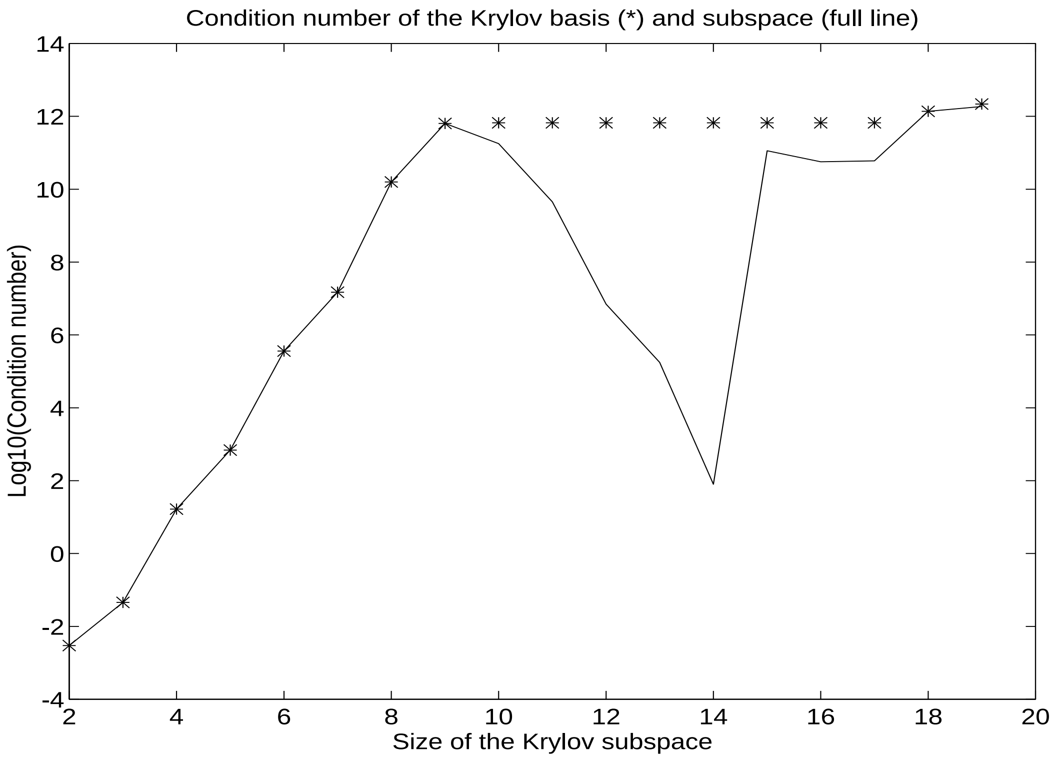
$$f = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 9 \\ 10 \\ 10 \\ 9 \\ \vdots \\ 2 \\ 1 \end{pmatrix} \in \mathbb{R}^{20}.$$



size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	2.570e+01	2.570e+01	2.570e+01	2.570e+01	0
3	6.144e+01	6.144e+01	6.144e+01	6.144e+01	1.8e-15
4	3.250e+03	3.250e+03	3.250e+03	3.250e+03	1.8e-15
5	1.260e+05	1.260e+05	1.260e+05	1.260e+05	2.5e-13
6	6.878e+07	6.879e+07	6.879e+07	6.879e+07	1.5e-11
7	2.689e+09	2.690e+09	2.692e+09	2.690e+09	7.5e-09
8	3.233e+11	2.091e+12	3.859e+12	2.091e+12	1.5e-08
9	??	2.075e+15	??	2.075e+15	2.7e-04
10	??	2.076e+15	??	7.123e+13	5.1e-04
11	??	2.076e+15	??	3.212e+07	5.1e-04
12	??	2.267e+17	??	2.267e+17	5.1e-04
13	??	3.594e+17	??	3.043e+17	1.6e-02
14	??	3.597e+17	??	3.982e+16	1.6e-02
15	??	3.597e+17	??	1.272e+17	1.6e-02
16	??	3.597e+17	??	9.517e+16	1.6e-02
17	??	3.597e+17	??	1.182e+17	1.6e-02
18	??	3.597e+17	??	1.077e+17	1.6e-02
19	??	3.597e+17	??	1.427e+17	1.6e-02

We still keep the same matrix, but we take the vector

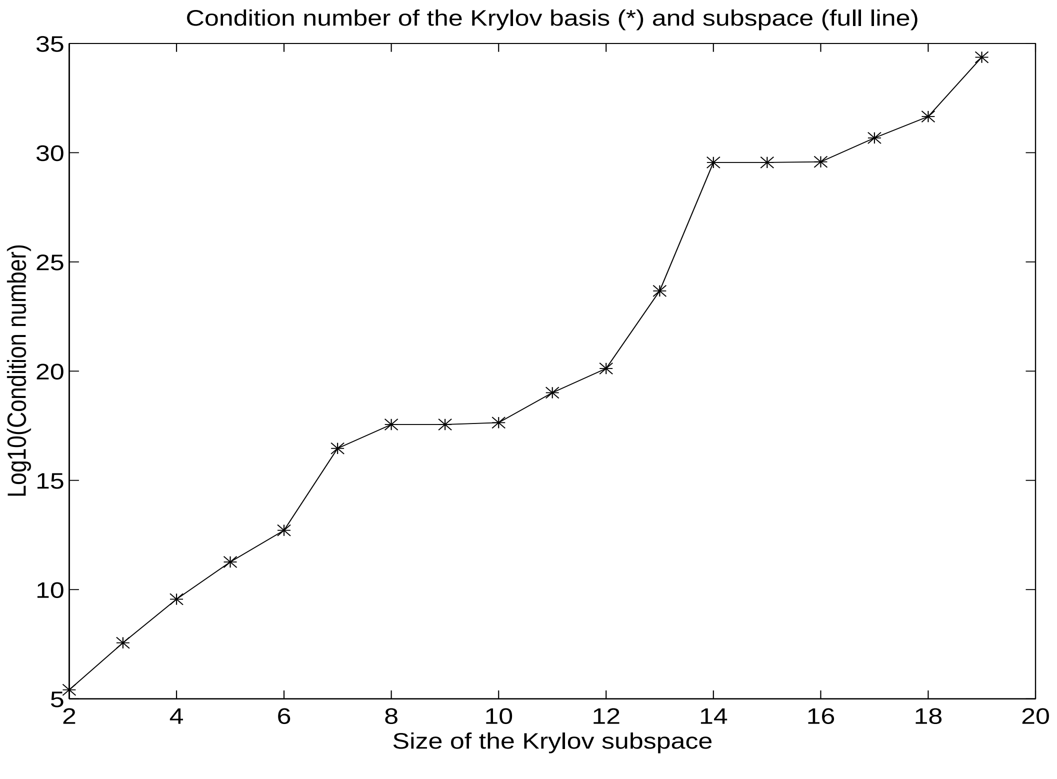
$$f = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ 20 \end{pmatrix} \in \mathbb{R}^{20}.$$



size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	4.261e+00	4.261e+00	4.261e+00	4.261e+00	0
3	6.521e+01	6.521e+01	6.521e+01	6.521e+01	4.4e-16
4	2.365e+04	2.365e+04	2.365e+04	2.365e+04	4.4e-16
5	9.812e+05	9.812e+05	9.812e+05	9.812e+05	1.1e-13
6	5.129e+08	5.130e+08	5.130e+08	5.130e+08	3.8e-12
7	2.114e+10	2.123e+10	2.131e+10	2.123e+10	1.5e-08
8	??	2.262e+13	??	2.262e+13	1.2e-07
9	??	9.078e+14	??	9.078e+14	9.9e-04
10	??	9.427e+14	??	2.551e+14	2.0e-03
11	??	9.427e+14	??	6.433e+12	2.0e-03
12	??	9.427e+14	??	9.982e+09	2.0e-03
13	??	9.427e+14	??	2.523e+08	2.0e-03
14	??	9.427e+14	??	1.133e+05	2.0e-03
15	??	9.427e+14	??	1.624e+14	2.0e-03
16	??	9.427e+14	??	8.094e+13	2.0e-03
17	??	9.427e+14	??	8.551e+13	2.0e-03
18	??	1.956e+15	??	1.956e+15	2.0e-03
19	??	3.104e+15	??	2.647e+15	2.0e-03

16.3 Example 3

With $A1 = diag(1,2,2,2,3,3,4,4,5,6,6,6,6) \in \mathbb{R}^{13 \times 13}$ and $f1 = (10^{-12}, 10^{-12}, 10^{-10}, 10^{-10}, 10^{-8}, 10^{-8}, 10^{-6}, 10^{-6}, 10^{-4}, 10^{-2}, 10^{-2}, 1, 1)^T \in \mathbb{R}^{13}$ we construct $A = Lanczos(A1, f1, 20)$, and we give here the condition numbers for A and $f = (1, 0, \ldots, 0)^T$.

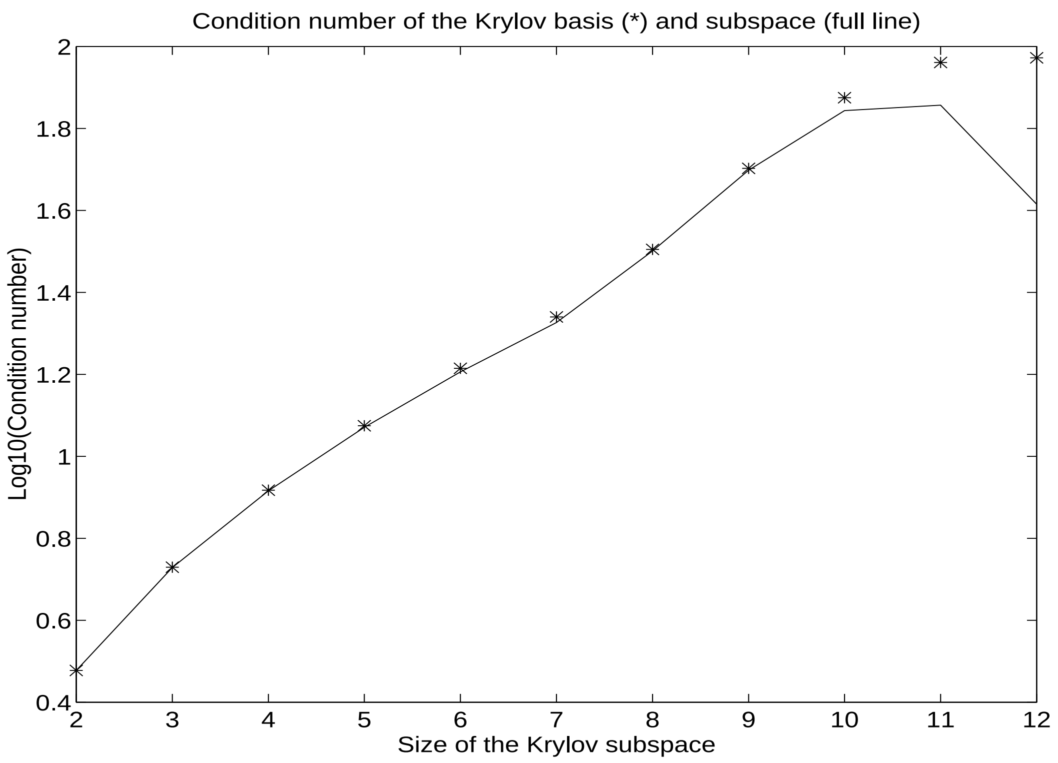


size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	2.584e+05	2.584e+05	2.584e+05	2.584e+05	0
3	3.654e+07	3.654e+07	3.654e+07	3.654e+07	7.3e-12
4	3.652e+09	3.653e+09	3.654e+09	3.653e+09	7.3e-12
5	1.805e+11	1.827e+11	1.848e+11	1.827e+11	6.3e-08
6	2.019e+12	5.174e+12	1.237e+13	5.174e+12	7.0e-08
7	??	2.884e+16	??	2.884e+16	2.4e-04
8	??	3.611e+17	??	3.611e+17	2.4e-04
9	??	3.613e+17	??	3.613e+17	2.4e-04
10	??	4.355e+17	??	4.355e+17	2.0e-03
11	??	1.031e+19	??	1.031e+19	4.0e+00
12	??	1.320e+20	??	1.320e+20	2.6e+02
13	??	4.683e+23	??	4.683e+23	2.6e+02
14	??	3.585e+29	??	3.585e+29	5.2e+05
15	??	3.585e+29	??	3.585e+29	5.4e+08
16	??	3.830e+29	??	3.830e+29	5.4e+08
17	??	4.734e+30	??	4.734e+30	2.7e+11
18	??	4.482e+31	??	4.482e+31	2.7e+11
19	??	2.339e+34	??	2.339e+34	1.4e+14

16.4 Example 4

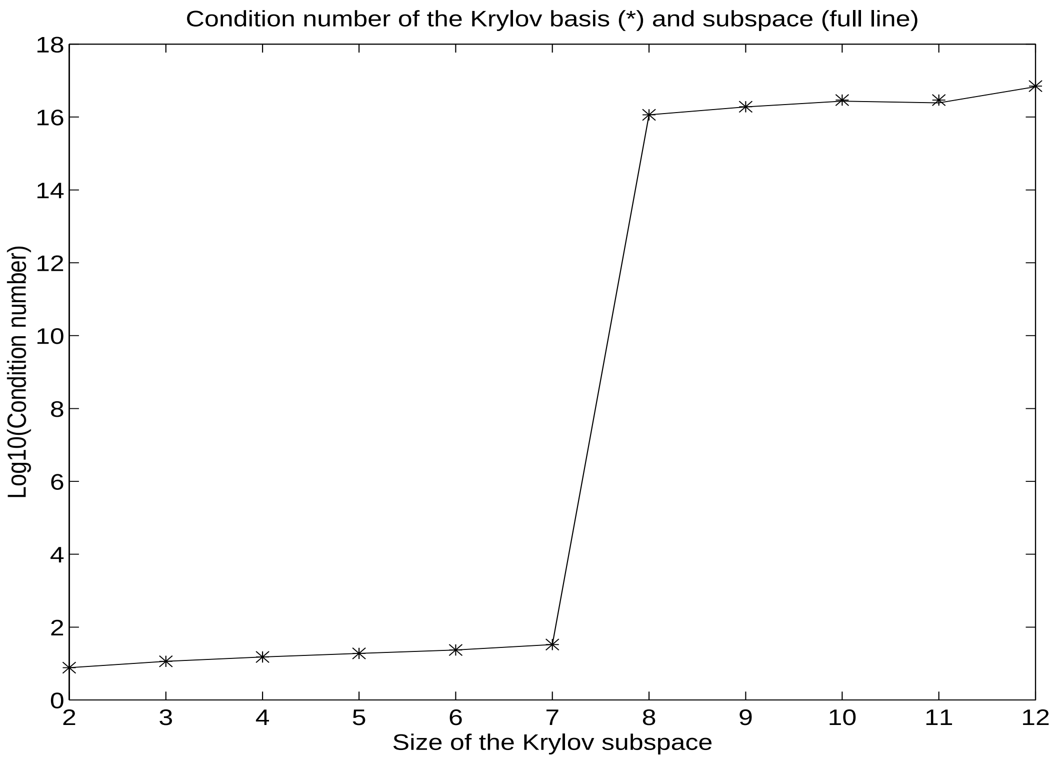
Let

$$A = \begin{pmatrix} 0 & 1 & & & \\ 12 & 0 & 2 & & \mathbf{0} \\ & \ddots & \ddots & \ddots & \\ \mathbf{0} & & 2 & 0 & 12 \\ & & & 1 & 0 \end{pmatrix} \in \mathbb{R}^{13\times 13} \quad \text{and} \quad f = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{13}$$



size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	3.005e+00	3.005e+00	3.005e+00	3.005e+00	0
3	5.364e+00	5.364e+00	5.364e+00	5.358e+00	1.4e-16
4	8.269e+00	8.269e+00	8.269e+00	8.239e+00	1.7e-16
5	1.188e+01	1.188e+01	1.188e+01	1.178e+01	1.9e-16
6	1.639e+01	1.639e+01	1.639e+01	1.608e+01	1.9e-16
7	2.189e+01	2.189e+01	2.189e+01	2.120e+01	3.2e-16
8	3.200e+01	3.200e+01	3.200e+01	3.170e+01	4.1e-16
9	5.043e+01	5.043e+01	5.043e+01	4.989e+01	4.1e-16
10	7.498e+01	7.498e+01	7.498e+01	6.977e+01	4.1e-16
11	9.141e+01	9.141e+01	9.141e+01	7.194e+01	4.1e-16
12	9.384e+01	9.384e+01	9.384e+01	4.121e+01	4.1e-16

With the previous matrix but with $f = (1 \dots 1)^T \in \mathbb{R}^{13}$ we obtain :



size	lower bound	$\mu_b\{\mathcal{K}_k(A,f)\}$	upper bound	$\mu\{\mathcal{K}_k(A,f)\}$	$(\|\Omega\|_2)_{mach}$
2	7.687e+00	7.687e+00	7.687e+00	7.687e+00	0
3	1.152e+01	1.152e+01	1.152e+01	1.152e+01	4.6e-16
4	1.511e+01	1.511e+01	1.511e+01	1.511e+01	5.1e-16
5	1.891e+01	1.891e+01	1.891e+01	1.891e+01	5.6e-16
6	2.347e+01	2.347e+01	2.347e+01	2.347e+01	5.6e-16
7	3.325e+01	3.325e+01	3.325e+01	3.325e+01	5.6e-16
8	??	1.148e+16	??	1.148e+16	6.0e-16
9	??	1.927e+16	??	1.896e+16	1.7e-01
10	??	2.928e+16	??	2.742e+16	1.7e-01
11	??	2.935e+16	??	2.440e+16	1.7e-01
12	??	7.038e+16	??	6.814e+16	1.7e-01

Chapter 17

Conclusion

We have provided an algorithm to measure the sensitivity of the Krylov subspace and the basis to matrix perturbations. This tool may be very useful for understanding instabilities of a Krylov subspace or basis. We plan to use it in various circumstances and to analyze the results thoroughly. Another direction of study is to understand the links between the stability of the Krylov subspace and the convergence of iterative methods in linear algebra exploiting these subspaces. For example, this algorithm could be used in the future to validate the computation of an invariant subspace by the Arnoldi process.

Part V

Conclusion

An important problem in scientific computing is the validation of numerical simulations. We have studied in this thesis how to control the accuracy for the following eigenproblem :

Problem (P) :

Given a square complex matrix $A \in \mathbb{C}^{n \times n}$, find some $\lambda \in \mathbb{C}$ and/or $x \in \mathbb{C}^n$ such that :

λ is an eigenvalue of A (i.e. $\det(A - \lambda I) = 0$)
and $x \neq 0$ is the associated eigenvector (i.e. $Ax = \lambda x$).

We have designed three new tools to assist the scientist in solving the problem (P).

Our first tool was SESAME (Expert System for the Selection and vAlidation of numerical MEthods) [6]. It is a knowledge based system, for the numerical library LAPACK [1], which can either select an adequate sequence of routines or validate a sequence of routines specified by the user for the solution of a given problem (P). Moreover, the system provides an estimate for the accuracy in the result. However, concerning the numerical quality of the results, some difficulties remain, in particular for defective eigenvalues. A defective eigenvalue is an eigenvalue whose geometric multiplicity (dimension of the largest associated invariant subspace) is strictly lower than its algebraic multiplicity (multiplicity as solution of the characteristic polynomial). The impact of defective eigenvalues on the resolution of the problem (P) is well understood in theory [10, 11] : infinite condition number and notion of Hölder-condition number. However, it is not easy to detect them practically since it requires the computation of the Jordan canonical form of the matrix and this computation is very difficult [24, 26]. Therefore, we cannot always explain a failure, like non convergence, when solving the problem (P).

In order to study the quality of the results of problem (P) for large sparse matrices, we developed in the second part of this thesis work a software tool for the computation of the spectral portrait of a large sparse matrix [7]. Spectral portraits [31, 46] are useful in numerous problems in scientific computing. Our tool relies on an algorithm, that computes the smallest singular value, based on a modification of Davidson's algorithm [15, 39]. The convergence can be improved by choosing a better preconditioner. Moreover, the computation time for a spectral portrait is relatively large and we have to find techniques to compute very quickly the interesting information : for instance a level line or the localization and magnitude of peaks ...

Finally, we discussed a theoretical tool for the measurment of the condition number of Krylov bases and subspaces. These subspaces are essential for many iterative methods in linear algebra [42], for instance, in the Arnoldi method [2] for the eigenproblem or in the GMRES method [43] for linear systems. The computation of these condition numbers cannot be guaranteed at this time. We then have to find a method to improve the quality of the result [9]. After that, we will use this tool in order to understand the reasons for a large condition number and to measure its impact on the convergence of different methods that use these Krylov subspaces.

Bibliography

[1] E. Anderson and al, *LAPACK Users' guide* SIAM, 1992.

[2] W.E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem.* Quart. Appl. Math., 9 :17-29, 1951.

[3] Z. Bai, J. Demmel, A. McKenney, *On the conditioning of the nonsymmetric eigenproblem : Theory and software.* LAPACK Working Note 13, 1989.

[4] F. Bauer, C. Fike, *Norms and exclusion theorems.* Num. Math. 2, 1960.

[5] C.M. O'Brien, *The GLIMPSE System.* NAG Technical Report TR 12/88. NAG Limited, Oxford, UK (September 1988).

[6] J-F. Carpraux, J. Erhel, *SESAME : a knowledge-based system for eigenvalue problems.* Third International Conference on Expert Systems for Numerical Computing, May 17-19, 1993, Purdue University, West-Lafayette, Indiana, USA. To appear in *Mathematics and Computers in Simulation 36 (1994).*

[7] J-F. Carpraux, J. Erhel, M. Sadkane, *Spectral portrait for non hermitian large sparse matrices.* SCAN-93, IMACS/GAMM International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, September 26-29, 1993, Technical University of Vienna, AUSTRIA. To appear in *Computing 52 (1994).*

[8] J-F. Carpraux, S.K. Godunov, S.V. Kuznetsov, *Stability of the Krylov bases and subspaces.* Third International Conference on Numerical Methods and Applications, August 21-26, 1994, Sofia, Bulgaria. IRISA Report N^o 834, 1994.

[9] J-F. Carpraux, *Stability of the algorithm computing the condition numbers of the Krylov bases and subspaces.* In preparation.

[10] F. Chatelin, *Valeurs propres de matrices.* Collection mathématiques appliquées pour la maîtrise, Masson, 1988.

[11] F. Chatelin, V. Frayssé, *Arithmetic reliability of Algorithms.* Symposium on High Performance Computers, Montpellier, Octobre 1991.

[12] F. Chatelin, V. Frayssé, *Qualitative Computing : Elements of a theory for finite precision computation.*, June 8-10, 1993, THOMSON-CSF, Laboratoire Central de recherches, Domaine de Corbeville, Orsay, France.

[13] P. Ciarlet, *Introduction à l'analyse matricielle.* Collection mathématiques appliquées pour la maîtrise, Masson, 1984.

[14] M. Crouzeix, B. Philippe, M. Sadkane, *The Davidson method.* SIAM J. Sci. Stat. Comput., Vol 15-1, January 1994.

[15] E.R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices.* Comp. Phys., 1975, pp. 87-94.

[16] J.W. Demmel, *Computing stable eigendecomposition of matrices.* Linear algebra and applications, N°79, 1986.

[17] I.S. Duff, R.G. Grimes, J.G. Lewis, *Sparse matrix test problems.* ACM Trans. Math. Softw. 15, 1-14, 1989.

[18] J. Erhel and B. Philippe, *Design of a toolbox to control arithmetic reliability,* in SCAN'91, Oldenburg, 1991.

[19] J. Erhel, *Statistical estimation of roundoff errors and condition numbers.* INRIA Report N°1490, 1991.

[20] *FOCUS Consortium, FOCUS : Internal reports of the FOCUS project.* Contact NAG Limited, Oxford UK (1988/1989).

[21] J.G.F. Francis, *The QR Transformation : A Unitary Analogue to the LR Transformation, Parts I and II.* Comp. J. 4, 265-72, 332-45, 1961.

[22] S.K. Godunov, *Spectral Portraits of Matrices and Criteria of Spectrum Dichotomy.* In Computer arithmetic and enclosure methods, J. Herzberger and L. Atanassova, eds., North-Holland and IMACS, 1992.

[23] S.K. Godunov, A.G. Antonov, O.P. Kiriljuk, V.I. Kostin, *Guaranteed Accuracy in Numerical Linear Algebra.* Kluwer Academic Publishers, 1993.

[24] G.H. Golub, J.H. Wilkinson, *Ill-conditioned eigensystems and the computation of the Jordan canonical form.* Stanford Tech. Rep. STAN-CS-75-478, Stanford Univ., Stanford, Calif, February 1975.

[25] G.H. Golub, C.F. Van Loan, *Matrix Computations.* The Johns Hopkins University Press, Baltimore and London, 1989.

[26] B. Kågström, A. Ruhe, *An algorithm for numerical computation of the Jordan normal form of a complex matrix.* ACM Trans. Math. Software 6 :398-419, 1980.

[27] W. Kahan, B.N. Parlett, E. Jiang, *Residual bounds on approximate eigensystems of nonnormal matrices.* SIAM J. Numer. Anal, Vol 19, 1982.

[28] *KASTLE : Internal Project Reports of The Teaching Company Scheme Project.* NAG Limited, Oxford, UK (1987/1988).

[29] T. Kato, *Perturbation theory of linear operators.* Springer-Verlag, Berlin, 1966.

[30] S. Konig, *An expert system shell for validated computation and its application to solving linear equations.* In SCAN 91, Oldenburg, 1991.

[31] V.I. Kostin, *On definition of matrices' spectra.* High Performance Computing II, 1991.

[32] A.N. Krylov, *On the numerical solution of equations whose solution determine the frequency of small vibrations of material systems (in russian).* Izv. Akad. Nauk. SSSR Otd Mat. Estest., 1 :491-539, 1931.

[33] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.* J. Res. Nat. Bur of Standards, 45 :255-282, 1950.

[34] P. Laug, *Pilotage d'un code modulaire d'éléments finis par un système expert.* INRIA Report N^o 653, Mars 1987.

[35] A.N. Malyshev, *Parallel aspects of some spectral problems in linear algebra.* CWI Report, July 1991.

[36] G. Masini, A. Napoli, *Les Langages a objets : langages de classes, langages de frames, langages d'acteurs.* InterEditions, 1989.

[37] B.N. Parlett, *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, 1980.

[38] G. Peters and J.H. Wilkinson, *Inverse Iteration, Ill-Conditioned Equations, and Newton's Method.* SIAM Review 21, 339-60, 1979.

[39] B. Philippe, M. Sadkane, *Computation of the singular subspace associated with the smallest singular values of large matrices.* INRIA Report N°2064, 1993.

[40] F. Rechenmann, P. Fontanille, P. Uvietta, *SHIRKA : Système de gestion de bases de connaissances centrées objets : manuel d'utilisation.* INRIA et laboratoire ARTEMIS/IMAG, 1990.

[41] F. Rechenmann and B. Rousseau, *A development shell for knowledge-based systems in scientific computing,* in Houstis E.N., Rice J.R. (éds), Expert Systems for Numerical Computing, Elsevier Science Publishers, Amsterdam, 1992.

[42] Y. Saad, *Numerical methods for large eigenvalue problems.* Manchester University Press, 1992.

[43] Y. Saad, M.H. Schultz, *GMRES : A generalized minimal residual algorithm for solving nonsymmetric linear systems.* SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.

[44] C. Saurel, *Contribution aux systèmes experts : développement d'un cas concret et étude du problème de la génération d'explications négatives.* Thèse d'informatique, Toulouse, 15 décembre 1987.

[45] G.W. Stewart, Ji-guang Sun, *Matrix Perturbation Theory.* Academic Press, INC, 1990.

[46] L.N. Trefethen, *Pseudospectra of matrices.* In 14th Dundee Biennal Conference on Numerical Analysis, D.F. Griffiths and G.A. Watson, eds, 1991.

[47] L.N. Trefethen, *Non-Normal Matrices and Pseudo-Eigenvalues.* Book to appear.

[48] J.M. Varah, *On the separation of two matrices.* SIAM J. Numer. Anal, Vol 16, N°2, 1979.

[49] J.H. Wilkinson, *Rounding errors in algebraic processes.* Englewoods Cliffs, N.J : Prentice-Hall, 1963.

[50] J.H. Wilkinson, *The algebraic eigenvalue problem.* Oxford University Press, 1965.

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399