Rotinas para resolver sistemas lineares, auto-problemas:

IMSL Fortran Numerical Library:

https://developer.nvidia.com/imsl-fortran-numerical-library

The IMSL Fortran Numerical Library is a comprehensive set of mathematical and statistical functions that developers can embed into their Fortran software applications. It offloads CPU work to NVIDIA GPU hardware where the cuBLAS library is utilized. Users with supported hardware are able to link the IMSL Fortran Library with cuBLAS to gain significant performance improvements for many linear algebra functions. The calling sequences for IMSL functions are untouched, so there is no learning curve and users can be immediately productive.

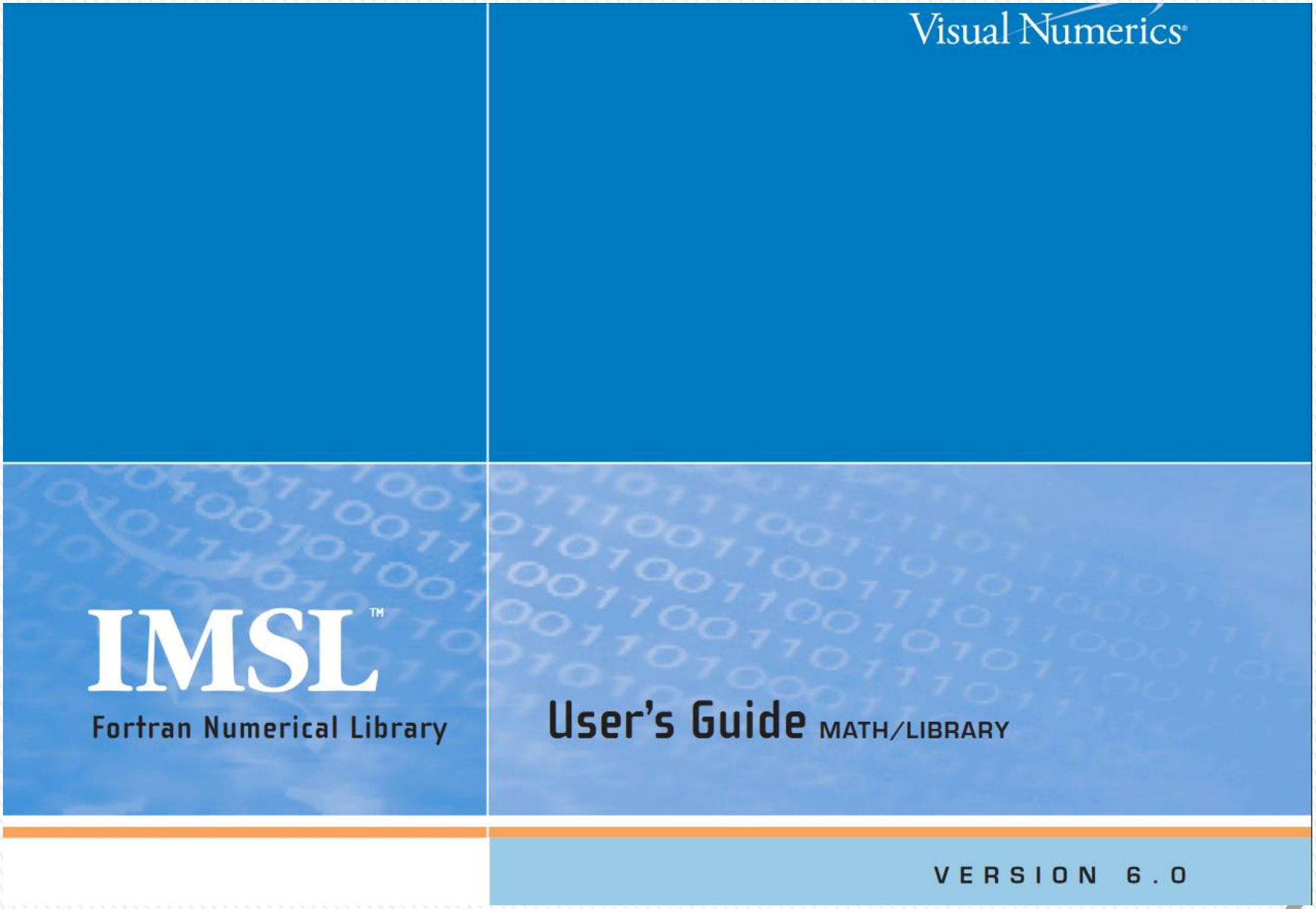
Key Features

- 1000+ accurate and highly reliable algorithms
- Support for the optional arguments of modern Fortran syntax
- Intuitive naming conventions eliminate the need for developers to learn or remember special function names
- Evolves easily with software and hardware upgrades

The IMSL Fortran Numerical Library provides analytical building blocks that elimi
scratch, and can save up to 95% of the time required to research and develop algo
algorithms free QA teams to focus on core application testing.

For more information about IMSL and other GPU Accelerated Libraries:

- Download evaluation copy
- Rogue Wave IMSL for Fortran features
- GPU Accelerated Libraries



SUBROUTINE LSLRG

SUBROUTINE DLSLRG

LAPACK — Linear Algebra PACKage

http://www.netlib.org/lapack/index.html

Presentation

LAPACK is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

The original goal of the LAPACK project was to make the widely used EISPACK and LINPACK libraries run efficiently on shared-memory vector and parallel processors. On these machines, LINPACK and EISPACK are inefficient because their memory access patterns disregard the multi-layered memory hierarchies of the machines, thereby spending too much time moving data instead of doing useful floating-point operations. LAPACK addresses this problem by reorganizing the algorithms to use block matrix operations, such as matrix multiplication, in the innermost loops. These block operations can be optimized for each architecture to account for the memory hierarchy, and so provide a transportable way to achieve high efficiency on diverse modern machines. We use the term "transportable" instead of "portable" because, for fastest possible performance, LAPACK requires that highly optimized block matrix operations be already implemented on each machine.

LAPACK routines are written so that as much as possible of the computation is performed by calls to the Basic Linear Algebra Subprograms (BLAS). LAPACK is designed at the outset to exploit the Level 3 BLAS — a set of specifications for Fortran subprograms that do various types of matrix multiplication and the solution of triangular systems with multiple right-hand sides. Because of the coarse granularity of the Level 3 BLAS operations, their use promotes high efficiency on many high-performance computers, particularly if specially coded implementations are provided by the manufacturer.

Highly efficient machine-specific implementations of the BLAS are available for many modern high-performance computers. For details of known vendor- or ISV-provided BLAS, consult the BLAS FAQ. Alternatively, the user can download ATLAS to automatically generate an optimized BLAS library for the architecture. A Fortran 77 reference implementation of the BLAS is available from netlib; however, its use is discouraged as it will not perform as well as a specifically tuned implementation.

Acknowledgments:

Since 2010, this material is based upon work supported by the National Science Foundation under Grant No. NSF-OCI-1032861. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF). Until 2006, this material was based upon work supported by the National Science Foundation under Grant No. ASC-9313958, NSF-0444486 and DOE Grant No. DE-FG03-94ER25219. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF) or the Department of Energy (DOE).

The LAPACK project is also sponsored in part by MathWorks and Intel since many years.

LAPACK — Linear Algebra PACKage

Software

Licensing

LAPACK is a freely-available software package. It is available from netlib via anonymous ftp and the World Wide Web at http://www.netlib.org/lapack . Thus, it can be included in commercial software packages (and has been). We only ask that proper credit be given to the authors.

The license used for the software is the modified BSD license, see:

- LICENSE

Like all software, it is copyrighted. It is not trademarked, but we do ask the following:

- If you modify the source for these routines we ask that you change the name of the routine and comment the changes made to the original.
- We will gladly answer any questions regarding the software. If a modification is done, however, it is the responsibility of the person who modified the routine to provide support.

LAPACK, version 3.7.1

- Download: lapack-3.7.1.tgz
- LAPACK 3.7.1 Release Notes
- Updated: June 25, 2015
- LAPACK GitHub Open Bug (Current known bugs)

Standard C language APIs for LAPACK

collaboration LAPACK and INTEL Math Kernel Library Team

- LAPACK C INTERFACE is now included in the LAPACK package (in the lapacke directory)
- LAPACKE User Guide
- Updated: November 16, 2013
- header files: lapacke.h, lapacke_config.h, lapacke_mangling.h, lapacke_utils.h

http://icl.cs.utk.edu/lapack-for-windows/index.html

LAPACK / CLAPACK / ScaLAPACK for Windows

Home LAPACK CLAPACK ScaLAPACK

LAPACK, CLAPACK AND SCALAPACK FOR WINDOWS

The LAPACK community has decided to extend its support to Microsoft Windows based users. The decision was taken due to the large amount of requests we received. The project included in this initiative are LAPACK, CLAPACK and ScaLAPACK . For more information on those libraries, please refer to their website.

Computational Software Development with Windows HPC

The Innovative Computing Laboratory (ICL) at the University of Tennessee has been engaged in High-performance computing (HPC) research through the development of applications, scientific codes, and computational libraries that have helped form the fabric of HPC computing in the 21st century.

One of our goal is to extend HPC applications, software and mathematical libraries to the Microsoft Windows Compute Cluster environment. To this end we hope to understand the requirements and opportunities for computational based research and innovation utilizing Windows high performance computing.

What you will find here

Here you should be able to able all Windows-related libraries and package for LAPACK (Fortran), CLAPACK (C), and ScaLAPACK (C and Fortran). The first question to ask you is what you really need .

- For a **basic** usage, we recommend to use **prebuilt libraries**.
- For an **advanced** usage requiring special compilation or optimization, we recommend to use the **packages** that will allow you to compile our libraries on your machine.

Pre-built libraries

This is the easy way to get the library installed on your machine. Read carefully the requirements that are needed.

LAPACK pre-built libraries
Requirement: Visual Studio, Intel Fortran compiler
CLAPACK pre-built libraries
Requirement: Visual Studio, Intel Fortran compiler
ScaLAPACK pre-built libraries
Requirement: Visual Studio, Microsoft MPI, Intel and C Fortran compiler

http://math.nist.gov/MatrixMarket/

A visual repository of test data for use in comparative studies of algorithms for numerical linear algebra, featuring nearly 500 sparse matrices from a variety of applications, as well as matrix generation tools and services.

Browse
by collection
by matrix name
by generator name
the top ten

Interactive Generation
via Java (the Deli)

Documentation
File Formats
File Compression
Matrix Structure Plots
Matrix Cityplots
3D Interactive Cityplots
Spectral Portraits

Search
by matrix properties
by application area
by contributor

Software
Matrix Market I/O:
... in C, Fortran, Matlab
Harwell-Boeing I/O:
... in C, Fortran, Matlab
BeBOP Format Conversion Utility

Other Resources
Bibliography
Glossary
Related sites

Background
Welcome
What's New
What's Coming
Credits

Sponsors
NIST
ITL
TEMSS
MCSD
DARPA

Contact Us
matrixmarket@nist.gov

http://www.netlib.org/cgi-bin/search.pl

[Home]

Search :

- Netlib Repository (standard interface)
- Netlib Repository (advanced interface)
- GAMS Problem Taxonomy
- Numerical Analysis Digest

To search the Netlib repository, enter your search query in the form below and then press the Go button.

linear systems

Look for : any of the words Go!

Query "linear systems" found 968 matches.

1. tennessee/ut-cs-91-145.ps

by: Bruce MacLennan,
title: Continuous Symbol Systems: The Logic of Connectionism,
ref: University of Tennessee Technical Report CS-91-145, September 1991.
for: It has been long assumed that knowledge and thought are most naturally represented as discrete symbol systems (calculi). Thus a major contribution of connectionism is that it provides an alternative model of knowledge and cognition that avoids many of the limitations of the traditional approach. But what idea serves for connectionism the same unifying role that the idea of a calculus served for the traditional theories? We claim it is the idea of a continuous symbol system., This paper presents a preliminary formulation of continuous symbol systems and indicates how they may aid the understanding and development of connectionist theories. It begins with a brief phenomenological analysis of the discrete and continuous; the aim of this analysis is to directly contrast the two kinds of symbols systems and identify their distinguishing characteristics. Next, based on the phenomenological analysis and on other observations of existing continuous symbol systems and connectionist models, I sketch a mathematical characterization of these systems. Finally the paper turns to some applications of the theory and to its implications for knowledge representation and the theory of computation in a connectionist context. Specific problems addressed include decomposition of connectionist spaces, representation of recursive structures, properties of connectionist categories, and decidability in continuous formal systems.
Score: 99%

2. tennessee/ornl-tm-12404.ps

title: Software Libraries for Linear Algebra Computation on High-Performance Computers
by: Jack J. Dongarra and David W. Walker
ref: Oak Ridge National Laboratory, ORNL TM-12404, August, 1993.
for: This paper discusses the design of linear algebra libraries for high performance computers. Particular emphasis is placed on the development of scalable algorithms for MIMD distributed memory concurrent computers. A brief description of the EISPACK, LINPACK, and LAPACK libraries is given, followed by an outline of ScaLAPACK, which is a distributed memory version of LAPACK currently under development. The importance of block-partitioned algorithms in reducing the frequency of data movement between different levels of hierarchical memory is stressed. The use of such algorithms helps reduce the message startup costs on distributed memory concurrent computers. Other key ideas in our approach are the use of distributed versions of the Level 3 Basic Linear Algebra Subprograms (BLAS) as computational building blocks, and the use of Basic Linear Algebra Communication Subprograms (BLACS) as communication building blocks. Together the distributed BLAS and the BLACS can be used to construct higher-level algorithms, and hide many details of the parallelism from the application developer. The block-cyclic data distribution is described, and adopted as a good way of distributing block-partitioned matrices. Block-partitioned versions of the Cholesky and LU factorizations are presented, and

Para resolver sistemas lineares esparsos, sugiro MA27

http://www.hsl.rl.ac.uk/ipopt/

HSL for IPOPT

HSL provides a number of linear solvers that can be used in IPOPT. We provide several different ways for IPOPT users to download our codes.

Free downloads

Are available from the boxes on the right. The personal licence on the HSL Archive package permits commercial use, but **not redistribution**.

A condition of the licence is that HSL is cited in any resulting publications or presentations:
"HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/"

Commercial Licencing

Details of commercial licencing, including incorporation licencing, are available on our licencing page. Alternatively, please contact us at hsl@stfc.ac.uk

Which solver?

For general use we recommend HSL_MA97. For small or highly sparse problems use MA57. For huge problems use HSL_MA86 (if factors fit in memory) or HSL_MA77 (if they don't).

Solver	Free to all	Free to academics	Problem size	Parallel	Repeatable answers	Notes
MA27	Yes	Yes	Small	No	Yes	Out dated, relatively slow
MA57		Yes	Small / Medium	Threaded BLAS	Yes	
HSL_MA77		Yes	Huge	Limited	Yes	Out-of-core
HSL_MA86		Yes	Large	Highly	No*	Designed for multicore
HSL_MA97		Yes	Small / Medium / Large	Yes	Yes	Slower than HSL_MA86 on large problems

http://www.hsl.rl.ac.uk/catalogue/

HSL Software Index

For information on licencing HSL please see our licencing page.

Please remember to cite HSL as:
"HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/"

HSL has moved to a rolling release schedule. New and upgraded packages are added continuously.
A **list of recent changes** is available.

Fortran | MATLAB | C

EIGENVALUES AND EIGENVECTORS

EA: Eigenvalues and eigenvectors of real symmetric matrices

EA16	Compute selected eigenpairs using rational Lanczos method
HSL_EA19	Sparse symmetric or Hermitian: leftmost eigenpairs
HSL_EA20	s-root of a sparse self-adjoint positive-definite pencil
EA22	Sparse symmetric: simultaneous iteration
EA25	Sparse symmetric: Lanczos for the spectrum

EB: Eigenvalues and eigenvectors of general matrices

EB13	Sparse unsymmetric: Arnoldi's method
EB22	Sparse unsymmetric: subspace iteration

EP: Parallel eigenvalues and eigenvectors of real symmetric matrices

EP25	Sparse symmetric: Lanczos for the spectrum

MATHEMATICAL FUNCTIONS

FA: Random numbers

FA14	Uniform distribution
HSL_FA14	Uniform distribution

FD: Simple Functions

FD15	Real-valued machine constants

SORTING

KB: Sorting numbers

KB05	Sort numbers into ascending order using Quicksort
KB06	Sort numbers into descending order using Quicksort

Respostas Dinâmicas

Resposta Estática: $[K]\cdot\{U\}=\{F\}$ Ações não variam no tempo

Resposta Dinâmica: $[K]\cdot\{U\}+[M]\cdot\{\ddot{U}\}=\{F\}$

Ações variam no tempo, inclusão das forças inerciais

[M]: matriz de massa da estrutura

Respostas Dinâmicas, duas classes a ser analisado, para obter:

- **as frequências naturais das vibrações livres e seus correspondentes modos:**

$$[K]\cdot\{U\}+[M]\cdot\{\ddot{U}\}=\{0\}$$

- **Movimentos e esforços devidos a ações prescritas forçadas:**

$$[K]\cdot\{U\}+[M]\cdot\{\ddot{U}\}=\{F\}$$

Respostas Dinâmicas

Matriz de Massa Consistente: Uma das formas mais usadas.

Advém do uso da funções de forma da matriz de rigidez. Assim, para cada elemento "e":

$$m^e = \int_0^{\ell} \rho \cdot [N]^t . [N] \, d\xi$$

N : *funções de forma*

$$m^e = \frac{\rho \cdot L}{420} \begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & -22 \cdot L & 0 & 54 & 13 \cdot L \\ 0 & -22 \cdot L & 4 \cdot L^2 & 0 & -13 \cdot L & -3 \cdot L^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & -13 \cdot L & 0 & 156 & 22 \cdot L \\ 0 & 13 \cdot L & -3 \cdot L^2 & 0 & 22 \cdot L & 4 \cdot L^2 \end{bmatrix}$$

Onde ρ é massa por comprimento (ρ =A.γ)

γ : kg/m^3 **(massa específica)**

Respostas Dinâmicas

Matriz de Massa *Lumped* (massa concentrada; “amontoadora”)

Matriz alternativa, concentrar a massa em seus nós, para cada elemento “*e*”.

$$m^e = \frac{\rho \cdot L}{2}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$m^e_{rot} = \frac{\rho \cdot L}{2}\left(\frac{I}{A} + \frac{L^2}{12}\right)\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz com vantagem p/ montagem, economia de memória e processamento.

Matriz de Massa *Lumped* (massa concentrada; "amontoadora")

É interessante seu uso para aplicações onde simulam-se massas não-estruturais, como na engenharia aeronáutica, considerando massas concentradas devidos a: cargas, combustíveis, etc..

The first appearance of a mass matrix in a journal article occurs in two early-1930s papers[5] by Duncan and Collar [34,35]. There it is called "inertia matrix" and denoted by $[m]$. The original example [34, p. 869] displays the 3×3 diagonal mass of a triple pendulum. In the book [55] the notation changes to A.

Diagonally lumped mass matrices (DLMM) dominate pre-1963 work. Computational simplicity was not the only reason. Direct lumping gives an obvious way to account for *nonstructural masses* in simple discrete models of the spring-dashpot-particle variety. For example, in a multistory building "stick model" wherein each floor is treated as one DOF in lateral sway under earthquake or wind action, it is natural to take the entire mass of the floor (including furniture, isolation, etc.) and assign it to that freedom. Nondiagonal masses pop-up ocassionally in aircraft matrix analysis — e.g. wing oscillations in [55, §10.11] — as a result of measurements. As such they necessarily account for nonstructural masses due to fuel, control equipment, etc.

Matriz de Massa Consistente

The formulation of the consistent mass matrix (CMM) by Archer [6,7] was a major advance. All CMMs displayed in §31.3 were first derived in those papers. The underlying idea is old. In fact it follows directly from the 18^{th}-Century Lagrange dynamic equations [84], a proven technique to produce generalized masses.

These had to wait until three things became well established by the early 1960s: (i) the Direct Stiffness Method, (ii) the concept of shape functions, and (iii) the FEM connection to Rayleigh-Ritz. The critical ingredient (iii) was established in Melosh's thesis [97] under Harold Martin. The link to dynamics was closed with Archer's contributions, and CMM became a staple of FEM. But only a loose staple. Problems persisted:

(a) Nonstructural masses are not naturally handled by CMM. In systems such as ships or aircraft, the structural mass is only a fraction (10 to 20%) of the total.

(b) It is inefficient in some solution processes, notably explicit dynamics.[6]

(c) It may not give the best results compared to other alternatives.[7]

(d) For elements derived outside the assumed-displacement framework, the stiffness shape functions may be unknown or be altogether missing.

Problem (a) can be addressed by constructing "rigid mass elements" accounting for inertia (and possibly gravity or centrifugal forces) but no stiffness. Nodes of these elements must be linked to structural (elastic)

Matriz de Massa Consistente

nodes by MFC constraints that enforce kinematic constraints. This is more of an implementation issue than a research topic, although numerical difficulties typical of rigid body dynamics may crop up.

Problems (b,c,d) can be attacked by parametrization. The father of NASTRAN, Dick MacNeal, was the first to observe [87,90] that averaging the DLMM and CMM of the 2-node bar element produced better results than using either alone. This idea was further studied by Belytschko and Mullen [163] using Fourier analysis. Krieg and Key [171] had emphasized that in transient analysis the introduction of a time discretization operator brings new compensation phenomena, and consequently the time integrator and the mass matrix should be not be chosen separately.

A good discussion of mass diagonalization schemes can be found in the textbook by Cook et al. [28].

The template approach addresses the problems by allowing and encouraging full customization of the mass to the problem at hand. It was first described in [47,48] for a Bernoulli-Euler plane beam using Fourier methods. It is presented in more generality in the following Chapter, where it is applied to other elements. The general concept of template as parametrized form of FEM matrices is discussed in [46].

Matriz de Massa: Combinação entre a *Lumped* e a Consistente

ELSEVIER

Computers & Structures

Volume 46, Issue 6, 17 March 1993, Pages 1041-1048

A review of mass matrices for eigenproblems

Ki-ook Kim

⊞ **Show more**

https://doi.org/10.1016/0045-7949(93)90090-Z

Get rights and content

Abstract

Various nonconsistent mass matrices have been presented to achieve more accurate natural frequencies in eigenproblems of the finite element analysis. The matrices are obtained as a linear combination of lumped and consistent mass matrices. For an improved accuracy, the consistent mass should be more weighted than the lumped mass. Instead of the mass combination, the interpolation functions can be combined to give nonconsistent mass matrices, which show the same tendency. To find a nonsingular lumped mass matrix for the bending vibration of beams, a translational inertia has been proposed for rotational degrees of freedom. The inertia effect is highly overestimated and hence lower natural frequencies are obtained. When combined with the consistent mass matrix, however, the modified lumped mass matrix gives a significant improvement for the natural frequencies of intermediate and higher modes. A simple corrective method was applied to get a better estimation of the natural frequencies through the use of the frequency dependent stiffness and mass matrices. The method shows high accuracy without complicated calculations.

Respostas Dinâmicas

Matriz de Massa Global:

Como se faz para a matriz de rigidez **sistema global**:

$$[M]^e_{6x6} = \left[[R]^T\right]^e \cdot [m]^e \cdot [R]^e$$

Matriz de Massa Global: Endereçamento idem.

$$[Mest]_{3nnx3nn} = \sum_{e=1}^{ne} [M]^e_{6x6}$$

Elemento ne:
Nó inicial → j
Nó final → k

Nó Global	Posição Local	Posição Global
j	1	3·j-2
	2	3·j-1
	3	3·j
k	4	3·k-2
	5	3·k-1
	6	3·k

Respostas Dinâmicas

As equações para o movimento então pode ser representadas matricialmente, após inserir as condições de contorno da estrutura, e são dadas por:

$$[K]\cdot\{U\}+[M]\cdot\{\ddot{U}\}=\{F\}$$

Onde as condições de contorno podem ser inseridas nas matrizes [K] e [M] da maneira usual, por exemplo, pela técnica de “zeros” e “uns”, não **usar penalidade.**

Análise de Vibração Livre:

As equações para o movimento livre, após inserir as condições de contorno da estrutura, são dadas por:

$$[K]\cdot\{U\}+[M]\cdot\{\ddot{U}\}=\{0\}$$

Considerando movimento harmônico: $U=\phi\cdot sen(\omega\cdot t)$

Com ϕ e ω sendo, respectivamente, os modos fundamentais de vibração e suas frequências naturais.

De modo que se leva a resolução do auto-problema:

$$\left([K]-\omega^2\cdot[M]\right)\cdot\phi=\{0\}$$

Como ϕ é não-trivial, assim:

Análise de Vibração Livre:

Como ϕ é não-trivial, assim: $\left|[K]-\omega^2\cdot[M]\right|=0$

E tomando: $\omega^2=\lambda$

Chega-se a equação característica: $\left|[K]-\lambda\cdot[M]\right|=0$

E λ são os auto-valores dessa equação. A estrutura tem seu modo de vibração, ϕ, associado a cada um *dos auto-valores,* λ :

Existem vários algoritmos para resolver esse auto-problema na literatura.

Análise de Vibração Livre:

- Os diferentes tipos de matriz de massa levam a diferenças suaves no cálculo dos auto-valores;

- Não há indicação definitiva de qual matriz é melhor para quaisquer problema em estruturas;

- Com o uso dessa matriz lumped (“amontoadora”), a convergência para a solução exata dos auto-valores acontece por baixo;

- Os auto-valores obtidos com o uso da matriz de massa consistente levam a valores maiores que os exatos. Esse tipo de matriz leva respostas mais precisas para problemas de flexão.

Análise de Vibração Livre:
Rotinas para cálculo de auto-problemas:

Lapack: DSYGV()

```
*  =========== DOCUMENTATION ===========
*
* Online html documentation available at
*            http://www.netlib.org/lapack/explore-html/
*
*> \htmlonly
*> Download DSYGV + dependencies
*> <a href="http://www.netlib.org/cgi-bin/netlibfiles.tgz?format=tgz&filename=/lapack/lapack_routine/dsygv.f">
*> [TGZ]</a>
*> <a href="http://www.netlib.org/cgi-bin/netlibfiles.zip?format=zip&filename=/lapack/lapack_routine/dsygv.f">
*> [ZIP]</a>
*> <a href="http://www.netlib.org/cgi-bin/netlibfiles.txt?format=txt&filename=/lapack/lapack_routine/dsygv.f">
*> [TXT]</a>
*> \endhtmlonly
*
*  Definition:
*  ===========
*
*       SUBROUTINE DSYGV( ITYPE, JOBZ, UPLO, N, A, LDA, B, LDB, W, WORK,
*                         LWORK, INFO )
*
*       .. Scalar Arguments ..
*       CHARACTER          JOBZ, UPLO
*       INTEGER            INFO, ITYPE, LDA, LDB, LWORK, N
*       ..
*       .. Array Arguments ..
*       DOUBLE PRECISION   A( LDA, * ), B( LDB, * ), W( * ), WORK( * )
*       ..
*
```

Análise de Vibração Livre:
Rotinas para cálculo de auto-problemas:

Lapack: DSYGV()

```
*> DSYGV computes all the eigenvalues, and optionally, the eigenvectors
*> of a real generalized symmetric-definite eigenproblem, of the form
*> A*x=(lambda)*B*x,  A*Bx=(lambda)*x,  or B*A*x=(lambda)*x.
*> Here A and B are assumed to be symmetric and B is also
*> positive definite.
*> \endverbatim
*
*  Arguments:
*  ==========
*
*> \param[in] ITYPE
*> \verbatim
*>          ITYPE is INTEGER
*>          Specifies the problem type to be solved:
*>          = 1:  A*x = (lambda)*B*x
*>          = 2:  A*B*x = (lambda)*x
*>          = 3:  B*A*x = (lambda)*x
*> \endverbatim
*>
*> \param[in] JOBZ
*> \verbatim
*>          JOBZ is CHARACTER*1
*>          = 'N':  Compute eigenvalues only;
*>          = 'V':  Compute eigenvalues and eigenvectors.
*> \endverbatim
*>
*> \param[in] UPLO
*> \verbatim
*>          UPLO is CHARACTER*1
*>          = 'U':  Upper triangles of A and B are stored;
```

Análise de Vibração Livre:

Rotinas para cálculo de auto-problemas:

Lapack: DSYGV()

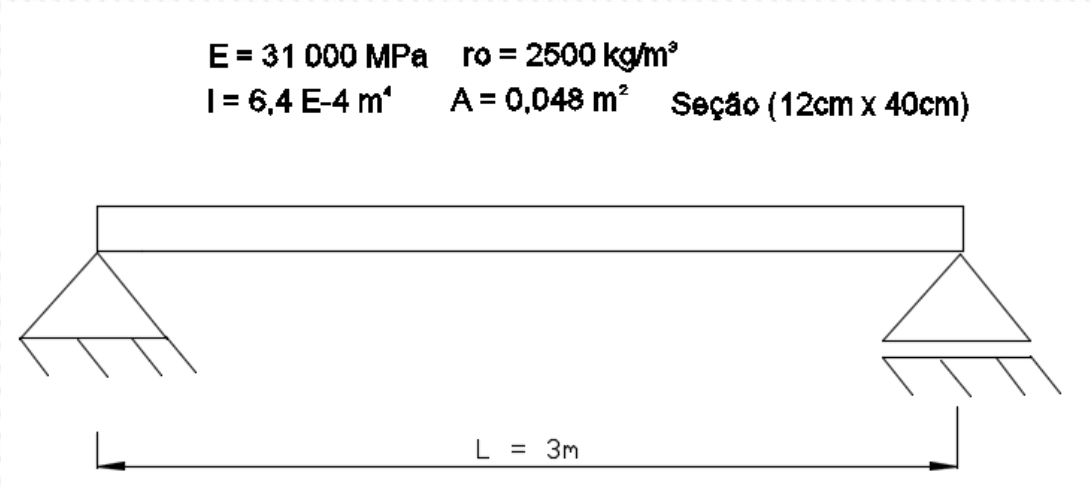
```
      SUBROUTINE AUTOPROBLEMA(A,M,N,IQ,ITER,RESFREQ)
! input: A e M: matrizes de rigidez e massa da estrutura, iq aarq a ser impresso auto-respostas
! output: A[,]: contem os autovetores, vetor da coluna i é o desloc da freq natural i; M[]:  contem os autovalores na 1a. coluna
! arquivo fort.55 imprime autovalores
      Real(8) A(N,N),M(N,N),RESFREQ(10),AUX
      Real(8), Allocatable :: AUTOVALOR(:), WORK(:)
      CHARACTER*1 JOBZ,UPLO
      INTEGER N
      ALLOCATE(AUTOVALOR(N))
      ! MATRIZES TEM QUE SER SIMETRICAS!
      ITYPE = 1   ! = 1:  A*x = (lambda)*B*x
      IF(ITER.EQ.1)JOBZ = 'V'  ! 'V'   = Compute eigenvalues and eigenvectors.
      IF(ITER.NE.1)JOBZ = 'N'  ! 'N'   = Compute only eigenvalues.
      UPLO = 'U'  ! Upper triangles of A and B are stored;  N =  The order of the matrices A and B.  N >= 0.
      LDA = N
      LDB = N
      INFO = 0
      LWORK = (N+2)*N
      ALLOCATE(WORK(LWORK))
      WORK = 0.D0
      !LWORK is INTEGER, The length of the array WORK.  LWORK >= max(1,3*N-1). For optimal efficiency, LWORK >= (NB+2)*N,
                        ! where NB is the blocksize for DSYTRD returned by ILAENV.
      ! AUTOVALOR is DOUBLE PRECISION array, dimension (N),  If INFO = 0, the eigenvalues in ascending order.
      CALL DSYGV(ITYPE,JOBZ,UPLO,N,A,LDA,M,LDB,AUTOVALOR,WORK,LWORK,INFO) ! dsygv.for
!OUT: INFO is INTEGER: = 0:  successful exit; < 0:  if INFO = -i, the i-th argument had an illegal value;
                  ! > 0:  DPOTRF or DSYEV returned an error code:;   <= N:  if INFO = i, DSYEV failed to converge i off-diagonal elements of an intermediate
                  ! > tridiagonal form did not converge to zero;
      IO = 0
      DO I =1,N
          AUTOVALOR(I) = DSQRT(AUTOVALOR(I))
          M(I,1) = AUTOVALOR(I)
          IF(M(I,1).GT.1.AND.IO.EQ.0)THEN
              RESFREQ(3) = I
              IO = 1
          ENDIF
      ENDDO
      IF(IQ.NE.0)CALL PRINT_AUTOVALOR(AUTOVALOR,N,IQ,ITER)
```

Análise de Vibração Livre: Comparar MEF com Solução analítica de uma viga bi-apoiada

Solução analítica de uma viga bi-apoiada:

$$\omega_n = \frac{n^2\pi^2}{L^2}\sqrt{\frac{EI}{\rho A}} \rightarrow n = 1,2...$$

E: Modulo de Young em N/m2;
ρ: massa especifica em kg/m3



Solução analítica de uma viga bi-apoiada:

$$\omega_n = \frac{n^2\pi^2}{L^2}\sqrt{\frac{EI}{\rho A}} \rightarrow n = 1,2...$$

E = 31E9 N/m^2 ; L = 3 m; I = 6,4E-4 m^4 ; A = 0,048 m^2 , ρ = 2500 kg/m^3

ω_1 = 445,9 rad/s ou $f_1 = \omega_1/(2\pi)$ = 70,97 Hz
ω_2 = 1.783,6 rad/s ou $f_2 = \omega_2/(2\pi)$ = 283,9 Hz

Matriz Consistente

MEF: 2 EF:

	Freq(rad/s)	Freq(Hz)
AUTOVALOR(1)=	570.80	90.85
AUTOVALOR(2)=	1891.46	301.04
AUTOVALOR(3)=	1979.65	315.07
AUTOVALOR(4)=	3902.52	621.11
AUTOVALOR(5)=	6607.61	1051.63
AUTOVALOR(6)=	9071.89	1443.84

MEF: 10 EF:

	Freq(rad/s)	Freq(Hz)
AUTOVALOR(1)=	1.00	0.16
AUTOVALOR(2)=	451.36	71.84
AUTOVALOR(3)=	1845.68	293.75
AUTOVALOR(4)=	1871.38	297.84
AUTOVALOR(5)=	4457.16	709.38
AUTOVALOR(6)=	5582.66	888.51
AUTOVALOR(7)=	8507.34	1353.99
AUTOVALOR(8)=	9457.30	1505.18
AUTOVALOR(9)=	13562.16	2158.48

Solução analítica de uma viga bi-apoiada:

E = 31E9 N/m^2 ; L = 3 m; I = 6,4E-4 m^4 ; A = 0,048 m^2 , ρ = 2500 kg/m^3

ω_1 = 445,9 rad/s ou $f_1 = \omega_1/(2\pi)$ = 70,97 Hz
ω_2 = 1.783,6 rad/s ou $f_2 = \omega_2/(2\pi)$ = 283,9 Hz

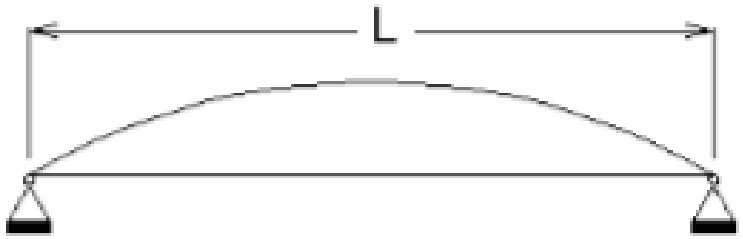
Matriz Lumped

MEF: 2 EF:

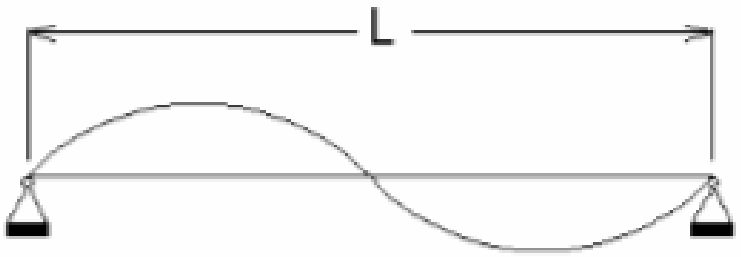
	Freq(rad/s)	Freq(Hz)
AUTOVALOR(1)=	401.99	63.98
AUTOVALOR(2)=	1209.77	192.54
AUTOVALOR(3)=	1796.76	285.96
AUTOVALOR(4)=	1883.95	299.84
AUTOVALOR(5)=	2095.37	333.49
AUTOVALOR(6)=	4337.75	690.38

MEF: 10 EF:

	Freq(rad/s)	Freq(Hz)
AUTOVALOR(1)=	440.89	70.17
AUTOVALOR(2)=	1707.01	271.68
AUTOVALOR(3)=	1841.89	293.15
AUTOVALOR(4)=	3650.90	581.06
AUTOVALOR(5)=	5480.31	872.22
AUTOVALOR(6)=	6079.71	967.62
AUTOVALOR(7)=	8790.46	1399.05
AUTOVALOR(8)=	8983.78	1429.81
AUTOVALOR(9)=	11582.15	1843.36
AUTOVALOR(10)=	12266.05	1952.20



primeiro modo



segundo modo