ELSEVIER

Mathematics and Computers in Simulation 36 (1994) 315–325

MATHEMATICS AND COMPUTERS IN SIMULATION

SESAME: a knowledge-based system for eigenvalue problems

Jean-François Carpraux *, Jocelyne Erhel

IRISA/INRIA – Campus de Beaulieu, 35042 Rennes Cedex, France

1. Introduction

This paper presents a knowledge-based system devoted to the LAPACK library [1] to help the user in selecting a numerical procedure and in validating a result. Many front-end systems such as GLIMPSE and KASTLE developed by NAG [9,10] help the user in selecting the right routine of a scientific library. Our system is also concerned with the numerical quality of the result. It can either select a routine or check the validity of a given routine to solve a user's problem. The accuracy of the result depends upon the error on the data, the numerical stability and the condition number. Our goal is to provide an error estimation thanks to the knowledge of these three components. It should be noted that, contrarily to other expert systems such as [11], we do not intend to generate code, only to advise the user. All the knowledge base is modelized by objects, with no explicit inference rules. This approach is well-suited to describe a scientific expertise and allows to extend or modify easily the base.

We have developed a first prototype at IRISA, to demonstrate the feasibility of our goals. We do not presently deal with the complete LAPACK library but with a subset of it which aims at solving eigenvalue problems and which contains about hundred routines. We claim however that this domain is large and complex enough to show the capabilities of our system.

2. Numerical research

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. $Ax = \lambda x$).

We only consider the matrices treated in Lapack, i.e. small order matrices (order less than 5000), and with dense, packed or band storage. The case of large sparse matrices is not present in Lapack. We don't 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:

* Corresponding author.

SSDI 0378-4754(93)E0031-Y

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

We begin by studying the algorithmic resolution of the eigenvalue problem [5,8,12,16]. For each step we give the number of associated Lapack's routines. The choice of the routine for each step depends on certain properties of the data (computation in simple or double precision, real or complex matrix, symmetric or not).

To compute the eigenvalues, we first reduce the input matrix (by Householder method) in a matrix which has more null elements and the same eigenvalues, in order to reduce the time of computation of the eigenvalues. One among 16 routines must be selected for this reduction. Next, we compute the eigenvalues (by a QR-type method or by bisections) of this reduced matrix, which is either an Hessenberg matrix (nonsymmetric input matrix) or a real tridiagonal symmetric matrix (symmetric or hermitian input matrix). The choice also depends on the number of eigenvalues wanted. To compute the eigenvectors (by back-substitution or by inverse iterations), we also have to know the number of eigenvectors wanted and sometimes, the method used to compute the eigenvalues. Eigenvalues and eigenvectors computations concern respectively 12 and 16 routines.

We give below a classification of the user's eigenproblem and the names of the routines solving them. For eigenvalues and eigenvectors computations, more than one routine may be available, that is to say the computation can be done by one of several routines. We give first the best one (which provides the best tradeoff between accuracy and complexity) then other available routines between brackets (these routines have to be considered for the validation of routines). For the matrix transformation there is only one available routine.

We put an "x" as first letter of the routine names, because it only depends on the type of computation (simple or double precision) and on the storage of the matrix elements (real or complex storage). Indeed, this letter is a "S" for simple real precision computation, a "D" for double real precision computation, a "C" for simple complex precision computation and a "Z" for double complex precision computation.

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

(1) **Matrix transformation**:
  (a) Nonsymmetric matrix: xGEHRD.
  (b) Symmetric matrix:
    (i) Dense storage: xSYTRD (real), xHETRD (complex).
    (ii) Packed storage: xSPTRD (real), xHPTRD (complex).
    (iii) Band storage: xSBTRD (real), xHBTRD (complex).

(2) **Eigenvalues computation**:
  (a) Hessenberg matrix: xHSEQR.
  (b) Real tridiagonal symmetric matrix:
    (i) Less than 25%: xSTEBZ [xSTERF,xSTEQR]
    (ii) More than 25%
      A. More than 25% of the eigenvectors are desired: xSTEQR [xSTEBZ, xSTERF]
      B. Less than 25% of the eigenvectors are desired: xSTERF [xSTEBZ, xSTEQR]

(3) **Eigenvectors computation**:
  (a) Hessenberg matrix

(i) Less than 25%: xHSEIN [xTREVC if the matrix is on Schur form]
(ii) More than 25%: xTREVC [xHSEIN]

(b) Real tridiagonal symmetric matrix
(i) Less than 25%: xSTEIN [xSTEQR if the eigenvalues were computed with itself]
(ii) More than 25%
A. Eigenvalues computed with xSTEBZ: xSTEIN
B. Eigenvalues computed with xSTEQR: xSTEQR [xSTEIN]

For standard eigenvalue problems, 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 which solve them. As before, we also give between brackets the name of the available routine which is not the best one.

(1) **Symmetric matrix**
(a) Computation of all the eigenvalues and, if required, all the eigenvectors
(i) Real dense storage: xSYEV [xSYEVX]
(ii) Complex dense storage: xHEEV [xHEEVX]
(iii) Real packed storage: xSPEV [xSPEVX]
(iv) Complex packed storage: xHPEV [xHPEVX]
(v) Real band storage: xSBEV [xSBEVX]
(vi) Complex band storage: xHBEV [xHBEVX]
(vii) Real tridiagonal storage: xSTEV [xSTEVX]
(b) Computation of a few eigenvalues and, if required, of the associated eigenvectors.
(i) Real dense storage: xSYEVX [xSYEV]
(ii) Complex dense storage: xHEEVX [xHEEV]
(iii) Real packed storage: xSPEVX [xSPEV]
(iv) Complex packed storage: xHPEVX [xHPEV]
(v) Real band storage: xSBEVX [xSBEV]
(vi) Complex band storage: xHBEVX [xHBEV]
(vii) Real tridiagonal storage: xSTEVX [xSTEV]

(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.

We want now to get an estimation of the error in the result. This error depends on the condition number of the eigenproblem, on the stability of the algorithm and on the error in the initial data. The condition number of an eigenproblem consists in a measure of the variation of the eigenvalues and eigenvectors through a matrix perturbation ΔA [3]. The idea of backward analysis [4,7,15] is to prove that the approximate solution is the exact solution of a perturbed matrix. Let B be the backward error, it is a measure of the distance between the perturbed matrix $\overline{A}$ and the accurate one A. We have $B\epsilon = \|\overline{A} - A\|_2$, where ϵ is the computer precision.

Consider the computation of $\Lambda = \{\lambda_i\}_{i=1}^n$, and let $\overline{\lambda}$ be the mean of Λ ($\overline{\lambda} = \frac{1}{n} * \sum_{i=1}^n \lambda_i$). We have $|\Delta\overline{\lambda}| \leq C_\Lambda * (B\epsilon + \|\Delta A\|_2)$, where $\Delta\overline{\lambda}$ is the error done on the computation of Λ and C_Λ is the condition number of the computation of Λ. In the same way, we have $\theta_{\max}(M, M') \leq C_\theta * (B\epsilon + \|\Delta A\|_2)$ where $\theta_{\max}(M, M')$ is the maximum angle between the exact invariant subspace M and the computed one M' and C_θ is the condition number of the computation of M.

The numbers C_Λ and C_θ are sometimes easy to know. If the matrix is symmetric, it is well known that $C_\Lambda = 1$ and $C_\theta = \frac{1}{\delta}$ where $\delta = \text{dist}(\text{sp}(A) - \Lambda, \Lambda)$; $\text{sp}(A)$ designs the A-spectrum. If the matrix is nonsymmetric and if we used an expert routine to compute the eigenvalues and/or eigenvectors, the condition numbers are known as result of that routine [2]. So in this case we do not have to compute these condition numbers again.

The next section is devoted to the translation of the mathematical expertise into the knowledge base.

3. Description of the knowledge base

Our system is based on the development shell SHIRKA [14], written in Le-Lisp, which 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. Multiple inheritance allows to describe complex objects and is well suited to mathematical entities. The inference rules are implicitly embedded into the knowledge base thanks to the organisation in hierarchies and to default values or attached procedures. No explicit rule such that "if-then-else" has to be written. The basic inference mechanism is the classification process which can take into account a given subset of attributes and which can infer values of another given subset of attributes. The process can explain any result when wanted. SHIRKA is easy to use thanks to a user interface which visualizes the objects and the actions taken on them.

So, we define an object to find out the first letter of the name of the routine. For this, we only have to know the computer precision used and the storage type of matrix (real or complex).

To find the rest of the name we have to know if the matrix is symmetric (or hermitian: we assume that hermitian = symmetric + complex) or not, and the storage of the matrix (dense, packed, band or real tridiagonal). We also have to know the number of eigenvalues and eigenvectors wanted (zero, a few (less than 25%) or a lot (more than 25%)). So we choose to create two objects: one for the matrix with its characteristics (real ?, symmetric ?, storage ?) and the second to define the eigenproblem (number of eigenvalues and eigenvectors wanted).

To solve an eigenproblem, the user can either select a driver routine or a sequence of routines corresponding to the three mathematical steps described above, namely matrix transformation, eigenvalues and eigenvectors computations. Specific objects are devoted to drivers and to each of these steps and can be used either to select or to validate a routine. Typically, standard eigenproblems refer to the driver routine object whereas others require a sequence among three objects. We chose to create an object for each of these three steps of the mathematical resolution to get more flexibility and to include easily particular cases. For instance, if no eigenvectors are to be computed we only need the two first objects. On the contrary, we only need the last one when only eigenvectors are desired (if the user already knows the eigenvalues).

Moreover, particular matrices may occur, such as Hessenberg matrices which of course do not require the transformation step so that only the last two objects are concerned.

The method to compute the eigenvectors depends on the final matrix after transformation and eigenvalue computations. The three objects for each step contain an attribute to describe the form of the matrix (tridiagonal, hessenberg, etc.). This attribute may vary after each step and is used by the system to take into account particular cases and to select the right third step for eigenvector computation. By this way, immediate answers such as with diagonal matrices can be given quickly.

The validation step must first check if the user's routine belongs effectively to Lapack. We want to avoid nonexistent names with false combinations of the first letter and the generic name, such as SHEEV where S means real single precision and HEEV exists only for complex matrices. Therefore we create an object providing a list of Lapack routines where the user must choose the routines to validate.

To give an estimation of the quality of the result, we have a first object which checks if we already know the condition number or if it has to be computed. In this case it provides the name of the routine which computes this number. Then we have a second object which estimates the error $(C * (B\epsilon + \Delta A))$ and provides it to the user. We have finally a last object which provides the user advises to improve the result, if required.

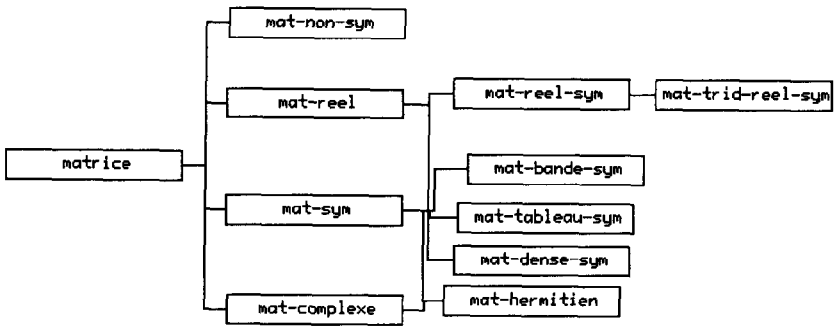
All the objects contained in the knowledge base are pictured below ; they are divided into three groups corresponding to their use:

- Description of the problem: the matrix (object *Matrice*), the requirements (number of eigenvalues and eigenvectors) (object *Pb*), the environment (precision) (object *Type*).
- Procedures: the driver routine (object *Driver-routine*), the procedure to transform the matrix (object *Transfo-matrice*), the procedure to compute the eigenvalues (object *Calc-val-prop*), the procedure to compute the eigenvectors (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*).

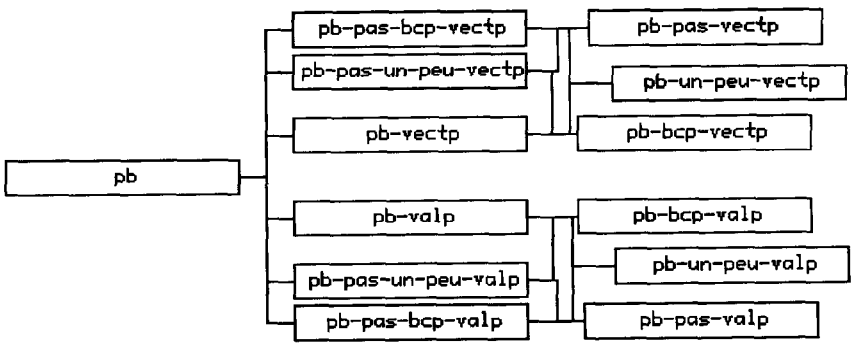
4. Use of the system

For both selection and validation of routine, the user begins to describe the context defined by a matrix, an eigenproblem and the type of the computation.

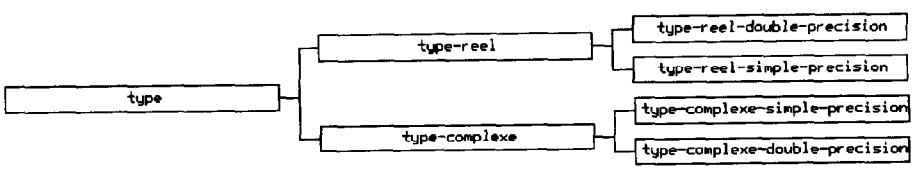
To select a routine, the system infers the procedure name thanks to a classification on a restricted set of the attributes. First, it finds the first letter of the routine name by classification with inference of the *Type* object. Then it classifies with inference the *Driver-routine* object. If it's a success the system stops. If not, it classifies successively the objects *Transfo-matrice, Calc-val-prop, Calc-vect-prop* to find the sequence of routines names.



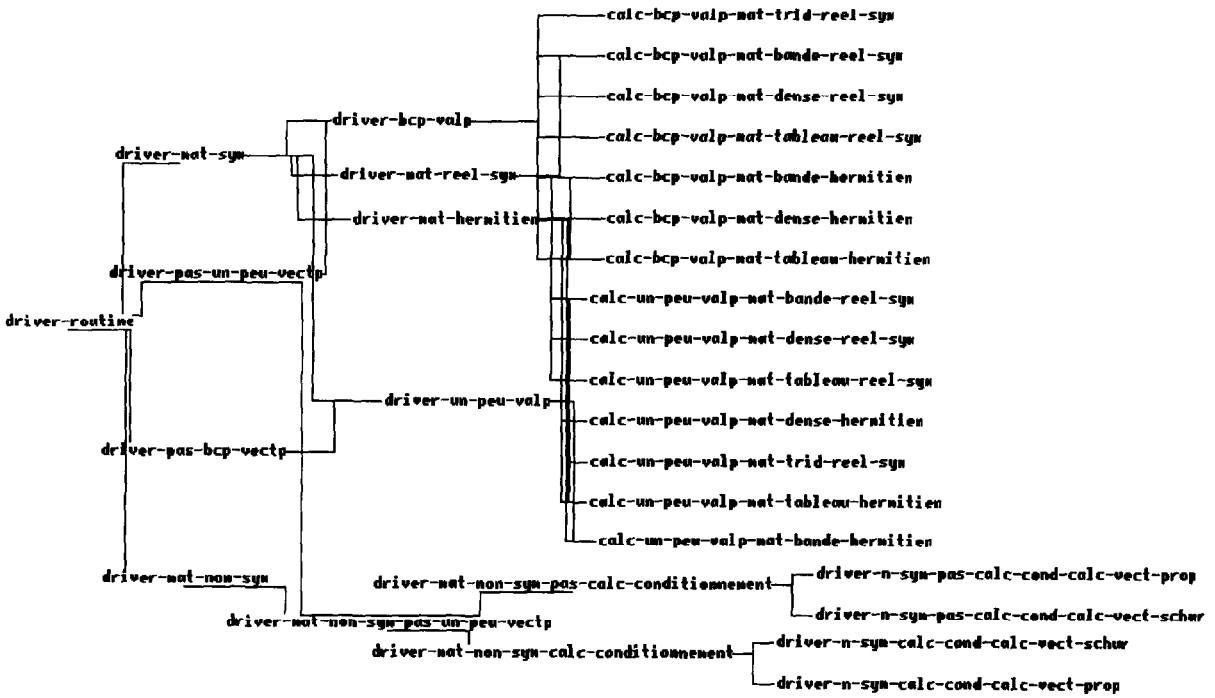
Object *Matrice.*



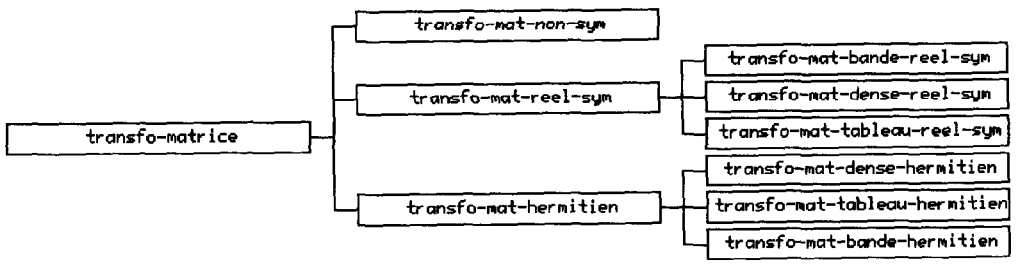
Object *Pb.*



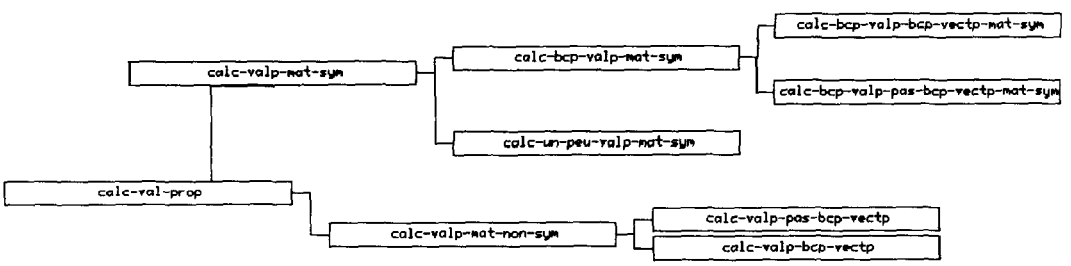
Object *Type.*



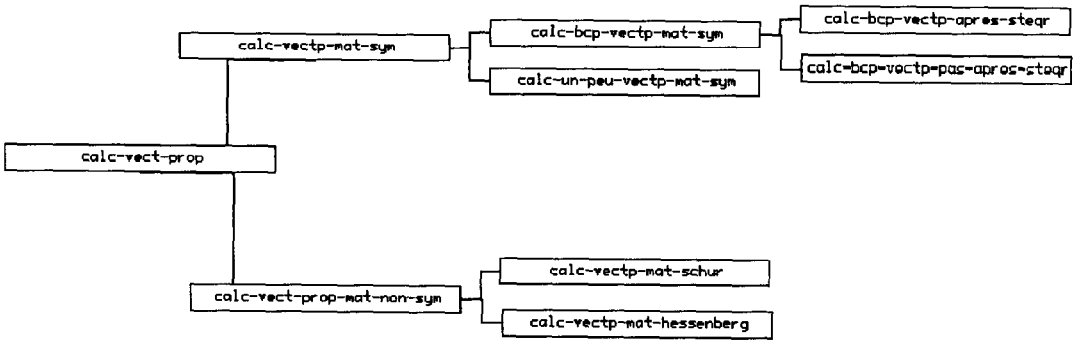
Object *Driver-routine.*



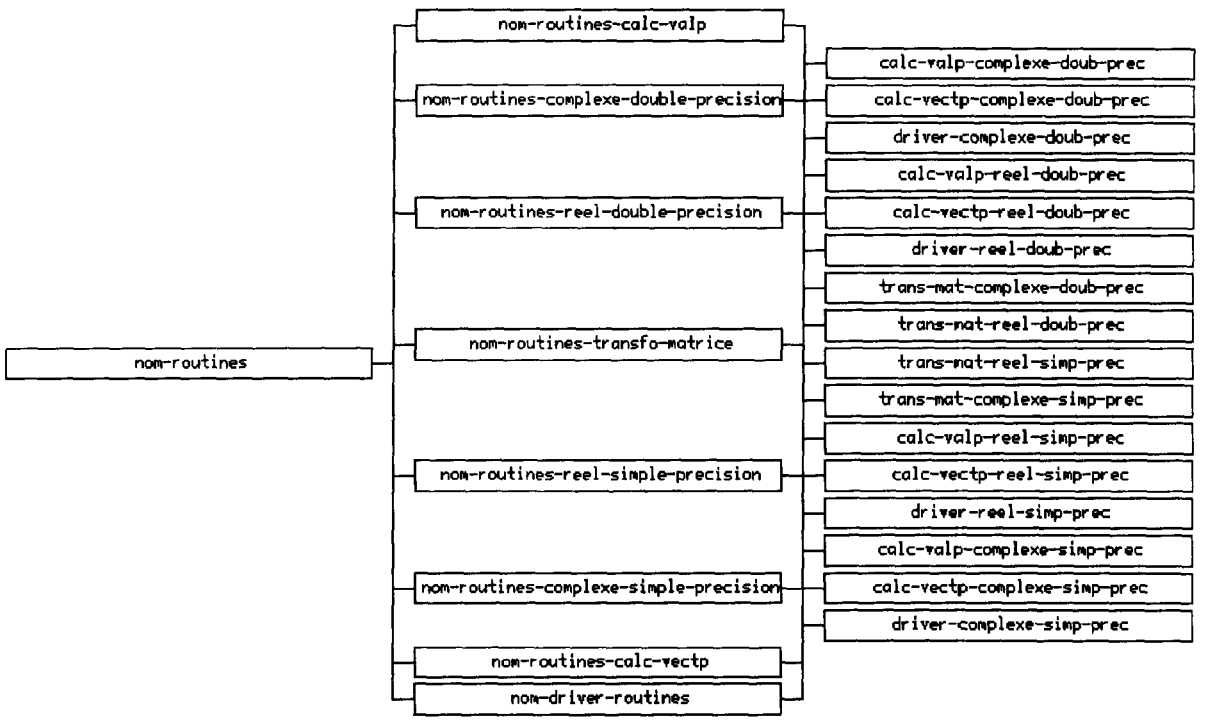
Object *Transfo-matrice.*



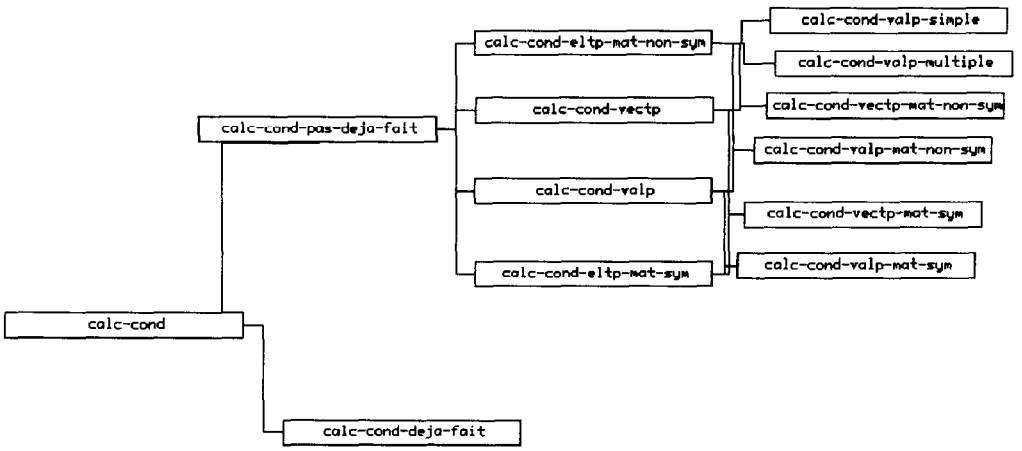
Object *Calc-val-prop.*



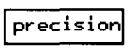
Object *Calc-vect-prop.*



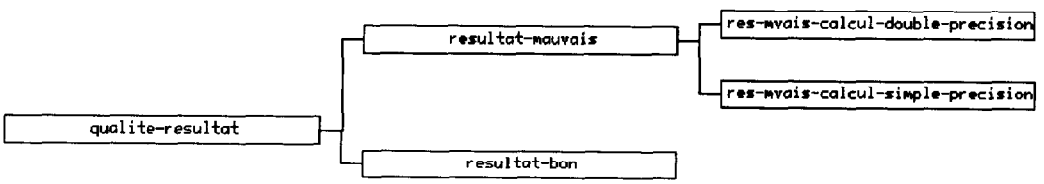
Object *Nom-routines.*



Object *Calc-cond.*



Object *Precision.*



Object *Qualite-resultat.*

On the other hand, to check a routine, the user provides also the procedure name. For this, the system classifies with inference the object *Nom-routines* to propose a list of existing routine names. The user has so to choose a name in the list. Then the system detects if the method is suitable or not after classification of those same four objects without inference on all the attributes. By comparing with its own selection, the system can also find if the method is the best one.

To validate a result, the user provides an estimation of the error on the data which may come from a previous estimation by the system and of the condition number which can be computed by the routine proposed by the system (classification with inference of the object *Calc-cond*). Then the system gives an error estimation (classification with inference of the object *Precision*). The analysis error is done by classification with inference of the object *Qualite-resultat*.

5. Conclusion

Knowledge-based systems are quite useful for helping the user of scientific libraries. We developed a prototype dealing with a subset of the Lapack library and providing help in selecting a routine, validating a user's routine and estimating the accuracy of the result. Our system is constructed upon a development-shell by defining objects corresponding to mathematical entities and by using a powerful classification mechanism on the hierarchies. This knowledge-base can be easily extended to the whole LAPACK library thanks to this approach. Currently, our system uses a general interface which is not well-suited for scientific computing. We plan to develop a specific interface to hide all the internal representations and to provide explanations directly in a scientific language.

Acknowledgments

The authors are indebted to F. Rechenmann for providing the system SHIRKA and for his technical assistance.

References

[1] E. Anderson et al., LAPACK Users' guide, SIAM (1992).
[2] Z. Bai, J. Demmel and A. McKenney, On the conditioning of the nonsymmetric eigenproblem: Theory and software, LAPACK Working Note 13, 1989.
[3] F. Chatelin, Valeurs propres de matrices, Collection mathématiques appliquées pour la maîtrise (Masson, Paris, 1988).
[4] F. Chatelin and V. Frayssé, Arithmetic reliability of algorithms, Symposium on High Performance Computers, Montpellier (1991).
[5] P. Ciarlet, Introduction à l'analyse matricielle, Collection mathématiques appliquées pour la maîtrise (Masson, Paris, 1984).
[6] J. Erhel and B. Philippe, Design of a toolbox to control arithmetic reliability, SCAN'91, Oldenburg (1991).
[7] J. Erhel, Statistical estimation of roundoff errors and condition numbers, Rapport de recherche INRIA No. 1490, 1991.
[8] G.H. Golub and C.F. Van Loan, Matrix Computations (Johns Hopkins University Press, 1983).
[9] C.M. O'Brien, The GLIMPSE System, NAG Technical Report TR 12/88, NAG Limited, Oxford, UK, September 1988.
[10] KASTLE: Internal project reports of the teaching company scheme project, NAG Limited, Oxford, UK (1987/1988).

[11] S. Konig, An expert system shell for validated computation and its application to solving linear equations, SCAN'91, Oldenburg (1991).
[12] B.N. Parlett, The Symmetric Eigenvalue Problem (Prentice-Hall, Englewood Cliffs, NJ, 1980).
[13] F. Rechenmann, P. Fontanille and P. Uvietta, SHIRKA: Système de gestion de bases de connaissances centrées objets: manuel d'utilisation, INRIA et Laboratoire ARTEMIS/IMAG (1990).
[14] F. Rechenmann and B. Rousseau, A development shell for knowledge-based systems in scientific computing, in: E.N. Houstis and J.R. Rice, eds., Expert Systems for Numerical Computing (Elsevier, Amsterdam, 1992).
[15] J.H. Wilkinson, Rounding errors in algebraic processes (Prentice-Hall, Englewoods Cliffs, NJ, 1963).
[16] J.H. Wilkinson, The algebraic eigenvalue problem (Oxford University Press, Oxford, 1965).