ALGEMA - SISTEMA COMPUTACIONAL PARA ALGEBRA DE MATRIZES

CARLOS TADEU DOS SANTOS DIAS

Orientador : Prof. Dr. CASSIO ROBERTO DE MELO GODOI

Dissertação apresentada à Escola
Superior de Agricultura "Luiz de
Queiroz" da Universidade de São
Paulo, para obtenção do título
de Mestre em Agronomia, Área de
Concentração : Estatística e
Experimentação Agronômica.

PIRACICABA

Estado de São Paulo - Brasil
- 1 9 8 8 -

A meus pais

Ubaldo e Francisca,

Dedico

A meus irmãos

Francisco, Ana, Silene e Verônica,

Ofereço

AGRADECIMENTOS

Expressamos os nossos agradecimentos a todos aqueles que contribuíram, direta ou indiretamente para a realização do presente trabalho e de maneira especial às seguintes pessoas e instituições:

- Ao Professor Dr. Cássio Roberto de Mello Godoi, pela orientação e ensinamentos recebidos, durante o desenvolvimento deste trabalho, bem como pela sua amizade.

- Aos Professores e funcionários do Departamento de Matemática e Estatística da ESALQ/USP pelos ensinamentos adquiridos e gentilezas concedidas, respectivamente.

- A todo pessoal do Centro de Informática na Agricultura (CIAGRI), em especial a Adriano Azevedo, Eder Polizel, José Celso, Gladys Pierri, Ariovaldo de Carvalho, Gabriel Sariés, Sérgio Roberto e Carlos Henrique pela colaboração e troca de idéias.

- A Professora Dra. Sheila Zambello de Pinho pelas críticas e sugestões.

- Aos colegas do Departamento de Bioestatística da UNESP-Botucatu pela oportunidade concedida.

- Ao Professor Roberto Rus Perez, do Departamento de Educação UNESP-Botucatu pela revisão do vernáculo.

- A Maria Célia Rodrigues, pelo constante incentivo nos momentos difíceis.

- A Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA), pela oportunidade de realização do curso.

- Ao CNPQ e CAPES pelo auxílio financeiro concedido.

INDICE

	Página
RESUMO	viii
SUMMARY	x
1 INTRODUÇÃO	1
2 REVISÃO DA LITERATURA	3
3 MÉTODO	8
3.1 MÓDULO ALGEDADO	10
3.2 MÓDULO ALGEGUAR	14
3.3 MÓDULO ALGECHAM	17
3.4 MÓDULO ALGETRAN	18
3.5 MÓDULO ALGEPROD	20
3.6 MÓDULO ALGEIMPRE	23
3.7 MÓDULO ALGEPRESC	25
3.8 MÓDULO ALGEINVE	26
3.9 MÓDULO ALGEOPER	31
3.10 MÓDULO ALGEPART	40
3.11 MÓDULO ALGEAPAG	42
3.12 MÓDULO ALGECONT	43
3.13 MÓDULO ALGEDIRE	44
3.14 MÓDULO ALGEBIBLI	45
3.15 MÓDULO ALGEAUTO	49
3.16 MÓDULO ALGEDIAG	69
3.17 MÓDULO ALGENORMA	71
3.18 MÓDULO ALGEORTO	72
3.19 MÓDULO ALGESWEEP	76

3.20 LINGUAGEM PASCAL	78
4 EXEMPLOS DE APLICAÇÃO.....	80
5 COMENTARIOS FINAIS	137
6 REFERENCIAS BIBLIOGRAFICAS	139
7 ANEXO (LISTAGEM DO SISTEMA ALGEMA)	144

ALGEMA - SISTEMA COMPUTACIONAL PARA ALGEBRA DE MATRIZES**Autor : CARLOS TADEU DOS SANTOS DIAS****Orientador : Prof. Dr. Cassio R. de M. Godoi****RESUMO**

Pretende-se com este trabalho dar subsídios àqueles profissionais que necessitem desenvolver trabalhos que envolvam operações com matrizes, em particular, aos estudantes de cursos de Pós-Graduação em Estatística, como auxílio aos princípios envolvidos em disciplinas como Álgebra de Matrizes, Modelos Lineares, Análise Multivariada e Regressão e Covariância.

O sistema foi desenvolvido para a linha de computadores PC - XT em linguagem Pascal, comportando-se em um disquete de 5 1/4".

No desenvolvimento do trabalho foram abordadas as seguintes operações matriciais.

- a) Transposição ;
- b) Soma e Subtração ;
- c) Produto de Matrizes ;
- d) Produto por Escalar ;
- e) Inversão de Matriz pelo Método Gauss - Jordan;
- f) Diagonalização por Congruência ;
- g) Ortonormalização por Gram - Schmidt ;
- h) Normalização;
- i) Autovalores e Autovetores para matriz simétrica pelo Método QR ;
- j) Operador "Sweep" ;

Além dessas operações tradicionais o sistema apresenta opções para:

- a) Entrar com dados via teclado ou arquivo;
- b) Correção de dados;
- c) Guardar na memória principal seis matrizes;
- d) Recuperar matrizes guardadas na memória principal;
- e) Operar com linhas e colunas;
- f) Particionar matrizes;
- g) Fornecer norma de matrizes;
- h) Fazer teste de simetria;
- i) Mostrar o conteúdo das matrizes armazenadas na memória principal;
- j) Imprimir dados no vídeo ou impressora;
- k) Apagar arquivo de dados;
- l) Mostrar o conteúdo do diretório;

O sistema foi testado pelos alunos do curso de doutorado em Estatística ESALQ/USP na disciplina Análise Multivariada permitindo que se estabelecessem correções experimentalmente. Também foram feitas comparações com outros sistemas, apresentando uma boa performance. Alunos do departamento de estatística e de genética têm feito uso do ALGEMA para resolver problemas dados em aula e em trabalhos como seminários, dissertações e teses. O autor tem feito uso do mesmo em seu local de trabalho para atendimento à comunidade.

ALGEMA - COMPUTATIONAL SYSTEM FOR MATRIX ALGEBRA

Author : Carlos Tadeu dos Santos Dias

Adviser: Prof. Dr. Cássio Roberto de Melo Godoi

SUMMARY

The objective of this work is to supply professionals involved in matrix algebra manipulations with a useful tool. In particular to the students from statistical areas at graduate level as an auxiliary tool to the principles involved in courses like Matrix Algebra, Linear Models, Multivariate Analysis, Multiple Regression, etc.

The ALGEMA system was developed for the IBM-PC line of computers in the Turbo Pascal compiler. One 5 1/4" flexible disk is capable to maintain the whole system.

The following matrix operations were treated in the work development:

- (a) Transposition;
- (b) Sum and Subtraction;
- (c) Product of Matrices;
- (d) Product by a scalar;
- (e) Gauss-Jordan Matrix Inversion;
- (f) Congruence Diagonalization;
- (g) Gram-Schmidt orthonormalization;
- (h) Columns normalization;
- (i) Eigenvalues and eigenvectors by QR method;

(j) "Sweep" operator.

Furthermore, the system presents option like:

- (a) Data input by console or disk;
- (b) Data correction;
- (c) Six main memories to store matrix;
- (d) Call for matrices from the six memories;
- (e) Lines and columns elementary operations;
- (f) Matrix partition;
- (g) Matrix norms;
- (h) Matrix symmetry test;
- (i) To show memories contents;
- (j) Screen or Printer output;
- (k) File erasing;
- (l) Showing of directory contents.

1 INTRODUÇÃO

Uma das principais ferramentas do estatístico é a álgebra de matrizes que, com suas operações, permite obter importantes informações.

Os conhecimentos teóricos de matrizes muitas vezes não são bem assimilados, em virtude de não se poder verificar algumas passagens ou realizar cálculos mais complicados que, executados à mão, seriam bastante laboriosos e desencorajadores. O que se deseja é um trabalho que associe a álgebra de matrizes com computação de forma a ajudar e estimular o estudante nos seus estudos e trabalhos.

O sistema é de simples operação e poderá ser de ampla utilidade para cursos de Pós-Graduação em Estatística, como auxílio aos princípios envolvidos em disciplinas como Álgebra de Matrizes, Modelos Lineares, Análise Multivariada e Regressão e Covariância. Mas espera-se que seja útil a outros profissionais.

Os principais objetivos deste trabalho são :

- 1) Dar subsídio àqueles profissionais que necessitem desenvolver trabalhos que envolvam operações com matrizes;
- 2) Estimular os estudantes em cursos que envolvam álgebra matricial.

Os principais motivos que levaram a realização deste trabalho foram os seguintes :

- 1) A necessidade de independência de pacotes estrangeiros, muitas vezes caros;

- 2) A não existência de "software" específico para área de estatística no que se refere a matrizes;
- 3) Contribuir para a melhoria do ensino em nosso país.

2 REVISAO DA LITERATURA

Kublanovskaya (1961) e Francis (1962) citados por Flanders (1984) publicaram independentemente um método iterativo para cálculo de autovalores e autovetores, conhecido como QR. Trata-se de um dos melhores e mais rápidos algoritmos para essa finalidade até o momento.

Poole e Borchers (1984) desenvolveram programas para adição, subtração e multiplicação escalar de matrizes, além de multiplicação de duas matrizes e inversão de matriz quadrada e não singular. São programas sem nenhuma ligação entre si.

Loiola (1985) desenvolveu rotinas para adição, subtração e multiplicação de duas matrizes, transposição e inversão de matriz quadrada não-singular, adição da linha de uma matriz à linha de outra, produto da linha de uma matriz por um escalar e adição a outra linha da mesma matriz, produto da coluna de uma matriz por um escalar e adição a outra coluna da mesma matriz, troca de linhas e troca de colunas, soma dos elementos das colunas formando vetor linha e, soma dos elementos das linhas formando vetor coluna. Esse trabalho mostra um pouco mais de criatividade, contudo, sem chegar ao ponto de interligação lógica das rotinas.

STATGRAPHICS (Statistical Graphics System) apresenta uma rotina para cálculo de autovalores e autovetores para matrizes reais simétricas, outra para multiplicação de duas matrizes e outra para inverter uma matriz quadrada não singular.

SAEG (Sistema de Análise Estatística e Genética) apresenta rotina para inversão de matriz não-singular.

SAS (Statistical Analysis System) apresenta rotinas para:

- Fazer a soma de duas matrizes.
- Trocar sinal dos elementos de uma matriz.
- Criar um vetor cujos elementos subsequentes são, a partir de certo valor definido, incrementados de uma unidade.
- Fazer produto de duas matrizes e produto de um escalar por uma matriz.
- Fazer o produto de duas matrizes, onde os elementos da matriz resultante são o produto dos correspondentes elementos das duas matrizes, permitindo ainda fazer o produto por escalar.
- Fazer o produto de Kronecker de duas matrizes (produto direto).
- Fazer a divisão de cada elemento de uma matriz pelo correspondente elemento de outra matriz, produzindo uma matriz de quocientes.
- Elevar uma matriz à uma potência inteira, maior ou igual a -1.
- Criar uma nova matriz cujos elementos são os elementos de uma matriz elevados à potência dos correspondentes elementos de outra matriz. Se algum valor na primeira matriz é negativo, o correspondente elemento na segunda matriz deve ser um inteiro.
- Fazer a comparação de duas matrizes, elemento por elemento, e produzir uma nova matriz que contém somente zeros e uns. Se a comparação entre os elementos é verdadeira, o correspondente elemento na nova matriz é 1, caso contrário é zero.

- Comparar cada elemento de uma matriz com o correspondente elemento da outra matriz. O maior dos dois valores será o elemento correspondente da nova matriz. Há uma outra rotina que faz a mesma coisa, somente que tomando o menor valor dos dois elementos comparados.
- Concatenar horizontalmente duas matrizes de mesma dimensão linha. Do mesmo modo, há uma outra que concatena verticalmente duas matrizes de mesma dimensão coluna.
- Comparar duas matrizes, elemento por elemento, para produzir uma nova matriz. Se os correspondentes elementos em ambas são nulos, o elemento da nova matriz é 1, caso contrário, é zero. Há uma outra que gera uma nova matriz em que se faz o inverso do descrito acima.
- Fazer o exame de cada elemento de uma matriz e produzir uma nova matriz cujos elementos são zeros e uns. Se um elemento da matriz é igual a zero, o correspondente elemento na nova matriz é 1. Se um elemento da matriz é não nulo, o correspondente elemento na nova matriz é zero.
- Transpor matriz.
- Produzir uma nova matriz cujos elementos são os valores absolutos de outra.
- Produzir uma matriz bloco-diagonal a partir de duas outras.
- Produzir uma matriz cujos elementos são a soma acumulada até a posição do elemento.
- Gerar uma matriz de delineamento a partir de um vetor definido pelo usuário. Cada valor desse vetor se refere à coluna da

matriz de delineamento. Essa coluna contém uns (1) nas posições cujo elemento correspondente no vetor geral são correntes, caso contrário são zeros.

- Calcular o determinante de uma matriz quadrada. O método usado é o produto dos autovalores da matriz.
- Criar uma matriz diagonal cujos elementos são os correspondentes elementos da diagonal de uma matriz quadrada.
- Calcular autovalor e autovetor de matriz simétrica. O método usado é o QL (Wilkinson e Reinsch, 1971, citado no SAS, 1985).
- Calcular a inversa de Moore-Penrose. Essa inversa é calculada usando a decomposição em valores singulares (Wilkinson e Reinsch, 1971; citado no SAS, 1985).
- Calcular a inversa regular de uma matriz quadrada e não singular além de uma outra que resolve sistema de equações simultâneas. O método usado foi a decomposição LU (Forsythe e Moler, 1967, citado no SAS, 1985).
- Gerar uma matriz com todos valores iguais.
- Permitir inserir uma matriz em outra desde que haja compatibilidade de linha ou coluna.
- Gerar uma matriz identidade.
- Decomposição de Cholesky de uma matriz simétrica e definida não negativa. A matriz resultante é triangular superior.
- Calcular a decomposição em valores singulares de uma matriz.
- Fazer a operação "sweep" (Goodnight, 1979).

Trata-se de um bom sistema, porém, seu uso em forma de comandos o torna menos popular.

Paniago et alii (1987) desenvolveram o SOC ("Software" Científico) que faz entre outras operações: soma, produto e cálculo de autovalores e autovetores. Segue a mesma linha do SAS.

3 METODO

O sistema dispõe de 21 módulos, sendo cada módulo composto por um ou mais algoritmos. Embora o sistema tenha sido desenvolvido para operações matriciais, trabalhou-se com variáveis tipo vetor.

No corpo principal do sistema, declara-se o número total de elementos das variáveis globais tipo vetor que é de 8.100, equivalente à uma matriz de 90 linhas por 90 colunas. Declara-se também o vetor A denominado Memória de Trabalho, no qual é realizada a maioria dos cálculos e cinco vetores M_1, M_2, M_3, M_4, M_5 , denominados memórias fixas onde guardam-se matrizes na memória principal do microcomputador.

O sistema apresenta um menu principal com várias opções, sendo algumas destas desdobradas em "sub-menus". Executada uma opção, o sistema retorna ao "menu" principal.

ALGEBRA DE MATRIZES

Menu

⟨A⟩ Entrar com dados	⟨L⟩ Operar linha / coluna
⟨B⟩ Corrigir dados	⟨M⟩ Particionar matriz
⟨C⟩ Guardar matriz	⟨N⟩ Autovetores - Autovalores
⟨D⟩ Recuperar matriz	⟨O⟩ Operador sweep
⟨E⟩ Transpor	⟨P⟩ Norma de matrizes
⟨F⟩ Produto de matrizes	⟨Q⟩ Teste de simetria
⟨G⟩ Produto por escalar	⟨R⟩ Conteudo memoria
⟨H⟩ Inverter matriz	⟨S⟩ Imprimir dados
⟨I⟩ Diagonalizar (congruencia)	⟨T⟩ Apagar arquivo
⟨J⟩ Ortonormalizar (Gram-Schmidt)	⟨U⟩ Mostrar directorio
⟨K⟩ Normalizar	⟨X⟩ Finalizar

A seguir descreve-se cada módulo.

3.1 MÓDULO ALGEDADO

3.1.1 Objetivo

A função desse módulo é realizar a entrada de dados e permitir que se façam correções dos mesmos.

3.1.2 Descrição

Os dados podem ser digitados via teclado ou via arquivo tipo texto.

3.1.2.1 Via teclado

A entrada de dados apresenta-se de forma a facilitar o trabalho do usuário, permitindo que se entre com toda ou com parte da matriz. Para tanto, há um sub-menu que solicita ao usuário informação sobre o tipo de matriz : Simétrica , Diagonal ou qualquer.

3.1.2.2 Via arquivo texto

Os dados devem estar previamente digitados em um componente de memória auxiliar, por exemplo, disquete flexível ou rígido. Ao sistema deve ser fornecido o drive onde se encontra o arquivo de dados, dois pontos e o nome do arquivo da mesma forma que foi criado, incluindo ou não, conforme o caso, a extensão.

No caso de se entrar com o nome errado do arquivo, o sistema emite uma mensagem de erro :

Arquivo <nome do arquivo> inexistente.

O usuário poderá verificar o nome correto, bastando para isso que solicite a opção "<U> Mostrar diretório" do Menu Principal, informando o drive onde está o arquivo. Assim, será impresso na

tela o nome completo de todos os arquivos daquele diretório.

Em ambos os casos, após terem sido incorporados os dados na memória principal, o usuário poderá fazer correções, bastando para isso que solicite a opção " Corrigir dados" do Menu Principal. Por ter-se trabalhado com linguagem compilada, que exige o pré-dimensionamento de conjuntos, esse módulo emite mensagem de erro quando as dimensões da matriz superam o tamanho do conjunto pré-dimensionado.

A matriz de dados pode ser simétrica, assimétrica ou diagonal. Nos dois primeiros tipos, o usuário deve entrar com os elementos por linha da matriz. Por exemplo:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ & a_{22} & a_{23} & \dots & a_{2n} \\ & & a_{33} & \dots & a_{3n} \\ & & & \dots & \\ & & & & a_{nn} \end{bmatrix}$$

SIMETRICA

$$B = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

ASSIMETRICA

No caso da matriz A , entra-se com os elementos por linha, sendo que o elemento da posição simétrica é alocado via sistema. Portanto, entra-se com os elementos a partir da diagonal principal para a direita da linha.

Se a matriz é assimétrica entra-se com todos os elementos também por linha.

No caso de matriz diagonal, é suficiente que se entre apenas com os elementos da diagonal, deixando a cargo do sistema a alocação dos elementos nulos fora da diagonal principal.

$$\begin{bmatrix}
 a_{11} & 0 & 0 & \dots & 0 \\
 & a_{22} & 0 & \dots & 0 \\
 & & a_{33} & \dots & 0 \\
 & & & \dots & \\
 & (SIM.) & & & a_{nn}
 \end{bmatrix}$$

DIAGONAL

3.1.3 Correções

O sistema apresenta ao usuário uma tela com as opções:

CORREÇÃO DE MATRIZ ...:

- 1 - Simétrica
- 2 - Assimétrica
- 3 - Nenhuma das opções

Escolhidas as opções 1 ou 2, o sistema solicita o índice da linha e coluna do elemento a ser corrigido, fornecendo na tela o valor atual e solicita ao usuário que entre com o valor corrigido. No caso da opção 1, não é necessário corrigir o elemento da posição simétrica; isso é feito automaticamente pelo sistema. Sem sair deste "loop", pode-se fazer tantas correções quantas forem necessárias. Em ambos os casos, se o usuário entrar com índices maiores que a dimensão da matriz de dados, o sistema emitirá mensagem de erro. O usuário tem a opção de voltar ao Menu Principal em duas circunstâncias: quando não desejar fazer mais correções, bastando para isso que se tecle o "N" de não, ao surgir a pergunta "Mais correções ? (S/N)" e quando solicitada a opção 3.

3.2 MÓDULO ALGEGUAR

3.2.1 Objetivo

Esse módulo tem por função armazenar matrizes que entraram via programa ou via arquivo padrão ASCII ou que foi operada através dos procedimentos existentes no sistema.

3.2.2 Descrição

Uma Matriz de dimensões dentro dos limites, pode ser armazenada em uma das cinco memórias fixas ou em arquivo. Uma vez lida a matriz de entrada ela fica na memória de trabalho e será destruída, se qualquer opção de operação matricial for solicitada. Para preservação dessa matriz deve-se guardá-la numa das cinco memórias principais. Portanto, uma vez solicitada a opção < C > "Guardar matriz" do menu principal, surgirá na tela:

GUARDAR ...:

- 1 - Memória principal
- 2 - Arquivo
- 3 - Nenhuma das opções

Escolhida a opção 1, surgem na tela as opções de guardar a matriz em cinco diferentes posições como a seguir.

GUARDAR ...:

- 1 -----
- 2 -----
- 3 -----
- 4 -----
- 5 -----
- 6 Nenhuma das opções

Os itens 1,2,3,4 e 5 representam locais da memória principal onde serão guardadas as matrizes por todo o tempo de processamento (memórias fixas), a não ser que se destrua a matriz guardada, com outra matriz ou que se guarde somando nessa posição. Se a opção 6 for solicitada, o sistema retorna ao Menu Principal sem guardar a matriz da memória de trabalho, porém ela permanece inalterada. Definida uma memória fixa, surge na tela a pergunta

Guardar somando ? (S/N),

em que o usuário deverá teclar ("S" ou "s") ou ("N" ou "n") quando se desejar guardar a matriz da memória de trabalho, somando com a que está na memória fixa ou não respectivamente. É nesse ponto do sistema que se poderá fazer soma ou diferença de matrizes. A soma de $A + B$, de mesmas dimensões com qualquer uma delas em uma memória fixa e a outra na memória de trabalho, é feita solicitando guardar na mesma posição de memória fixa e teclando "S" ou "s" quando surgir a pergunta acima. A diferença entre A e B é feita com A ou B em uma memória fixa e a outra, multiplicada por (-1) , somando nessa memória.

Se as duas matrizes a somar forem de dimensões distintas, o sistema emite a mensagem "Dimensões incompatíveis".

Escolhida a opção 2 (guardar em arquivo) o usuário deverá informar o drive (a, b, c etc.), colocar dois pontos (:) e o nome a se dar à matriz. Por exemplo :

b : X'X.DAD

Em seguida é solicitado o número de casas decimais

e inteiras desejadas para o arquivamento da matriz. Esse módulo prevê alguns erros como no caso do usuário dar o nome de um arquivo já existente. Para tanto o sistema emitirá a mensagem

"Arquivo <nome do arquivo> Existente",

indicando a existência de arquivo com mesmo nome, voltando em seguida ao "menu" principal. Caso o usuário deseje guardar com o nome do arquivo já existente, ele poderá solicitar a opção "<T> Apagar Arquivo" do "menu" principal e após ter apagado solicita a opção "G.- Guardar matriz" guardando com o nome desejado. São previstos alguns erros de entrada de dados como disco cheio, diretório cheio, tamanho do arquivo maior que o máximo permitido. O sistema retorna ao "menu" principal depois de guardar a matriz.

3.3 MÓDULO ALGECHAM

3.3.1 Objetivo

A função deste módulo é realizar a operação de recuperar uma matriz da memória fixa para a memória de trabalho.

3.3.2 Descrição

Uma vez que se tenha guardado uma matriz em alguma das memórias fixas, pode-se acessá-la através desse módulo. O sistema emite um Menu com o conteúdo atual das memórias fixas, permitindo ao usuário ver em qual posição de memória está a matriz desejada. O usuário tem a opção de voltar ao Menu Principal caso desista de chamar alguma matriz.

O sistema tem defesa (i.é., evita o aborto do processamento do sistema) no caso de se solicitar uma matriz de uma memória vazia. Neste caso ele emite a mensagem

"Conteúdo inexistente"

Após a operação de chamar a matriz ter se completado o sistema retorna ao Menu Principal.

Essa rotina utiliza as seguintes variáveis:

L - número de linhas da matriz a ser chamada;

C - número de colunas da matriz a ser chamada.

Como resultado, a rotina fornece a matriz que se deseja de dimensões $L \times C$.

3.4 MÓDULO ALGETRAN

3.4.1 Objetivo

Este módulo tem por função realizar a operação tradicional de transposição.

3.4.2 Descrição

Com uma matriz de dimensões permitidas na memória de trabalho, pode-se solicitar do Menu Principal a opção "<E> Transpor", que consiste em alterar a posição relativa das linhas e colunas de uma matriz cujas linhas e colunas são respectivamente iguais às colunas e linhas da matriz original.

Exemplo:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

$$A' = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}$$

Esta rotina utiliza as seguintes variáveis :

L - número de linhas da matriz

C - número de colunas da matriz

A - matriz A a ser transposta de dimensões L x C
L C

Como resultado, a "procedure" retorna ao "menu" principal tendo na memória de trabalho a transposta da matriz original de dimensões C x L.

3.5 MÓDULO ALGEPROD

3.5.1 Objetivo

A função deste módulo é realizar o produto entre duas matrizes.

3.5.2 Descrição

Considere-se duas matrizes localizadas em duas das memórias: fixas ou de trabalho. A primeira possui dimensões $L \times C$ e a segunda $C \times M$, ou seja, necessariamente o número de colunas da primeira deve ser igual ao número de linhas da segunda.

Escolhida a opção "<F> Produto de matrizes", surge na tela um menu com o conteúdo das memórias fixas e da memória de trabalho para lembrar ao usuário que matrizes estão ali guardadas.

```

1 - A
2 - B
3 - .....
4 - .....
5 - .....
6 - Memória de Trabalho.
PRODUTO DE =>
COM =>

```

Em seguida aparece a frase "Produto de =>" onde o usuário deverá indicar o número da memória onde está a matriz que pré-

multiplica. Depois surge a palavra "COM =>" onde se deve indicar a matriz que pós-multiplica. Deve-se planejar as operações antes de entrar no sistema para poupar o uso abusivo das memórias fixas e para forçar mais o uso da memória de trabalho. Caso as dimensões das matrizes não sejam compatíveis para o produto, o sistema emite a mensagem "Produto incompatível" voltando ao Menu Principal após teclada a chave <ENTER>. Assim, tem-se a liberdade de averiguar o porquê da incompatibilidade, bastando para isso que se solicite a opção "<R> Conteúdo da memória" onde são mostradas as dimensões das matrizes.

Memória	Nome	Linha	Coluna
1	A	4	3
2	A'	3	4
3		0	0
4		0	0
5		0	0
6	MEMÓRIA DE TRABALHO	3	4

3.5.3 Descrição de Regra de Processamento

Quando se deseja obter o produto entre duas matrizes, o número de colunas da primeira matriz deve ser igual ao número de linhas da segunda, obtendo-se uma matriz resultante com o mesmo número de linhas da primeira e o mesmo número de colunas da segunda, ou seja:

$$\begin{matrix} A & B & = & R \\ L & C & M & L & M \end{matrix}$$

Sendo cada elemento da matriz resultante, definido por:

$$r_{ij} = \sum_{k=1}^C a_{ik} b_{kj}$$

Essa rotina utiliza as seguintes variáveis:

L - número de linhas da matriz A;

C - número de colunas da matriz A, que é igual ao número de linhas da matriz B;

M - número de colunas da matriz B;

$\begin{matrix} A & e & B \\ L & C & C & M \end{matrix}$ - matrizes A e B a serem multiplicadas,

possuindo respectivamente dimensões

L x C e C x M;

Como resultado, a rotina fornece a matriz R de dimensões L x M. Este resultado retorna, ao Menu Principal na memória de trabalho.

3.6 MÓDULO ALGEIMPRE

3.6.1 Objetivo

Esse módulo tem por função realizar a impressão no vídeo ou na impressora de qualquer matriz que esteja na memória de trabalho.

3.6.2 Descrição

Matriz de qualquer dimensão, desde que esteja na memória de trabalho, pode ser impressa por essa rotina. Solicitada a opção "5 - Imprimir dados", surge no vídeo um sub-menu com as opções de impressão:

IMPRESSAO ...:

1 - Vídeo

2 - Impressora

Caso solicite-se a opção 1, surgirá na tela a primeira coluna da matriz. Para se verem as demais linhas e colunas tem-se as seguintes opções:

<CTRL D> ou "->"	---> Avança uma coluna
<CTRL S> ou "<-"	---> Regride uma coluna
<CTRL C> ou "PG DN"	---> Avança quinze linhas
<CTRL R> ou "PG UP"	---> Regride quinze linhas
<CTRL X> ou "↓"	---> Avança uma linha
<CTRL E> ou "↑"	---> Regride uma linha
<CTRL F>	---> Avança cinco colunas
<CTRL A>	---> Regride cinco colunas

Para retornar ao Menu Principal basta que se digite <ENTER>. Caso solicite-se a opção 2, o sistema pedirá ao usuário um nome que identifique os dados, o número de casas decimais e o número de casas inteiras com que os elementos da matriz deverão ser impressos. Se por algum motivo deseja-se interromper a impressão, basta que se pressione <ESC>, retornando-se ao "menu" principal. Os elementos são impressos por linha da matriz sem indicação de posição relativa na matriz.

Exemplo:

A matriz

$$A = \begin{bmatrix} 40 & -30 \\ 20 & 325,5 \end{bmatrix}$$

será assim impressa:

Matriz A

40.0000	-30.0000
20.0000	325.5000

3.7 MÓDULO ALGEPRESC

3.7.1 Objetivo

A função deste módulo é fazer o produto de uma matriz por um escalar.

3.7.2 Descrição

Com uma matriz de quaisquer dimensões na memória de trabalho, pode-se solicitar a opção do Menu Principal " <G> Produto (divisão) por constante", no qual pede-se ao usuário que digite a constante. A operação é efetuada e, a seguir, retorna ao Menu Principal com a matriz na memória de trabalho já alterada.

As seguintes variáveis são fornecidas à rotina em questão:

L - número de linhas da matriz

C - número de colunas da matriz

k - valor do escalar

A - matriz cujos elementos serão alterados por
L C

k.

3.8 MÓDULO ALGEINVE

3.8.1 Objetivo

A função dessa rotina é realizar a inversão generalizada pelo método (modificado) de Gauss - Jordan.

3.8.2 Descrição

3.8.2.1 Inversão de matriz quadrada

Sendo A uma matriz quadrada diz-se que a sua inversa generalizada é A^{-} se possuir a seguinte propriedade:

$$AA^{-}A = A$$

Para realizar a inversão, utiliza-se da matriz identidade I de idêntica ordem à da matriz da memória de trabalho. O processo inicia-se procurando o maior elemento da primeira coluna em valor absoluto e guardando o número da linha onde ele se situa. Caso o maior (em módulo) elemento estiver fora da diagonal principal, faz-se a troca de linhas tanto em A como em I de tal forma que o maior elemento de A fique na posição diagonal (pivô). Se não existir, passa-se à próxima coluna e a partir da diagonal repete-se o descrito acima.

Posto na posição diagonal o maior elemento em módulo, divide-se a linha i da matriz A pelo pivô a partir da coluna i . A mesma operação é feita na matriz I só que em toda a linha i .

Logo após, é feita a redução a zero da coluna i

abaixo da diagonal, segundo a formulação abaixo:

$$a_{kj} = a_{kj} + a_{ij} (-a_{kj}) \quad \text{para } k \neq i$$

e repete-se a operação em toda a linha i , a linha $(i-1)$ fica inalterada. A mesma operação é feita na matriz I (identidade), sem que necessariamente se zere a posição abaixo da diagonal. Este processo tem continuidade com as linhas e colunas subsequentes, até que em A se tenha a forma hermitiana da matriz original e nesse momento a matriz I se transforma na inversa generalizada.

3.8.2.1.1 Considerações sobre o método apresentado

a) Durante a utilização deste método, são feitos os cálculos de determinante, traço e posto da matriz original, à medida que se obtém a inversa.

O determinante é obtido através da formulação:

$$\det(A) = \prod_{i=1}^n a_{ii} \times S$$

onde:

$\det(A)$ - é o determinante da matriz original;

$\prod_{i=1}^n a_{ii}$ - é o produtório dos elementos da diagonal principal antes da divisão pelo pivô.

S - é o sinal associado ao número de trocas de linhas, isto é, a cada troca de linha o determinante muda de sinal.

O posto é obtido contando-se o número de elementos da posição diagonal diferente de zero.

O traço é obtido pela definição, qual seja :

$$\text{tr}(A) = \sum_{i=1}^E a_{ii} ,$$

onde E é o número de colunas.

b) Quando ocorre pivô nulo, então passa-se para a próxima linha.

Essa rotina utiliza as seguintes variáveis :

L - número de linhas da matriz

A - matriz A a ser invertida, de dimensões L x L

Como resultado a rotina fornece a matriz A^{-1} de dimensões L x L. Se a matriz é singular obtém-se a inversa generalizada que goza da propriedade $AA^{-1}A = A$. Caso seja não singular obtém-se a inversa comum que goza da propriedade

$$AA^{-1} = I.$$

3.8.2.2 Inversão de matriz retangular

Para esse tipo de inversão há dois casos:

a) Número de linhas é maior que o de colunas
($L > C$)

b) número de linhas é menor que o de colunas
($L < C$)

Em ambos os casos, completa-se a matriz com zeros, a fim de torná-la quadrada. Feito isso, entra-se na rotina para inversão de matriz quadrada como descrito anteriormente.

Exemplo:

caso a) $L > C$:

$$\begin{array}{c}
 \begin{array}{cccc|cccc}
 a & a & \dots & a & 0 & 0 & \dots & 0 \\
 11 & 12 & & 1C & & & & \\
 a & a & \dots & a & 0 & 0 & \dots & 0 \\
 21 & 22 & & 2C & & & & \\
 \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
 a & a & \dots & a & 0 & 0 & \dots & 0 \\
 L1 & L2 & & LC & & & &
 \end{array} \\
 L \quad \quad \quad L
 \end{array}$$

caso b) $L < C$:

3.9 MÓDULO ALGEOPER

3.9.1 Objetivo

A função deste módulo é realizar algumas operações especiais úteis quando se opera com matrizes. As operações são :

- Troca de linhas;
- Troca de colunas;
- Produto de linhas de uma matriz por escalares e adição a outra linha da mesma matriz;
- Produto de colunas de uma matriz por escalares e adição a outra coluna da mesma matriz;
- Eliminação de linhas;
- Eliminação de colunas;
- Divisão de linha por escalar;
- Divisão de coluna por escalar;
- Justaposição de matrizes de mesma dimensão linha;
- Justaposição de matrizes de mesma dimensão coluna.

3.9.2 Descrição

Cada uma das operações a ser realizada necessita de um determinado tipo de entrada, como se vê adiante:

3.9.2.1 Troca de linhas

Dada uma matriz A , entra-se com $l1$ e $l2$, os
L C

índices das linhas a serem trocadas, ter-se-á como resultado da operação uma nova matriz, idêntica à matriz A, com exceção das posições das linhas l1 e l2. Após executada a troca de linhas o sistema retorna ao Menu Principal.

Essa rotina utiliza as seguintes variáveis:

L - número de linhas da matriz;

C - número de colunas da matriz;

A - matriz A de dimensões L x C;
L C

l1, l2 - número das linhas a serem trocadas;

No caso do usuário solicitar uma linha fora das dimensões da matriz localizada na memória de trabalho, o sistema voltará a solicitar as novas linhas sem avisar ao usuário o porquê.

3.9.2.2 Troca de colunas

Dada uma matriz A, entra-se com c1 e c2 os
L C

índices das colunas a serem trocadas, teremos como resultado da operação, uma nova matriz idêntica à matriz A, a menos das posições das colunas c1 e c2.

No caso do usuário solicitar uma coluna fora das dimensões da matriz da memória de trabalho, o sistema voltará a solicitar as novas dimensões sem avisar o usuário o porquê.

Essa rotina utiliza as seguintes variáveis:

L - número de linhas da matriz

C - número de colunas da matriz

A - matriz A possuindo dimensões L x C
L C

c1,c2 - número das colunas a serem trocadas

Como resultado a rotina fornece uma matriz de dimensões L x C, idêntica à matriz A, à exceção das colunas trocadas.

3.9.2.3 Produto de linhas por escalares e adição a uma linha da mesma matriz

Seja uma matriz de dimensões permitidas, dois escalares, o índice da linha que se deseja multiplicar pelo primeiro escalar e o índice da outra linha que se deseja multiplicar pelo segundo escalar e na qual se substituirá o resultado da operação.

Dada uma matriz A na memória de trabalho, o
L C

sistema solicita que se defina dois escalares e_1 e e_2 , e as linhas

l1 e l2 as quais devem ser operadas. Ter-se-á como resultado da operação uma nova matriz R idêntica a A, a menos da linha l2 definida como a seguir:

$$r_{l2,j} = e_1 a_{l1,j} + e_2 a_{l2,j}, \quad j = 1, 2, \dots, C$$

onde C é o número de colunas da matriz.

Antes da execução da operação é apresentado na tela

o resumo da operação e a pergunta se confirma ou não. Caso o usuário não confirme será solicitado nova operação, não sendo executada a anterior. Essa rotina utiliza as seguintes variáveis

L - número de linhas da matriz;

C - número de colunas da matriz;

e₁ - valor do primeiro escalar;

e₂ - valor do segundo escalar;

l1 - número da linha que será multiplicada pelo primeiro escalar;

l2 - número da linha que será multiplicada pelo segundo escalar e que receberá o resultado da operação.

A_{L C} - a matriz A possuindo dimensões L x C.

Após executada a operação, o sistema retorna ao "menu" principal.

3.9.2.4 Produto de colunas por escalares e adição a uma coluna da mesma matriz

A descrição é análoga a 3.9.2.3 substituindo-se linha por coluna.

3.9.2.5 Eliminação de linhas

Seja uma matriz de dimensões permitidas e o número

da linha a eliminar (l). Com uma matriz A na memória de $L \times C$ trabalho, a rotina solicita que se defina o índice da linha a eliminar. Ter-se-á como resultado da operação uma matriz R idêntica a A , a menos da linha l .

Depois de executada a operação de eliminação de linha, tem-se a opção de eliminar mais linhas, caso desejado.

Esta rotina utiliza as seguintes variáveis:

L - número de linhas da matriz;

C - número de colunas da matriz;

A - a matriz A possuindo dimensões $L \times C$;
 $L \times C$

l - número da linha a eliminar.

Como resultado, a rotina fornece uma matriz de dimensões $(L - 1) \times C$, ou seja diminui uma linha a cada eliminação que for solicitada dentro da rotina.

Após executada a operação e não se deseje mais eliminação de linhas, o sistema retorna ao Menu Principal.

3.9.2.6 Eliminação de colunas

As mesmas considerações, feitas para eliminação de linhas, são válidas para colunas.

3.9.2.7 Divisão de linha por escalar

Seja uma matriz de dimensões permitidas, o número

da linha e o escalar.

Com uma matriz A na memória de trabalho, a
 $L \ C$

rotina solicita que se entre com a linha que se quer dividir e em seguida o escalar. O escalar não pode ser nulo e caso se entre com esse valor, o sistema emite mensagem de erro, permitindo que se entre novamente com a linha e escalar. No caso de se desejar sair dessa rotina sem alterar a matriz da memória de trabalho basta que se divida qualquer linha pela unidade (1). Em caso de se solicitar um número de linha diferente do que existe, o sistema emite a mensagem "Linha fora dos limites".

Essa rotina utiliza as seguintes variáveis:

L - número de linhas da matriz;

C - número de colunas da matriz;

A - a matriz A possuindo dimensões $L \times C$;
 $L \ C$

l - número da linha a dividir;

e - escalar pelo qual será dividida a linha l .

Como resultado tem-se uma matriz de dimensões $L \times C$ idêntica a A com exceção da linha l .

Após a execução da operação e não se deseje mais eliminação de linha, o sistema retorna ao Menu Principal.

3.9.2.8 Divisão de coluna por escalar

As mesmas considerações, feitas para divisão de linhas, são válidas para colunas.

3.9.2.9 Justaposição de matrizes de mesmo número de linhas

Com duas matrizes de mesmo número de linhas, forma-se uma matriz por justaposição.

Dada uma matriz A na memória de trabalho, o
 $L \quad C$

sistema solicita que se indique com qual matriz da memória fixa deseja-se fazer a justaposição. Em dois casos o sistema emite mensagem de erro: no caso em que o número de elementos total das duas matrizes ultrapassa o número pré-fixado para a matriz da memória de trabalho e no caso de dimensões linhas incompatíveis.

Ter-se-á como resultado da justaposição uma nova matriz com o mesmo número de linhas L , mas com o número de colunas igual à soma dos números de colunas das duas que foram justapostas.

Essa rotina utiliza-se das seguintes variáveis:

L - número de linhas da matriz localizada na memória de trabalho;

C - número de colunas da matriz localizada na memória de trabalho;

$L1$ - número de linhas da matriz localizada na memória fixa e igual a L .

$C1$ - número de colunas da matriz localizada na memória fixa;

A - a matriz possuindo dimensões $L \times C$ localizada
 $L \quad C$

na memória de trabalho;

B - a matriz possuindo dimensões $L1 \times C1$

localizada em uma das memórias fixas.

Como resultado a rotina fornece a matriz de dimensões $L \times (C + C1)$ que é a justaposição de A com B , ficando o resultado na memória de trabalho. O bloco inicial da nova matriz é o que estava na memória de trabalho.

Exemplo:

Sejam as matrizes

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1k} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2k} \\ \dots & \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & b_{m3} & \dots & b_{mk} \end{bmatrix}$$

dará como resultado da justaposição a matriz

$$R = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_{11} & b_{12} & b_{13} & \dots & b_{1k} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_{21} & b_{22} & b_{23} & \dots & b_{2k} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} & b_{m1} & b_{m2} & b_{m3} & \dots & b_{mk} \end{bmatrix}$$

3.9.2.10 Justaposição de matrizes de mesmo número de colunas

Segue a mesma explicação de 3.9.2.9 para coluna.

3.10 MÓDULO ALGEPART

3.10.1 Objetivo

A função deste módulo é a obtenção de uma submatriz

3.10.2 Descrição

Sejam uma matriz de quaisquer dimensões, os índices iniciais e finais das linhas e das colunas delimitado o bloco que se deseja obter da matriz original. Ter-se-á uma matriz R de dimensões $(L2 - L1 + 1) \times (C2 - C1 + 1)$ na memória de trabalho.

Casos como índice de linha fora do intervalo $[1, L]$, índice de coluna fora do intervalo $[1, C]$, índice de linha invertido, índice de coluna invertido são evitados no sistema e em todos os casos emite-se uma mensagem de erro.

Essa rotina utiliza as seguintes variáveis:

L - número de linhas da matriz A;

C - número de colunas da matriz A;

l1 - número da linha inicial de A onde deve iniciar a partição;

l2 - número da linha de A onde deve finalizar a partição;

c1 - número da coluna inicial de A onde deve iniciar a partição;

c2 - número da coluna final de A onde deve finalizar a partição.

Como resultado, a rotina fornece a matriz R de dimensões $(L2 - L1 + 1) \times (C2 - C1 + 1)$, destruindo A .

Exemplo:

Seja a matriz a seguir e $l1 = 2$, $l2 = 3$, $c1 = 3$, $c2$

$= n$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

a matriz resultante será

$$R = \begin{bmatrix} a_{23} & \dots & a_{2n} \\ a_{33} & \dots & a_{3n} \end{bmatrix}$$

3.11 MÓDULO ALGEAPAG

3.11.1 Objetivo

A função desse módulo é apagar um arquivo de dados gravado em disco flexível ou rígido.

3.11.2 Descrição

O sistema solicita que o usuário entre com o drive (a,b,c), dois pontos (:) e o nome do arquivo a apagar incluindo extensão caso exista. A seguir surge a pergunta "Confirma? (S/N)". No caso de confirmar o arquivo será apagado retornando em seguida ao Menu Principal. Caso contrário, não será apagado o arquivo e o sistema retornará ao Menu Principal. Caso se entre com o nome errado surgirá na tela a mensagem de arquivo inexistente.

Exemplo: Suponha-se que no drive A exista o arquivo MATB.DAD, mas não exista o arquivo MATX.DAD.

Entre drive, (:) e nome do arquivo a apagar ...: A:MATB.DAD

Após digitar <ENTER> surgirá a frase "Confirma? (S/N)". Se for teclada a "S", o sistema apagará o arquivo MATB.DAD e retornará ao menu principal.

Entre drive, (:) e nome do arquivo a apagar ...: A:MATX.DAD

ARQUIVO "MATX.DAD" INEXISTENTE

Tecla <ENTER> Para continuar ...

3.12 MÓDULO ALGECONT

3.12.1 Objetivo

A função desse módulo é mostrar na tela o conteúdo das memórias fixas e da memória de trabalho.

3.12.2 Descrição

Ao pedir a opção (R), surge na tela uma tabela indicando o número da memória, o nome que se deu à matriz, o número de linhas e colunas desta. Para retornar ao Menu Principal digitar (ENTER). Esta rotina poderá ser útil quando se desejar saber as dimensões e identificar as matrizes que se deseja operar.

=====					
=	Memória	Nome	Linha - Coluna		=
=====					
=	1	A	3	3	=
=	2	A11	2	2	=
=	3	A12	2	1	=
=	4	A21	1	2	=
=	5	A22	1	1	=
=	6	Memória de Trabalho	1	1	=
=====					

Tecle (ENTER) para continuar ...

3.13 MÓDULO ALGEDIRE

3.13.1 Objetivo

A função desse módulo é mostrar todos os arquivos do diretório de um disquete ou de um sub-diretório do disco rígido.

3.13.2 Descrição

Para utilização, basta que se indique o drive, (:) e o nome do arquivo.

Diretório do drive B:*.*

```
-----  
ALGEMA.COM      ALGEOPER.PAS    ALGEPART.PAS    ALGEIMPR.PAS  
ALGEBIBL.PAS    ALGEAPAG.PAS    ALGENORM.PAS    ALGEDIRE.PAS  
ALGETRAN.PAS    ALGEPROD.PAS    TURBO.COM        TURBO.MSG  
ALGEGUAR.PAS    ALGESWEE.PAS    ALGEMA.PAS       ALGEDIAS.PAS  
ALGEORTO.PAS    ALGECHAM.PAS    ALGECONT.PAS     TURBOBCD.COM  
ALGEPRES.PAS    ALGESIME.PAS    ALGEMA.000       ALGEAUTO.PAS  
ALGEINVE.PAS    ALGECHAM.PAS    ALGECONT.PAS     ALGEDADO.PAS  
-----
```

16384 Bytes livres

Tecla <ENTER> para continuar ...

3.14 MÓDULO ALGEBIBLI

3.14.1 Objetivo

A função deste módulo é declarar algumas rotinas que são utilizadas frequentemente na execução do sistema ou que são utilizadas de uma maneira global. São as seguintes as rotinas usadas neste módulo:

- Rotina Limpatela;

Tem por função limpar a tela entre duas posições-linhas definidas.

- Rotina Diagrama;

Permite que se imprima na tela qualquer conjunto de caracteres numa determinada posição linha-coluna.

- Rotina Cabeçalho;

Tem a função de imprimir a frase com o nome do sistema ALGEMA - algebra de matrizes.

- Rotina Teclenter;

Tem a função de parar a execução do sistema nos casos de emissão de mensagem no video. Para continuar a execução é necessário que se tecele a chave <ENTER> do teclado.

- Função Número;

Tem a função de proteger a entrada de dados

numéricos, ou seja só aceitar números reais ou inteiros. Assim, na entrada de dados numéricos se entrar com letras o sistema não aceita.

- Rotina Limpalinha;

Tem a função de limpar a linha a partir de determinada coluna da tela até a última. Dentro das rotinas que a usam são passados parâmetros que dão as coordenadas do início da linha na tela.

- Rotina Simetria;

A função dessa rotina é realizar o teste de simetria de matriz, teste este que pode ser solicitado através do Menu Principal ou ser chamado por rotinas que só operam matrizes simétricas como é o caso da rotina que calcula autovalores e autovetores.

Ela testa também se a matriz é nula, evitando-se que matriz nula seja operada em várias partes do sistema. Se for solicitada através do Menu Principal, surgirá no vídeo a mensagem de ocorrência ou não de simetria.

A matriz a ser verificada deve estar na memória de trabalho e o teste é definido como segue.

Se $|a_{ij} - a_{ji}| > 0,00001$ a $|a_{ji}| > 0,00001$,

para $i = 1, 2, \dots, L$ e $j = 1, 2, \dots, C$ onde $L = C$
for verdadeiro para pelo menos um par i, j ,
então a matriz é assimétrica.

- Rotina Normaliza;

A função dessa rotina é realizar a normalização das colunas de matrizes. Com uma matriz de quaisquer dimensões na memória de trabalho, pode-se solicitar a normalização de suas colunas através da seguinte metodologia: Primeiro é feita a soma de quadrados dos elementos da coluna, a seguir cada elemento da coluna é dividido pela raiz quadrada dessa soma; o resultado desta operação é, então, substituído na posição correspondente. Em resumo, temos:

Passo 1:

$$S = \sum_{i=1}^L a_{ij}^2, \text{ para } j \text{ fixo}$$

Passo 2:

$$a_{kj} := a_{kj} / \sqrt{S}, \quad k = 1, 2, \dots, L \text{ e } j \text{ fixo.}$$

Após os passos acima repetirem-se para todas as colunas, ter-se-á como resultado uma matriz de mesma dimensão, porém com suas colunas normalizadas.

3.15 MÓDULO ALGEAUTO

3.15.1 Objetivo

Os métodos que são apresentados na literatura para determinação de autovalores de uma matriz são para avaliar uma equação polinomial e calcular suas raízes. Há duas sérias objeções para esse procedimento quando se determinam autovalores de matriz em computadores digitais. Uma delas refere-se à ineficiência do processo se a ordem da matriz é maior que quatro ou cinco. A outra é que ele é mal-condicionado em um sentido semelhante ao definido em conexão com equações lineares simultâneas. Esses resultados são inexatos devido aos erros de arredondamento, ainda que os autovalores sejam bem definidos pela matriz dada.

A função desse módulo é realizar a determinação de autovalores e autovetores de uma matriz simétrica, pelo método QR.

3.15.2 Descrição

É necessário apenas ter-se uma matriz simétrica A e sua dimensão, sendo que A deve estar na memória de trabalho.

3.15.3 Metodologia

Antes da discussão sobre as regras de processamento de cada rotina, far-se-á uma discussão geral do método QR.

No presente momento esse é o método mais eficiente e largamente usado para o cálculo dos autovalores de uma matriz. O método foi primeiramente publicado em 1961 por J. G.F. Francis, citado por Flanders (1984) e, tem sido alvo de intensa pesquisa.

Dada uma matriz A , pelo processo de ortonormalização de Gram-Schmidt, A pode ser fatorada em $A = QR$, com R triangular superior ($t_{ij} = 0$ para $i > j$) e Q ortogonal. Seja $A_1 = A$, e defina-se uma sequência de matrizes A_m , Q_m e R_m por:

$$A_m = Q_m R_m \quad (1)$$

e

$$A_{m+1} = R_m Q_m, \quad m = 1, 2, \dots$$

A sequência A_m convergirá ou para uma matriz triangular com os autovalores de A na sua diagonal ou uma matriz quase triangular da qual os autovalores podem ser facilmente convergentes. Uma técnica será introduzida e ilustrada mais adiante para acelerar este processo.

Algumas propriedades de (1) podem ser derivadas. De (1),

$$R_m = Q_m' A_m$$

e então

$$A_{m+1} = Q_m' A_m Q_m \quad (2)$$

Portanto, A_{m+1} é ortogonalmente similar a A e por indução a

$A_1 = A$. Dai os autovalores de A_{m+1} também o são de A .

Se o par $\tilde{v}, \lambda$ são características para A_{m+1} tal que

$$A_{m+1} \tilde{v} = \lambda \tilde{v}, \text{ então}$$

$$\lambda \tilde{v} = A_{m+1} \tilde{v} = (Q' A Q)_m \tilde{v} = Q' A (Q \tilde{v})$$

então

$$A (Q \tilde{v}) = \lambda (Q \tilde{v})$$

Assim $Q \tilde{v}, \lambda$ são os pares característicos de A_m . Com estas

explanções, pode-se investigar vetores característicos através deste processo.

Note que se A_1 é simétrica, $A_2, A_3, \dots$ também o

são. Realmente, temos por hipótese que

$$A'_1 = (Q_1 R_1)' = Q_1 R_1$$

Dai

$$R'_1 Q'_1 = Q_1 R_1,$$

$Q'_1 R'_1 Q'_1 = Q'_1 Q_1 R_1 = R_1$, devido à ortogonalidade de Q .

Então $Q'_1 R'_1 = R_1 Q_1$, isto é $A'_2 = A_2$.

O método QR pode ser relativamente demorado porque a fatorização QR gasta tempo quando repetida muitas vezes. Para diminuir o gasto de tempo a matriz é preparada para o método QR por redução a uma forma simples e única para que a fatorização QR seja mais rápida. Para tanto, suponha-se A_1 simétrica e tridiagonal. Para provar isto, há um lema que diz que o produto de uma matriz triangular e uma matriz de Hessenberg é outra Hessenberg. Note-se que

$$Q_1 = A_1 R_1^{-1} = A_1 \begin{bmatrix} \text{triangular} \\ 0 \end{bmatrix} = \begin{bmatrix} \text{triangular} \\ 0 \end{bmatrix} \quad \text{Hessenberg}$$

Q_1 tem zeros abaixo da subdiagonal. Daí,

$$A_2 = R_1 Q_1 = \begin{bmatrix} \text{triangular} \\ 0 \end{bmatrix} \begin{bmatrix} \text{triangular} \\ 0 \end{bmatrix} = \begin{bmatrix} \text{triangular} & 0 \\ 0 & \text{triangular} \end{bmatrix}$$

mas A_2 é simétrica, daí tridiagonal.

Finalmente, precisa-se de uma via eficiente para executar a fatoração

$$A = QR \quad \text{de } A,$$

onde A é simétrica e tridiagonal. Isso é obtido por uma sequência de rotações no plano. Por exemplo, seja

$$U' = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \oplus I_{n-2}$$

$$U' = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 & 0 & \dots & 0 \\ -\sin \theta & \cos \theta & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Se o bloco 2×2 superior em A é

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

então $U'A$ será:

$$\begin{bmatrix} a \cos \theta + b \sin \theta & b \cos \theta + c \sin \theta & \dots \\ -a \sin \theta + b \cos \theta & -b \sin \theta + c \cos \theta & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

$U'A$ tem o elemento $2,1$

$$b_{12} = -a \sin \theta + b \cos \theta$$

que pode ser anulado pela seleção

$$\cos \theta = \frac{a}{\sqrt{a^2 + b^2}} \quad \sin \theta = \frac{b}{\sqrt{a^2 + b^2}}$$

Neste caso, pode-se pré-multiplicar A pela matriz de rotação e gradualmente mudá-la para a forma triangular superior.

O algoritmo QR é geralmente aplicado com uma substituição a princípio para os autovalores ordenados com o propósito de incrementar a rapidez da convergência. Para uma sequência de constantes $\{c_m\}$, define-se

$$A_1 = A \quad e$$

$$A_m - c_m I = Q_m R_m$$

$$A_{m+1} = c_m I + R_m Q_m, \quad m = 1, 2, \dots$$

As matrizes A_m são semelhantes a A_1 , desde que

$$R_m = Q_m' (A_m - c_m I)$$

$$A_{m+1} = c_m I + Q_m' (A_m - c_m I) Q_m$$

$$= c_m I_m + Q_m' A_m Q_m - c_m I_m$$

$$A_{m+1} = Q_m' A_m Q_m \quad m \geq 1$$

Os autovalores de A_{m+1} são os mesmos que os de A_m ,

e daí os mesmos de A . Para ser mais específico na escolha da substituição $\{c_m\}$, consideremos somente uma matriz simétrica

tridiagonal A . Seja

$$A_m = \begin{bmatrix} a_1^{(m)} & b_1^{(m)} & 0 & \dots & 0 \\ b_1^{(m)} & a_2^{(m)} & b_2^{(m)} & \dots & 0 \\ 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & b_{n-1}^{(m)} \\ 0 & \cdot & \cdot & \cdot & b_{n-1}^{(m)} & a_{n-1}^{(m)} \end{bmatrix}$$

Há dois métodos pelos quais $\{c_m\}$ é escolhido

$$(1) \quad c_m = a_n^{(m)}$$

$$(2) \quad c_m \text{ é o autovalor de}$$

$$\begin{bmatrix} (m) & & \\ a & & (m) \\ n-1 & & b \\ & & n-1 \end{bmatrix} \begin{bmatrix} (m) & & \\ b & & (m) \\ & & a \\ & & n \end{bmatrix} (*)$$

o qual é próximo a $a_{n-1}^{(m)}$. A segunda escolha é preferida, mas em qualquer dos casos a matriz A_m converge para uma matriz bloco diagonal na qual os blocos tem ordem 1 ou 2.

Qualquer seleção de $\{c\}$ garante

$$\begin{bmatrix} (m) & (m) \\ b & b \\ n-1 & n-2 \end{bmatrix} \rightarrow 0 \text{ quando } m \rightarrow \infty$$

geralmente em uma razão mais rápida que o método QR original. Da

expressão acima temos que qualquer dos dois $b_{n-1}^{(m)} \rightarrow 0$ ou

$b_{n-2}^{(m)} \rightarrow 0$ quando $m \rightarrow \infty$. No caso (1), $a_n^{(m)}$ converge para

um autovalor de A , e no caso (2) os dois autovalores podem facilmente ser extraídos da submatriz (*).

Uma vez que um ou dois autovalores tem sido obtido

devido a $b_{n-1}^{(m)}$ ou $b_{n-2}^{(m)}$ essencialmente zero, a matriz A_m

pode ser reduzida na ordem por uma ou duas linhas, respectivamente. Seguindo este raciocínio, o método QR com substituição pode ser aplicado para a matriz reduzida. A escolha das substituições faz com que a convergência a zero seja mais

rápida para $b_{n-1}^{(m)}$ do que para os elementos remanescentes fora da diagonal da matriz. Neste caso, o método QR torna-se um método

rápido, mais que qualquer outro método até o presente momento.

3.15.4 Descrição das Sub-rotinas

O presente módulo inicia-se com as declarações básicas de constantes, tipos e variáveis. Os tipos MATSPD, MATCS incluem uma componente extra com índice 0 (zero). Isto evita testes extras posteriores para iniciar e finalizar os "loops".

Somente a parte triangular da matriz A é usada. A matriz U é ortogonal, e consequentemente suas colunas serão vetores característicos de A, correspondendo às raízes RT[1], ..., RT[N]. Após A ser reduzida à forma tridiagonal, somente sua diagonal DG e diagonal superior SPD são utilizadas.

Veja-se agora a redução de Householder. As matrizes de entrada são: A simétrica e U identidade. As de saída são a matriz ortogonal U^* tal que U^*AU^* é (simétrica) tridiagonal com diagonal DG e diagonal superior SPD. A primeira subprocedure, STEP, é o primeiro passo da redução de Householder, isto é, formam as $(i + 1)$ -ésimas linhas através da n-ésima linha de HAH' onde H tem a forma

$$I - 2\underset{\sim}{v}\underset{\sim}{v}' \quad \text{com } \underset{\sim}{v}^{(k)}_{\sim{i}} = 0$$

para $i < k$, e $i > k + 1$, $1 \leq k \leq n - 1$

Agora

$$\begin{aligned} HAH' &= (I - 2\underset{\sim}{v}\underset{\sim}{v}')A(I - 2\underset{\sim}{v}\underset{\sim}{v}') \\ &= A - \underset{\sim}{v}(2\underset{\sim}{v}'A) - 2A\underset{\sim}{v}\underset{\sim}{v}' + (4\underset{\sim}{v}'A\underset{\sim}{v})\underset{\sim}{v}\underset{\sim}{v}' \end{aligned}$$

Fazendo-se

$$\underline{p} = 2A\underline{v} \text{ e } z = 2\underline{v}'A\underline{v} = 2\underline{v}'\underline{p},$$

então

$$\begin{aligned} HAH' &= A - \underline{v}\underline{p}' - \underline{p}\underline{v}' + z\underline{v}\underline{v}' \\ &= A - \underline{v}\underline{p}' - (\underline{p} - z\underline{v})\underline{v}' \end{aligned}$$

Este é precisamente o cálculo na rotina STEP. A i -ésima linha é precisamente $[a_{ii}, c, 0, 0, \dots, 0]$ que é calculada no corpo da rotina TRIDI.

A segunda rotina, UPDATU, troca U por UH' . Agora $H' = H$ e

$$\begin{aligned} UH' &= U(I - 2\underline{v}\underline{v}') \\ &= U - (2U\underline{v})\underline{v}' \\ &= U - \underline{p}\underline{v}' \quad \text{onde } \underline{p} = 2U\underline{v} \end{aligned}$$

A próxima rotina SHIFT, calcula a substituição que é próximo de uma raiz característica. Ela é aplicada a um bloco 2×2 de A . Agora o polinômio característico de

$$\begin{bmatrix} a & b \\ d & c \end{bmatrix}$$

é

$$x^2 - tx + d$$

onde

$$t = a + c$$

é o traço, e

$$d = ac - b^2$$

o determinante.

Pela fórmula quadrática, as raízes são

$$\begin{aligned}\lambda &= -\frac{1}{2} \left(t \pm \sqrt{t^2 - 4d} \right) \\ &= -\frac{1}{2} \left(t \pm \sqrt{(a+c)^2 - 4(ac - b^2)} \right) \\ &= -\frac{1}{2} \left(t \pm \sqrt{(a-c)^2 + 4b^2} \right)\end{aligned}$$

Observa-se que o produto das raízes é $\lambda_1 \lambda_2 = d$.

A quantidade u , que mantém-se da substituição previamente calculada, é usada para decidir se a substituição corrente é uma das raízes ou 0 (zero). Se a raiz característica λ de menor valor absoluto é relativamente próxima a u , então λ é escolhida para a substituição e $u = \lambda$. Se isso não ocorrer, a maior raiz λ^* é relativamente próxima a u , então λ^* é escolhida e $u = \lambda^*$. De outra forma 0 (zero) é escolhido e u é feito igual a menor raiz.

A rotina final é o algoritmo QR, assim chamado devido à escolha original da notação $A = QR$ para fatoração em Q

ortogonal e R triangular superior.

Mas atualmente, Q é um produto

$$Q = Q_1 Q_2 \dots Q_{n-1}$$

de matrizes de rotação elementar. Cada fator elimina um elemento adicional da subdiagonal de A. Escrevendo

$$Q'_{n-1} Q'_{n-2} \dots Q'_1 A = R$$

Então Q'_1 fará uma rotação nas primeiras duas coordenadas. Q

próximo, Q'_2 fará rotacionar a segunda e terceira coordenadas.

Q'_3 a terceira e quarta coordenadas. Gradualmente vai-se mudando A

por estas vias até todos os elementos da subdiagonal serem eliminados e ela tornar-se a triangular superior. Cada multiplicação por Q'_j introduz um elemento na segunda

diagonal, superior e de sobra as sucessivas mudanças em A não são por agora simétricas. Para ver precisamente quais elementos trocam, suporemos que após ter multiplicado à esquerda por Q'_1 ,

Q'_2 até Q'_{j-1} que o bloco diagonal de A até a j-ésima linha é

$$\begin{bmatrix}
 u_1 & w & 0 & 0 & \dots \\
 v_1 & u_2 & v_2 & 0 & \dots \\
 0 & v_2 & u_3 & v_3 & \dots \\
 0 & 0 & v_3 & u_4 & \dots \\
 \dots & \dots & \dots & \dots & \dots
 \end{bmatrix}$$

Para forma Q'_j fazemos

$$z = \sqrt{u_1^2 + v_1^2} \quad C = \frac{u_1}{z} \quad S = \frac{v_1}{z}$$

Então definimos

$$Q'_j = \begin{bmatrix}
 C & S & 0 & 0 & \dots \\
 -S & C & 0 & 0 & \dots \\
 0 & 0 & 1 & 0 & \dots \\
 0 & 0 & 0 & 1 & \dots \\
 \dots & \dots & \dots & \dots & \dots
 \end{bmatrix}$$

temos

$$\begin{bmatrix}
 C_u + S_v & C_w + S_u & S_v & 0 & \dots \\
 & & & & \\
 0 & -S_w + C_u & C_v & 0 & \dots \\
 & & & & \\
 0 & & v_2 & u_3 & v_3 & \dots \\
 & & & & & \\
 0 & & 0 & v_3 & u_4 & \dots \\
 & & & & & \\
 \dots & \dots & \dots & \dots & \dots & \dots
 \end{bmatrix}$$

Observa-se que $C_u + S_v = (u_1^2 + v_1^2) / z = z$. Assim

S_v foi adicionado à j -ésima linha na segunda diagonal superior. Então o próximo bloco diagonal, partindo-se da $(j + 1)$ -ésima linha, tem a mesma forma que o anterior.

O primeiro elemento na diagonal superior foi trocado de v_1 para C_v . A subrotina FACTOR efetua estes cálculos.

O resultado final é R. Necessita-se em adição a diagonal DG e a diagonal superior SPD, outro vetor T2 para conter a segunda diagonal superior de R. Precisa-se guardar todos os senos e cossenos em ordem para formar RQ. A "procedure" FACTOR produz a matriz triangular superior

$$R = \begin{bmatrix} Q' & \dots & Q' \\ & & \\ & & \end{bmatrix} \begin{bmatrix} Q'A \\ & \\ & \end{bmatrix},$$

isto é,

$$A = Q_1 Q_2 \dots Q_{n-1} R.$$

O algoritmo QR precisa produzir

$$A_1 = RQ_1 \dots Q_{n-1}.$$

Ao efetuar a próxima subrotina, MULT, inicialmente calcula-se os produtos sucessivos

$$RQ_1, RQ_1 Q_2, \dots$$

Sabe-se que o resultado final A_1 é simétrica e tridiagonal,

assim pode-se guardá-lo nos dois vetores DG e SPD. Ao mesmo tempo, MULT atualiza a matriz ortogonal U que vem sendo determinada. Assim ela terá finalmente nas colunas vetores característicos da matriz original A.

Deve-se examinar cuidadosamente se se multiplica na esquerda ou na direita, por Q_i ou por Q_i' . Dado A, o primeiro passo é determinar $QAQ' = A_1$ sendo $U = Q'$

onde A_1 é tridiagonal. Então faz-se uma sequência de passos

$$A_1 - \lambda_1 I = Q_1 R_1$$

$$A_2 = R_1 Q_1 + \lambda_1 I$$

$$A_2 - \lambda_2 I = Q_2 R_2$$

$$A_3 = R_2 Q_2 + \lambda_2 I$$

.....

$$A_{r-1} - \lambda_{r-1} I = Q_{r-1} R_{r-1}$$

$$A_r = R_{r-1} Q_{r-1} + \lambda_{r-1} I$$

Espera-se que A_r seja aproximadamente diagonal, assim seus elementos diagonais são as raízes características de A . Para calcular os vetores característicos de A , necessitamos expressar A em termos de A_r . Por indução,

$$A \begin{pmatrix} Q_1 & \dots & Q_{r-1} \end{pmatrix} = \begin{pmatrix} Q_1 & \dots & Q_{r-1} \end{pmatrix} A_r \quad \text{assim,}$$

$$QAQ' \begin{pmatrix} Q_1 & \dots & Q_{r-1} \end{pmatrix} = \begin{pmatrix} Q_1 & \dots & Q_{r-1} \end{pmatrix} A_r$$

Em termos de U isto pode ser escrito

$$A(UQ_1 \dots Q_{r-1}) = (UQ_1 \dots Q_{r-1})A_r$$

Estas equações significam que as colunas de $UQ_1 \dots Q_{r-1}$ são os vetores característicos de A .

Conclusão: Se pré-multiplicar A por um Q' , então precisa pós-multiplicar a U corrente por Q .

Isso mostra que U deve ser atualizado.

A primeira parte de MULT pós-multiplica R . Checando os cálculos no presente estágio, tem-se que:

$$\begin{bmatrix} u_1 & v_1 & w_1 & 0 & 0 \\ t_1 & u_2 & v_2 & w_2 & 0 \\ 0 & 0 & u_3 & v_3 & w_3 \\ 0 & 0 & 0 & u_4 & v_4 \\ 0 & 0 & 0 & 0 & u_5 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & C & -5 & 0 & 0 \\ 0 & 5 & C & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix}
 -u_1 & v_1 C + w_1 S & -v_1 S + w_1 C & 0 & 0 \\
 t_1 & v_2 C + v_2 S & -u_2 S + v_2 C & w_2 & 0 \\
 0 & u_3 S & u_3 C & v_3 & w_3 \\
 0 & 0 & 0 & w_4 & v_4 \\
 0 & 0 & 0 & 0 & u_5
 \end{bmatrix}$$

Quando estiverem acabados os cálculos, o resultado será simétrico e tridiagonal. A segunda diagonal superior desaparecerá. Nota-se que w_1 contribui para o termo $v_1 C + w_1 S$. O termo $-v_1 S + w_1 C$ não

contribui para outros termos; ele já é provavelmente zero. Por essa razão, não se precisa calcular isto. O termo da subdiagonal t_1 não contribui para outros termos; desta forma não é necessário

calcular o próximo u_3 . Em consequência disso, deve-se calcular

somente os termos que são de necessidade no resultado final ou são de necessidade para a próxima rotação elementar.

Em adição ao cálculo de autovalores e autovetores de matrizes simétricas, apresenta-se como opção, o cálculo da k -ésima potência da matriz A como também, seu determinante, posto e traço.

A k -ésima potência é calculada usando a propriedade

$$A^k = U W^k U',$$

onde U é a matriz ortogonal dos autovalores da mesma.

O determinante foi calculado pelo produto dos λ_i

(autovalores) $i = 1, 2, \dots, n$.

$$\det(A) = \prod_{i=1}^n \lambda_i$$

o traço pela soma dos λ_i

$$\text{Tr}(A) = \sum_{i=1}^n \lambda_i$$

e o posto pela contagem dos λ_i diferentes de zero. Considerou-se como zero os valores absoluto de λ menores que 10^{-5} .

Este módulo está protegido para os casos em que se deseje operar com matrizes não simétricas e nulas.

É feito o cálculo da matriz de autovalores e autovetores. No entanto, se for solicitada a opção "<3> Autovetores e autovalores" deve-se guardar o primeiro em uma das memórias. Após isso, o sistema se encarrega de passar os autovalores para a memória de trabalho, voltando ao "menu" principal, ficando à escolha do usuário a opção de guardá-los ou não.

Uma propriedade interessante é: ao tomar-se

$K = -1$ obtém-se A^{-1} que possui as quatro propriedades da inversa de Moore-Penrose, que se passa agora a demonstrar:

Tem-se que $A' = A$

Hipótese : $A = UWU'$

Tese : $A = UW^+ U'^{-1}$

$$\begin{aligned}
 \text{Prova : (a) } AA^{-1} &= UWU'(UW^{-1}U')UWU' \\
 &= UW(U'U)W^{-1}(U'U)WU' \\
 &= UWW^{-1}WU' \\
 &= UWU' = A
 \end{aligned}$$

$$\begin{aligned}
 \text{(b) } A^{-1}AA^{-1} &= UW^{-1}U'(UWU')^{-1}UW^{-1}U' \\
 &= UW^{-1}(U'U)W(U'U)W^{-1}U' \\
 &= UW^{-1}WW^{-1}U' \\
 &= UW^{-1}U' = A^{-1}
 \end{aligned}$$

$$\begin{aligned}
 \text{(c) } AA^{-1} &= UW^{-1}U'UWU' \\
 &= UW^{-1}WU' \\
 &= UU' \text{ (SIMETRICA)}
 \end{aligned}$$

$$\begin{aligned}
 \text{(d) } AA^{-1} &= UWU'UW^{-1}U' \\
 &= UWW^{-1}U' \\
 &= UU' \text{ (SIMETRICA)}.
 \end{aligned}$$

3.16 MÓDULO ALGEDIAG

3.16.1 Descrição Geral

A função deste módulo é realizar a diagonalização de uma matriz simétrica e definida positiva localizada na área de trabalho.

3.16.2 Descrição de Regras de Processamento

O processo de diagonalização por congruência, consiste em justapor a matriz original A à matriz identidade I e a seguir executa-se uma sequência de operações elementares simultaneamente em A e I . Essas operações visam a zerar os elementos abaixo da diagonal em A , tornando-a uma matriz triangular superior. No final das operações teremos em I a matriz das operações C' . Para se entender melhor este processo, descreve-se a seguir o algoritmo utilizado.

1. Passo: Define-se a constante

$$C_{ik} = -a_{ik} / a_{kk} \quad \text{para } a_{kk} > 0$$

2. Passo: Calcula-se

$$a_{ij} = C_{ik} a_{kj} + a_{ij} \quad e$$

$$C_{ij} = C_{ik} C_{kj} + C_{ij}$$

3. Passo: Incrementa-se j , repetindo-se o passo 2.

4. Passo: Incrementa-se i , repetindo-se os passos 1 e 2.

5. Passo: Incrementa-se k , repetindo-se os passos 1 e 2.

Visto que a matriz A é definida positiva, a matriz das operações é não singular o que garante que $C'AC$ é uma diagonal D com todos elementos $d_{ii} > 0$, $i = 1, 2, \dots, L$. Pode-se calcular pelo

ALGEMA a matriz

$$F = C' D^{-1/2},$$

sendo C' invertida pelo processo de Gauss-Jordan e $D^{-1/2}$ pela opção (N). 4. O produto FF' é:

$$\begin{aligned} FF' &= C' D^{-1/2} D^{1/2} C \\ &= C' DC \\ &= A \end{aligned}$$

A matriz F é dita raiz quadrada generalizada de A .

Ao solicitar a diagonalização, o usuário poderá optar pela matriz C' ou D ou F .

3.17 MÓDULO ALGENORMA

3.17.1 Objetivo

A função desse módulo é realizar o cálculo de normas de matrizes.

3.17.2 Descrição

Com uma matriz de quaisquer dimensões na memória de trabalho, pode-se obter as seguintes normas:

Seja A uma matriz $L \times C$, então

$$(i) \|A\|_1 = \max_j \sum_{i=1}^L |a_{ij}| \quad (\text{máximo das somas dos valores absolutos dentro de colunas});$$

$$(ii) \|A\|_{\infty} = \max_i \sum_{j=1}^C |a_{ij}| \quad (\text{máximo das somas dos valores absolutos dentro de linhas});$$

$$(iii) \|A\|_f = \sqrt{\sum_{i=1}^L \sum_{j=1}^C |a_{ij}|^2} \quad (\text{soma de quadrado dos elementos})$$

Observe-se que a norma (iii) de Frobenius de uma matriz A é simplesmente a norma euclideana de A considerada um elemento de

LC . Após a execução desta rotina o sistema retorna ao Menu Principal com a matriz original inalterada na memória de trabalho.

3.18 MÓDULO ALGEORTO

3.18.1 Objetivo

Este módulo tem por função realizar o processo de ortogonalização de Gram-Schmidt a partir de uma matriz de qualquer dimensão na memória de trabalho.

3.18.2 Descrição

Sejam $u_1, \dots, u_s$ vetores que formam um conjunto linearmente independente em um espaço vetorial V com produto interno $\langle \cdot, \cdot \rangle$. Pode-se então construir um conjunto ortogonal de vetores $x_1, \dots, x_s$ tal que, para cada i com $1 \leq i \leq s$, o conjunto dos vetores $x_1, \dots, x_i$ gera precisamente o mesmo subespaço que o conjunto de vetores $u_1, \dots, u_i$.

No algoritmo faz-se primeiramente o cálculo de um conjunto ortogonal de vetores não nulos $v_1, \dots, v_s$ e então é feita a normalização.

$$x_i = \frac{1}{\|v_i\|} v_i$$

Para iniciar, exige-se que v_1 seja um múltiplo de u_1 afim de que eles gerem o mesmo espaço de maneira que toma-se

$$v_{\sim 1} = u_{\sim 1}$$

Em seguida, escolhe-se o segundo vetor $u_{\sim 2}$ do conjunto original, e

dele subtraímos um múltiplo de $v_{\sim 1}$

$$v_{\sim 2} = u_{\sim 2} - k_1 v_{\sim 1},$$

onde k_1 , é escolhido de tal maneira que $v_{\sim 1}$ e $v_{\sim 2}$ sejam ortogo-

nais, como exigido pelo teorema:

$$0 = \langle v_{\sim 1}, v_{\sim 2} \rangle = \langle v_{\sim 1}, u_{\sim 2} \rangle - k_1 \langle v_{\sim 1}, v_{\sim 1} \rangle$$

isso fornece

$$k_1 = \frac{\langle v_{\sim 1}, u_{\sim 2} \rangle}{\langle v_{\sim 1}, v_{\sim 1} \rangle}$$

e

$$v_{\sim 2} = u_{\sim 2} - \frac{\langle v_{\sim 1}, u_{\sim 2} \rangle}{\langle v_{\sim 1}, v_{\sim 1} \rangle} v_{\sim 1}$$

Para formar o vetor $v_{\sim 3}$, subtrai-se múltiplos de $v_{\sim 1}$ e $v_{\sim 2}$ de $u_{\sim 3}$

$$v_{\sim 3} = u_{\sim 3} - k_2 v_{\sim 2} - c_1 v_{\sim 1},$$

onde k_2 e c_1 são determinadas de tal maneira que $v_{\sim 3}$ seja orto-

gonal a $v_{\sim 1}$ e $v_{\sim 2}$. Isso fornece

$$0 = \langle v_{\sim 1}, v_{\sim 3} \rangle = \langle v_{\sim 1}, u_{\sim 3} \rangle - c_1 \langle v_{\sim 1}, v_{\sim 1} \rangle,$$

$$0 = \langle v_{\sim 2}, v_{\sim 3} \rangle = \langle v_{\sim 2}, u_{\sim 3} \rangle - k_2 \langle v_{\sim 2}, v_{\sim 2} \rangle$$

Portanto

$$c_1 = \frac{\langle v_{\sim 1}, u_{\sim 3} \rangle}{\langle v_{\sim 1}, v_{\sim 1} \rangle} \text{ e } k_2 = \frac{\langle v_{\sim 2}, u_{\sim 3} \rangle}{\langle v_{\sim 2}, v_{\sim 2} \rangle}$$

assim,

$$v_{\sim 3} = u_{\sim 3} - \frac{\langle v_{\sim 2}, u_{\sim 3} \rangle}{\langle v_{\sim 2}, v_{\sim 2} \rangle} v_{\sim 2} - \frac{\langle v_{\sim 1}, u_{\sim 3} \rangle}{\langle v_{\sim 1}, v_{\sim 1} \rangle} v_{\sim 1}$$

A fórmula para $v_{\sim r}$ geral é

$$v_{\sim r} = u_{\sim r} - \frac{\langle v_{\sim r-1}, u_{\sim r} \rangle}{\langle v_{\sim r-1}, v_{\sim r-1} \rangle} v_{\sim r-1} - \frac{\langle v_{\sim r-2}, u_{\sim r} \rangle}{\langle v_{\sim r-2}, v_{\sim r-2} \rangle} v_{\sim r-2} - \dots - \frac{\langle v_{\sim 1}, u_{\sim r} \rangle}{\langle v_{\sim 1}, v_{\sim 1} \rangle} v_{\sim 1}$$

Por último é feita a normalização dos vetores $v_{\sim i}$.

Assim,

$$x_{\sim i} = \frac{1}{\|v_{\sim i}\|} v_{\sim i}, \quad i = 1, \dots, n$$

são os vetores normalizados,

Para uma interpretação geométrica do processo de Gram-Schmidt quando $\tilde{u}_1$, $\tilde{u}_2$ e $\tilde{u}_3$ são vetores coluna 3×1 no espaço tridimensional, considere a figura abaixo:

Os vetores $\overline{OP}$, $\overline{OQ}$ representam respectivamente $\tilde{u}_1$ e $\tilde{u}_2$. $\tilde{x}_1$ é vetor unitário ao longo de $\overline{OP}$. A reta $\overline{QR}$ é perpendicular a $\overline{OP}$ de maneira que

$$\overline{OR} = \overline{OQ} \cos \theta = \langle \tilde{x}_1, \tilde{u}_2 \rangle \tilde{x}_1, \text{ onde } \theta \text{ é o ângulo entre}$$

$\tilde{u}_1$ e $\tilde{u}_2$. Portanto, $\langle \tilde{x}_1, \tilde{u}_2 \rangle \tilde{x}_1$ é um vetor na direção de $\tilde{x}_1$ de

comprimento $\overline{OR}$, ou seja, é o vetor $\overline{OR}$ da Figura. O vetor $\overline{OR}$ é chamado de projeção de $\overline{OQ}$ sobre $\overline{OP}$.

Vê-se que

$$\tilde{v}_2 = \tilde{u}_2 - \langle \tilde{x}_1, \tilde{u}_2 \rangle \tilde{x}_1$$

é o vetor $\overline{RQ}$ que, geometricamente, é perpendicular a $\overline{OP}$.

3.19 MÓDULO ALGESWEEP

3.19.1 Objetivo

A função deste módulo é realizar a operação "sweep". Este operador é talvez o mais versátil de todos operadores estatísticos. Ele pode ser útil em regressão múltipla, modelo linear, análise da variância multivariada, correlações parciais, solução de sistemas de equações e outros. O aspecto mais importante do operador "sweep" diz respeito ao seu uso como ferramenta educacional. É que cada elemento da matriz resultante é prontamente identificável e tem significado estatístico. Além disso representa um algoritmo econômico do ponto de vista da utilização da memória do computador, pois a operação é processada no lugar, isto é, na mesma variável usada para guardar a matriz.

3.19.2 Descrição

Com uma matriz de qualquer dimensão na memória de trabalho, pode-se solicitar a opção "<0> Operador "Sweep"", na qual é solicitada ao usuário que indique a coluna inicial e final a ser operada. Terminada a operação, pode-se fazer nova operação em cima da matriz alterada ou retornar ao "menu" principal com a matriz alterada na memória de trabalho.

Para se ter uma idéia do que faz este operador, suponha-se que esteja na memória de trabalho a matriz $[A]$ e operando-se todas as colunas de A ter-se-á no final a inversa

condicional de A ($[A]^{-1}$). Suponha-se ainda o modelo

$$\tilde{y} = X\tilde{b} + \tilde{e}$$

se a matriz na memória de trabalho for

$$\begin{bmatrix} X'X & X'y \\ y'X & y'y \end{bmatrix}$$

e operando-se as colunas de $X'X$ ter-se-á no final a matriz

$$\begin{bmatrix} (X'X)^{-1} & (X'X)^{-1}X'y \\ -y'X(X'X)^{-1} & y'y - y'X(X'X)^{-1}X'y \end{bmatrix}$$

onde cada bloco tem seu significado estatístico

$$\begin{bmatrix} (X'X)^{-1} & \hat{b} \\ -\hat{b} & \text{S.Q. Residuo} \end{bmatrix}$$

As seguintes variáveis são fornecidas à rotina em questão:

- L - número de linhas da matriz;
- C - número de colunas da matriz;
- J1 - primeira coluna a ser operada;
- J2 - última coluna a ser operada;
- A - matriz a ser operada;
- L C

Como resultado, a "procedure" fornece a matriz R de dimensões L x C.

3.20 LINGUAGEM PASCAL

Pascal foi a linguagem utilizada para desenvolver o presente trabalho. Ela foi desenvolvida como uma ferramenta para o ensino de bons hábitos de programação, tornando fácil escrever programas estruturados e claros, confiáveis e facilmente modificáveis.

A linguagem Pascal foi definida originalmente por Nicklauss Wirth, em 1971 e mais tarde acrescida de facilidades, que possibilitam implementações eficientes e confiáveis e, ainda, tornar disponível uma linguagem própria para o ensino de programação.

Entre as características mais importantes da Pascal podemos apontar a grande facilidade para estruturação dos dados; a possibilidade de o programador definir seus próprios tipos escalares, além dos tipos-padrão oferecidos pela linguagem e um conjunto completo de comandos estruturados.

O Pascal Turbo é uma extensão do Pascal que combina um editor, um compilador, criando um ambiente de desenvolvimento de "software" muito produtivo. O editor e as ferramentas de depuração estão integrados no compilador, o que facilita correções de erros ou modificações nas operações, via editor, facilmente. Outra característica do Pascal Turbo é a alta capacidade de resolução gráfica. Trabalhou-se aqui com esta extensão do Pascal.

O Pascal-SC (Pascal for Scientific Computation)

apresentado em 1980, é outra extensão da linguagem, visando acompanhar os avanços da velocidade e capacidade de processamento, que outras linguagens que tratam problemas numéricos não acompanharam. Um exemplo é a multiplicação de matrizes, que só poderia ser feita via chamada de sub-rotina, sendo feita em Pascal-SC apenas por um operador. O Pascal-SC tem o conceito geral de operadores para tipos de dados, ou seja, além dos tipos inteiros e reais, também aceita tipos de dados e operadores para números complexos, vetores, matrizes e intervalos. Expressões para estes tipos de dados podem, entretanto, ser escritas na notação costumeira da matemática.

4 EXEMPLOS DE APLICAÇÃO

4.1 Análise da Variância de Blocos Incompletos Desbalanceados

Em um ensaio comparativo de seis variedades de soja, em que se usou o delineamento em blocos casualizados, com 4 repetições, os rendimentos em grãos por parcela foram os da tabela seguinte (MARKUS, 1974, pg 227):

VARIEDADES	BLOCOS			
	I	II	III	IV
A	44	38	-	41
B	44	-	-	52
C	68	55	49	61
D	34	32	24	30
E	33	34	32	33
F	49	45	43	45

onde: (-) representa uma parcela perdida.

Deseja-se obter a análise da variância e a matriz de variância e covariância para as estimativas dos efeitos estimados de contrastes de tratamentos.

Considerando-se o modelo estatístico

$$y_{ij} = u + t_i + b_j + e_{ij}, \quad i = 1, 2, \dots, 6 \\ j = 1, 2, \dots, 4$$

onde

y_{ij} : observação do tratamento i no bloco j ;

u : média geral;

t_i : efeito do i -ésimo tratamento;

b_j : efeito do j -ésimo bloco;

e_{ij} : erro,

matricialmente tem-se o modelo linear dado por

$$\underset{\sim}{Y} = \underset{\sim}{X} \underset{\sim}{B} + \underset{\sim}{E}$$

onde

$$\underset{\sim}{Y} = \begin{bmatrix} y_{11} \\ y_{12} \\ . \\ . \\ y_{64} \end{bmatrix}; \quad \underset{\sim}{B} = \begin{bmatrix} \underset{\sim}{u} \\ \underset{\sim}{\tau} \\ \underset{\sim}{b} \end{bmatrix}; \quad \underset{\sim}{E} = \begin{bmatrix} e_{11} \\ e_{12} \\ . \\ . \\ e_{64} \end{bmatrix}; \quad \underset{\sim}{\tau} = \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ t_6 \end{bmatrix};$$

$$\underset{\sim}{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Sejam

b : número de blocos ($b = 4$);

k_j : número de tratamentos no bloco j ;

v : número total de tratamentos;

r_i : número de repetições do tratamento i .

Soma de quadrado do residuo .

$$\sum \hat{Y}'\hat{Y} - \sum \hat{B}'X'\hat{Y}$$

onde $\hat{B} = (X'X)^{-1}X'Y$

a matriz de planejamento X justaposta ao Y é

u	t	t	t	t	t	t	b	b	b	b	Y
	1	2	3	4	5	6	1	2	3	4	~
1	1	0	0	0	0	0	1	0	0	0	44
1	1	0	0	0	0	0	0	1	0	0	38
1	1	0	0	0	0	0	0	0	0	1	41
1	0	1	0	0	0	0	1	0	0	0	44
1	0	1	0	0	0	0	0	0	0	1	52
1	0	0	1	0	0	0	1	0	0	0	68
1	0	0	1	0	0	0	0	1	0	0	55
1	0	0	1	0	0	0	0	0	1	0	49
1	0	0	1	0	0	0	0	0	0	1	61
1	0	0	0	1	0	0	1	0	0	0	34
1	0	0	0	1	0	0	0	1	0	0	32
1	0	0	0	1	0	0	0	0	1	0	24
1	0	0	0	1	0	0	0	0	0	1	30
1	0	0	0	0	1	0	1	0	0	0	33
1	0	0	0	0	1	0	0	1	0	0	34
1	0	0	0	0	1	0	0	0	1	0	32
1	0	0	0	0	1	0	0	0	0	1	33
1	0	0	0	0	0	1	1	0	0	0	49
1	0	0	0	0	0	1	0	1	0	0	45
1	0	0	0	0	0	1	0	0	1	0	43
1	0	0	0	0	0	1	0	0	0	1	45

Inicialmente calcula-se o vetor

$$\hat{B} = (X'X)^{-1}X'Y$$

entrando-se com a matriz aumentada $[X|Y]$, gravada no arquivo

MATXY.BID, através da opção <A>.2 do Menu Principal e guarda-a na memória 1 através da opção <C>. Para isolar a matriz X do Y

usa-se a opção <M> particionando-se assim a matriz $[X|Y]$ que estava na M.T. (memória de trabalho), da linha 1 à 21 e da coluna 1 à 11. Uma vez com a matriz X na M.T., faz-se a transposição dela através da opção <E> e, com o uso da opção <C>, guarda-se a X' na memória 2. Ainda com X' na M.T. faz-se a sua transposição através da opção <E> obtendo-se na M.T. a X e em seguida faz-se o produto de X' , que está na memória 2, com X, que está na M.T., com o uso da opção <F>.

Opção F

```

=====
|| Memoria -           Nome           - Linhas - Colunas ||
|| =====
||      1 - XIY           21           12           ||
||      2 - X'            11           21           ||
||      3 - -----          0           0           ||
||      4 - -----          0           0           ||
||      5 - -----          0           0           ||
||      6 - Memoria de Trabalho  21           11           ||
||      7 - Retorno                          ||
|| =====
=====
|| PRODUTO DE => 2 ||
|| COM => 6         ||
=====

```

Feito o produto, o resultado $X'X$ permanece na M.T. que com uso da opção <S>.2 obteve-se a sua impressão

21	3	2	4	4	4	4	6	5	4	6
3	3	0	0	0	0	0	1	1	0	1
2	0	2	0	0	0	0	1	0	0	1
4	0	0	4	0	0	0	1	1	1	1
4	0	0	0	4	0	0	1	1	1	1
4	0	0	0	0	4	0	1	1	1	1
4	0	0	0	0	0	4	1	1	1	1
6	1	1	1	1	1	1	6	0	0	0
5	1	0	1	1	1	1	0	5	0	0
4	0	0	1	1	1	1	0	0	4	0
6	1	1	1	1	1	1	0	0	0	6

Pode-se notar que a matriz $X'X$ possui a seguinte estrutura

$$\begin{bmatrix} n & r' & k' \\ \sim & \sim & \sim \\ r & R & N \\ \sim & \sim & \sim \\ k & N' & K \\ \sim & \sim & \sim \end{bmatrix}$$

onde

n : número de observações;

r : vetor do número de repetições dos tratamentos;

k : vetor do número de tratamentos dos blocos;

R : matriz diagonal com o número de repetições dos tratamentos na diagonal;

N : matriz $[n \times j]$ de incidência composta dos números de observações de cada tratamento i em cada bloco j ;

K : matriz diagonal $[K \times j]$ com o número de tratamentos por bloco na diagonal.

Identificar essa estrutura é útil nos cálculos subsequentes em termos de ganho de memória. Continuando, obtém-se a inversa de

(Moore Penrose de $X'X$ com o uso da opção <N>.4 com $k = -1$ e guarda-se na memória 3 (opção <C>.1.3). Verifica-se, pela opção <N>, a singularidade de $X'X$

Determinante : 0.000000000E+00
 Traco : 6.300000000E+01
 Posto : 9

Pela opção <D> chama-se a matriz $[X|Y]$ para a M.T. com o intuito de isolar $\tilde{Y}$, o que é feito usando a opção <M> particionando-se da linha 1 à linha 21 e da coluna 12 à 12. Com $\tilde{Y}$ na M.T. guarda-se na memória 1, sem somar (uso da opção <C>). Ainda com $\tilde{Y}$ na M.T. faz-se uso da opção <E> para transpô-la e da opção <F> para calcular o produto do conteúdo da M.T. com o da memória 1, obtendo-se a soma de quadrado total que se guarda na memória 5 (opção <C>). Faz-se em seguida o produto da memória 2 com a 1 (opção <F>) obtendo-se a $X'\tilde{Y}$ que se guarda na memória 4. Faz-se novamente o uso da opção <F> para fazer o produto da matriz da memória 3 com a da M.T. obtendo-se o vetor das estimativas dos parâmetros. Com o uso da opção <S> obtém-se a impressão

29.60294
 2.43382
 8.43382
 21.24632
 -7.00368
 -4.00368
 8.49632
 10.79657
 6.96324
 2.71324
 9.12990

Resumindo os passos até aqui, tem-se o seguinte quadro:

Descrição dos passos	Opção	Memória de Trabalho
Entra com $X_{\sim} y$	$\langle A \rangle.2$	$X_{\sim} y$
Guarda $X_{\sim} y$ na M1	$\langle C \rangle.1$	$X_{\sim} y$
Isola X	$\langle M \rangle$	X
Transpõe X	$\langle E \rangle$	X'
Guarda X' na M2	$\langle C \rangle.2$	X'
Transpõe X'	$\langle E \rangle$	X
Faz o produto $X'.X$	$\langle F \rangle$	$X'X$
Imprime $X'X$	$\langle S \rangle.2$	$X'X$
		-1
Eleva $X'X$ à -1	$\langle N \rangle.4$	$(X'X)$
		-1
Guarda $(X'X)$ na M3	$\langle C \rangle.1.3$	$(X'X)$
Chama $X_{\sim} y$ da M1	$\langle D \rangle.1$	$X_{\sim} y$
Isola y	$\langle M \rangle$	y
Guarda y na M1	$\langle C \rangle.1$	y
Transpõe y	$\langle E \rangle$	y'
Faz o produto $y'.y$	$\langle F \rangle$	$y'y$
Guarda $y'y$ na M5	$\langle C \rangle$	$y'y$
Faz o produto $X'.y$	$\langle F \rangle$	$X'y$
Guarda $X'y$ na M4	$\langle C \rangle.4$	$X'y$
		^
Faz o produto $(X'X)^{-1} . X'y$	$\langle F \rangle$	B
		$\sim$

Com o vetor B na M.T. calcula-se a transposta (opção $\langle E \rangle$) e o produto da matriz da M.T. com a da memória 4 (opção $\langle F \rangle$) e em seguida faz-se o produto da M.T. por -1 (opção $\langle G \rangle$)

guardando-se o resultado $\hat{B}'X'Y$ na memória 5 (opção <C>), desta vez somando, obtendo-se $\hat{Y'Y} - \hat{B}'X'Y$, ou seja, a soma de quadrado de residuo. Para ter acesso ao resultado chama-se o conteúdo da memória 5 (uso da opção <D>) que com o uso da opção <S> imprime-se resultando:

S.Q.Residuo

170.22917

Resumindo tem-se o quadro:

Descrição dos passos	Opção	Memória de Trabalho
Transpõe $\hat{B}$	<E>	$\hat{B}'$
Faz o produto $\hat{B}' \cdot X' y$	<F>	$\hat{B}'X'y$
Faz o produto $(\hat{B}'X'y)$ por -1	<G>.M	$-\hat{B}'X'y$
Guarda $-\hat{B}'X'y$ na M5	<C>.5.5	$-\hat{B}'X'y$
Chama $\hat{y'y} - \hat{B}'X'y$ da M5	<D>.5	$\hat{y'y} - \hat{B}'X'y$
Imprime $\hat{y'y} - \hat{B}'X'y$	<S>	$\hat{y'y} - \hat{B}'X'y$

Soma de quadrado de tratamento ajustado:

$\hat{Y'q}$

onde

$\tilde{t}$: vetor das estimativas dos efeitos de tratamentos

facilmente obtido de $\tilde{B}$.

$$\tilde{q} = (\tilde{t} - N\tilde{K}^{-1}\tilde{l}).$$

em que

$\tilde{t}$: vetor dos totais de tratamentos e

$\tilde{l}$: vetor dos totais de blocos.

ambos facilmente obtidos do vetor $X'Y$.

Inicialmente chama-se o $X'Y$ da memória 4 (opção

<D>) e com o uso da opção <M> particiona-se da linha 2 à 7 e da coluna 1 à 1 obtendo-se o vetor $\tilde{t}$ que se guarda na memória 1 (opção <C>). Chama-se em seguida a matriz X' da memória 2 (opção <D>) e obtém-se a transposta (opção <E>) e, em seguida faz-se o produto da memória 2 com a M.T. (opção <F>) obtendo-se $X'X$ que se guarda na memória 2 (opção <C>). A matriz N é obtida particionando-se $X'X$ da linha 2 à 7 e da coluna 8 à 11 (opção

<M>) guardando-se na memória 5. A matriz K^{-1} é obtida chamando-se a $X'X$ da memória 2 (opção <D>) e, em seguida, particionando-se da linha 8 à 11 e da coluna 8 à 11 (opção <M>) e, usando-se a opção <H>, obtém-se a K^{-1} . Faz-se em seguida o produto da memória

-1

5 com a M.T. (opção <F>) obtendo-se a NK que se guarda na memória 5 (opção <C>). Em seguida, chama-se o conteúdo da memória 4 ($X'Y$) e faz-se o particionamento da linha 8 à linha 11 e da coluna 1 à 1 (opção <M>) obtendo-se o vetor $\tilde{l}$ e, em seguida, o produto da memória 5 com a M.T., obtendo-se NK $\tilde{l}$ que com o uso da opção <G> fica multiplicada por -1 e guardada na memória 1 (opção <C>), desta vez somando, obtém-se o vetor $\tilde{q}$.

Para obter-se o vetor $\tilde{r}$ faz-se o produto da memória 3 com memória 4 (opção <F>) obtendo-se o $\tilde{B}$ que com o uso da opção <M> faz-se a partição da linha 2 à 7 e da coluna 1 à 1 de onde obtém-se o $\tilde{r}$ que com o uso da opção <E> obtém-se a sua transposta. Com esse resultado na M.T. faz-se uso da opção <F> para calcular o produto da M.T. com a memória 1 obtendo-se a soma de quadrado de tratamentos ajustada. Com o uso da opção <S> faz-se a sua impressão

S.Q.Tratamento (aj.)

2045.2375

Resumindo tem-se o quadro:

Descrição dos passos	Opção	Memória de Trabalho
Chama $X'y$ da M4	$\langle D \rangle .4$	$X'y$
Isola t de $X'y$	$\langle D \rangle .4$	$X'y$
Guarda t na M5	$\langle C \rangle .1$	t
Chama X' da M2	$\langle D \rangle .2$	X'
Transpõe X'	$\langle E \rangle$	X
Faz o produto $X'.X$	$\langle F \rangle$	$X'X$
Guarda na M2	$\langle C \rangle .2$	$X'X$
Isola N de $X'X$	$\langle M \rangle$	N
Guarda N na M5	$\langle C \rangle .5$	N
Chama $X'X$ da M2	$\langle D \rangle .2$	$X'X$
Isola K de $X'X$	$\langle M \rangle$	K
-1		-1
Calcula K	$\langle H \rangle$	K
-1		-1
Faz o produto $N.K$	$\langle F \rangle$	NK
-1		-1
Guarda NK na M5	$\langle C \rangle .5$	NK
Chama $X'y$ da M4	$\langle D \rangle .4$	$X'y$
Isola l	$\langle M \rangle$	l
-1		-1
Faz o produto $NK \cdot l$	$\langle F \rangle$	$NK \cdot l$
-1		-1
Multiplica $NK \cdot l$ por -1	$\langle G \rangle .M$	$-NK \cdot l$
-1		-1
Guarda $-NK \cdot l$ na M1		$-NK \cdot l$
obtendo-se q		
Faz o produto $(X'X)^{-1} \cdot X'y$	$\langle F \rangle$	B
Isola γ	$\langle M \rangle$	γ
Transpõe γ	$\langle E \rangle$	γ'
Faz o produto $\gamma' \cdot q$	$\langle F \rangle$	$\gamma' \cdot q$

Imprime $\tilde{Y}' \cdot q$

| <S>
|

| $\tilde{Y}' \cdot q$
|

Soma de quadrado de blocos:

Obtém-se a S.Q. de Blocos pelo produto $\tilde{L}' K^{-1} \tilde{L} - C$

em que

$$C = \frac{\tilde{Y}' \tilde{u} \tilde{u}' \tilde{Y}}{n}$$

sendo

$\tilde{u}$: vetor de uns (1) de dimensões 21×1 e

n : o número total de observações ($n = 21$).

O vetor $\tilde{L}$ é obtido chamando-se a memória 4 ($X'Y$) e, em seguida, particionando-se da linha 8 à 11 e da coluna 1 à 1. Com o uso da opção <E> faz-se a sua transposição e guarda-se na

memória 1 (opção <C>). Para obtenção da K^{-1} chamou-se a memória 2 ($X'X$) e particiona-se da linha 8 à 11 e da coluna 8 à 11, e em

seguida, com o uso da opção <H> obtém-se a K^{-1} . Solicita-se a opção <F> faz-se o produto da memória 1 com a M.T., obtendo-se

a $\tilde{L}' K^{-1}$ que guarda-se na memória 5. Chama-se o $\tilde{L}'$ da memória 1, transpõe-se e calcula-se o produto da memória 5 com a

M.T. obtendo-se $\tilde{L}' K^{-1} \tilde{L}$ que se guarda na memória 1.

Para obtenção de C (que poder-se-ia ter obtido elevando-se ao quadrado o primeiro elemento de $X'Y$ e dividindo-se por $n = 21$), faz-se uso da opção $\langle A \rangle$ para ler via arquivo a matriz $[XIY]$ e, uma vez com ela na M.T., particiona-se da linha 1 à 21 e da coluna 12 à 12 obtendo-se o Y que, com o uso da opção $\langle E \rangle$ faz-se a sua transposta e guarda-se na memória 3. Novamente faz-se uso da opção $\langle A \rangle$ desta vez para entrar-se com o vetor u via teclado. Com o vetor u na M.T. guarda-o na memória 4 e em seguida faz-se o produto da memória 3 com a 4, obtendo-se $Y'u$ que se guarda na memória 5. Com o uso da opção $\langle D \rangle$ chamou-se o u da memória 4, faz-se a sua transposta e com o uso da opção $\langle F \rangle$ faz-se o produto da memória 5 com a M.T. obtendo-se $Y'uu'$ que se guarda na memória 5. Chama-se a Y' da memória 3 transpõe-se e faz-se o produto da memória 5 com a M.T. obtendo-se $Y'uu'Y$ que com o uso da opção $\langle G \rangle$ divide-se por -21 obtendo-se $-C$ que guarda-se somando, na memória 1 e em seguida guarda-se $-C$ na memória 4. Na memória 1 obtém-se por fim a S.Q.Blocos. Para ter-se acesso a esse resultado faz-se uso da opção $\langle D \rangle.1.1$ e com a opção $\langle S \rangle$ tem-se a sua impressão via impressora

S.Q.Blocos

189.77143

Resumindo tem-se o seguinte quadro:

Descrição dos passos	Opção	Memória de Trabalho
Chama-se $X'y$ da M4	$\langle D \rangle .4$	$X'y$
Isola $\tilde{l}$	$\langle M \rangle$	$\tilde{l}$
Transpõe $\tilde{l}$	$\langle E \rangle$	$\tilde{l}'$
Guarda $\tilde{l}'$ na M1	$\langle C \rangle .1$	$\tilde{l}'$
Chama $X'X$ da M2	$\langle D \rangle .2$	$X'X$
Isola K	$\langle M \rangle$	K
Inverte K	$\langle H \rangle$	K
Faz o produto $\tilde{l}'K$	$\langle F \rangle$	$\tilde{l}'K$
Guarda $\tilde{l}'K$ na M5	$\langle C \rangle .5$	$\tilde{l}'K$
Chama $\tilde{l}'$ da M1	$\langle D \rangle .4$	$\tilde{l}'$
Transpõe $\tilde{l}'$	$\langle E \rangle$	$\tilde{l}$
Faz o produto $\tilde{l}'K \cdot \tilde{l}$	$\langle F \rangle$	$\tilde{l}'K \cdot \tilde{l}$
Guarda $\tilde{l}'K \cdot \tilde{l}$ na M1	$\langle C \rangle .1$	$\tilde{l}'K \cdot \tilde{l}$
Entra com Xly	$\langle A \rangle .2$	Xly
Isola y	$\langle M \rangle$	y
Transpõe y	$\langle E \rangle$	y'
Guarda y' na M3	$\langle C \rangle .3$	y'
Entra com u	$\langle A \rangle .1$	u
Guarda u na M4	$\langle C \rangle .4$	u
Faz o produto $y'u$	$\langle F \rangle$	$y'u$
Guarda $y'u$ na M5	$\langle C \rangle .5$	$y'u$

Descrição dos passos	Opção	Memória de Trabalho
Chama $\underset{\sim}{u}$ da M4	<D>.4	$\underset{\sim}{u}$
Transpõe $\underset{\sim}{u}$	<E>	$\underset{\sim}{u}'$
Faz o produto $\underset{\sim}{y}'\underset{\sim}{u}\underset{\sim}{.}\underset{\sim}{u}'$	<F>	$\underset{\sim}{y}'\underset{\sim}{uu}'$
Guarda $\underset{\sim}{y}'\underset{\sim}{u}\underset{\sim}{.}\underset{\sim}{u}'$ na M5	<C>.5	$\underset{\sim}{y}'\underset{\sim}{uu}'$
Chama $\underset{\sim}{y}'$ da M3	<D>.3	$\underset{\sim}{y}'$
Transpõe $\underset{\sim}{y}'$	<E>	$\underset{\sim}{y}'$
Faz o produto $\underset{\sim}{y}'\underset{\sim}{uu}'\underset{\sim}{.}\underset{\sim}{y}$	<F>	$\underset{\sim}{y}'\underset{\sim}{uu}'\underset{\sim}{.}\underset{\sim}{y}$
Divide $\underset{\sim}{y}'\underset{\sim}{uu}'\underset{\sim}{y}$ por -21	<G>.D	-C
Guarda -C na M1	<C>.1.5	-C
Guarda -C na M4	<C>.4	-C
Chama $\underset{\sim}{l}'K$ $\underset{\sim}{l}-C$ da M1	<D>.1	$\underset{\sim}{l}'K$ $\underset{\sim}{l}-C$
Imprime $\underset{\sim}{l}'K$ $\underset{\sim}{l}-C$	<S>.2	$\underset{\sim}{l}'K$ $\underset{\sim}{l}-C$

Soma de quadrado total:

A soma de quadrado total obtém-se chamando a $\underset{\sim}{Y}'$ da memória 3, transpondo-se e com uso da opção <F> faz-se o produto da memória 3 com a M.T. obtendo-se $\underset{\sim}{Y}'\underset{\sim}{Y}$ que se guarda somando na memória 4 onde estava a -C obtendo-se assim a soma de quadrado total. Para ter acesso à esse resultado faz-se uso da opção <D> de onde chama-se da memória 4 a $\underset{\sim}{Y}'\underset{\sim}{Y} - C$ que com uso da opção <S> tem-se a sua impressão

S.Q.Total

2405.2381

A seguinte tabela apresenta os passos resumidamente para obtenção da Soma de Quadrado Total:

Descrição dos passos	Opção	Memória de Trabalho
Chamar $y'_{\sim}$ da M3	<D>	$y'_{\sim}$
Transpõe $y'_{\sim}$	<E>	$y_{\sim}$
Faz o produto $y'_{\sim} \cdot y_{\sim}$	<F>	$y'_{\sim} y_{\sim}$
Guarda $y'_{\sim} y_{\sim}$ na M4 obtendo $y'_{\sim} y_{\sim}-C$	<C>.4.5	$y'_{\sim} y_{\sim}$
Chama $y'_{\sim} y_{\sim}-C$ da M4	<D>.4	$y'_{\sim} y_{\sim}-C$
Imprime $y'_{\sim} y_{\sim}-C$	<S>.2	$y'_{\sim} y_{\sim}-C$

Resumindo-se os resultados tem-se a seguinte tabela:

Análise da Variância

Fonte	G.L.	S.Q.	Q.M.	F
Tratamentos	5	2.045,2375	409,0475	28.8350
Blocos	3	189,7714	63,2571	
Resíduo	12	170,2291	14,1857	
Total	20	2.405,2381		

A estimativa da matriz de variância e covariância foi obtida por

$$\hat{\text{Var}}(\hat{D}'\hat{\gamma}) = (D'C^{-1}D) \text{Q.M. Resíduo}$$

em que

D' : é a matriz de contrastes estimáveis de tratamentos (quando há perda de parcelas certos contrastes podem não serem estimáveis) e

C^- : qualquer inversa generalizada de C .

Inicialmente entra-se com a matriz D' via teclado e guarda-a na memória 1

$$D' = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & -1 & 5 & -1 & -1 & -1 \end{bmatrix}$$

faz-se a sua transposta e guarda-a na memória 3. Em seguida chama-se a $X'X$ da memória 2 e com uso da opção <L>.4 faz-se a eliminação da primeira linha. Faz-se uso da opção <L>.5 para eliminar a primeira coluna ficando na M.T. a matriz com a seguinte estrutura

$$\begin{bmatrix} R & N \\ \hline N' & K \end{bmatrix}$$

Em seguida faz-se uso da opção <O> operando da coluna 7 à 10, obtém-se no bloco superior esquerdo a matriz $C = R - NK^{-1}N'$ que com o uso da opção <M> particiona-se da linha 1 à 6 e da coluna 1 à 6 e com o uso da opção <H> obtém-se a C^- . Faz-se em seguida o produto da memória 1 com a M.T. obtendo-se $D'C^-$ e novamente

faz-se o produto da M.T. com a memória 3 obtendo-se $D'C^-D$, que é invariante para qualquer inversa de C. Com o uso da opção <G> faz-se o produto da M.T. pelo Q.M. do Resíduo (14,18576) obtendo-se a matriz das estimativas das variâncias e covariâncias que, através da opção <S> obtém-se a impressão:

```

12.2943 -7.5657  0.0000  0.0000  2.8371
          11.4077 -3.5464  0.0000 -25.8890
                    7.0928 -3.5464  21.2786
                          7.0928  0.0000
                                112.7767

```

Resumindo-se os passos, tem-se a seguinte tabela:

Descrição dos passos	Opção	Memória de Trabalho
Entra com D'	<A>.1.3	D'
Guarda D' na M1	<C>.1	D'
Transpõe D'	<E>	D
Guarda D na M3	<C>.3	D
Chama X'X da M2	<D>.2	X'X
Elimina linha 1 de X'X	<L>.4	(X'X) *
Elimina coluna 1 de		
* (X'X)	<L>.5	(X'X) **
"Sweep" nas colunas de		
** 7 à 10 em (X'X)	<O>	(X'X) ***
Isola C	<M>	C
Inverte C	<H>	C ⁻
Faz o produto D'.C ⁻	<F>	D'C ⁻
Multiplica D'C ⁻ D.14,8576	<G>.M	(D'C ⁻ D)14,8576
Imprime (D'C ⁻ D)14,8576	<S>.2	(D'C ⁻ D)14,8576

4.2 Regressão por Componentes Principais Quando a Matriz X não é de Posto Completo

Para o seguinte conjunto de dados, seja ajustar a função

$$y = b_0 + b_1 X_1 + b_2 X_2$$

para um vetor

$$\underset{\sim}{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

de norma mínima.

$$X = \begin{bmatrix} 1 & 5 & 0,2 \\ 1 & 10 & 0,4 \\ 1 & 15 & 0,6 \\ 1 & 20 & 0,8 \\ 1 & 25 & 1,0 \\ 1 & 30 & 1,2 \\ 1 & 35 & 1,4 \\ 1 & 40 & 1,6 \end{bmatrix} \quad \underset{\sim}{y} = \begin{bmatrix} 8,04 \\ 9,91 \\ 10,80 \\ 9,35 \\ 7,57 \\ 8,67 \\ 9,90 \\ 8,45 \end{bmatrix}$$

Note-se que a terceira coluna de X é 0,04 vezes a segunda.

A matriz X pode ser expressa por

$$\underset{8 \times 3}{X} = \underset{8 \times 2}{U} \underset{2 \times 2}{\Theta} \underset{2 \times 3}{V'}$$

onde

$$1) \Theta = \text{diag} \{ \theta_1, \theta_2 \}$$

$\theta_1, \theta_2, \dots, \theta_n$ são as raízes quadradas dos autovalores de XX' ,

chamados de valores singulares.

2) O posto de X é 2.

3) $U'U = V'V = I_{(2)}$, ou seja, as colunas de U e as de V são

ortogonais.

4) As colunas de U são autovetores normalizados de XX' e as linhas de V' são autovetores normalizados de $X'X$.

Usando o ALGEMA inicialmente entra-se com a matriz X através da opção $\langle A \rangle.1$ e guarda-a na memória 1. Solicita-se a transposta de X que esta na M.T. e faz-se o produto da memória 1 com a M.T. obtendo-se assim a XX'

	26,04	51,08	76,12	101,16	126,20	151,24	176,28	201,34	
		101,16	151,24	201,16	251,40	301,48	351,56	401,64	
			226,36	301,48	376,60	451,72	526,84	601,96	
				401,64	501,80	601,96	702,12	802,28	
					627,00	752,20	877,40	1002,60	
						902,44	1052,68	1202,92	
							1227,96	1403,24	
								1603,56	

Com o uso da opção $\langle N \rangle.3$ determina-se a matriz U de autovetores e guarda-a na memória 2. Automaticamente passa-se para a M.T.

a matriz θ^2 , que, pela opção $\langle N \rangle.4$, eleva-se à potência $k =$

$-0,5$ obtendo-se a θ^{-1} que é guardada na memória 3. A matriz V é obtida da expressão

$$V = X'U\theta^{-1}$$

Chamando-se a matriz X da memória 1, transpõe-se e faz-se o produto da M.T. com a memória 2 e, novamente, o produto da M.T., onde estava $X'U$, pela memória 3, obtendo-se V que se guarda na memória 4. Para obter-se a solução

$$\hat{b} = (V\theta^{-1} U')y$$

faz-se o produto da M.T., onde estava V , pela memória 3 obtendo-

se a $V\theta^{-1}$ que guarda-se na memória 5. Em seguida chama-se a U da memória 2 (opção <D>.2), transpõe-se e faz-se o produto da memória 5 com a M.T. obtendo-se a $V\theta^{-1} U'$ que é guardada na memória 5. Com uso da opção <A>.1.3 entra-se com os dados do vetor y e faz-se o produto da memória 5 com a M.T. obtendo-se

$$\hat{b} = \begin{bmatrix} 9,3728 \\ -0,0127 \\ -0,0005 \end{bmatrix}$$

cuja norma é

$$\|\hat{b}\| = 9,3728$$

obtida através da opção <P>, extaindo a raiz quadrada da norma de Frobenius. Portanto,

$$y = 9,3728 - 0,0127x_1 - 0,0005x_2$$

O cálculo dos valores estimados da produção $\hat{y}$ foi obtido através do produto

$$\hat{y} = Xb$$

que obtém-se facilmente fazendo-se o produto da memória 1 com a M.T.

$$\hat{y} = \begin{bmatrix} 9,3091 \\ 9,2454 \\ 9,1817 \\ 9,1180 \\ 9,0544 \\ 8,9907 \\ 8,9270 \\ 8,8633 \end{bmatrix}$$

A matriz X é tal que existe um relacionamento exato entre a segunda e a terceira coluna, que pode ser representado por uma reta. O problema da regressão é achar um plano em (x_1, x_2, y) que melhor se ajusta aos pontos y observados. Porém, como todos os pontos y ocorrem nos pontos (x_1, x_2) que estão sobre uma reta, o plano desejado é definido apenas por uma reta. Este plano, portanto, pode girar livremente em torno dessa reta e um ponto qualquer não situado na reta definida por (x_1, x_2) não dará um único valor para $\hat{y}$.

A estimativa de y é invariante. Esse fato pôde ser verificado pelo ALGEMA, de onde obtêve-se a mesma estimativa $\hat{y}$ usando tanto o operador "sweep" como o método de Gauss-Jordan para inverter $X'X$ na obtenção de $\hat{b}$.

Resumindo-se os passos no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com X	(A).1	X
Guarda X na M1	(C).1	X
Transpõe X	(E)	X'
Faz o produto X.X'	(F)	XX'
Calcula U	(N).3	U
Guarda U na M2	(C).2	U
		2
Automaticamente tem-se	-----	θ
2		-1
Eleva θ à -0,5	(N).4	θ
-1		-1
Guarda θ na M3	(C).3	θ
Chama X da M1	(D).1	X
Transpõe X	(E)	X'
Faz o produto X'.U	(F)	X'U
-1		
Faz o produto X'U. θ	(F)	V
Guarda V na M4	(C).4	V
-1		-1
Faz o produto V. θ	(F)	V θ
-1		-1
Guarda V θ na M5	(C).5	V θ
Chama U da M2	(D).2	U
Transpõe U	(E)	U'
-1		-1
Faz o produto V θ U'	(F)	V θ U'
-1		-1
Guarda V θ U' na M5	(C).5	V θ U'
Entra com y	(A).1.3	y
$\sim$		$\sim$
-1		$\wedge$
Faz o produto V θ U'. y	(F)	b
$\sim$		$\sim$

Descrição dos passos	Opção	Memória de Trabalho
Imprime $\hat{b}$	$\langle S \rangle . 2$	$\hat{b}$
Calcula norma de $\hat{b}$	$\langle P \rangle$	$\hat{b}$
Faz o produto $X \cdot \hat{b}$	$\langle F \rangle$	$\hat{y}$

4.3 Modelo Simples de Crescimento de Populações ao Longo do Tempo

Sejam duas populações diferentes que se interagem. Representando-se o número dessas populações por $n_{1,i}$ e $n_{2,i}$, em que identifica-se os $n_{1,i}$ de número de bem-te-vis e os $n_{2,i}$ de número de insetos. Supondo que os insetos, sem nenhum bem-te-vi para comê-los, teriam uma taxa de natalidade excedendo a de mortalidade; por exemplo, poder-se-ia ter (NOBLE & DANIEL, 1986, pg 31)

$$n_{2,i+1} = 1,2n_{2,i}$$

Sem haver inseto que lhes sirvam de alimento, supõe-se que a taxa de mortalidade dos bem-te-vis excederá a de natalidade; por exemplo, poder-se-ia ter

$$n_{1,i+1} = 0,6n_{1,i}$$

Deseja-se modelar o que acontece quando os bem-te-vis conseguem por vezes comer insetos. Presumivelmente, isso permitirá um crescimento na população de bem-te-vis, proporcional ao número de insetos disponíveis, de modo que ter-se-á

$$n_{1,i+1} = 0,6n_{1,i} + 0,5n_{2,i}$$

por exemplo. Os insetos, por sua vez, sofreriam um decréscimo de

população, devido aos bem-te-vis predadores, de maneira que

$$n_{2,i+1} = 1,2n_{2,i} - kn_{1,i}$$

onde k reflete a taxa de morte de insetos pelos bem-te-vis, que chama-se de taxa de predação; k está arbitrário para que se possa estudar o efeito, sobre o modelo, de mudanças da taxa k de predação. Supondo-se, especificamente, que há inicialmente 1000 insetos e 100 bem-te-vis, temos o modelo

$$\begin{aligned} n_{1,i+1} &= 0,6n_{1,i} + 0,5n_{2,i} \\ n_{2,i+1} &= -kn_{1,i} + 1,2n_{2,i} \end{aligned} \quad , \text{ para } i \geq 1,$$

onde $n_{1,1} = 100$ e $n_{2,1} = 1000$. Desejando-se analisar o comportamento destas populações ao longo do tempo, pode-se escrever matricialmente o sistema acima da seguinte forma

$$\tilde{x}_i = \begin{bmatrix} n_{1,i} \\ n_{2,i} \end{bmatrix}$$

$$A = \begin{bmatrix} 0,6 & 0,5 \\ -k & 1,2 \end{bmatrix}$$

de modo que o modelo se torna

$$\tilde{x}_{i+1} = A\tilde{x}_i \quad , \text{ para } i \geq 1$$

sendo que

$$\underset{\sim}{x}_0 = \begin{bmatrix} 100 \\ 1000 \end{bmatrix}$$

Facilmente vê-se que

$$\underset{\sim}{x}_{i+1} = A^i \underset{\sim}{x}_1$$

de maneira que o estudo do comportamento do $\underset{\sim}{x}_i$ é equivalente ao estudo do comportamento das potências A^i . É intuitivo que, se a taxa de predação for suficientemente pequena, a população de insetos crescerá sem limites, permitindo assim que a população de bem-te-vis também cresça. Se no entanto, a taxa de predação for muito alta, os insetos seriam extintos e então os bem-te-vis teriam igualmente ocasionado sua própria destruição.

Para ilustrar, consideremos $k = 0,1$, de maneira que

$$A = \begin{bmatrix} 0,6 & 0,5 \\ -0,1 & 1,2 \end{bmatrix}$$

Inicialmente entra-se via opção <A>.1.3 do ALGEMA com A e guarda-se na memória 1 (uso da opção <C>). Em seguida entra-se com o vetor $\underset{\sim}{x}_1$ e guarda-o na memória 2. Com o uso da opção <F> faz-se o produto da memória 1 com a 2 obtendo-se $\underset{\sim}{x}_2$ que com uso da opção <S>.2 faz-se a sua impressão. A seguir faz-se o

produto da memória 1 com ela mesma obtendo-se A^2 que guarda-se na memória 3. Em seguida faz-se o produto da memória 3 com a 2 obtendo-se $x_{\sim 3}$ e assim por diante. Dessa forma obtém-se a seguinte tabela que mostra aproximadamente os números $n_{1,i}$ e $n_{2,i}$

de bem-te-vis e insetos no i -ésimo instante de tempo.

i	0	1	2	3	4	5	6	7
n _{1,i}	100	560	931	1244	1523	1784	2037	2294
n _{2,i}	1000	1190	1372	1553	1739	1935	2143	2368
i	11	16	29	99				
n _{1,i}	3473	5625	19432	15346591				
n _{2,i}	3490	5628	19432	15346591				

Experimentalmente, verifica-se o que se esperava: para uma taxa de predação pequena, tanto $n_{1,i}$ quanto $n_{2,i}$ tendem para o infinito. Pode-se verificar pela tabela que, eventualmente, as duas populações se igualam.

Resumindo-se os passos, tem-se o seguinte quadro:

Descrição dos passos	Opção	Memória de Trabalho
Entra com A	$\langle A \rangle.1.3$	A
Guarda A na M1	$\langle C \rangle.1$	A
Entra com $x_{\sim 1}$	$\langle A \rangle.1.3$	$x_{\sim 1}$

Descrição dos passos	Opção	Memória de Trabalho
Guarda $x_{\sim 1}$ na M2	$\langle C \rangle.2$	$x_{\sim 1}$
Faz o produto $A.x_{\sim 1}$	$\langle F \rangle$	$x_{\sim 2}$
Imprime $x_{\sim 2}$	$\langle S \rangle.2$	$x_{\sim 2}$
Faz o produto $A.A_2$	$\langle F \rangle$	A_2
Guarda A_2 na M3	$\langle C \rangle.3$	A_2
Faz o produto $A.x_{\sim 1}$	$\langle F \rangle$	$x_{\sim 2}$
..... etc.		

Considerando-se uma taxa de predação maior, digamos $k = 0,18$, de modo que

$$A = \begin{bmatrix} 0,6 & 0,5 \\ -0,18 & 1,2 \end{bmatrix}$$

A tabela a seguir mostra o desenvolvimento das populações ao longo do tempo. Ela é obtida de forma idêntica à anterior: chamando-se a matriz A da memória 1 e através da opção $\langle B \rangle$ faz-se a correção do elemento 2,1 e em seguida guardando-a na memória 1.

i	0	1	2	3	4	5	6	7
n	100	560	927	1215	1436	1600	1718	1796
1,i								
n	1000	1182	1317	1414	1478	1515	1530	1527
2,i								
i	15	30	40	60	90	100		
n	1633	668	310	56	3	0		
1,i								
n	1173	441	200	35	2	0		
2,i								

Mais uma vez a experimentação verifica a expectativa intuitiva: para uma taxa de predação maior, tanto a população dos insetos quanto a dos bem-te-vis desaparecem, embora haja uma expansão inicial.

Supondo-se que se tivesse interessado no número total de indivíduos no ambiente, poder-se-ia usar a opção <P> com a matriz A na M.T., e ver-se que

$$\|A\|_c = 1,7$$

de maneira que a população total pode ser multiplicada por 1,7 em um período de tempo. Se se tivesse interessado no número máximo de animais de um só tipo qualquer, então

$$\|A\|_l = 1,38$$

obtida na mesma opção, de maneira que a maior das populações (insetos, no caso) pode ser multiplicada por 1,38 em um período.

Resumindo-se os passos no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Chama A da M1	<C>.1	A
		*
Corrige elemento 2,1		A
*		*
Guarda A na M1	<D>.1	A
..... Repetição dos passos		
	*	
..... anteriores, em A		
*		*
Chama A da M1	<C>.1	A
		*
Calcula norma	<P>	A

4.4 Uso da Diagonalização por congruência

Uma aplicação interessante da congruência é na determinação de máximos e mínimos de funções a duas ou mais variáveis independentes. Sabe-se que, numa função $f(x_1, x_2, \dots, x_n)$ as condições necessárias de existência de máximo podem ser resumidas:

$$\begin{array}{ll} \text{a)} & \frac{df}{dx_i} = 0 \\ \text{b)} & \frac{d^2 f}{dx_i^2} < 0 \end{array}$$

com $i = 1, 2, \dots, n$. E para existência de mínimo:

$$\begin{array}{ll} \text{a)} & \frac{df}{dx_i} = 0 \\ \text{b)} & \frac{d^2 f}{dx_i^2} > 0 \end{array}$$

para $i = 1, 2, \dots, n$.

Assim, dada a função f , pode-se obter o sistema das derivadas primeiras igualadas a zero. As derivadas parciais, de segunda ordem, conduzem à uma forma quadrática, que, se definida positiva, indica um mínimo de f , se definida negativa, um máximo daquela função.

Supondo-se que se deseje verificar a existência de máximo ou mínimo, na função

$$f(x, y) = 2x^2 + 3xy + 2y^2 + x + 6y + 3$$

tem-se

$$\begin{aligned}df(x,y) &= 4xdx + 3ydx + 3xdy + 4ydy + dx + 6dy \\&= (4x + 3y + 1)dx + (3x + 4y + 6)dy\end{aligned}$$

A primeira condição necessária de existência de ponto crítico é

$$\begin{aligned}4x + 3y + 1 &= 0 \\3x + 4y + 6 &= 0\end{aligned}$$

que matricialmente tem-se

$$\begin{array}{ccc}\begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix} & \begin{bmatrix} x \\ y \end{bmatrix} & = \begin{bmatrix} -1 \\ -6 \end{bmatrix} \\ A & \underset{\sim}{x} & \underset{\sim}{b}\end{array}$$

Para resolvê-lo entra-se com a matriz A (opção <A>.1.1) e guarda-se na memória 1, calcula-se a sua inversa (opção <H>) e guarda-a na memória 2. Em seguida entra-se com o vetor $\underset{\sim}{b}$ (opção <A>.1.3) e faz-se o produto da memória 2 com a M.T. obtendo-se a solução

$$\underset{\sim}{x} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

que substituindo na equação original tem-se

$$f(2,-3) = -5$$

Determinou-se a

$$d^2 f(x,y) = 4dx^2 + 3dydx + 3dxdy + 4dy^2$$

Desse modo, a matriz dos coeficientes de $d^2 f(x,y)$, será

$$\begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix}$$

que por coincidência é igual à matriz anterior. chamando-se a memória 1 e em seguida pedindo-se a opção <I>.2 obtém-se a matriz

$$D = \begin{bmatrix} 4 & 0 \\ 0 & 1,75 \end{bmatrix}$$

isto é, uma forma quadrática definida positiva, o que caracteriza um ponto de mínimo.

Então, f passa por um mínimo no ponto $(2, -3, -5)$.

Resumindo-se os passos executados no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com A	<A>.1.1	A
Guarda A na M1	<C>.1	A
		-1
Inverte A	<H>	A
-1		-1
Guarda A na M2	<C>.2	A
Entra com b	<A>.1.3	b
~		~
-1		
Faz o produto A .b	<F>	x
~		~
Chama A da M1	<D>.1	A
Diagonaliza A	<I>.2	D

Nota-se de relance, regularidade na sequência de bases no início (esquerda) e no final (direita). Além disso, as bases parecem suceder-se numa ordem "aleatória". Pode-se investigar o tipo de casualidade pela técnica desenvolvida por A.A.Markov(1856-1922), matemático russo. Considera-se a cadeia como um resultado de um processo estocástico, onde cada ligação da cadeia está em um dos quatros estados A,C,G,U. Uma mudança de uma ligação para a próxima, isto é, de um estado para o outro, é chamada transição. Com quatro estados temos 16 transições diferentes A → A, A → C, A → G, A → U, C → A, C → C etc. As 16 transições possíveis ocorrem com as seguintes frequências, colocadas em forma de matriz:

	A	C	G	U
A	4	14	13	0
C	7	23	32	11
G	18	25	26	12
U	2	12	9	9

Nota-se que as transições que ocorrem mais frequentemente são C → G, G → G, G → C e C → C, enquanto que A → A e U → A são raras. A transição A → U nunca ocorre nesta sequência espacial de ARN. A partir das frequências absolutas derivam-se as frequências relativas em cada linha. Para obter-se essas frequências, uma vez com a matriz A na memória de trabalho, solicita-se a opção <L>.6 e faz-se a divisão de cada linha pelo seu total, por exemplo, a primeira por 31 etc. Ao término dessa operação obtém-se a matriz das frequências relativas:

$$F = \begin{bmatrix} 0,129 & 0,452 & 0,419 & 0,000 \\ 0,096 & 0,315 & 0,438 & 0,151 \\ 0,222 & 0,309 & 0,321 & 0,148 \\ 0,063 & 0,375 & 0,281 & 0,281 \end{bmatrix}$$

A essa matriz dá-se o nome de matriz de transição. Observa-se que a soma dos elementos em cada linha é igual a 1. Para verificar essa propriedade, entra-se com um vetor coluna de uns (1) e pós multiplica-se a F.

Estendendo a investigação a transições de duas etapas tais como $A \rightarrow C \rightarrow A$, $G \rightarrow C \rightarrow U$ etc. pode-se perguntar: quais são as frequências relativas e/ou probabilidades das 16 transições de duas etapas $A \rightarrow . \rightarrow A$, $A \rightarrow . \rightarrow C$ etc. ? O ponto significa que não-se está interessado no estado particularmente intermediário. A transição $A \rightarrow . \rightarrow A$ contém os quatros seguintes casos:

$A \rightarrow A \rightarrow A$

$A \rightarrow C \rightarrow A$

$A \rightarrow G \rightarrow A$

$A \rightarrow U \rightarrow A$

Nas cadeias de Markov admite-se que transições consecutivas são independentes uma das outras no sentido da teoria das probabilidades.

Poder-se-ia pensar em uma matriz de transição teórica

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$$

p_{11} é a probabilidade de transição $A \rightarrow A$ e é estimada pelo valor empírico 0,129 e assim por diante. Para cada linha postula-se que

$$\sum_{k=1}^4 p_{ik} = 1, \quad (i = 1, 2, 3, 4)$$

Sob a hipótese das cadeias de Markov a regra da multiplicação é válida, e obtém-se

$$P(A \rightarrow A \rightarrow A) = P(A \rightarrow A) \cdot P(A \rightarrow A) = p_{11} p_{11},$$

$$P(A \rightarrow C \rightarrow A) = P(A \rightarrow C) \cdot P(C \rightarrow A) = p_{12} p_{21},$$

$$P(A \rightarrow G \rightarrow A) = P(A \rightarrow G) \cdot P(G \rightarrow A) = p_{13} p_{31},$$

$$P(A \rightarrow U \rightarrow A) = P(A \rightarrow U) \cdot P(U \rightarrow A) = p_{14} p_{41}.$$

Tem-se a propriedade de $A \rightarrow . \rightarrow A$ usando a regra de adição

$$P(A \rightarrow . \rightarrow A) = p_{11} p_{11} + p_{12} p_{21} + p_{13} p_{31} + p_{14} p_{41}$$

Isto não é nada mais que o produto interno da primeira linha pela primeira coluna da matriz P . Pelo mesmo argumento podemos calcular a probabilidade de qualquer outra transição de duas fases. O resultado está sempre compreendido na fórmula "linha x

coluna". Todas as 16 probabilidades são, portanto, geradas pela

matriz de multiplicação P^2 .

Pode-se testar se a molécula particular de ARN tem a propriedade de Markov. Para este propósito tomem-se as probabilidades de transição, desconhecidas, dadas na matriz P^2 , pelos valores empíricos dados por F^2 . Então com a F na memória de trabalho, solicita-se a opção <F> e faz-se o produto da M.T. por ela mesma. Executando-se no ALGEMA obtém-se

$$F^2 = \begin{bmatrix} 0,153 & 0,330 & 0,387 & 0,130 \\ 0,149 & 0,335 & 0,361 & 0,155 \\ 0,139 & 0,352 & 0,373 & 0,136 \\ 0,124 & 0,339 & 0,360 & 0,177 \end{bmatrix}$$

Finalmente, confrontando-se os resultados deste cálculo com as frequências relativas das transições de duas etapas obtém-se por contagem direta na Fig.1

$$\begin{bmatrix} 0,065 & 0,323 & 0,548 & 0,064 \\ 0,153 & 0,444 & 0,208 & 0,195 \\ 0,161 & 0,222 & 0,457 & 0,160 \\ 0,156 & 0,437 & 0,313 & 0,094 \end{bmatrix}$$

Calculando-se a estatística

$$\sum_{i=1}^4 \sum_{j=1}^4 \frac{n_i (p_{ij} - p_{ij}^o)^2}{p_{ij}^o} \sim \chi^2_{(3 \times 4)}$$

onde

n_i : somatória dos elementos da linha i da matriz A ;

p_{ij} : estimativa das probabilidades de transição do estado i para o estado j ;

p_{ij}^o : valor esperado das probabilidades de transição do estado i para o estado j .

obtem-se

$$\chi^2 = 8,847$$

da tabela tem-se

$$P(\chi^2 \geq 8,847) < 0,005$$

Conclui-se que a molécula de ARN não possui a propriedade de Markov, o que significa nesse caso, que as transições subsequentes dependem umas das outras.

Resumindo-se os passos no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com A	(A).1.3	A
		*
Divide linha 1 por 31	(L).6	A
		**
Divide linha 2 por 73	(L).6	A

Divide linha 3 por 81	(L).6	A
Divide linha 4 por 32	(L).6	F
Guarda F na M1	(C).1	F
Entra com 1	(A).1.3	1
~		~

Descrição dos passos	Opção	Memória de Trabalho
Faz o produto F.1 ~	⟨F⟩	F.1 ~
Faz o produto F.F 2	⟨F⟩	F 2
Imprime F	⟨S⟩.2	

4.6 Processo de Inversão de Matrizes

Seja inverter a matriz (Boldrini, 1980, pg 86)

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & 3 \end{bmatrix}$$

pelo processo prático baseado nas operações com as linhas da matriz. Esse processo em termos de cálculo é bastante vantajoso. Na prática opera-se simultaneamente com as matrizes A e I (identidade), através de operações elementares, até chegar-se à matriz I na posição correspondente à matriz A.

$$[A|I] \xrightarrow{-1} [I|A^{-1}]$$

Para isso, entra-se via opção <A> com a matriz A juxtaposta à matriz identidade

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \end{bmatrix}$$

com a opção <L>.0, troca-se a primeira pela segunda linha obtendo-se

$$\begin{bmatrix} 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \end{bmatrix}$$

com a opção <L>.2 somou-se à quarta a primeira linha e sem sair

dessa opção soma-se à segunda a primeira linha multiplicada por -2 obtendo-se

$$\begin{array}{cccc|cccc} 1 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 & -2 & 1 & -2 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 & 4 & 0 & 1 & 0 & 1 \end{array}$$

ainda com a opção <L>.2 subtrai-se a segunda linha da terceira obtendo-se

$$\begin{array}{cccc|cccc} 1 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 & -2 & 1 & -2 & 0 & 0 \\ 1 & 0 & 0 & -1 & 3 & -1 & 2 & 1 & 0 \\ 1 & 0 & 0 & -1 & 4 & 0 & 1 & 0 & 1 \end{array}$$

com a opção <L>.6 dividi-se a terceira linha por -1 trocando-se assim o seu sinal obtendo-se

$$\begin{array}{cccc|cccc} 1 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 & -2 & 1 & -2 & 0 & 0 \\ 1 & 0 & 0 & 1 & -3 & 1 & -2 & -1 & 0 \\ 1 & 0 & 0 & -1 & 4 & 0 & 1 & 0 & 1 \end{array}$$

com a opção <L>.2 soma-se a terceira linha com a primeira e pedindo-se mais operação, multiplica-se a terceira linha por -2 e soma-a à segunda anulando-se assim uma parte da terceira coluna obtendo-se

$$\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & -2 & 1 & -1 & -1 & 0 \\ 1 & 0 & 1 & 0 & 4 & -1 & 2 & 2 & 0 \\ 1 & 0 & 0 & 1 & -3 & 1 & -2 & -1 & 0 \\ 1 & 0 & 0 & -1 & 4 & 0 & 1 & 0 & 1 \end{array}$$

com a opção <L>.2 soma-se a terceira linha à quarta obtendo-se

$$\begin{array}{cccc|cccc} 1 & 0 & 0 & -2 & 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & 4 & -1 & 2 & 2 & 0 \\ 0 & 0 & 1 & -3 & 1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 \end{array}$$

com a opção <L>.2 e sem sair dela anula-se a quarta coluna nas três primeiras linhas, multiplicando a quarta linha por 2 e somando à primeira; por -4 e somando à segunda; por 3 e somando à terceira linha. Obtém-se

$$\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 3 & -3 & -3 & 2 \\ 0 & 1 & 0 & 0 & -5 & 6 & 6 & -4 \\ 0 & 0 & 1 & 0 & 4 & -5 & -4 & 3 \\ 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 \end{array}$$

onde a segunda matriz bloco é a inversa de A.

Resumindo-se os passos executados no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com AII	<A>.1	AII
Troca linha 1 com a 2	<L>.0	---
Multiplica linha 1 por 1	<L>.2	---
e linha 4 por 1	<L>.2	---
Multiplica linha 1 por -2	<L>.2	---
e linha 2 por 1	<L>.2	---
Multiplica linha 2 por 1	<L>.2	---
e linha 3 por -1	<L>.2	---
Divide a linha 3 por -1	<L>.6	---
Multiplica linha 1 por 1	<L>.2	---
e linha 3 por 1	<L>.2	---
Multiplica a linha 3 por	<L>.2	---
-2 e a linha 2 por 1	<L>.2	---
Multiplica linha 3 por 1	<L>.2	---
e linha 4 por 1	<L>.2	---
Multiplica a linha 4 por	<L>.2	---
2 e a linha 1 por 1	<L>.2	---
Multiplica a linha 4 por		-1
3 e a linha 3 por 1	<L>.2	IIA

Descrição dos passos	Opção	Memória de Trabalho
-1		-1
Imprime IIA	(S).2	IIA

4.7 Equações Lineares Simultâneas

Uma aplicação interessante do operador "sweep" é na resolução de um sistema de equações simultâneas, com uma matriz dos termos independentes ao invés de um vetor. Dado um sistema

$$Ax = b_1, b_2, b_3$$

que se dispor em uma única matriz na forma

$$[A | b_1 | b_2 | b_3]$$

e operando-se as colunas de A tem-se

$$[A^{-1} | x_1 | x_2 | x_3]$$

Para ilustrar, considere-se o sistema linear

$$\begin{aligned} 3x_1 + x_2 - x_3 &= 2, -0,1, 0 \\ x_1 - 2x_2 + x_3 &= -9, 0,1, 7 \\ 4x_1 + 3x_2 + 2x_3 &= 1, 3,2, 25 \end{aligned}$$

Matricialmente tem-se

$$\begin{bmatrix} 3 & 1 & -1 & | & 2 & -0,1 & 0 \\ 1 & -2 & 1 & | & -9 & 0,1 & 7 \\ 4 & 3 & 2 & | & 1 & 3,2 & 25 \end{bmatrix}$$

Para resolver o sistema entra-se com essa matriz via opção <A>.1.3 e em seguida solicita-se a opção <0> operando-a da

coluna 1 à 3 obtém-se como resultado a matriz

```

| 0,2333  0,1667  0,0333 | -1,0000  0,1000  2,0000 |
| -0,0667 -0,3333  0,1333 |  3,0000  0,4000  1,0000 |
| -0,3667  0,1667  0,2333 | -2,0000  0,8000  7,0000 |

```

em que as três primeiras linhas e colunas constituem a inversa de A e as três colunas seguintes, as soluções do sistema.

Resumindo-se os passos executados no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com A b b b	<A>.1.3	A b b b
~ ~ ~		~ ~ ~
"Sweep" da coluna 1 à 3	<0>	A x x x
		~1 ~2 ~3

4.8 Variáveis Centradas

Quando se trabalha com computadores digitais é de grande importância a simplificação dos cálculos para se obter uma melhor exatidão dos resultados. Em regressão, o uso de variáveis centradas torna os cálculos bem mais simplificados. Seja $[X|y]$ a matriz ampliada de observações

$$X|y = \begin{bmatrix} \bar{X} & X & Y \\ 11 & 21 & 1 \\ X & X & Y \\ 12 & 22 & 2 \\ X & X & Y \\ 13 & 23 & 3 \\ \dots & \dots & \dots \\ X & X & Y \\ 1n & 2n & n \end{bmatrix}$$

e a matriz quadrada

$$A = I - \frac{1}{n} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \end{bmatrix},$$

onde

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

portanto

$$D = A(X|y) = \begin{bmatrix} x_{11} & x_{21} & y_1 \\ x_{12} & x_{22} & y_2 \\ x_{13} & x_{23} & y_3 \\ \dots & \dots & \dots \\ x_{1n} & x_{2n} & y_n \end{bmatrix}$$

onde

$$x_{ij} = X_{ij} - \bar{X}_i$$

$$y_j = Y_j - \bar{Y}, \quad i = 1, 2 \text{ e } j = 1, 2, \dots, n$$

O produto $D'D$ é a matriz de soma de quadrados e de produtos das variáveis centradas

$$\begin{bmatrix} \sum_{j=1}^n (x_{1j})^2 & \sum_{j=1}^n (x_{1j})(x_{2j}) & \sum_{j=1}^n (x_{1j})(y_j) \\ \sum_{j=1}^n (x_{2j})^2 & \sum_{j=1}^n (x_{2j})(y_j) & \sum_{j=1}^n y_j^2 \end{bmatrix}$$

Para ilustrar tomaram-se os seguintes dados (NETER & WASSERMAN, 1974, pg 391):

X	X	Y
1j	2j	j
5,5	31	79,3
2,5	55	200,1
8,0	67	163,2
3,0	50	200,1
3,0	38	146,0
2,9	71	177,7
8,0	30	30,9
9,0	56	291,9
4,0	42	160,0
6,5	73	339,4
5,5	60	159,6
5,0	44	86,3
6,0	50	237,5
5,0	39	107,2
3,5	55	155,0

Esses dados foram gravados em um arquivo de nome CENTRAD.DAD de onde é lido pelo ALGEMA via arquivo texto pela opção <A>.2 e guardado na memória 1. Em seguida, para construir a matriz A, entra-se inicialmente com a matriz diagonal I (opção <A>.1.2) ¹⁵

e em seguida guarda-a na memória 2. Entra-se com o vetor de uns ^{~15} e guarda-o na memória 3 que, solicitando-se a

transposta faz-se o produto da memória 3 com a M.T. obtendo-se ^{ee'} que com o uso da opção <G>.D dividi-se pela constante -15. Em seguida guarda-se esse resultado na memória 2, somando, obtém-se a matriz A que com uso da opção <D> chama-a da memória 2, faz-se o produto da M.T. com ela mesma obtendo-se como resultado a própria matriz A, verificando assim a idempotência de A. Em seguida faz-se o produto da memória 2 com a 1 obtendo-se a matriz D

0,3400	-19,7333	-89,8467	
-2,6600	4,2667	31,1533	
2,8400	16,2667	-5,7467	
-2,1600	-0,7333	31,1533	
-2,1600	-12,7333	-22,9467	
-2,2600	20,2667	8,7533	
2,8400	-20,7333	-138,0467	
3,8400	5,2667	122,9533	
-1,1600	-8,7333	-8,9467	
1,3400	22,2667	170,4533	
0,3400	9,2667	-9,3467	
-0,1600	-6,7333	-82,6467	
0,8400	-0,7333	68,5533	
-0,1600	-11,7333	-61,7467	
-1,6600	4,2667	-13,9467	

que guarda-se na memória 4 e em seguida solicita-se a sua transposta e faz-se o produto da M.T. com a memória 4 obtendo-se

D'D =	59,2760	11,1400	252,3580	
	11,1400	2562,9333	10723,0867	
	252,3580	10723,0867	89547,1173	

Resumindo-se os passos executados no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com X_{ly}	$\langle A \rangle.2$	X_{ly}
Guarda X_{ly} na M1	$\langle C \rangle.1$	X_{ly}
Entra com I	$\langle A \rangle.1.2$	I
Guarda I na M2	$\langle C \rangle.2$	I
Entra com e	$\langle A \rangle.1.3$	e
Guarda e na M3	$\langle C \rangle.3$	e
Transpõe e	$\langle E \rangle$	e'
Faz o produto e.e'	$\langle F \rangle$	ee'
Divide ee' por -15	$\langle G \rangle.D$	-ee' / 15

Descrição dos passos	Opção	Memória de Trabalho
Guarda -ee'/15 na M2 ~~	<C>.2.5	-ee'/15 ~~
Chama A da M2	<D>.2	A
Faz o produto A.A	<F>	A
Faz o produto A.Xly ~	<F>	D
Imprime D	<S>	D
Guarda D na M4	<C>.4	D
Transpõe D	<E>	D'
Faz o produto D'.D	<F>	D'D

4.9 Autovalores de uma matriz não simétrica

Supondo-se que em uma economia com três setores tem-se

$$A = \begin{bmatrix} 0,1 & 0,0 & 0,3 \\ 0,1 & 0,0 & 0,2 \\ 0,0 & 0,1 & 0,1 \end{bmatrix}$$

$$a_{\sim n} = \begin{bmatrix} 0,1 & 0,1 & 0,1 \end{bmatrix}$$

$$d_{\sim} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

onde

a_{ij} : elemento genérico de A , é a quantidade da mercadoria i necessária para a produção de uma unidade da mercadoria j .

a_{nj} : elemento genérico de $a_{\sim n}$, é a quantidade de trabalho diretamente empregada na produção de uma unidade da mercadoria j .

$d_{\sim}$: vetor coluna correspondente à cesta de mercadorias recebidas pelo trabalhador por unidade de trabalho.

A matriz

$$A^* = A + da_{\sim n}$$

possui autovalores reais.

*
 A é obtida entrando-se com A através da opção (A).1.3 e guardando-a na memória 1, entrando-se com $\tilde{a}$ e guardando-o na memória 2 e por último entrando-se com o vetor $\tilde{d}$ e fazendo-se o produto da M.T. com a memória 2 obtendo-se $\tilde{a}$ que guarda-se na memória 1, somando, obtendo-se assim a matriz

$$A = \begin{pmatrix} -0,2 & 0,1 & 0,4 \\ 0,1 & 0,0 & 0,2 \\ -0,1 & 0,2 & 0,2 \end{pmatrix}$$

A dificuldade existente é obter os autovalores dessa matriz, o que foi feito pelo ALGEMA utilizando-se a opção (J).

Pelo processo de ortonormalização de Gram-Schmidt, uma matriz real não singular

$$A = A$$

pode ser fatorada como

$$A = U T$$

onde U é ortogonal e T é triangular superior. Fazendo

$$A = T U$$

pode-se fatorar

$$A = U T$$

e obter-se por produto

$$A_2 = T_{11} U_1$$

A sequência $\{A_n\}$ converge para uma matriz triangular superior, e seus elementos diagonais são os autovalores de A.

Assim procedendo, chama-se a matriz A^* da memória 1 e através da opção <J> obtém-se a U_o que se guarda na memória 2. Em seguida chama-se a matriz A^* da memória 1 e novamente faz-se uso da opção <J> desta vez para obter-se a T_o . Com T_o na M.T. faz-se o produto da M.T. com a memória 2 obtendo-se a A_1 que se guarda na memória 3. A matriz A_1 foi novamente fatorada, reiniciando o processo acima descrito.

Os resultados obtidos foram:

$$A_1 = \begin{bmatrix} 0,467 & 0,303 & 0,000 \\ 0,062 & -0,067 & 0,000 \\ 0,000 & 0,000 & 0,000 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0,505 & -0,226 & 0,000 \\ 0,014 & -0,105 & 0,000 \\ 0,000 & 0,000 & 0,000 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0,499 & 0,243 & 0,000 \\ 0,003 & -0,099 & 0,000 \\ 0,000 & 0,000 & 0,000 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 0,500 & -0,240 & 0,000 \\ 0,001 & -0,100 & 0,000 \\ -0,000 & 0,000 & 0,000 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 0,500 & 0,241 & 0,000 \\ 0,000 & -0,100 & 0,000 \\ -0,000 & 0,000 & 0,000 \end{bmatrix}$$

Dessa última iteração tem-se que os autovalores de A^* são $\lambda_1 = 0,5$, $\lambda_2 = -0,1$ e $\lambda_3 = 0$.

Resumindo-se os passos executados no ALGEMA tem-se:

Descrição dos passos	Opção	Memória de Trabalho
Entra com A	$\langle A \rangle.1.3$	A
Guarda A na M1	$\langle C \rangle.1$	A
Entra com $a_{\sim n}$	$\langle A \rangle.1.3$	$a_{\sim n}$
Guarda na M2	$\langle C \rangle.2$	$a_{\sim n}$
Entra com $d_{\sim}$	$\langle A \rangle.1.3$	$d_{\sim}$
Faz o produto $d_{\sim} a_{\sim}$	$\langle F \rangle$	$d_{\sim} a_{\sim}$
Guarda $d_{\sim} a_{\sim}$ na M1	$\langle C \rangle.1.5$	$d_{\sim} a_{\sim}$
$*$		$*$
Chama A da M1	$\langle D \rangle.1$	A
$*$		
Ortonormaliza A	$\langle J \rangle$	U
Guarda U na M2	$\langle C \rangle.2$	U
o		o
$*$		$*$
Chama A da M1	$\langle D \rangle.1$	A
$*$		
Ortonormaliza A	$\langle J \rangle$	T
Faz o produto $T \cdot U$	$\langle F \rangle$	o
$o \quad o$		1

Descrição dos passos	Opção	Memória de Trabalho
Guarda A ₁ na M3	(C).3	A ₁
..... Repete-se com A ₁ os mesmos passos feitos em A [*]		
..... até A ₅		

5 COMENTÁRIOS FINAIS

Sugere-se para uma nova etapa do desenvolvimento do ALGEMA que:

- 1) Seja apresentada opção para cálculo de autovalores e autovetores reais de matriz não simétrica.
- 2) Seja apresentada opção para inversa de Moore-Penrose para qualquer matriz.
- 3) Seja apresentada opção que faça produto Kronecker e soma direta.

Um problema que surge quando se opera com matrizes em equipamentos de baixa precisão (tamanho pequeno da mantissa) é no que se refere ao condicionamento de uma matriz. Matrizes mal condicionadas podem dar resultados errôneos. Para ilustrar seja a seguinte matriz

	-1843,1667	-425,5000	-591,6667	-826,0000	
		312,5000	-2,0000	115,0000	
			681,8333	-88,1667	
				799,1667	

Nota-se que a coluna quatro é igual à soma das outras três com sinal trocado. Porém, as opções <H> e <N> fornecem o valor 15.886,701 para o determinante. Esse problema é, para esses equipamentos, insolúvel. No entanto, pode-se estudar melhor essa matriz pelo ALGEMA verificando, por exemplo, os seus autovalores. O menor autovalor é -0,000025, um valor pequeno, mostrando uma tendência a zero o que significaria a singularidade

da matriz.

O presente trabalho já está sendo usado por alunos da Pós-Graduação para desenvolver trabalhos como dissertação (SILVA), e seminário (SOUZA) e, em disciplinas como Análise Multivariada. Alunos do curso de genética também têm feito uso do mesmo. Está sendo usado como módulo independente no sistema MOLIMU desenvolvido por GODOI (1987).

O autor tem utilizado o ALGEMA em seu local de trabalho no preparo de arquivos de dados para exercícios de alunos de estatística além de atendimento aos colegas como ajuda em trabalhos científicos. Outro trabalho está sendo desenvolvido pelo autor, seguindo a mesma filosofia deste.

6 REFERENCIAS BIBLIOGRÁFICAS

AMARAL, D. L. Congruência de Matrizes. Piracicaba, 1975. 11p.

(Seminário - Escola Superior de Agricultura "Luiz de Queiroz"
/USP).

ATKINSON, K. E. An Introduction to Numerical Analysis. USA,
1978. 587p.

BATSCHLET, E. Introdução à Matemática para Biocientistas. 2. ed.
Rio de Janeiro, EDUSP, 1984. 596p.

BHAT, U. N. Elements of Applied Stochastic Processes. USA. John
Wiley & Sons, Inc., 1972. 414p.

BOLDRINI, J. L. et alii. Algebra Linear. 3. ed. São Paulo,
Editora Harbra, 1980. 411p.

BORLAND INTERNATIONAL. Turbo Pascal - Reference Manual. 3.
versão. 376p.

- CLAUDIO, D. M. et alii. Fundamentos de Matemática Computacional. 1. ed. Porto Alegre, D. C. Luzatto Editores Ltda, 1987. 194p.
- DIAS, C. T. S. Álgebra Matricial: Um Programa Didático Aplicado à Estatística. Piracicaba, 1985. 15p. (Seminário - Escola Superior de Agricultura "Luiz de Queiroz" /USP).
- FINKBEINER, D. T. Introdução às Matrizes e Transformações Lineares. 2. ed. Rio de Janeiro. Ao Livro Técnico, 321p.
- FLANDERS, H. Scientific Pascal. Virginia, USA. Reston Publ. 1984. 563p.
- GODOI, C. R. M. Análise Estatística Multidimensional. 1. Simpósio de Estatística Aplicada à Experimentação Agronômica; 30. Reunião Anual da Região Brasileira da Sociedade Internacional de Biometria, Piracicaba, 1985. 187p.
- GODOI, C. R. M. et alii. Sistema de Análise de Modelos Lineares Multidimensionais e Álgebra de Matrizes - MOLIMU. 2. Simpósio de Estatística Aplicada à Experimentação Agronômica; 32. Reunião Anual da Região Brasileira da Sociedade Internacional de Biometria, Londrina, 1987. 72p.
- GOODNIGHT, J. H. The Operator Sweep: Its Importance in Statistical Computing. NC, SAS Institute Inc., 1978. 41p.

- HOFFMANN, R. & VIEIRA, S. Análise de Regressão; Uma Introdução à Econometria. 2. ed. São Paulo, Hucitec, 1983. 379p.
- LINHARES, O. L. Classes de Matrizes Quadradas para Teste de Algoritmos Computacionais; Notas de I.C.M.S. São Carlos, 1985. 36p.
- LINHARES, O. L. Matrizes de Teste Inteiras, com Inversas Inteiras e Autovalores e Autovetores Inteiros, Facilmente Explicitáveis ; Notas do I.C.M.S.C. São Carlos, 1987. 40p.
- LOIOLA, C. R. A. Rotinas Matemáticas em Basic para Micros. 1. ed. Rio de Janeiro, Campus, 1985. 225p.
- MARCONI, M. A. & LAKATOS, E. M. Técnicas de Pesquisa. 1. ed. São Paulo, Atlas, 1986. 205p.
- MARKUS, R. Elementos de Estatística Aplicada. 1. ed. Porto Alegre , Editado pelo Diretório Acadêmico "Leopoldo Cortez", 1974. 329p.
- NETER, J. & WASSERMAN, W. Applied Linear Statistical Models. 1. ed. USA, Irwin, 1974, 842p.
- NOBLE, B. Applied Linear Algebra. Englewood Cliffs, N.J., USA. Prentice-Hall, 1969. 523p.

- NOBLE, B. & DANIEL, J. W. Algebra Linear Aplicada. 2. ed. Rio de Janeiro, Prentice-Hall do Brasil Ltda, 1986. 378p.
- PACITTI, T. & ATKINSON, C. P. Programação e Métodos Computacionais. 2. ed. Rio de Janeiro, LTC S.A., 1983. 665p.
- POOLE, L. & BORCHERS, M. Programas Usuais em Basic. São Paulo, McGraw-Hill do Brasil, 1984. 180p.
- RAO, C. R. Linear Statistical Inference and Its Applications. 2. ed. New York, USA. 1973. 625p.
- SAEG -Análises Estatísticas e Genéticas. Viçosa. (Manual - Universidade Federal de Viçosa). 252p.
- SAS - Statistical Analysis System. Guide for Personal Computers. 6. ed. Cary, North Carolina. SAS Institute Inc., 1985. 243p.
- SILVA, E. A. Análise de Covariância em Delineamentos de Blocos Completos Aumentados (Blocos de Federer). Piracicaba, 1987. 106p. (Mestrado - Escola Superior de Agricultura "Luiz de Queiroz"/USP).
- SOC - "Software" Científico. Campinas. (Manual - DMQ e NTIA da EMBRAPA). 1987.

- SOUZA, L. A Decomposição em Valores Singulares da Matriz de Planejamento na Análise de Regressão. Piracicaba, 1986. 33p.
(Seminário - Escola Superior de Agricultura "Luiz de Queiroz")
- SCHMITZ, E. A. & TELES, A. A. S. Pascal e Técnicas de Programação LTC, 1986. 205p.
- STARK, P. A. Introdução aos Métodos Numéricos. Rio de Janeiro, Interciência, 1979. 337p.
- STATGRAPHICS - Statistical Graphics System. USA, by Statistical Graphics Corporation. 1985.
- TURBO PASCAL SYSTEM. Borland Inc. 1984. (Editor de Texto para Microcomputadores).
- WEBER, J. E. Matemática para Economia e Administração. 3. ed. São Paulo, Harper & Row do Brasil Ltda, 1976. 649p.
- WIRTH, N. Programação Sistemática em Pascal. Rio de Janeiro, Campus, 1985. 197p.

ANEXO

(Listagem do Sistema ALGEMA)

ESQUEMA DO ALGEMA

```

{... Arquivo ALGEMA.PAS ...}
program ALGEMA;
const
{... NO = NÚMERO DE ELEMENTOS DOS VETORES ...}
  NO
    = 8100;
type
{... TIPOS DOS VETORES ...}
  POIA      = ^VETA;
  POIM1     = ^VETM1;
  POIM2     = ^VETM2;
  POIM3     = ^VETM3;
  POIM4     = ^VETM4;
  POIM5     = ^VETM5;
  POIF      = ^VETF;
  POID      = ^VETD;
  VETA      = array [1..NO] of real;
  VETM1     = array [1..NO] of real;
  VETM2     = array [1..NO] of real;
  VETM3     = array [1..NO] of real;
  VETM4     = array [1..NO] of real;
  VETM5     = array [1..NO] of real;
{... TIPO DE CADEIA DE CARACTERE ...}
  STR100    = string [100];
{... TIPO DE RECORD ...}
  REG       = record
                nome : string [20];
              end; { record }

var
{... VARIÁVEIS USADAS NO ARQUIVO ALGEDIRE ...}
  KEYASCII,
  KEYSKAN,
{... L = NÚMERO DE LINHAS DA MATRIZ A ...}
{... C = NÚMERO DE COLUNAS DA MATRIZ A ...}
  L,
  C,
  II       : integer;
{... A = VETOR MEMÓRIA DE TRABALHO ...}
  A       : POIA;
{... M1...M5 = VETORES MEMÓRIA FIXA ...}
  M1      : POIM1;
  M2      : POIM2;
  M3      : POIM3;
  M4      : POIM4;
  M5      : POIM5;
  N,
{... VETORES QUE GUARDAM DIMENSÕES LINHA E COLUNA ...}
  LIN,
  COL      : array [1..6] of integer;
{... GUARDA INFORMAÇÃO QUAL MEMÓRIA FOI USADA ...}
  USOU MEMORIA : array [1..5] of boolean;

```

```

{... SINALIZA, QUANDO OPÇÃO DO "MENU" PRINCIPAL
... NÃO PODE SER SOLICITADA ...}
    SINCALCULO,
    SIM,
{... INFORMA SOBRE NULIDADE DA MATRIZ ...}
    NULA,
{... INFORMA SOBRE SIMETRIA DA MATRIZ ...}
    AVISIM,
{... INFORMA SOBRE ESGOTAMENTO DA MEMÓRIA "HEAP" ...}
    HEAPCHEIO      : boolean;
{... GUARDA A OPÇÃO DO "MENU" PRINCIPAL SOLICITADA ...}
    OPCAO,
{... INFORMA QUANDO O DETERMINANTE DEVE SER IMPRESSO ...}
    IMPDET          : char;
{... GUARDA NOME DAS MEMÓRIAS ...}
    MEMO            : array [1..6] of REG;

{... DIRETIVA DE LIGAÇÃO DOS ARQUIVOS COM O ALGEMA ...}
{$i ALGEBIBL.PAS}    { Modulo Procedures globais }
{$i ALGEDADO.PAS}    { Modulo Entrada e correcao de dados }
{$i ALGEGUAR.PAS}    { Modulo Guardar dados }
{$i ALGECHAM.PAS}    { Modulo Chamar dados }
{$i ALGETRAN.PAS}    { Modulo Transpor dados }
{$i ALGEIMPR.PAS}    { Modulo Impressao dados }
{$i ALGEPROD.PAS}    { Modulo Produto de dados }
{$i ALGEPRES.PAS}    { Modulo Produto por escalar }
{$i ALGEINVE.PAS}    { Modulo Inversa generalizada }
{$i ALGEOPER.PAS}    { Modulo Operacao linha /coluna }
{$i ALGEPART.PAS}    { Modulo Particionar matriz }
{$i ALGECONT.PAS}    { Modulo Conteudo memorias }
{$i ALGEAPAG.PAS}    { Modulo Apagar arquivo de dados }
{$i ALGEDIRE.PAS}    { Modulo Mostrar diretorio }
{$i ALGEAUTO.PAS}    { Modulo Autovalores/autovetores }
{$i ALGEDIAG.PAS}    { Modulo Diagonalizacao (Congruencia) }
{$i ALGESWEEP.PAS}   { Modulo Operador sweep }

```

```

{$i ALGEORTO.PAS}      { Modulo. Ortogonalizacao (Gram-Schmidt) }

{$i ALGENORM.PAS}      { Modulo Normas de matriz }

{... PROGRAMA PRINCIPAL ...}

begin { ALGEMA }
  TESTAHEAP (int (NO));
  if HEAPCHEIO
    then
      exit;
  {... GERA O VETOR A E, INICIALIZA ALGUMAS VARIAVEIS ...}
  new (A);
  for II := 1 to 5 do
    USOUMEMORIA [III] := false;
  clrscr;
  CABECALHO;
  IMPDET := 'S';
  AVISIM := false;
  SINCALCULO := false;
  for II := 1 to 5 do
    begin { for }
      N [III] := 0;
      LIN [III] := 0;
      COL [III] := 0;
      MEMO [III].NOME := '-----';
    end; { for }
  MEMO [6].NOME := 'Memoria de Trabalho';
  repeat
    LIMPATELA (2,6,78,23);
    QUADRO (32,06,47,08);
    DIAGRAMA (34,07,'MENU INICIAL');
    QUADRO (05,09,76,21);
    DIAGRAMA (07,10,'<A> Entrar com a matriz');
    DIAGRAMA (07,11,'<B> Corrigir a matriz');
    DIAGRAMA (07,12,'<C> Memorizar ou arquivar a matriz');
    DIAGRAMA (07,13,'<D> Chamar matriz das memorias');
    DIAGRAMA (07,14,'<E> Transpor a matriz');
    DIAGRAMA (07,15,'<F> Multiplicar matrizes');
    DIAGRAMA (07,16,'<G> Produto (divisao) por escalar');
    DIAGRAMA (07,17,'<H> Inverter matriz');
    DIAGRAMA (07,18,'<I> Diagonalizar por congruencia');
    DIAGRAMA (07,19,'<J> Ortonormalizar (Gram-Schmidt)');
    DIAGRAMA (07,20,'<K> Normalizar');
    DIAGRAMA (44,10,'<L> Operar linhas (colunas)');
    DIAGRAMA (44,11,'<M> Particionar a matriz');
    DIAGRAMA (44,12,'<N> Autovalores e autovetores');
    DIAGRAMA (44,13,'<O> Operador "sweep"');
    DIAGRAMA (44,14,'<P> Normas da matriz');
    DIAGRAMA (44,15,'<Q> Teste de simetria');
    DIAGRAMA (44,16,'<R> Conteudo das memorias');

```

```

DIAGRAMA (44,17,'(S) Mostrar ou imprimir a matriz');
DIAGRAMA (44,18,'(T) Apagar arquivo');
DIAGRAMA (44,19,'(U) Mostrar diretorio');
DIAGRAMA (44,20,'(X) Finalizar ');
read (kbd,OPCAO);
OPCAO := upcase (OPCAO);
if (OPCAO in ['B'..'S']) AND (not SINCALCULO)
then
    write (#7)
else
    begin
        LIMPATELA (2,6,78,23);
        case OPCA0 of
            'A' : ENTDAADOS;
            'B' : CORRECAO;
            'C' : GUARDAR;
            'D' : CHAMAR;
            'E' : TRANSPOR;
            'F' : PRODUTO;
            'G' : PROESC;
            'H' : INVERSA;
            'I' : DIAGONALIZAR;
            'J' : GRAM_SCHMIDT;
            'K' : NORMALIZA;
            'L' : OPERACAO;
            'M' : PARTICIONAR;
            'N' : ESPECTRO;
            'O' : SWEEP;
            'P' : NORMA;
            'Q' : begin
                    AVISIM := true;
                    SIMETRIA;
                end;
            'R' : CONTEUDO;
            'S' : IMPRESSAO;
            'T' : APAGAR;
            'U' : DIRETORIO;
        end; { case }
    end; { else }
until OPCA0 in ['X'];
dispose (A);
if USOU MEMORIA [1]
then
    dispose (M1);
if USOU MEMORIA [2]
then
    dispose (M2);
if USOU MEMORIA [3]
then
    dispose (M3);
if USOU MEMORIA [4]

```

```
    then
      dispose (M4);
  if USOUMEMORIA [5]
    then
      dispose (M5);
end. { algema }
```



```

{... Arquivo ALGEDADO.PAS ...}
procedure CORRECAO;
{... CORRIGE ELEMENTOS DA MATRIZ ...}
var
  I,
  J,
  IJ,
  JI      : integer;
{... OPCAO = DEFINE O TIPO DE MATRIZ A CORRIGIR ...}
{... CH = DEFINE NOVA CORREÇÃO OU NAO MAIS...}
  OPCAO,
  CH,
  LIXO      : char;
begin { CORRECAO }
  repeat
    QUADRO (27,10,52,18);
    DIAGRAMA (29,11,'CORRECAO DE MATRIZ ...');
    DIAGRAMA (29,13,'1 - Simetrica');
    DIAGRAMA (29,15,'2 - Qualquer');
    DIAGRAMA (29,17,'3 - Retorno ');
    read (kbd,OPCAO);
    if OPCAO in ['1','2']
    then
      begin
        repeat
          LIMPATELA (2,6,78,21);
          QUADRO (19,8,61,20);
          DIAGRAMA (21,9,'Entre com o indice linha... ');
          I := trunc (NUMERO (49,9,60));
          DIAGRAMA (21,11,'Entre com o indice coluna... ');
          J := trunc (NUMERO (50,11,60));
          if (I in [1..L]) and (J in [1..C])
          then
            begin
              IJ := (I-1) * C + J;
              DIAGRAMA (21,13,'Valor anterior: ');
              write (A^[IJ]:7:4);
              DIAGRAMA (21,15,'');
              write ('Entre o novo A(',I:4,',',J:4,') : ');
              A^[IJ] := NUMERO (49,15,60);
              if OPCAO = '1'
              then
                begin
                  JI := (J-1) * C + I;
                  A^[JI] := A^[IJ];
                end; { OPCAO = '1' }
            repeat
              DIAGRAMA (24,17,'Nova mudanca ? (S/N) ');
              read (kbd,CH);
              CH := upcase (CH);
            until CH in ['S','N'];
          end;
        end;
      end;
    end;
  end;

```

```

        end { if (I in [1..L]) and (J in [1..C]) }
    else
        begin
            DIAGRAMA (21,19,'Indice(s) difere(m) da(s)
                        dimensao(oes)');

            write (#7);
            TECLENTER(24,21);
            CH := 'N';
            end; { else }
        until CH in ['N'];
    end; { if OPCAO }
until OPCAO in ['1','2','3'];
end; { CORRECAO }

procedure ENTDAOS;
{... PERMITE A ENTRADA DE MATRIZ
... VIA TECLADO OU ARQUIVO ...}
var
    ENT,
    OPCAO      : char;
    ARQUIVO    : text;
    NOME_ARQ   : STR100;

procedure INFORMA;
{... EMITE MENSAGEM DE NUMERO DE DADOS
... SOLICITADOS MAIOR QUE O PERMITIDO ...}
begin { INFORMA }
    LIMPATELA (2,6,78,21);
    QUADRO (14,11,65,13);
    gotoxy (16,12);
    write ('Dimensoes superam os ',NO,' elementos declarados');
    write (#7);
    TECLENTER (16,14);
end; { INFORMA }

procedure DADSIM;
{... ENTRADA DE MATRIZ SIMETRICA, VIA TECLADO...}
var
{... VARIÁVEIS DE CONTROLE ...}
    I,
    J,
    IJ,
    JI      : integer;
begin { DADSIM }
    LIMPATELA (2,6,78,21);
    QUADRO (22,11,52,13);
    I := 1;
    while I <= L do
        begin { while I <= L }
            J := I;
            while J <= C do

```

```

begin { while J <= C }
  IJ := (I-1) * C + J;
  JI := (J-1) * C + I;
  DIAGRAMA (24,12,'');
  write ('A(',I:4,',',J:4,') = ');
  A^[IJ] := NUMERO (39,12,51);
  A^[JI] := A^[IJ];
  J := J + 1;
end; { while J <= C }
I := I + 1;
end; { while I <= L }
end; { DADSIM }

procedure DADIAG;
{... ENTRADA DE MATRIZ DIAGONAL, VIA TECLADO...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  J,
  IJ :integer;
begin { DADIAG }
  LIMPATELA (2,6,78,21);
  QUADRO (22,11,52,13);
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I-1) * C + J;
          if (I = J)
            then
              begin { if (I = J) }
                DIAGRAMA (24,12,'');
                write ('A(',I:4,',',J:4,') = ');
                A^[IJ] := NUMERO (39,12,51);
              end { if (I = J) }
            else
              A^[IJ] := 0;
              J := J + 1;
            end; { while J <= C }
          I := I + 1;
        end; { while I <= L }
      end; { procedure DADIAG }

procedure DADASSIM ;
{... ENTRADA DE QUALQUER MATRIZ, VIA TECLADO ...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  J,

```

```

IJ      :integer;
begin { DADASSIM }
  LIMPATELA (2,6,78,21);
  QUADRO (22,11,52,13);
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I-1) * C + J;
          DIAGRAMA (24,12,'');
          write ('A(',I:4,',',J:4,') = ');
          A^[IJ] := NUMERO (39,12,51);
          J := J + 1;
        end; { while J <= C }
        I := I + 1;
      end; { while I <= L }
    end; { DADASSIM }

procedure ARQSIM;
{... ENTRADA DE MATRIZ-SIMETRICA, VIA ARQUIVO TEXTO ...}
var
{... VARIAVEIS DE CONTROLE ...}
  K,
  I,
  J,
  IJ,
  JI :integer;
begin { ARQSIM }
  I := 1;
  J := 1;
  LIMPATELA (2,6,78,21);
  QUADRO (29,10,52,14);
  DIAGRAMA (30,11,'      LENDO PARTE      ');
  DIAGRAMA (30,12,'    SUPERIOR DA MATRIZ ');
  DIAGRAMA (30,13,'      SIMETRICA      ');
  for I := 1 to C do
    for J := I to C do
      begin { for J := I to C }
        IJ := (I - 1) * C + J;
        JI := (J - 1) * C + I;
        read (ARQUIVO, A^[IJ]);
        A^[JI] := A^[IJ];
      end; { for J := I to C }
    close (ARQUIVO);
  end; { ARQSIM }

procedure ARQDIAG;
{... ENTRADA DE MATRIZ DIAGONAL, VIA ARQUIVO ...}
var

```

```

(... VARIÁVEIS DE CONTROLE ...)
I,
II      :integer;
begin { ARQDIAG }
  I := 1;
  while I < L * C do
    begin { while I < L * C }
      A^[I] := 0;
      I := I + 1;
    end; { while I < L * C }
  LIMPATELA (2,6,78,21);
  QUADRO (29,10,50,14);
  DIAGRAMA (30,11,' LENDO OS ELEMENTOS');
  DIAGRAMA (30,12,'   NAO NULOS DA');
  DIAGRAMA (30,13,'  MATRIZ DIAGONAL ');
  I := 0;
  while not eof (ARQUIVO) do
    begin { while not eof (ARQUIVO) }
      while not eoln (ARQUIVO) do
        begin { while not eoln (ARQUIVO) }
          I := I + 1;
          II := (I - 1) * C + I;
          read (ARQUIVO, A^[II]);
        end; { while not eoln (ARQUIVO) }
        I := I + 1;
        II := (I - 1) * C + I;
        read (ARQUIVO, A^[II]);
      end; { while not eof (ARQUIVO) }
    close (arquivo)
  end; { ARQDIAG }

procedure ARQASSIM;
(... ENTRADA DE QUALQUER MATRIZ, VIA ARQUIVO TEXTO ...)
var
(... I = VARIÁVEL DE CONTROLE ...)
I      :integer;
begin { ARQASSIM }
  LIMPATELA (2,6,78,21);
  QUADRO (34,10,45,14);
  DIAGRAMA (36,11,' LENDO ');
  DIAGRAMA (36,12,'ARQUIVO ');
  DIAGRAMA (36,13,'COMPLETO');
  I := 0;
  while not eof (ARQUIVO) do
    begin { while not eof (ARQUIVO) }
      while not eoln (ARQUIVO) do
        begin { while not eoln (ARQUIVO) }
          I := I + 1;
          read (ARQUIVO, A^[I]);
        end; { while not eoln (ARQUIVO) }
        I := I + 1;
      end;
    end;
  end;

```

```

        read (ARQUIVO, A^[I]);
    end; { while not eof (ARQUIVO) }
    close (ARQUIVO);
end; { ARQASSIM }
begin { ENTDAOS }
    repeat
        QUADRO (28,9,51,17);
        DIAGRAMA (30,10,'ENTRAR DADOS VIA...');
        DIAGRAMA (30,12,'1 - Teclado');
        DIAGRAMA (30,14,'2 - Arquivo de dados');
        DIAGRAMA (30,16,'3 - Retorno ');
        read (kbd,ENT);
    until ENT in ['1'..'3'];
    if ENT in ['2']
    then
        begin { if ENT in ['2'] }
            DIAGRAMA (22,19,'Entre drive,(:) e nome do arquivo ');
            read (NOME_ARQ);
            {$i-}
            assign (ARQUIVO,NOME_ARQ);
            reset (ARQUIVO);
            {$i+}
        end; { if ENT in ['2'] }
    if ENT <> '3'
    then
        begin { if ENT <> '3' }
            if (ioresult = 0) or (ENT in ['1'])
            then
                begin { if (ioresult = 0) or (ENT in ['1']) }
                    repeat
                        LIMPATELA (2,6,78,21);
                        QUADRO (31,9,47,19);
                        DIAGRAMA ( 33,10,'MATRIZ ...');
                        DIAGRAMA ( 33,12,'1 - Simetrica');
                        DIAGRAMA ( 33,14,'2 - Diagonal');
                        DIAGRAMA ( 33,16,'3 - Qualquer');
                        DIAGRAMA ( 33,18,'4 - Retorno ');
                        read (kbd,OPCAO);
                        LIMPATELA (2,6,78,21);
                        QUADRO (23,11,60,15);
                        case OPCAO of
                            '1' : begin { case OPCAO = '1' }
                                DIAGRAMA (25,12,'Entre com o n. de
                                    linhas... ');
                                repeat
                                    L := trunc(NUMERO (53,12,59));
                                until L > 0;
                                C := L;
                                if (L * C) <= NO
                                then
                                    if ENT in ['1']

```

```

        then
            DADSIM
        else
            ARQSIM
        else
            INFORMA
    end; { case OPCAO = '1' }
'2' : begin { case OPCAO = '2' }
        DIAGRAMA (25,12,'Entre com o n. de
                    linhas...: ');
        repeat
            L := trunc(NUMERO (53,12,59));
        until L > 0;
        C := L;
        if (L * C) <= NO
        then
            if ENT in ['1']
            then
                DADIAG
            else
                ARQDIAG
            else
                INFORMA
        end; { case OPCAO = '2' }
'3' : begin { case OPCAO = '3' }
        DIAGRAMA (25,12,'Entre com o n. de
                    linhas...: ');
        repeat
            L := trunc(NUMERO (54,12,59));
        until L > 0;
        DIAGRAMA (25,14,'Entre com o n. de
                    colunas...: ');
        repeat
            C := trunc(NUMERO (54,14,59));
        until C > 0;
        if (L * C) <= NO
        then
            if ENT in ['1']
            then
                DADASSIM
            else
                ARQASSIM
            else
                INFORMA
        end; { case OPCAO = '3' }
'4' :
    end; { case }
until OPCAO in ['0'..'4'];
SINCALCULO := true;
end { if ioresult = 0 or ENT in ['1'] }
else

```

```
begin
  LIMPATELA (2,6,78,21);
  gotoxy (24,10);
  writeln ('ARQUIVO "',NOME_ARQ,'" INEXISTENTE');
  TECLENTER (24,12);
end; { else }
end { if ENT <> '3' }
end; { ENTDADOS }
```



```

{... Arquivo ALGEGUAR.PAS ...}
procedure GUARDAR;
{... GUARDA MATRIZ EM MEMÓRIAS AUXILIARES E FIXAS ...}
var
{... E = NÚMERO DA MEMÓRIA EM USO ...}
{... V = 'S' GUARDA SOMANDO, V = 'N' GUARDA SEM SOMAR ...}
{... OPCAO = NÚMERO DA MEMÓRIA AUXILIAR ONDE GUARDAR MATRIZ ...}
{... IND = VARIÁVEL DE CONTROLE QUE INFORMA
... NÚMERO DA MEMÓRIA FIXA ...}
{... CT = NÚMERO TOTAL DE DÍGITOS DE UM NÚMERO REAL ...}
{... CD = NÚMERO DE DÍGITOS DECIMAIS DE UM NÚMERO REAL ...}
{... CI = NÚMERO DE DÍGITOS INTEIROS DE UM NÚMERO REAL ...}
{... ARQUIVO = NOME ASSOCIADO AO ARQUIVO ...}
{... NOME_ARQ = NOME DE ARQUIVO TEXTO DADO PELO USUÁRIO ...}
E,
V,
OPCAO      : char;
IND,
LIXO,
CT,
CD,
CI         : integer;
ARQUIVO    : text;
NOME_ARQ   : STR100;

```

```

procedure GUARMEMO;
{... GUARDA MATRIZ EM UMA DAS CINCO MEMÓRIAS FIXAS ...}
var
{... VARIÁVEIS DE CONTROLE ...}
I,
J,
IJ         : integer;

```

```

begin { GUARMEMO }
  TESTAHEAP (int(NO));
  if HEAPCHEIO
  then
    exit;
  LIMPATELA (2,6,78,21);
  QUADRO (34,12,46,16);
  DIAGRAMA (36,13,'GUARDANDO');
  DIAGRAMA (36,14,' MEMORIA');
  DIAGRAMA (36,15,'PRINCIPAL ');
  case IND of
    1 : if not USOUMEMORIA [1]
        then
          begin { if not USOUMEMORIA [1] }
            USOUMEMORIA [1] := true;
            new (M1);
            end; { if not USOUMEMORIA [1] }
    2 : if not USOUMEMORIA [2]

```

```

    then
        begin { if not USOUMEMORIA [2] }
            USOUMEMORIA [2] := true;
            new (M2);
        end; { if not USOUMEMORIA [2] }
3 : if not USOUMEMORIA [3]
    then
        begin { if not USOUMEMORIA [3] }
            USOUMEMORIA [3] := true;
            new (M3);
        end; { if not USOUMEMORIA [3] }
4 : if not USOUMEMORIA [4]
    then
        begin { if not USOUMEMORIA [4] }
            USOUMEMORIA [4] := true;
            new (M4);
        end; { if not USOUMEMORIA [4] }
5 : if not USOUMEMORIA [5]
    then
        begin { if not USOUMEMORIA [5] }
            USOUMEMORIA [5] := true;
            new (M5);
        end; { if not USOUMEMORIA [5] }
end; { case }
N [IND] := N [IND] + 1;
if (N [IND] = 1) or (V in ['N'])
then
    begin { if (N [IND] = 1) or (V in ['N']) }
        I := 1;
        while I <= L do
            begin { while I <= L }
                J := 1;
                while J <= C do
                    begin { while J <= C }
                        IJ := (I-1) * C + J;
                        case E of
                            '1' : M1^[IJ] := 0;
                            '2' : M2^[IJ] := 0;
                            '3' : M3^[IJ] := 0;
                            '4' : M4^[IJ] := 0;
                            '5' : M5^[IJ] := 0;
                        end; { case E }
                        J := J + 1;
                    end; { while J <= C }
                    I := I + 1;
                end; { while I <= L }
            end; { if (N [IND] = 1) or (V in ['N']) }
if (V in ['5']) and (L = LIN [IND]) and (C = COL [IND])
or (V in ['N']) or (N [IND] = 1)
then
    begin

```

```

I := 1;
while I <= L do
begin { while I <= L }
  J := 1;
  while J <= C do
  begin { while J <= C }
    IJ := (I-1) * C + J;
    case E of
      '1' : M1^[IJ] := M1^[IJ] + A^[IJ];
      '2' : M2^[IJ] := M2^[IJ] + A^[IJ];
      '3' : M3^[IJ] := M3^[IJ] + A^[IJ];
      '4' : M4^[IJ] := M4^[IJ] + A^[IJ];
      '5' : M5^[IJ] := M5^[IJ] + A^[IJ];
    end; { case E }
    J := J + 1;
  end; { while J <= C }
  I := I + 1;
end; { while I <= L }
LIN [IND] := L;
COL [IND] := C;
end { if }
else
begin
  LIMPATELA (2,6,78,21);
  QUADRO (26,9,52,11);
  DIAGRAMA (28,10,'DIMENSOES INCOMPATIVEIS ');
  TECLENTER (28,13);
  N [IND] := N [IND] - 1;
end; { else }
end; { GUARMEMO }

procedure GUARARQ;
{... GUARDA MATRIZ EM ARQUIVO TEXTO ...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  J,
  IJ : integer;
{... DEFINE TIPO DE MATRIZ A ARQUIVAR ...}
  TECLA : char;

procedure INFORMA (ERRO : byte);
{... EMITE MENSAGEM, INFORMANDO O TIPO DE ERRO
... DE ENTRADA E/OU SAÍDA DE DADOS ...}
begin { INFORMA }
  if ERRO > 0 then
  begin { if ERRO > 0 }
    LIMPATELA (2,6,78,21);
    case ERRO of
      $F0 : DIAGRAMA (5,14,'Erro de escrita -> Provavelmente
disco cheio ');
    end;
  end;
end;

```

```

$F1 : DIAGRAMA (5,14,'Diretorio esta cheio -> Nao ha mais
      espaco no diretorio do disco ');
$F2 : DIAGRAMA (5,14,'Tamanho do arquivo maior que maximo
      permitido -> 65535 registros ');
$FF : DIAGRAMA (5,14,'Arquivo desaparecido -> Close num
      arquivo que nao esta no diretorio ');
else
  DIAGRAMA (30,14,'Erro de escrita no disco ');
end; { case ERRO }
TECLENTER (10,16);
I := L + 1;
J := C + 1;
end; { if ERRO > 0 }
end; { INFORMA }

procedure ARQUIVAMENTO;
begin { ARQUIVAMENTO }
  {$i-}
  write (ARQUIVO,A^[IJ]:CT:CD,' ');
  {$i+}
  INFORMA (ioresult);
end; { ARQUIVAMENTO }

procedure MENSAGEMARQUIVAMENTO;
begin { MENSAGEMARQUIVAMENTO }
  LIMPATELA (2,6,78,21);
  QUADRO (32,12,44,16);
  DIAGRAMA (34,13,'GUARDANDO');
  DIAGRAMA (34,14,'      EM      ');
  DIAGRAMA (34,15,'  ARQUIVO  ');
end; { MENSAGEMARQUIVAMENTO }

begin { GUARARQ }
  LIMPATELA (2,6,78,21);
  repeat
    QUADRO (28,9,50,19);
    DIAGRAMA (30,10,'ARQUIVAR MATRIZ ...');
    DIAGRAMA (30,12,'1 - Simetrica');
    DIAGRAMA (30,14,'2 - Diagonal');
    DIAGRAMA (30,16,'3 - Qualquer');
    DIAGRAMA (30,18,'4 - Retorno ');
    read (kbd,TECLA);
  until TECLA in ['1','2','3','4'];
  close (ARQUIVO);
  rewrite (ARQUIVO);
  if TECLA in ['1','2','3']
  then
    MENSAGEMARQUIVAMENTO;
  I := 1;
  while I <= L do
    begin { while I <= L }

```

```

J := 1;
while J <= C do
begin { while J <= C }
  IJ := (I - 1) * C + J;
  case TECLA of
    '1': if I <= J
          then
            ARQUIVAMENTO;
    '2': if I = J
          then
            ARQUIVAMENTO;
    '3': ARQUIVAMENTO;
    '4':
      end; { case TECLA }
      J := J + 1;
  end; { while J <= C }
  if (TECLA <> '4')
  then
    writeln (ARQUIVO);
    I := I + 1;
  end; { while I <= L }
close (ARQUIVO);
end; { GUARARQ }

begin { GUARDAR }
repeat
  QUADRO (26,8,50,16);
  DIAGRAMA (28,9,'GUARDAR...');
  DIAGRAMA (28,11,'1 - Memoria principal');
  DIAGRAMA (28,13,'2 - Arquivo');
  DIAGRAMA (28,15,'3 - Retorno ');
  read (kbd,OPCAO);
until OPCAO in ['1','2','3'];
LIMPATELA (2,6,78,21);
case OPCAO of
  '1' : begin { case OPCAO = '1' }
        repeat
          QUADRO (26,7,53,16);
          DIAGRAMA (28,8,'GUARDAR NA MEMORIA...');
          for IND := 1 to 5 do
            begin { for IND := 1 to 5 }
              gotoxy (28,9 + IND);
              writeln(IND,' - ', MEMO [IND].NOME );
            end; { for IND := 1 to 5 }
          DIAGRAMA (28,15,'6 - Retorno ');
          read (kbd,E);
        until E in ['1'..'6'];
        if E in ['1'..'5']
        then
          begin { if E in ['1'..'5'] }
            repeat

```

```

        QUADRO (26,17,72,21);
        DIAGRAMA (28,18,'Guardar somando ? (S/N) ');
        read (kbd,V);
        V := upcase (V);
        if V = '6'
            then
                exit;
            until V in ['S','N'];
        DIAGRAMA (28,20,'Entre nome identifique
                    dados ');
        val (E,IND,LIXD);
        read (MEMO[IND].NOME);
        case E of
            '1'..'5' : GUARMEMO;
        end; { case E }
    end; { if E in ['1'..'5'] }
end; { case OPCAO = '1' }
'2' : begin { case OPCAO = '1' }
        QUADRO (13,10,70,16);
        DIAGRAMA (15,11,'Entre o drive,(:) e nome
                    arquivo... ');
        read (NOME_ARQ);
        {$i-}
        assign (ARQUIVO,NOME_ARQ);
        reset (ARQUIVO);
        {$i+}
        if ioresult <> 0
            then
                begin { if ioresult <> 0 }
                    DIAGRAMA (15,13,'Entre numero casas
                                decimais... ');
                    CD := trunc (NUMERO (47,13,69));
                    DIAGRAMA (15,15,'Entre numero casas
                                inteiras... ');
                    CI := trunc (NUMERO (47,15,69));
                    CT := CI + CD + 1;
                    GUARARQ
                end { if ioresult <> 0 }
            else
                begin
                    LIMPATELA (2,6,78,23);
                    QUADRO (22,09,54,11);
                    gotoxy (24,10);
                    writeln ('ARQUIVO "',NOME_ARQ,'" JA EXISTE');
                    TECLENTER (24,13);
                    close (ARQUIVO);
                end; { else }
            end; { case OPCAO = '2' }
end; { case global }
end; { GUARDAR }

```

```

{... ARQUIVO ALGECHAM.PAS ...}
procedure CHAMAR;
{... CHAMAR MATRIZ DA MEMORIA FIXA OU ARQUIVO TEXTO ...}
var
{... FF = NUMERO CARACTERE DA MEMORIA FIXA A CHAMAR ...}
{... IND = NUMERO INTEIRO DA MEMORIA FIXA A CHAMAR ...}
{... LIXO = VARIÁVEL AUXILIAR QUE GUARDA RESULTADO INTERMEDIÁRIO
... DA TRANSFORMAÇÃO DE CARACTERE PARA NUMÉRICO ...}
  FF      : char;
  IND,
  LIXO    : integer;

procedure CHAMEMO;
{... CHAMAR MATRIZ DA MEMORIA FIXA ...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  J,
  IJ      : integer;
begin { CHAMEMO }
  LIMPATELA (2,6,78,21);
  VAL (FF,IND,LIXO);
  if (LIN [IND] <> 0) and (COL [IND] <> 0)
  then
    begin { if (LIN [IND] <> 0) and (COL [IND] <> 0) }
      LIMPATELA (2,6,78,21);
      QUADRO (32,12,44,16);
      DIAGRAMA (34,13,'CHAMANDO');
      DIAGRAMA (34,14,' MEMORIA');
      DIAGRAMA (34,15,'PRINCIPAL');
      L := LIN [IND];
      C := COL [IND];
      I := 1;
      while I <= L do
        begin { while I <= L }
          J := 1;
          while J <= C do
            begin { while J <= C }
              IJ := (I-1) * C + J;
              case FF of
                '1' : A^[IJ] := M1^[IJ];
                '2' : A^[IJ] := M2^[IJ];
                '3' : A^[IJ] := M3^[IJ];
                '4' : A^[IJ] := M4^[IJ];
                '5' : A^[IJ] := M5^[IJ];
              end; { case FF }
              J := J + 1;
            end; { while J <= C }
            I := I + 1;
          end; { while I <= L }
        end { if (LIN [IND] <> 0) and (COL [IND] <> 0) }
    end

```

```

else
  begin
    QUADRO (28,9,51,11);
    DIAGRAMA (30,10,'Conteudo inexistente');
    TECLENTER (25,13);
  end; { else }
end; { CHAMEMO }

begin { CHAMAR }
  LIMPATELA (2,6,78,21);
  repeat
    QUADRO (24,7,52,16);
    DIAGRAMA (26,8,'CHAMAR DA MEMORIA ...');
    IND := 1;
    while IND <= 5 do
      begin { while IND <= 5 }
        gotoxy (26,9 + IND);
        writeln (IND,' - ',MEMO [IND].NOME);
        IND := IND + 1;
      end; { while IND <= 5 }
    DIAGRAMA (26,9 + IND,'6 - Retorno ');
    read (kbd,FF);
  until FF in ['1'..'6'];
  if FF in ['1'..'5']
  then
    CHAMEMO;
end; { CHAMAR }

```



```

{... Arquivo ALGETRAN.PAS ...}
procedure TRANSPOR;
{... TRANSPOR MATRIZ DE QUALQUER DIMENSSAO ...}
type
{... DEFINIÇÃO DE UM TIPO PONTEIRO ...}
  POIB = ^VETB;
  VETB = array [1..NO] of real;
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  J,
  IJ,
  JI,
{... AUX = VARIÁVEL AUXILIAR NA TRANSPOSIÇÃO ...}
{... B = VETOR AUXILIAR DO TIPO PONTEIRO ...}
{... LIMTRAN = VARIÁVEL AUXILIAR PARA LIBERAR ÁREA
... USADA PELO VETOR B ...}
  AUX      : integer;
  B        : POIB;
  LIMTRAN   : ^integer;
begin
  TESTAHEAP (int(NO));
  if HEAPCHEIO
    then
      exit;
  QUADRO (31,12,44,14);
  DIAGRAMA (33,13,'TRANSPONDO ');
  mark (LIMTRAN);
  new (B);
  JI := 0;
  J := 1;
  while J <= C do
    begin { while J <= C }
      I := 1;
      while I <= L do
        begin { while I <= L }
          JI := JI + 1;
          IJ := (I-1) * C + J;
          B^[JI] := A^[IJ];
          I := I + 1;
        end; { while I <= L }
      J := J + 1;
    end; { while J <= C }
{... PASSAGEM DOS ELEMENTOS TRANSPOSTOS PARA VETOR A ...}
  IJ := 1;
  AUX := L*C;
  while IJ <= AUX do
    begin
      A^[IJ] := B^[IJ];
      IJ := IJ + 1;
    end; { while }

```

```
AUX := C;  
C := L;  
L := AUX;  
  release (LIMTRAN);  
end; { TRANSPOR }
```

```

{... Arquivo ALGEPROD.PAS ...}
procedure PRODUTO;
{... FAZ PRODUTO DE DUAS MATRIZES ...}
type
  POIAX = ^VETAX;
  VETAX = array [1..NO] of real;
  POILIMPA = ^integer;
var
  {... U = NUMERO CARACTERE DA PRIMEIRA MATRIZ A MULTIPLICAR ...}
  {... Y = NUMERO CARACTERE DA SEGUNDA MATRIZ A MULTIPLICAR ...}
  {... AX = VETOR-AUXILIAR DO TIPO POIAX, QUE RECEBE OS ELEMENTOS
    ... DE A ...}
  {... I,J,K,IJ,KJ,IK = VARIÁVEIS DE CONTROLE ...}
  {... INDU = VARIÁVEL DE CONTROLE DO VETOR LIN (NUMERO DE LINHAS)
    ... DAS MEMÓRIAS FIXAS ...}
  {... INDY = VARIÁVEL DE CONTROLE DO VETOR COL (NUMERO DE COLUNAS)
    ... DAS MEMÓRIAS FIXAS ...}
  {... LIXO = VARIÁVEL AUXILIAR DA TRANSFORMAÇÃO CARACTERE
    ... EM NUMERO ...}
  {... AUXU = VARIÁVEL AUXILIAR, LIGADA A PRIMEIRA MATRIZ
    ... A SOMAR ...}
  {... AUXY = VARIÁVEL AUXILIAR, LIGADA A SEGUNDA MATRIZ
    ... A SOMAR ...}
  {... LIMPA = VARIÁVEL USADA PARA LIBERAR ÁREA UTILIZADA POR
    ... VARIÁVEIS DO TIPO PONTEIRO ...}
  U,
  Y      : char;
  AX      : POIAX;
  I,
  J,
  K,
  IJ,
  KJ,
  IK,
  INDU,
  INDY,
  LIXO    : integer;
  AUXU,
  AUXY    : real;
  LIMPA   : POILIMPA;
procedure PRODADOS;
begin { PRODADOS }
  if (U = '6') or (Y = '6')
  then
    begin { if (U = '6') or (Y = '6') }
      TESTAHEAP (int (NO));
      if HEAPCHEIO
      then
        exit;
      mark (LIMPA);
      new (AX);
    end
  end

```

```

I := 1;
while I <= L do
  begin { while I <= L }
    J := 1;
    while J <= C do
      begin { while J <= C }
        IJ := (I-1) * C + J;
        AX^[IJ] := A^[IJ];
        J := J + 1;
      end; { while J <= C }
      I := I + 1;
    end; { while I <= L }
  end; { if (U = '6') or (Y = '6') }
LIMPATELA (2,6,78,23);
QUADRO (32,12,46,15);
DIAGRAMA (34,13,'PROCESSANDO');
DIAGRAMA (34,14,' PRODUTO ');
IJ := 0;
I := 1;
while I <= LIN [INDU] do
  begin { while I <= LIN [INDU] }
    J := 1;
    while J <= COL [INDY] do
      begin { while J <= COL [INDY] }
        IJ := IJ + 1;
        A^[IJ] := 0;
        K := 1;
        while K <= COL [INDU] do
          begin { while K <= COL [INDU] }
            IK := (I-1) * COL [INDU] + K;
            KJ := (K-1) * COL [INDY] + J;
            case U of
              '1' : AUXU := M1^[IK];
              '2' : AUXU := M2^[IK];
              '3' : AUXU := M3^[IK];
              '4' : AUXU := M4^[IK];
              '5' : AUXU := M5^[IK];
              '6' : AUXU := AX^[IK];
            end; { case U }
            case Y of
              '1' : AUXY := M1^[KJ];
              '2' : AUXY := M2^[KJ];
              '3' : AUXY := M3^[KJ];
              '4' : AUXY := M4^[KJ];
              '5' : AUXY := M5^[KJ];
              '6' : AUXY := AX^[KJ];
            end; { case Y }
            if (AUXU <> 0) and (AUXY <> 0)
              then
                A^[IJ] := A^[IJ] + AUXU * AUXY;
            K := K + 1;
          end; { while K <= COL [INDU] }
        end; { while J <= COL [INDY] }
      end; { while I <= LIN [INDU] }
    end; { if (U = '6') or (Y = '6') }
  end; { if (U = '6') or (Y = '6') }
end; { if (U = '6') or (Y = '6') }

```

```

        end; { while K <= COL [INDU] }
        J := J + 1;
    end; { while J <= COL [INDY] }
    I := I + 1;
end; { while I <= LIN [INDU] }
L := LIN [INDU];
C := COL [INDY];
if (U = '6') or (Y = '6')
then
    release (LIMPA);
end; { PRODADOS }

begin { PRODUTO }
    LIN [6] := L;
    COL [6] := C;
    QUADRO (13,7,65,17);
    DIAGRAMA (15,8,'Memoria -           Nome           - Linhas -
                Colunas');
    DIAGRAMA (15,9,'=====
                ===');
    I := 1;
    while I <= 6 do
        begin { while I <= 6 }
            DIAGRAMA (20,9+I,' ');
            writeln (I,' - ',MEMO [I].NOME);
            gotoxy (48,9+I);
            write (LIN [I]);
            gotoxy (57,9+I);
            write (COL [I]);
            I := I + 1;
        end; { while I <= 6 }
    DIAGRAMA (21,16,'7 - Retorno');
    QUADRO (13,18,31,21);
    DIAGRAMA (15,19,'PRODUTO DE => ');
    repeat
        LIMPALINHA (29,19,30);
        read (U);
    until U in ['1'..'7'];
    if U <> '7'
    then
        begin { if U <> '7' }
            DIAGRAMA (15,20,'COM => ');
            repeat
                read (Y);
                LIMPALINHA (22,20,30);
            until Y in ['1'..'7'];
            if Y <> '7'
            then
                begin { if Y <> '7' }
                    val (U,INDU,LIXO);

```

```

val (Y,INDY,LIXO);
if (INDU = 6) or (INDY = 6)
then
  begin { if (INDU = 6) or (INDY = 6) }
    LIN [6] := L;
    COL [6] := C;
    end; { if (INDU = 6) or (INDY = 6) }
if (COL [INDU] <> LIN [INDY]) or ((COL [INDU] = 0)
and (LIN [INDY] = 0))
then
  begin
    LIMPATELA (2,6,78,23);
    QUADRO (22,10,45,12);
    DIAGRAMA (24,11,'PRODUTO INCOMPATIVEL');
    TECLENTER (24,14);
  end { if }
else
  PRODADOS;
end; { if Y <> '7' }
end; { if U <> '7' }
end; { PRODUTO }

```

```

{... Arquivo ALGEIMPR.PAS ...}
procedure IMPRESSAO;
{... IMPRIMIR MATRIZ NO VIDEO OU NA IMPRESSORA ...}
var
{... SAIDA = CARACTERE QUE DEFINE O EQUIPAMENTO DE SAIDA ...}
{... CT = NUMERO TOTAL DE DIGITOS A IMPRIMIR VIA IMPRESSORA ...}
{... CI = NUMERO DE DIGITOS INTEIROS A IMPRIMIR VIA
... IMPRESSORA ...}
{... CD = NUMERO DE DIGITOS DECIMAIS A IMPRIMIR VIA
... IMPRESSORA ...}
{... CADEIA = NOME DADO A MATRIZ PELO USUARIO PARA IMPRESSAO
... VIA IMPRESSORA ...}
SAIDA : char;
CT,
CI,
CD : integer;
CADEIA : string [100];

procedure IMPRIVIDEO;
{... IMPRIMIR MATRIZ NO VIDEO ...}
var
{... VARIÁVEIS DE CONTROLE ...}
I,
J : integer;
{... CC = CARACTERE DA TECLA APERTADA NA IMPRESSAO VIA VIDEO ...}
CC : char;

Procedure MOSTRAMATRIZ;
{... MOSTRAR MATRIZ NO VIDEO ...}
var
{... VARIÁVEIS DE CONTROLE ...}
F : byte;
II,
IJ : integer;
begin { MOSTRAMATRIZ }
F := 1;
II := 1;
while (F <= 15) and (II <= L) do
begin
gotoxy(20,F + 6);
IJ := (II - 1) * C + J;
write ('A(',II:4,',',',J:4,') = ', A^[IJ]:24:10);
II := II + 1;
F := F + 1;
end; { while (F <= 15) and (II <= L) }
LIMPATELA (2,F + 6,78,23);
end; { MOSTRAMATRIZ }

begin { IMPRIVIDEO }
I := 1;
J := 1;

```

```

MOSTRAMATRIZ;
repeat
  textbackground (2);
  DIAGRAMA (3,23,'^E/^W/8/9 => Sobe  ^X,^C,2,3 => Dece');
  DIAGRAMA (39,23,'  ^S/^A/4 => Esq.  ^D/^F/6 => Dir. ');
  textbackground (0);
  KEYBOARD(CC);
  case CC of
    #52,#175,^S : if J > 1
                  then
                    J := J - 1;
    #54,#177,^D : if J < C
                  then
                    J := J + 1;
    #51,#181,^C : if I + 15 <= L
                  then
                    I := I + 15;
    #57,#173,^R : if I - 15 > 1
                  then
                    I := I - 15
                  else
                    I := 1;
    #56,#172,^E : if I > 1
                  then
                    I := I - 1;
    #50,#180,^X : if I < L
                  then
                    I := I + 1;
    ^F           : if J + 5 <= C
                  then
                    J := J + 5
                  else
                    J := C;
    ^A           : if J - 5 > 1
                  then
                    J := J - 5
                  else
                    J := 1;

    #13,#27 ;;
  else
    write (#7);
  end; { case CC }
MOSTRAMATRIZ;
until (CC in [#27,#13]);
end; { IMPRIVIDEO }

procedure IMPRESSORA;
(... IMPRIMIR MATRIZ NA IMPRESSORA ...)
var
[... VARIÁVEIS DE CONTROLE ...]
I,

```



```

J,
IJ      : integer;
{... CAR = CARACTERE QUE SE IGUAL A #27 (ESC) PARA A
... IMPRESSAO NA IMPRESSORA ...}
CAR      : char;
begin { IMPRESSORA }
  LIMPATELA (2,6,78,23);
  DIAGRAMA (20,16,'Nome para a matriz: ');
  read (CADEIA);
  QUADRO (32,12,45,15);
  DIAGRAMA (34,13,'Imprimindo');
  DIAGRAMA (34,14,'Impressora');
  textbackground (2);
  DIAGRAMA (26,23,'Pressione <ESC> p/ abortar ');
  textbackground (0);
  writeln (lst); writeln (lst);
  writeln (lst,CADEIA);
  writeln (lst);
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I - 1) * C + J;
          if abs (A^[IJ]) < 1e-5
            then
              A^[IJ] := 0;
          write (lst,A^[IJ]:CT:CD);
          if keypressed
            then
              begin
                read (kbd,CAR);
                if CAR = #27
                  then
                    begin
                      J := C;
                      I := L;
                    end; { if CAR = #27 }
                end; { if keypressed }
              J := J + 1;
            end; { while J <= C }
          writeln (lst);
          I := I + 1;
        end; { while I <= J }
      end; { IMPRESSORA }
    begin { IMPRESSAO }
      repeat
        QUADRO (30,11,47,17);
        DIAGRAMA (32,12,'IMPRESSAO...:');

```

```
DIAGRAMA (32,14,'1 - Video');
DIAGRAMA (32,16,'2 - Impressora ');
read (kbd,SAIDA);
LIMPATELA (2,6,78,23);
until SAIDA in ['1','2'];
if SAIDA = '1'
then
  IMPRIVIDEO
else
  begin
    DIAGRAMA (15,12,'Entre numero casas decimais... ');
    CD := trunc (NUMERO (46,12,78));
    DIAGRAMA (15,14,'Entre numero casas inteiras... ');
    CI := trunc (NUMERO (46,14,78));
    CT := CI + CD + 2;
    IMPRESSORA
  end; { else }
end; { impressao }
```

```

{... Arquivo ALGEPRES.PAS ...}
procedure PROESC;
{... MULTIPLICAR OU DIVIDIR MATRIZ POR CONSTANTE ...}
var
{... CT = CONSTANTE QUE MULTIPLICA OU DIVIDE A MATRIZ ...}
{... I,J,IJ = VARIÁVEIS DE CONTROLE ...}
{... TECLA = VARIÁVEL QUE DEFINE SE MULTIPLICA, OU DIVIDE ...}
  CT      : real;
  I,
  J,
  IJ      : integer;
  TECLA   : char;
begin { PROESC }
  QUADRO (18,11,60,15);
  DIAGRAMA (20,12,'Entre com a constante ');
  CT := NUMERO (43,12,59);
  repeat
    DIAGRAMA (20,14,'Multiplicar ou dividir por ela ? (M/D) ');
    read (kbd,TECLA);
    TECLA := upcase (TECLA);
    if (TECLA = 'D') and (CT = 0)
    then
      begin { if (TECLA = 'D') and (CT = 0) }
        DIAGRAMA (20,14,'Divisao impossivel !!!');
        LIMPALINHA (42,14,59);
        write (#7);
        TECLENTER (20,17);
        exit;
      end; { if (TECLA = 'D') and (CT = 0) }
  until (TECLA = 'M') or (TECLA = 'D');
  LIMPATELA (2,6,78,23);
  QUADRO (31,12,45,16);
  DIAGRAMA (33,13,'PROCESSANDO');
  DIAGRAMA (33,14,' PRODUTO P/');
  DIAGRAMA (33,15,' ESCALAR ');
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I-1) * C + J;
          case TECLA of
            'M': A^[IJ] := A^[IJ] * CT;
            'D': A^[IJ] := A^[IJ] / CT;
          end; { case TECLA }
          J := J + 1;
        end; { while J <= C }
        I := I + 1;
      end; { while I <= L }
    end; { PROESC }

```

```

{... Arquivo ALGEINVE.PAS ...}
procedure INVERSA;
type
  VET = array [1..90] of integer;
var
  {... FOLGA = TOLERANCIA ...}
  {... DET = DETERMINANTE DA MATRIZ ORIGINAL ...}
  {... TRACO = TRAÇO DA MATRIZ ORIGINAL ...}
  {... POSTO = POSTO DA MATRIZ ORIGINAL ...}
  FOLGA,
  DET,
  TRACO      : real;
  POSTO      : integer;

procedure MENSAGEM;
begin { MENSAGEM }
  QUADRO (31,12,48,15);
  DIAGRAMA (33,13,' Metodo');
  DIAGRAMA (33,14,' Gauss-Jordan');
end; { MENSAGEM }

procedure INVQUAD;
{... INVERTER MATRIZ QUADRADA ...}
var
  LINHA,
  COLUNA :VET;
  I,
  J,
  K,
  N,
  NF,
  KK,
  NK,
  IJ,
  IK,
  KI,
  JP,
  JK,
  KJ,
  JI,
  JQ,
  JR,
  IZ      :integer;
  MAIA,
  LIXO    :real;

procedure ZERAGEM;
begin
  for I := K + 1 To N Do
    For J := 1 To K Do
      begin

```

```

        IJ := N*(I-1)+J;
        A^[IJ] := 0;
        IJ := N * (J -1) + I;
        A^[IJ] := 0;
    end;
end;

```

```

procedure RETROCA;
Label Inicio;
begin
    K := NF;
    Inicio:
    K := K - 1;
    If K <= 0
    then
        exit;
    I := LINHA [K];
    if I > K
    then
        begin
            JQ := N * (K - 1);
            JR := N * (I - 1);
            for J := 1 To N Do
                begin
                    JK := JQ + J;
                    LIXO := A^[JK];
                    JI := JR + J;
                    A^[JK] := -A^[JI];
                    A^[JI] := LIXO;
                end;
            End;
        end;
    J := COLUNA [K];
    KI := K -N;
    for I := 1 To N Do
        begin
            KI := KI + N;
            LIXO := A^[KI];
            JI := KI - K + J;
            A^[KI] := -A^[JI];
            A^[JI] := LIXO;
        end;
    goto Inicio;
end;

```

```

begin
    MENSAGEM;
    N := L;
    FOLGA := 1E-8;
    POSTO := 0;
    TRACO := 0;
    for I := 1 To N Do

```

```

begin
  IJ := N * (I - 1) + 1;
  TRACO := TRACO + A^[IJ];
end;
DET := 1;
NK := -N;
for K := 1 To N Do
begin
  NK := NK + N;
  LINHA [K] := K;
  COLUNA [K] := K;
  KK := NK + K;
  MAIA := A^[KK];
  if MAIA > FOLGA
  then
    POSTO := POSTO + 1;
{... BUSCA DO MAIOR VALOR ...}

  for J := K To N Do
  begin
    IZ := N * (J - 1);
    for I := K To N Do
    begin
      IJ := IZ + I;
      if Abs (MAIA) < Abs (A^[IJ])
      then
        begin
          MAIA := A^[IJ];
          LINHA [K] := I;
          COLUNA [K] := J;
        end;
      end;
    end;
    if Abs (MAIA) <= FOLGA
    then
      begin
        DET := 0;
        ZERAGEM;
        NF := K;
        RETROCA;
        exit;
      end;
    end;
  end;

  J := LINHA [K];
  if J > K
  then
    begin
      KI := K - N;
      for I := 1 To N Do
        begin

```

```

{... TROCA DE LINHAS ...}
    KI := KI + N;
    LIXO := -A^[KI];
    JI := KI - K + J;
    A^[KI] := A^[JI];
    A^[JI] := LIXO;
end;
end;

I := COLUNA [K];
if I > K
then
begin
    JP := N * (I - 1);
    for J := 1 To N Do
begin
{... TROCA DE COLUNAS ...}
    JK := NK + J;
    JI := JP + J;
    LIXO := -A^[JK];
    A^[JK] := A^[JI];
    A^[JI] := LIXO;
end;
end;

{... DIVIDE COLUNA POR MENOS O PIVO ...}
for I := 1 To N Do
if I <> K
then
begin
    IK := NK + I;
    A^[IK] := A^[IK] / (-MAIA);
end;

for I := 1 To N Do
begin
    IK := NK + I;
    LIXO := A^[IK];
    IJ := I - N;
    For J := 1 To N Do
{... OPERA A MATRIZ ...}
begin
    IJ := IJ + N;
    if (I <> K) And (J <> K)
then
begin
    KJ := IJ - I + K;
    A^[IJ] := A^[IJ] + LIXO * A^[KJ];
end;
end;
end;
end;

```

```

KJ := K - N;
for J := 1 To N Do
begin
  KJ:=KJ+N;
  if J<>K
  then
    A^[KJ]:=A^[KJ]/MAIA;
end;

DET := DET * MAIA;
A^[KK] := 1 / MAIA;
end;

NF := N;
RETROCA;
end;

procedure INVRET1;
type
  POIB      = ^VETB;
  VETB      = array [1..NO] of real;
var
  B          : POIB;
  I,
  J,
  IJ,
  LAUX       : Integer;
begin { INVRET1 }
  I := L + 1;
  LAUX := L;
  L := C;
  while I <= L Do
  begin
    J := 1;
    While J <= C Do
    begin
      IJ := (I-1) * C + J;
      A^[IJ] := 0;
      J := J + 1;
    end; { While }
    I := I + 1;
  end; { While }
  INVQUAD;

  TESTAHEAP (Int(NO));
  if HEAPCHEIO
  then
    exit;
  new(B);
  L := C;
  C := LAUX;

```



```

II := 1;
I := 1;
while I <= L Do
begin
  J := 1;
  while J <= C Do
  begin
    IJ := (I - 1) * L + J;
    B^[IJ] := A^[IJ];
    II := II + 1;
    J := J + 1;
  end; { while J <= C }
  I := I + 1;
end; { while I <= L }
for II := 1 To L*C Do A^[II] := B^[II];
Dispose(B);
end; { INVRET1 }

```

```

procedure INVRET2;
type
  POIB      = ^VETB;
  VETB      = array [1..NO] of real;
  POINIC    = ^integer;

```

```

Var
  I,
  IO,
  J,
  IJ,
  CAUX,
  K      : Integer;
  B      : POIB;
  INIC   : POINIC;
begin { INVRET2 }

```

```

  TESTAHEAP (Int(NO));
  if HEAPCHEIO
  then
    exit;

```

```

  mark (INIC);
  new (B);
  CAUX := C;
  C := L;
  I := 1;
  IO := L * CAUX;
  while I <= IO Do
  begin
    B^[I] := A^[I];
    I := I + 1;
  end;
  K := 1;

```

```

I := 1;
while I <= L Do
begin
  J := 1;
  while J <= C Do
  begin
    IJ := (I-1) * C + J;
    If J > CAUX then A^[IJ]:=0
    else
    begin
      A^[IJ] := B^[K];
      K := K + 1;
    end;
    J := J + 1;
  end;
  I := I + 1;
end;
release (INIC);
INVQUAD;
C := L;
L := CAUX;
end; { INVRET2 }

begin { INVERSA }
  if L < C
  then
    INVRET1
  else
    if L > C
    then
      INVRET2
    else
      begin
        INVQUAD;
        if IMPDET = 'S'
        then
          begin
            QUADRO (23,17,55,21);
            DIAGRAMA (25,18,'Determinante: ');
            Write (DET:15);
            DIAGRAMA (25,19,'          Traco: ');
            Write (TRACO:15);
            DIAGRAMA (25,20,'          Posto: ');
            Write (POSTO:15);
            TECLENTER (25,22);
          end;
        end;
      end;
    LIMPATELA (2,6,23,78);
  end; { INVERSA }

```

```

{... Arquivo ALGEOPER.PAS ...}
procedure OPERACAO;
{... FAZ OPERAÇÕES TRADICIONAIS E NAO TRADICIONAIS COM LINHAS E
... COLUNAS ...}
{... H = OPÇÃO DA OPERAÇÃO A REALIZAR ...}
{... RES = OPÇÃO QUE CONFIRMA OU NAO O PRODUTO DE LINHAS OU
... COLUNAS POR CONSTANTE E ADIÇÃO ...}
{... RESP = OPÇÃO QUE DEFINE MAIS PRODUTO DE LINHAS OU COLUNAS
... POR CONSTANTE E ADIÇÃO OU NAO ...}
{... L1 = PRIMEIRA LINHA A TROCAR E PRIMEIRA LINHA A FAZER O
... PRODUTO POR CONSTANTE ...}
{... L2 = SEGUNDA LINHA A TROCAR E SEGUNDA LINHA A FAZER O
... PRODUTO POR CONSTANTE ...}
{... C1 = PRIMEIRA COLUNA A TROCAR E PRIMEIRA COLUNA A FAZER O
... PRODUTO POR CONSTANTE ...}
{... C2 = SEGUNDA COLUNA A TROCAR E SEGUNDA COLUNA A FAZER O
... PRODUTO POR CONSTANTE ...}
{... NL = NÚMERO DA LINHA A ELIMINAR ...}
{... NC = NÚMERO DA COLUNA A ELIMINAR ...}
{... IND = NÚMERO DA MEMÓRIA FIXA A JUSTAPOR ...}
{... I = VARIÁVEL DE CONTROLE ...}
{... E1 = PRIMEIRA CONSTANTE PARA O PRODUTO DE LINHAS OU COLUNAS
... E ADIÇÃO ...}
{... E2 = SEGUNDA CONSTANTE PARA O PRODUTO DE LINHAS OU COLUNAS
... E ADIÇÃO ...}
{... DIVISOR = CONSTANTE DIVISORA DE LINHA E COLUNA ...}
var
  H,
  RES,
  RESP          : char;
  L1,
  L2,
  C1,
  C2,
  NL,
  NC,
  IND,
  I              : integer;
  E1,
  E2,
  DIVISOR        : real;

procedure TROCLIN;
{... TROCA LINHAS ...}
var
{... J,L1J,L2J = VARIÁVEIS DE CONTROLE ...}
{... AUX = VARIÁVEL AUXILIAR PARA TROCA DE LINHAS ...}
  J,
  L1J,
  L2J          : integer;
  AUX          : real;

```

```
begin { TROCLIN }
```

```
  J := 1;
```

```
  while J <= C do
```

```
    begin { while J <= C }
```

```
      L1J := (L1 - 1) * C + J;
```

```
      L2J := (L2 - 1) * C + J;
```

```
      AUX := A^[L1J];
```

```
      A^[L1J] := A^[L2J];
```

```
      A^[L2J] := AUX;
```

```
      J := J + 1;
```

```
    end; { while J <= C }
```

```
end; { TROCLIN }
```

```
procedure TROCCOL;
```

```
{... TROCA COLUNAS ...}
```

```
var
```

```
{... I,IC1,IC2 = VARIÁVEIS DE CONTROLE ...}
```

```
{... AUX = VARIÁVEL AUXILIAR PARA TROCA DE COLUNAS ...}
```

```
  I,
```

```
  IC1,
```

```
  IC2    : integer;
```

```
  AUX    : real;
```

```
begin { TROCCOL }
```

```
  I := 1;
```

```
  while I <= C do
```

```
    begin { while I <= C }
```

```
      IC1 := (I - 1) * C + C1;
```

```
      IC2 := (I - 1) * C + C2;
```

```
      AUX := A^[IC1];
```

```
      A^[IC1] := A^[IC2];
```

```
      A^[IC2] := AUX;
```

```
      I := I + 1;
```

```
    end; { while I <= C }
```

```
end; { TROCCOL }
```

```
procedure OPERLIN;
```

```
{... PRODUTO DE CONSTANTES POR LINHAS E ADIÇÃO ...}
```

```
var
```

```
{... J,L1J,L2J = VARIÁVEIS DE CONTROLE ...}
```

```
  J,
```

```
  L1J,
```

```
  L2J    : integer;
```

```
begin { OPERLIN }
```

```
  J := 1;
```

```
  while J <= C do
```

```
    begin { while J <= C }
```

```
      L1J := (L1 - 1) * C + J;
```

```
      L2J := (L2 - 1) * C + J;
```

```
      A^[L2J] := E1 * A^[L1J] + E2 * A^[L2J];
```

```
      J := J + 1;
```

```
    end; { while J <= C }
```

```
end; ( OPERLIN )
```

```
procedure OPERCOL;
```

```
{... PRODUTO DE CONSTANTES POR COLUNAS E ADIÇÃO ...}
```

```
var
```

```
{... I,IC1,IC2 = VARIAVEIS DE CONTROLE ...}
```

```
I,
```

```
IC1,
```

```
IC2 : integer;
```

```
begin ( OPERCOL )
```

```
I := 1;
```

```
while I <= C do
```

```
begin ( while I <= C )
```

```
IC1 := (I - 1) * C + C1;
```

```
IC2 := (I - 1) * C + C2;
```

```
A^[IC2] := E1 * A^[IC1] + E2 * A^[IC2];
```

```
I := I + 1;
```

```
end; ( while I <= C )
```

```
end; ( OPERCOL )
```

```
procedure ELILIN;
```

```
{... ELIMINA LINHA ...}
```

```
var
```

```
{... II,I,J,IJ = VARIAVEIS DE CONTROLE ...}
```

```
II,
```

```
I,
```

```
J,
```

```
IJ : integer;
```

```
begin ( ELILIN )
```

```
LIMPATELA (10,10,74,21);
```

```
II := 1;
```

```
I := 1;
```

```
while I <= L do
```

```
begin ( while I <= L )
```

```
if (I = NL)
```

```
then
```

```
I := I + 1;
```

```
J := 1;
```

```
while J <= C do
```

```
begin ( while J <= C )
```

```
IJ := (I - 1) * C + J;
```

```
A^[IJ] := A^[IJ];
```

```
J := J + 1;
```

```
II := II + 1;
```

```
end; ( while J <= C )
```

```
I := I + 1;
```

```
end; ( while I <= L )
```

```
L := L - 1;
```

```
if L = 0
```

```
then
```

```
begin ( if L = 0 )
```

```

    C := 0;
    DIAGRAMA (15,15,'Nao ha mais linhas');
    TECLENTER (15,17);
    exit;
  end; { if L = 0 }
end; { ELILIN }

procedure ELICOL;
{... ELIMINA COLUNA ...}
var
{... II,I,J,IJ = VARIÁVEIS DE CONTROLE ...}
  II,
  I,
  J,
  IJ : integer;
begin { ELICOL }
  LIMPATELA (10,10,74,21);
  II := 1;
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          if (J <> NC)
            then
              begin { if (J <> NC) }
                IJ := (I - 1) * C + J;
                A^[IJ] := A^[IJ];
                II := II + 1;
              end; { if J <> NC }
            J := J + 1;
          end; { while J <= C }
        I := I + 1;
      end; { while I <= L }
    C := C - 1;
    if C = 0
      then
        begin { if C = 0 }
          L := 0;
          DIAGRAMA (15,15,'Nao ha mais colunas');
          TECLENTER (15,17);
          exit;
        end; { if C = 0 }
  end; { ELICOL }

procedure DIVILIN;
{... DIVIDE LINHA POR CONSTANTE ...}
var
{... J,NLJ = VARIÁVEIS DE CONTROLE ...}
  J,

```

```

NLJ      : integer;
begin { DIVILIN }
  J := 1;
  while J <= C do
    begin { while J <= C }
      NLJ := (NL - 1) * C + J;
      A^[NLJ] := A^[NLJ] / DIVISOR;
      J := J + 1;
    end; { while J <= C }
  end; { DIVILIN }

procedure DIVICOL;
{... DIVIDE COLUMNA POR CONSTANTE ...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  INC      : integer;
begin { DIVILIN }
  I := 1;
  while I <= C do
    begin { while I <= C }
      INC := (I - 1) * C + NC;
      A^[INC] := A^[INC] / DIVISOR;
      I := I + 1;
    end; { while I <= C }
  end; { DIVILIN }

procedure JUSTLIN;
{... JUSTAPOE MATRIZES DE MESMA DIMENSÃO LINHA ...}
type
  POIAUXI = ^VETAUXI;
  VETAUXI = array [1..NO] of real;
var
{... I,J,K,IJ,IK = VARIÁVEIS DE CONTROLE ...}
{... AUXI = VETOR AUXILIAR A JUSTAPOSIÇÃO ...}
{... LIMJUSLIN = VARIÁVEL AUXILIAR A TRANSFORMAÇÃO DE
... CARACTERE EM NÚMERO ...}
  I,
  J,
  K,
  IJ,
  IK      : integer;
  AUXI    : POIAUXI;
  LIMJUSLIN : ^integer;
begin { JUSTLIN }
  TESTAHEAP (int (NO));
  if HEAPCHEIO
  then
    exit;
  mark (LIMJUSLIN);
  new (AUXI);

```

```

I := 1;
while I <= L * C do
  begin { while I <= L * C }
    AUXI^[I] := A^[I];
    I := I + 1;
  end; { while I <= L * C }
I := 1;
while I <= L do
  begin { while I <= L }
    J := 1;
    while J <= (C + COL [IND]) do
      begin { while J <= (C + COL [IND]) }
        if J <= C
          then
            begin { if J <= C }
              K := 1;
              while K <= C do
                begin { while K <= C }
                  IJ := (I - 1) * (C + COL [IND]) + J;
                  IK := (I - 1) * C + K;
                  A^[IJ] := AUXI^[IK];
                  J := J + 1;
                  K := K + 1;
                end; { while K <= C }
              end { if J <= C }
            else
              begin { else }
                K := 1;
                while K <= COL [IND] do
                  begin { while K <= COL [IND] }
                    IJ := (I - 1) * (C + COL [IND]) + J;
                    IK := (I - 1) * COL [IND] + K;
                    case IND of
                      1 : A^[IJ] := M1^[IK];
                      2 : A^[IJ] := M2^[IK];
                      3 : A^[IJ] := M3^[IK];
                      4 : A^[IJ] := M4^[IK];
                      5 : A^[IJ] := M5^[IK];
                    end; { case IND }
                    K := K + 1;
                    J := J + 1;
                  end; { while K <= COL [IND] }
                end; { else if J <= C }
              end; { while J <= C + COL [IND] }
            I := I + 1;
          end; { while I <= L }
        C := C + COL [IND];
        release (LIMJUSLIN);
      end; { JUSTLIN }

procedure JUSTCOL;

```



```

{... JUSTAPOE  MATRIZES DE MESMA DIMENSSAO COLUNA ...}
var
{... I,J,K,IJ,KJ = VARIÁVEIS DE CONTROLE ...}
  I,
  J,
  K,
  IJ,
  KJ      : integer;
begin { JUSTCOL }
  K := 1;
  I := L + 1;
  while I <= (L + LIN [IND]) do
    begin { while I <= (L + LIN [IND]) }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I - 1) * C + J;
          KJ := (K - 1) * COL [IND] + J;
          case IND of
            1 : A^[IJ] := M1^[KJ];
            2 : A^[IJ] := M2^[KJ];
            3 : A^[IJ] := M3^[KJ];
            4 : A^[IJ] := M4^[KJ];
            5 : A^[IJ] := M5^[KJ];
          end; { case IND }
          J := J + 1;
        end; { while J <= C }
        K := K + 1;
        I := I + 1;
      end; { while I <= (L + LIN [IND]) }
      L := L + LIN [IND];
    end; { JUSTCOL }

begin { OPERACAO }
  if (L = 0) or (C = 0)
  then
    begin { if (L = 0) or (C = 0) }
      QUADRO (16,9,62,11);
      DIAGRAMA (18,10,'Matriz com numero de linhas ou colunas
                    zero');
      TECLENTER (18,13);
      exit;
    end; { if (L = 0) or (C = 0) }
  repeat
    QUADRO (13,07,65,18);
    DIAGRAMA (15,08,'0 - Trocar  linhas');
    DIAGRAMA (15,09,'1 - Trocar  colunas');
    DIAGRAMA (15,10,'2 - Produto de linhas por escalar e
                    adicao');
    DIAGRAMA (15,11,'3 - Produto de colunas por escalar e
                    adicao');
  end;

```

```

DIAGRAMA (15,12,'4 - Eliminar linhas');
DIAGRAMA (15,13,'5 - Eliminar colunas');
DIAGRAMA (15,14,'6 - Dividir linhas por escalar');
DIAGRAMA (15,15,'7 - Dividir colunas por escalar');
DIAGRAMA (15,16,'8 - Justapor matrizes de mesma dimensao
linha');
DIAGRAMA (15,17,'9 - Justapor matrizes de mesma dimensao
coluna ');

read (kbd,H);
until H in ['0'..'9'];
LIMPATELA (2,07,78,22);
if H in ['0'..'7']
then
  QUADRO (09,09,75,22);
case H of
  '0' : begin
    repeat
      DIAGRAMA (11,10,'Entre com as linhas a trocar :');
      DIAGRAMA (11,12,'Linha -----> ');
      L1 := trunc (NUMERO (27,12,74));
      DIAGRAMA (11,14,'Pela linha -> ');
      L2 := trunc (NUMERO (27,14,74));
      until (L1 in [1..L]) and (L2 in [1..L]);
      TROCLIN;
    end; { case H = '0' }
  '1' : begin
    repeat
      DIAGRAMA (11,10,'Entre com as colunas a trocar :');
      DIAGRAMA (11,12,'Coluna -----> ');
      C1 := trunc (NUMERO (28,12,74));
      DIAGRAMA (11,14,'Pela coluna -> ');
      C2 := trunc (NUMERO (28,14,74));
      until (C1 in [1..C]) and (C2 in [1..C]);
      TROCCOL;
    end; { case H = '1' }
  '2' : begin
    repeat
      repeat
        DIAGRAMA (11,10,'Entre com as constantes :');
        DIAGRAMA (11,12,'Constante 1 -> ');
        E1 := NUMERO (28,12,74);
        DIAGRAMA (11,14,'Constante 2 -> ');
        E2 := NUMERO (28,14,59);
        repeat
          DIAGRAMA (11,17,'Entre com as linhas :');
          DIAGRAMA (11,19,'Linha -> ');
          L1 := trunc (NUMERO (22,19,74));
          DIAGRAMA (11,21,'Linha -> ');
          L2 := trunc (NUMERO (22,21,74));
          until (L1 in [1..L]) and (L2 in [1..L]);
        LIMPATELA (10,10,74,21);
      end;
    end;
  end;
end;

```

```

    DIAGRAMA (10,14,' Operacao -> ');
    writeln (E1:6:2,' x Lin ',L1,' + ',E2:6:2,
            ' x Lin ',L2,' -> Lin ',L2);
    DIAGRAMA (24,16,' Confirma ? (S/N) ');
    read (kbd,RES);
    LIMPATELA (10,10,74,16);
    RES := upcase (RES);
until RES in ['S'];
OPERLIN;
repeat
    DIAGRAMA (24,10,'Mais operacao (S/N) ');
    read (kbd,RESP);
    RESP := upcase (RESP);
until RESP in ['S','N'];
LIMPATELA (10,10,74,12);
RESP := upcase (RESP);
until RESP in ['N']
end; { case H = '2' }
'3' : begin
    repeat
        repeat
            DIAGRAMA (11,10,'Entre com as constantes :');
            DIAGRAMA (11,12,'Constante 1 -> ');
            E1 := NUMERO (28,12,74);
            DIAGRAMA (11,14,'Constante 2 -> ');
            E2 := NUMERO (28,14,74);
            repeat
                DIAGRAMA (11,17,'Entre com as calunas :');
                DIAGRAMA (11,19,'Coluna -> ');
                C1 := trunc (NUMERO (23,19,74));
                DIAGRAMA (11,21,'Coluna -> ');
                C2 := trunc (NUMERO (23,21,74));
            until (C1 in [1..C1]) and (C2 in [1..C1]);
            LIMPATELA (10,10,74,21);
            DIAGRAMA (10,14,' Operacao -> ');
            writeln (E1:6:2,' x Col ',C1,' + ',E2:6:2,
                    ' x Col ',C2,' -> Col ',C2);
            DIAGRAMA (24,16,' Confirma ? (S/N) ');
            read (kbd,RES);
            LIMPATELA (10,10,74,16);
            RES := upcase (RES);
        until RES in ['S'];
        OPERCOL;
        repeat
            DIAGRAMA (11,10,'Mais operacao (S/N) ');
            read (kbd,RESP);
            RESP := upcase (RESP);
        until RESP in ['S','N'];
        LIMPATELA (10,10,74,12);
    until RESP in ['N']
end; { case H = '3' }

```

```

'4' : begin
    repeat
        repeat
            DIAGRAMA (11,10,'Entre com a linha a eliminar ->
                        ');
            NL := trunc (NUMERO (45,10,74));
        until NL in [1..L];
        ELILIN;
        if L = 0
        then
            exit;
        DIAGRAMA (11,20,'Mais eliminacao de linha ? (S/N)
                    ');
        read (kbd,RES);
        RES := upcase (RES);
        until RES in ['N'];
    end; { case H = '4' }
'5' : begin
    repeat
        repeat
            DIAGRAMA (11,10,'Entre com a coluna a eliminar
                        -> ');
            NC := trunc (NUMERO (46,10,74));
        until NC in [1..C];
        ELICOL;
        if C = 0
        then
            exit;
        DIAGRAMA (11,20,'Mais eliminacao de coluna ? (S/N)
                    ');
        read (kbd,RES);
        RES := upcase (RES);
        until RES in ['N'];
    end; { case H = '5' }
'6' : begin
    repeat
        repeat
            DIAGRAMA (11,10,'Entre com a linha a dividir -->
                        ');
            NL := trunc (NUMERO (45,10,74));
            DIAGRAMA (11,12,'Entre com a constante ----->
                        ');
            DIVISOR := NUMERO (45,12,74);
            if DIVISOR = 0
            then
                begin { if DIVISOR = 0 }
                    DIAGRAMA (11,14,'A constante nao pode ser
                                    nula ');
                    TECLENTER (11,16);
                end; { if DIVISOR = 0 }
            if (NL < 1) or (NL > L)

```

```

        then
            begin { if (NL < 1) or (NL > L) }
                DIAGRAMA (11,18,'Linha fora dos limites ');
                TECLENTER (11,20);
            end; { if (NL < 1) or (NL > L) }
        LIMPATELA (10,10,74,21);
    until (NL in [1..L]) and (DIVISOR <> 0);
    DIVILIN;
    DIAGRAMA (11,20,'Mais divisao de linha ? (S/N) ');
    read (kbd,RES);
    RES := upcase (RES);
    LIMPATELA (10,10,74,21);
until RES in ['N'];
end; { case H = '6' }
'7' : begin
    repeat
        repeat
            DIAGRAMA (11,10,'Entre com a coluna a dividir -->
                ');
            NC := trunc (NUMERO (46,10,74));
            DIAGRAMA (11,12,'Entre com a constante ----->
                ');
            DIVISOR := NUMERO (45,12,74);
            if DIVISOR = 0
                then
                    begin { if DIVISOR = 0 }
                        DIAGRAMA (11,14,'A constante nao pode ser
                            nula ');
                        TECLENTER (11,16);
                    end; { if DIVISOR = 0 }
            if (NC < 1) or (NC > C)
                then
                    begin { if (NC < 1) or (NC > C) }
                        DIAGRAMA (11,18,'Coluna fora dos limites');
                        TECLENTER(11,20);
                    end; { if (NC < 1) or (NC > C) }
            LIMPATELA (10,10,74,21);
        until (NC in [1..C]) and (DIVISOR <> 0);
        DIVICOL;
        DIAGRAMA (11,20,'Mais divisao de coluna ? (S/N).');
        read (kbd,RES);
        RES := upcase (RES);
        LIMPATELA (10,10,74,21);
    until RES in ['N'];
end; { case H = '7' }
'8' : begin
    LIN [6] := L;
    COL [6] := C;
    QUADRO (13,7,67,17);
    DIAGRAMA (15,8,'Memoria - Nome
        Linhas - Colunas');

```

```

DIAGRAMA (15,9,'=====
=====');
I := 1;
while I <= 6 do
begin { while I <= 6 }
  DIAGRAMA (20,9 + I,' ');
  writeln (I,' - ',MEMO [I].NOME);
  gotoxy (50,9 + I);
  write (LIN [I]);
  gotoxy (59,9 + I);
  write (COL [I]);
  I := I + 1;
end; { while I <= 6 }
QUADRO (13,19,54,22);
repeat
  DIAGRAMA (15,20,'Justapor (linha) com ---> ');
  IND := trunc (NUMERO(42,20,53));
until IND in [1..5];
if ((C + COL [IND]) * L <= NO) and (L = LIN [IND])
then
begin
  JUSTLIN;
end { if ((C + COL [IND]) * L <= NO) and
      (L = LIN [IND]) }
else
begin
  if (C + COL [IND]) * L > NO
  then
    DIAGRAMA (15,21,'A justaposicao ultrapassa
as dimensoes');
    if L <> LIN [IND]
    then
      DIAGRAMA (15,21,'Dimensoes linhas
incompativeis');
      TECLENTER (15,23);
    end; { else }
  end; { case H = '8' }
'9' : begin
  LIN [6] := L;
  COL [6] := C;
  QUADRO (13,7,67,17);
  DIAGRAMA (15,8,'Memoria - Nome -
Linhas - Colunas');
  DIAGRAMA (15,9,'=====
=====');
  I := 1;
  while I <= 6 do
begin { while I <= 6 }
  DIAGRAMA (20,9 + I,' ');
  writeln (I,' - ',MEMO [I].NOME);
  gotoxy (50,9 + I);

```

```

        write (LIN [I]);
        gotoxy (59,9 + I);
        write (COL [I]);
        I := I + 1;
    end; { while I <= 6 }
    QUADRO (13,19,54,22);
    repeat
        DIAGRAMA (15,20,'Justapor (coluna) com --> ');
        IND := trunc (NUMERO(42,20,53));
    until IND in [1..5];
    if ((L + LIN [IND]) * C <= NO) and (C = COL [IND])
    then
        begin {if ((L + LIN [IND]) * C <= NO) and
            (C = COL [IND]) }
            JUSTCOL;
        end { if ((L + LIN [IND]) * C <= NO) and
            (C = COL [IND]) }
    else
        begin
            if (L + LIN [IND]) * C > NO
            then
                DIAGRAMA (15,21,'A justaposicao ultrapassa
                    as dimensoes');
                if C <> COL [IND]
                then
                    DIAGRAMA (15,21,'Dimensoes colunas
                        incompativeis');
                    TECLENTER (15,23);
                end; { else }
            end; { case H = '9' }
        end; { case H }
    end; { OPERACAO }

```

```

{... Arquivo ALGEPART.PAS ...}
procedure PARTICIONAR;
{... SELECIONA UM BLOCO DA MATRIZ ...}
type
  POIX      = ^VETX;
  VETX      = array [1..NO] of real;
  POINI     = ^integer;
var
  {... L1 = LINHA INICIAL DO BLOCO A SELECIONAR ...}
  {... L2 = LINHA FINAL DO BLOCO A SELECIONAR ...}
  {... C1 = COLUNA INICIAL DO BLOCO A SELECIONAR ...}
  {... C2 = COLUNA FINAL DO BLOCO A SELECIONAR ...}
  {... INDL = NÚMERO DE LINHAS DO BLOCO SELECIONADO ...}
  {... INDC = NÚMERO DE COLUNAS DO BLOCO SELECIONADO ...}
  {... I,J,II,IJ = VARIÁVEIS DE CONTROLE ...}
  {... X = VETOR AUXILIAR DO TIPO POIX ...}
  {... INI = VARIÁVEL AUXILIAR NA LIBERAÇÃO DE VETORS DO TIPO
  ... PONTEIRO ...}
  L1,
  L2,
  C1,
  C2,
  INDL,
  INDC,
  I,
  J,
  II,
  IJ      : integer;
  X       : POIX;
  INI     : POINI;
begin { PARTICIONAR }
  TESTAHEAP (int (NO));
  if HEAPCHEIO
  then
    exit;
  mark (INI);
  new (X);
  repeat
    LIMPATELA (2,6,78,23);
    QUADRO (22,09,72,19);
    DIAGRAMA (24,10,'Linha inicial... ');
    L1 := trunc (NUMERO (41,10,71));
    DIAGRAMA (24,11,'Linha final.... ');
    L2 := trunc (NUMERO (41,11,71));
    if not (L1 in [1..L]) or not (L2 in [1..L])
    then
      begin { if not (L1 in [1..L]) or not (L2 in [1..L]) }
        gotoxy (24,13);
        write ('Numero de linha fora do intervalo : [',1,'...',
              L,',']');
        TECLENTER (24,20);
      end
  end

```



```

        end; { if not (L1 in [1..L]) or not (L2 in [1..L]) }
if L1 > L2
then
    begin { if L1 > L2 }
        DIAGRAMA (24,15,'Numero de linha invertido');
        TECLENTER (24,20);
    end; { if L1 > L2 }
until (L1 <= L2) and (L1 in [1..L]) and (L2 in [1..L]);
INDL := L2 - L1 + 1;
repeat
    DIAGRAMA (24,13,'Coluna inicial..: ');
    C1 := trunc (NUMERO (42,13,71));
    DIAGRAMA (24,14,'Coluna final....: ');
    C2 := trunc (NUMERO (42,14,71));
    if not (C1 in [1..C]) or not (C2 in [1..C])
    then
        begin { if not (C1 in [1..C]) or not (C2 in [1..C]) }
            gotoxy (24,16);
            write ('Numero de coluna fora do intervalo : [',1,',',
                C,']');
            TECLENTER (24,20);
            LIMPALINHA (24,16,71);
        end; { if not (C1 in [1..C]) or not (C2 in [1..C]) }
    if C1 > C2
    then
        begin { if C1 > C2 }
            DIAGRAMA (24,18,'Numero de coluna invertido');
            TECLENTER (24,20);
            LIMPALINHA (24,18,71);
        end; { if C1 > C2 }
until (C1 <= C2) and (C1 in [1..C]) and (C2 in [1..C]);
LIMPATELA (2,6,78,23);
QUADRO (30,12,46,14);
DIAGRAMA (32,13,'PARTICIONANDO');
INDC := C2 - C1 + 1;
II := 1;
I := L1;
while I <= L2 do
    begin { while I <= L2 }
        J := C1;
        while J <= C2 do
            begin { while J <= C2 }
                IJ := (I - 1) * C + J;
                X^[II] := A^[IJ];
                II := II + 1;
                J := J + 1;
            end; { while J <= C2 }
            I := I + 1;
        end; { while I <= L2 }
    I := 1;
while I <= INDL do

```

```
begin { while I <= INDL }  
  J := 1;  
  while J <= INDC do  
    begin { while J <= INDC }  
      IJ := (I - 1) * INDC + J;  
      A^[IJ] := X^[IJ];  
      J := J + 1;  
    end; { while J <= INDC }  
    I := I + 1;  
  end; { while I <= INDL }  
  L := INDL;  
  C := INDC;  
  release (INI)  
end; { PARTICIONAR }
```

```

{... Arquivo ALGEAPAG.PAS ...}
procedure APAGAR;
{... APAGA ARQUIVO TIPO TEXTO ...}
var
{... NOME_ARQ = NOME DO ARQUIVO A APAGAR DADO PELO USUARIO ...}
{... ARQUIVO = NOME DO ARQUIVO DECLARADO NO ALGEMA ...}
{... CON = RESPOSTA DE CONFIRMAÇÃO SE APAGA OU NAO ...}
    NOME_ARQ    : STR100;
    ARQUIVO     : text;
    CON         : char;
begin { APAGAR }
    QUADRO (11,09,73,13);
    DIAGRAMA (13,10,'Entre drive,(:) e nome do arquivo a apagar ...:
                ');
    read (NOME_ARQ);
    {$i-}
        assign (ARQUIVO,NOME_ARQ);
        reset (ARQUIVO);
    {$i+}
    if ioresult = 0
    then
        begin { if ioresult = 0 }
            close(ARQUIVO);
            repeat
                DIAGRAMA (13,12,'Confirma ? (S/N) ');
                read (kbd,CON);
                CON := upcase (CON);
            until CON in ['S','N'];
            if CON in ['S']
            then
                begin { if CON in ['S'] }
                    LIMPATELA (2,6,78,23);
                    QUADRO (32,12,43,15);
                    DIAGRAMA (34,13,'APAGANDO');
                    DIAGRAMA (34,14,'ARQUIVO ');
                    erase (ARQUIVO);
                end; { if CON in ['S'] }
            end { if ioresult = 0 }
        else
            begin { else }
                close(ARQUIVO);
                gotoxy (13,12);
                write ('ARQUIVO "',NOME_ARQ,'" INEXISTENTE');
                TECLENTER (13,15);
            end; { else }
        end; { APAGAR }

```

```

{... Arquivo ALGECONT.PAS ...}
procedure CONTEUDO;
{... MOSTRA O CONTEUDO DAS MEMORIAS FIXAS ...}
var
{... I = VARIÁVEL DE CONTROLE ...}
  I      : integer;
begin { CONTEUDO }
  QUADRO (14,08,66,17);
  DIAGRAMA (16,09,'= Memoria -           Nome           - Linha -
                  Coluna =');
  DIAGRAMA (16,10,'=====
                  ==');
  I := 1;
  LIN [6] := L;
  COL [6] := C;
  while I <= 6 do
    begin { while I <= 6 }
      gotoxy (16,10 + I);
      write ('= ',I,' ',MEMO [I].NOME);
      gotoxy (49,10 + I);
      write (LIN [I]);
      gotoxy (57,10 + I);
      write (COL [I]);
      gotoxy (64,10 + I);
      writeln ('=');
      I := I + 1;
    end; { while I <= 6 }
  TECLENTER (23,21);
end; { CONTEUDO }

```

```

{... Arquivo ALGEDIRE ...}
overlay procedure DIRETORIO;
{... MOSTRA NA TELA O DIRETORIO DO DISCO FLEXIVEL OU RIGIDO ...}
type
  Arq12      = array[1..47] of char;
  RegRec     = record
                    AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags : Integer;
                end;
  Str25      = String[25];
  Str100     = String[100];

Var
  Regs       : Regrec;
  Mascara    : Arq12;
  DTA        : array[1..43] of Byte;
  Nome       : Str25;
  Tracks,
  TotalTracks,
  Drive,
  Bytes,
  Sectors,
  Erro,
  i,
  k,
  l,
  col        : integer;
  NomeDrive,
  Car        : char;
  BytesUsados,
  Used,TotalBytes : Real;

begin { DIRETORIO }
  k:=1;
  l:=8;
  FillChar(DTA,SizeOf(DTA),0);
  FillChar(Mascara,SizeOf(Mascara),0);
  FillChar(Nome,SizeOf(Nome),0);
  Regs.AX:=$1A00;
  Regs.DS:=Seg(DTA);
  Regs.DX:=Ofs(DTA);
  MSDos(Regs);
  LIMPATELA (2,6,74,23);
  DIAGRAMA (3,10,'Entre com o Drive ..... ');
  readln(mascara);
  mascara[1] := UpCase(Mascara[1]);
  if (length(mascara) > 0) and (mascara[1] = 'A') or
  (mascara[1]='B') or (Mascara[1] = 'C') then
  begin { if }
    LIMPATELA (2,10,78,11);
    if (pos(':',mascara)>0) and (pos('.',mascara)=0) then
    begin { if }

```

```

l := pos(' ', mascara);
mascara[l] := '\';
mascara[l+1] := '*';
mascara[l+2] := '.';
mascara[l+3] := '*';
l := 8;
end; { if }
if (pos(':', mascara)=0) or (pos('.', mascara)=0) then
begin { if }
  mascara[2] := ':';
  mascara[3] := '*';
  mascara[4] := '.';
  mascara[5] := '*';
end; { if }
gotoxy (2, L-2);
write (' Diretorio do drive ', Mascara);
QUADRO (2, L - 1, 78, 21);
Regs.AX := $4E00;
Regs.DS := Seg(Mascara);
Regs.DX := OfS(Mascara);
Regs.CX := 22;
MSDos(Regs);
Erro := Regs.AX and $FF;
I := 1;
If (Erro=0) then
repeat
Nome[i] := Char(Mem[Seg(DTA):OfS(DTA)+29+I]);
I := I+1;
until not (Nome[i-1] in [' ' '..' '~']) or (I>20);
Nome[0] := char(i-1);
While (Erro = 0) do
begin { While (Erro = 0) }
  Erro := 0;
  Regs.AX := $4F00;
  Regs.CX := 22;
  MSdos(Regs);
  Erro := Regs.AX and $FF;
  I := 1;
  repeat
    Nome[i] := chr(Mem[Seg(DTA):OfS(DTA)+29+I]);
    I := I+1;
  until not (Nome[i-1] in [' ' '..' '~']) or (I>20);
  Nome[0] := chr(i-1);
  if (Erro=0) then
  begin { if (Erro=0) }
    col := (k-1)*15+3; gotoxy(col+1, l);
    Write(Nome);
    k := k+1;
    if k=6 then
    begin { if k=6 }
      k := 1;

```

```

l:=l+1;
if l=23 then
begin { l=23 }
  TECLENTER (4,22);
  for i := 7 to 22 do
  begin { for i := 7 to 22 }
    gotoxy(2,i);
    clreol;
  end; { for i := 7 to 22 }
  gotoxy(2,22);
  clreol;
  l:=7;
end; { if l=23 }
end; { if k=6 }
end; { if (Erro=0) }
end; { While (Erro =0) }
Drive := Ord(Mascara[1])-64;
if Drive > 3 then
begin { if Drive > 3 }
  GetDir(0, Nome);
  Drive := Ord(Nome[1])-64;
end; { if Drive > 3 }
Regs.AX := $3600;
Regs.DX := Drive;
MsDos(Regs);
Tracks := Regs.BX;
TotalTracks := Regs.DX;
Bytes := Regs.CX;
Sectors := Regs.AX;
Used := ((TotalTracks-Tracks)/TotalTracks)*100;
TotalBytes := ((Sectors*Bytes*1.0)*Tracks);
BytesUsados:=((Sectors*Bytes*1.0)*(TotalTracks-Tracks));
gotoxy(3,23);
write(TotalBytes:14:0, ' Bytes livres ');
TECLENTER (40,23);
end; { if }
end; { DIRETORIO }

```

```

{... Arquivo ALGEBIBLI.PAS ...}
procedure KEYBOARD (var CAR : char);
begin { KEYBOARD }
    inline ($9C/$B8/$00/$00/$CD/$16/$A2/KEYASCII/$88/$26/KEYSCAN/
           $9D);
    CAR := chr(KEYASCII);
    if KEYASCII = 0 then
        car := chr (KEYSCAN + 100);
end; { KEYBOARD }

procedure LIMPATELA (ESQ,ALTO,DIR,BAIXO : byte);
{... LIMPAR A TELA ...}
{... ESQ = COLUNA INICIAL PARA LIMPAR ...}
{... ALTO = LINHA INICIAL PARA LIMPAR ...}
{... DIR = COLUNA FINAL PARA LIMPAR ...}
{... BAIXO = LINHA FINAL PARA LIMPAR ...}
begin { LIMPATELA }
    window (ESQ,ALTO,DIR,BAIXO);
    clrscr;
    window (1,1,79,23);
end; { LIMPATELA }

procedure TESTAHEAP (X : real);
{... TESTA CAPACIDADE DA MEMORIA HEAP ...}
var
    MEMORIA : real;
begin { TESTAHEAP }
    HEAPCHEIO := false;
    if memavail >= 0
    then
        MEMORIA := int (memavail)
    else
        MEMORIA := 65535. + int (memavail);
    if int (16 * MEMORIA/6) < X
    then
        begin
            LIMPATELA (2,6,78,22);
            gotoxy (25,11);
            write ('RAM insuficiente ');
            HEAPCHEIO := true;
            read;
        end;
end; { TESTAHEAP }

procedure RISCO (TIPO : char; ALTURA,COMEÇO,FIM : byte; SIMBOLO :
char);
{... FAZ RISCO CONTINUO NA TELA ...}
var
    I : byte;
begin { RISCO }
    if (TIPO = 'H') or (TIPO = 'h')

```



```

    then
      for I := COMECO to FIM do
        begin { I := COMECO to FIM }
          gotoxy (I,ALTURA);
          write (#205);
        end; { I := COMECO to FIM }
      if (TIPO = 'V') or (TIPO = 'v')
      then
        for I := COMECO to FIM do
          begin { I := COMECO to FIM }
            gotoxy (ALTURA,I);
            write (#186);
          end; { I := COMECO to FIM }
        end; { RISCO }

procedure QUADRO (X0,Y0,X1,Y1 : byte);
{... FAZ QUADRO NA TELA ...}
{... X0 = COLUNA INICIAL DE FORMAÇÃO DO QUADRO ...}
{... Y0 = LINHA INICIAL DE FORMAÇÃO DO QUADRO ...}
{... X1 = COLUNA FINAL DE FORMAÇÃO DO QUADRO ...}
{... Y1 = LINHA FINAL DE FORMAÇÃO DO QUADRO ...}
begin { QUADRO }
  gotoxy (X0,Y0);
  write (#201);
  RISCO ('H',Y0,X0+1,X1-1,#176);
  gotoxy (X1,Y0);
  write (#187);
  RISCO ('V',X1,Y0+1,Y1-1,#176);
  gotoxy (X1,Y1);
  write (#188);
  RISCO ('H',Y1,X0+1,X1-1,#176);
  gotoxy (X0,Y1);
  write (#200);
  RISCO ('V',X0,Y0+1,Y1-1,#176);
end; { QUADRO }

procedure DIAGRAMA(X,Y : byte; NOME : STR100);
{... FAZ IMPRESSÃO DE CARACTERES. USO INTERNO
... DOS ARQUIVOS DO ALGEMA...}
begin { DIAGRAMA }
  gotoxy(X,Y);
  write (nome);
end; { DIAGRAMA }

procedure CABECALHO;
{... IMPRIME CABEÇALHO DO TOPO ...}
begin { CABECALHO }
  QUADRO (1,1,79,24);
  QUADRO (18,2,60,4);
  DIAGRAMA (20,3,'ALGEMA - Sistema de Algebra de Matrizes ');
end; { CABECALHO }

```

```

procedure TECLENTER (X,Y:byte);
{... IMPOE UMA PAUSA NA EXECUCAO DO PROGRAMA ...}
{... X = COLUNA INICIAL PARA IMPRIMIR MENSAGEM ...}
{... Y = LINHA FINAL PARA IMPRIMIR MENSAGEM ...}
var
  LIXOS : char;
begin { TECLENTER }
  DIAGRAMA (X,Y,'Tecle <ENTER> para continuar... ');
  write (#7);
  repeat
    read (kbd,LIXOS);
  until LIXOS = #13;
end; { TECLENTER }

function NUMERO (LI,CO,COL : integer) : real;
{... FAZ LEITURA DE DADOS NUMERICOS,
... COM DEFESA PARA NAO NUMERICOS ...}
{... LI = LINHA ONDE DEVE FICAR O CURSOR ...}
{... CO = COLUNA ATÉ ONDE DEVE SER LIMPA A LINHA ...}
{... COL = COLUNA ONDE DEVE FICAR O CURSOR ...}
var
  NUM : real;
begin { function NUMERO }
  repeat
    gotoxy (LI,CO);
    window (2,2,COL,23);
    clreol;
    window (1,1,79,23);
    NUM := 0;
    {$i-}
    read (NUM);
    {$i+}
  until ioresult = 0;
  NUMERO := NUM;
end; { function NUMERO }

procedure LIMPALINHA (REX,REY,COL : integer);
{... FAZ LIMPEZA DE LINHA ...}
{... REX = NUMERO DA COLUNA A INICIAR A LIMPEZA ...}
{... REY = NUMERO DA LINHA A INICIAR A LIMPEZA ...}
{... COL = NUMERO DA COLUNA ONDE FINALIZAR A LIMPEZA ...}
begin { LIMPALINHA }
  gotoxy (REX,REY);
  window (2,2,COL,23);
  clreol;
  window (1,1,79,23);
end; { LIMPALINHA }

procedure SIMETRIA;
{... REALIZA TESTE DE SIMETRIA ...}
var

```

```

{... VARIÁVEIS DE CONTROLE ...}
I,
J,
IJ,
JI : integer;
begin
  QUADRO (32,12,44,15);
  DIAGRAMA (34,13,'Testando');
  DIAGRAMA (34,14,'Simetria ');
  NULA := true;
  if L = C
  then
    begin
      SIM := true;
      I := 1;
      while I <= C do
        begin
          J := I;
          while J <= C do
            begin
              IJ := (I-1) * C + J;
              JI := (J-1) * C + I;
              if abs (A^[IJ] - A^[JI]) > 1e-5 * abs (A^[JI]) >
                1e-5
              then
                begin
                  SIM := false;
                  J := C + 1;
                  I := C + 1;
                end; { if }
                J := J + 1;
              end; { while J <= C }
              I := I + 1;
            end; { while I <= C }
          end { if L = C }
        else SIM := false;
        I := 1;
        while I <= (C * C) do
          begin
            if abs (A^[I]) > 1e-5
            then
              begin
                NULA := false;
                I := (C * C) + 1;
              end;
            I := I + 1;
          end; { while I <= (C * C) }
        LIMPATELA (2,12,78,15);
        if (SIM and AVISIM)
        then
          begin

```

```

        QUADRO (18,11,36,13);
        DIAGRAMA (20,12,'Matriz simetrica');
        TECLENTER (20,15);
    end;
    if ((SIM = false) and AVISIM)
    then
        begin
            QUADRO (18,11,40,13);
            DIAGRAMA (20,12,'Matriz nao simetrica');
            TECLENTER (20,15);
        end;
    AVISIM := false;
end; { SIMETRIA }

```

```

procedure NORMALIZA;
{... NORMALIZA AS COLUNAS DA MATRIZ ...}
var
{... VARIÁVEIS DE CONTROLE ...}
    J,
    I,
    K,
    IJ,
    KJ      : integer;
{... SOMAUX = SOMA DE QUADRADO DE LINHA ...}
    SOMAUX   : real;
begin { NORMALIZA }
    LIMPATELA (2,6,78,21);
    QUADRO (32,12,47,14);
    DIAGRAMA (34,13,'Normalizando');
    J := 1;
    while J <= C do
    begin
        SOMAUX := 0;
        I := 1;
        while I <= L do
        begin
            IJ := (I - 1) * C + J;
            SOMAUX := SOMAUX + sqr (A^[IJ]);
            I := I + 1;
        end;
        if abs (SOMAUX) > 1e-6
        then
            begin
                K := 1;
                while K <= L do
                begin
                    KJ := (K - 1) * C + J;
                    A^[KJ] := A^[KJ] / sqrt (SOMAUX);
                    K := K + 1;
                end;
            end;
    end;
end;

```

```
      J := J + 1;  
    end;  
end; { NORMALIZA }
```

```

{... Arquivo ALGEAUTO.PAS ...}
overlay procedure ESPECTRO;
{... CALCULA AUTOVALORES AUTOVETORES E K-ESIMA POTENCIA DE
... UMA MATRIZ REAL E SIMETRICA ...}
const
{... NOV = NUMERO DE ELEMENTOS DE VETORES AUXILIARES ...}
{... TOL = TOLERANCIA ...}
  NOV      = 100;
  TOL      = 2.5E-14;
type
  POIU      = ^MATU;
  MATU      = array [1..NOV] of real;
  POIRT     = ^VETP;
  VETP      = array [1..NOV] of real;
  POIDG     = ^MATDG;
  MATDG     = array [1..NOV] of real;
  POISPD    = ^MATSPD;
  MATSPD    = array [0..NOV] of real;
  POIV      = ^MATV;
  MATV      = array [1..NOV] of real;
  POIP      = ^MATP;
  MATP      = array [1..NOV] of real;
  POISN     = ^MATSN;
  MATSN     = array [0..NOV] of real;
  POICS     = ^MATCS;
  MATCS     = array [0..NOV] of real;
  POIT2     = ^MATT2;
  MATT2     = array [1..NOV] of real;
var
{... I,J,K,KJ,KI,JK,II,IK,IJ,IKK,JJ = VARIÁVEIS DE CONTROLE ...}
{... POSTO = É O POSTO DA MATRIZ ORIGINAL ...}
{... X,Y,NORM,EPS = VARIÁVEIS AUXILIARES ...}
{... DET = DETERMINANTE DA MATRIZ ...}
{... TRACO = TRAÇO DA MATRIZ ...}
{... OPC = OPÇÃO DE RESULTADO DO CÁLCULO ...}
{... U,DG,SPD,RT,V,P,SN,CS,T2 = VETORES AUXILIARES ...}
{... LIMPAUTO = VARIÁVEL AUXILIAR PARA LIBERAÇÃO DE VETORES
... DO TIPO PONTEIRO ...}
  I,
  J,
  K,
  KJ,
  KI,
  JK,
  II,
  IK,
  IJ,
  IKK,
  JJ,
  POSTO      : Integer;
  X,

```

```

Y,
NORM,
EPS,
DET,
TRACO           : Real;
CH,
OPC             : Char;
U               : POIU;
DG              : POIDG;
SPD             : POISPD;
RT              : POIRT;
V               : POIV;
P               : POIP;
SN              : POISN;
CS              : POICS;
T2              : POIT2;
LIMPAUTO        : ^integer;

```

```

procedure PAVET;
{... PASSA AUTOVETORES PARA A MEMORIA DE TRABALHO ...}
begin { PAVET }
  if L > 1
  then
    begin { if L > 1 }
      LIMPATELA (2,6,78,23);
      QUADRO (30,12,45,15);
      DIAGRAMA (32,13,'Chamando');
      DIAGRAMA (32,14,'Autovetores ');
      for I := 1 to L do
        for J := 1 to L do
          begin { for J := 1 to L }
            IJ := (I - 1) * C + J;
            A^[IJ] := U^[IJ];
          end; { for J := 1 to L }
        end; { if L > 1 }
      else
        A^[1] := 0;
    end; { PAVET }

```

```

procedure PAVAL;
{... PASSA AUTOVALORES PARA A MEMORIA DE TRABALHO ...}
begin { PAVAL }
  LIMPATELA (2,6,78,23);
  QUADRO (30,12,45,15);
  DIAGRAMA (32,13,'Chamando');
  DIAGRAMA (32,14,'Autovalores ');
  I := 1;
  II := 0;
  while I <= L do
    begin { while I <= L }
      II := II + 1;

```

```

J := 1;
while J <= C do
  begin { while J <= C }
    IJ := (I - 1) * C + J;
    if I = J
      then
        A^[IJ] := RT^[III]
      else
        A^[IJ] := 0;
    J := J + 1;
  end; { while J <= C }
  I := I + 1;
end; { while I <= L }
end; { PAVAL }

```

```

procedure DETRAPO;
{... CALCULA O DETERMINANTE, TRAÇO E POSTO DA MATRIZ ...}
begin { DETRAPO }
  DET := 1;
  POSTO := 0;
  TRACO := 0;
  I := 1;
  while I <= L do
    begin { while I <= L }
      DET := DET * RT^[I];
      TRACO := TRACO + RT^[I];
      if abs (RT^[I]) > 1e-5
        then
          POSTO := POSTO + 1;
      I := I + 1;
    end; { while I <= L }
  end; { DETRAPO }

```

```

procedure POTENCIA;
{... CALCULA A K-ÉSIMA POTENCIA DA MATRIZ ...}
var
{... P = POTENCIA REAL ...}
{... NEGATIVO = INFORMA, QUANDO VERDADEIRA ,SE HÁ AUTOVALOR ... NEGATIVO ...}
{... I,J,K,IJ,IK,JK = VARIÁVEIS DE CONTROLE ...}
  P          : real;
  NEGATIVO   : boolean;
  I,
  J,
  K,
  IJ,
  IK,
  JK         : integer;
begin { POTENCIA }
  LIMPATELA (2,6,78,23);
  NEGATIVO := false;

```



```

I := 1;
while I <= C do
begin { while I <= C }
  if (RT^[I] < -1e-7)
  then
    NEGATIVO := true;
    I := I + 1;
  end; { while I <= C }
if NEGATIVO = false
then
begin { if NEGATIVO = false }
  QUADRO (8,11,56,18);
  DIAGRAMA (10,12,'Entre com a k-esima potencia real ');
  P := NUMERO (46,12,55);
  LIMPATELA (2,6,78,23);
  QUADRO (30,12,49,15);
  DIAGRAMA (32,13,'Processando');
  DIAGRAMA (32,14,'K-esima potencia');
  K := 1;
  while K <= C do
    begin { while K <= C }
      V^[K] := 0;
      if abs (RT^[K]) >= 1E-7
      then
        V^[K] := exp (P * ln (abs (RT^[K])));
      K := K + 1;
    end; { while K <= C }
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I-1) * C + J;
          A^[IJ] := 0;
          K := 1;
          while K <= C do
            begin { while K <= C }
              IK := (I - 1) * C + K;
              JK := (J - 1) * C + K;
              A^[IJ] := A^[IJ] + U^[IK] * V^[K] * U^[JK];
              K := K + 1;
            end; { while K <= C }
            J := J + 1;
          end; { while J <= C }
          I := I + 1;
        end; { while I <= L }
      end { if NEGATIVO = false }
    else
      begin { else }
        LIMPATELA (2,6,78,23);
      end;
    end;
  end;
end;

```

```

    QUADRO (30,11,53,14);
    DIAGRAMA (32,12,'Autovalor negativo');
    DIAGRAMA (32,13,'Potencia indefinida ');
    TECLENTER (32,17);
  end; { else }
end; { POTENCIA }

```

```

procedure TRIDI;
{... TRIDIAGONALIZA A MATRIZ A POR HOUSEHOLDER ...}
var
{... CE,T,X,Y = VARIÁVEIS AUXILIARES ...}
{... I,J,K = VARIÁVEIS DE CONTROLE ...}
  CE,
  T,
  X,
  Y
  :Real;
  I,
  J,
  K
  :Integer;

```

```

procedure STEP (I :Integer);
{... PRIMEIRO PASSO DA REDUÇÃO DE HOUSEHOLDER ...}
{... I = VARIÁVEL DE CONTROLE ...}
var
{... Z = VARIÁVEL AUXILIAR ...}
  Z
  :Real;
begin { STEP }
  for J := I + 1 to L do
    begin { for J := I + 1 to L }
      T := 0;
      for K := I + 1 to J do
        begin { for K := I + 1 to J }
          KJ := (K - 1) * C + J;
          T := T + A^[KJ] * V^[K];
        end; { for K := I + 1 to J }
      for K := J + 1 to L do
        begin { for K := J + 1 to L }
          JK := (J - 1) * C + K;
          T := T + A^[JK] * V^[K];
        end; { for K := J + 1 to L }
      P^[J] := 2 * T
    end; { for J := I + 1 to L }
  Z:=0;
  for J := I + 1 to L do
    Z := Z + P^[J] * V^[J];
  Z := 2 * Z;
  for J := I + 1 to L do
    for K := J to L do
      begin { for K := J to L }
        JK := (J - 1) * C + K;
        A^[JK] := A^[JK] - V^[J] * P^[K] - (P^[J] - Z * V^[J])

```



```

        IJ := (I - 1) * C + J;
        V^[IJ] := Y * A^[IJ];
    end; { for J := I + 2 to L }
    STEP(I); UPDATU(I);
end { if CE <> 0 }
end; { for I := 1 to L - 2 }
DG^[L - 1] := A^[L * L - L - 1];
DG^[L] := A^[L * L];
SPD^[L - 1] := A^[L * L - L];
SPD^[L] := 0
end; { TRIDI }

```

```

procedure SHIFT (A, B, CE :Real; var LAM, MU :Real);
{...CALCULA MAIOR AUTOVALOR DE UM BLOCO 2 x 2 DE A ...}

```

```

var
    TR,
    DET,
    LO,
    B2,
    SRT :Real;
begin { SHIFT }
    B2 := sqr (B);
    DET := A * CE - B2;
    TR := A + CE;
    SRT := sqrt (sqr (A - CE) + 4 * B2);
    if TR >= 0
    then
        LAM := TR + SRT
    else
        LAM := TR - SRT;
    LAM := 0.5 * LAM; {... MAIOR AUTOVALOR ...}
    LO := DET / LAM; {... MENOR AUTOVALOR ...}
    if abs (LO - MU) < 0.5 * abs (LO)
    then
        begin { if abs (LO - MU) < 0.5 * abs (LO) }
            LAM := LO;
            MU := LO
        end { if abs (LO - MU) < 0.5 * abs (LO) }
    else
        if abs (LAM - MU) < 0.5 * abs (LAM)
        then MU := LAM
        else
            begin { else }
                MU := LO;
                LAM := 0
            end { else }
    end; { SHIFT }

```

```

procedure QR;
{... FATORA A MATRIZ A EM QR, SENDO Q ORTOGONAL E R TRIANGULAR
... SUPERIOR ...}

```

```

var
  CE,
  S,
  X,
  X1,
  Y,
  Z,
  UU,
  LAM,
  MU      :Real;
  I,
  J,
  K,
  M      :Integer;

procedure FACTOR (I, M :Integer);
{... PRODUZ A MATRIZ R TRIANGULAR SUPERIOR ...}
begin { FACTOR }
  DG^[I] := DG^[I] - LAM;
  UU := SPD^[I];
  for J := I to M - 1 do
    begin { for J := I to M - 1 }
      X := DG^[J];
      X1 := DG^[J + 1] - LAM;
      Y := SPD^[J];
      Z := sqrt (sqr (X) + sqr (UU));
      CE := X / Z;
      S := UU / Z;
      CS^[J] := CE;
      SN^[J] := S;
      DG^[J] := Z;
      DG^[J + 1] := -S * Y + CE * X1;
      SPD^[J] := CE * Y + S * X1;
      UU := SPD^[J + 1];
      SPD^[J + 1] := CE * UU;
      T2^[J] := S * UU
    end {-for J := I to M - 1 }
  end; { FACTOR }

procedure MULT ( I, M :Integer);
{... CALCULA OS AUTOVETORES E AUTOVALORES DE A ...}
begin { MULT }
  for K := I to M - 1 do
    begin { for K := I to M - 1 }
      CE := CS^[K];
      S := SN^[K];
      X := DG^[K];
      Y := SPD^[K];
      SPD^[K - 1] := SPD^[K - 1] * CE + T2^[K - 1] * S;
      DG^[K] := X * CE + Y * S + LAM;
      SPD^[K] := -X * S + Y * CE;
    end
  end

```

```

    DG^[K + 1] := DG^[K + 1] * CE;
  for J := 1 to L do
    begin { for J := 1 to L }
      JK := (J - 1) * C + K;
      X := U^[JK];
      Y := U^[JK + 1];
      U^[JK] := X * CE + Y * S;
      U^[JK + 1] := -X * S + Y * CE
    end { for J := 1 to L }
  end; { for K := I to M - 1 }
  DG^[M] := DG^[M] + LAM
end; { MULT }

begin { QR }
  for I := 0 to L do
    begin { for I := 0 to L }
      SN^[I] := 0;
      CS^[I] := 0
    end; { for I := 0 to L }
  M := L;
  MU := 0;
  while M > 0 do
    begin { while M > 0 }
      if Abs (SPD^[M - 1]) < EPS
      then
        begin { if Abs (SPD^[M - 1]) < EPS }
          RT^[M] := DG^[M];
          M := M - 1
        end { if Abs (SPD^[M - 1]) < EPS }
      else
        begin { else }
          I := M - 1;
          repeat
            I := I - 1
          until Abs (SPD^[I]) < EPS;
          I := I + 1;
          Shift (DG^[M - 1], SPD^[M - 1], DG^[M], LAM, MU);
          Factor (I, M);
          T2^[I - 1] := 0;
          SPD^[I - 1] := 0;
          Mult (I, M)
        end { else }
      end { while M > 0 }
    end; { QR }

procedure ORDENA;
[... ORDENA DECRESCENTEMENTE AUTOVALORES E AUTOVETORES DE A ...]
var
  K      :Byte;
  AAR    :Real;
  KK     :Integer;

```

```

begin { ORDENA }
  K:=0;
  for J:=1 to L - 1 do
    if RT^[J + 1] > RT^[J]
    then
      begin { if RT^[J + 1] > RT^[J] }
        K := 1;
        AAR := RT^[J];
        RT^[J] := RT^[J + 1];
        RT^[J + 1] := AAR;
        KK := J;
        for I := 1 to L do
          begin { for I := 1 to L }
            IKK := (I - 1) * C + KK;
            AAR := U^[IKK];
            U^[IKK] := U^[IKK + 1];
            U^[IKK + 1] := AAR;
          end; { for I := 1 to L }
        end; { if RT^[J + 1] > RT^[J] }
    if K = 1
    then
      ORDENA
  end; { ORDENA }

begin { ESPECTRO }
  HEAPCHEIO := false;
  SIMETRIA;
  if NULA = false
  then
    begin { if NULA = false }
      if SIM = true
      then
        begin { if SIM = true }
          TESTAHEAP (int (NO) + 8* (sqrt(NO) + 10));
          if HEAPCHEIO
          then
            exit;
          mark (LIMPAUTO);
          new (U);
          new (RT);
          new (DG);
          new (SPD);
          new (V);
          new (P);
          new (SN);
          new (CS);
          new (T2);
          QUADRO (30,12,44,16);
          DIAGRAMA (32,13,'Processando');
          DIAGRAMA (32,14,'Autovetores');
          DIAGRAMA (32,15,'Autovalores');
        end;
      end;
    end;
  end;

```

```

for J := 1 to L do
  for K := 1 to L do
    begin { for K := 1 to L }
      JK := (J - 1) * C + K;
      U[JK] := 0;
    end; { for K := 1 to L }
  for J:=1 to L do
    begin { for J:=1 to L }
      JJ := (J - 1) * C + J;
      U[JJ] := 1;
    end; { for J:=1 to L }
  RT[1] := A[1];
  if L > 1
  then
    begin { if L > 1 }
      TRIDI;
      NORM := 0;
      Y := 0;
      for J:=1 to L do
        begin { for J:=1 to L }
          X := Y + Abs (DG[J]);
          Y := Abs (SPD[J]);
          X := X + Y;
          if X > NORM
          then
            NORM := X
          end; { for J:=1 to L }
      EPS := NORM * TOL;
      QR;
      ORDENA;
    end; { if L > 1 }
  DETRAPO;
  repeat
    LIMPATELA (2,6,78,23);
    QUADRO (22,8,56,18);
    DIAGRAMA (24,9,'Desejas...');
    DIAGRAMA (24,11,'1 - Autovetores');
    DIAGRAMA (24,13,'2 - Autovalores');
    DIAGRAMA (24,15,'3 - Autovetores e autovalores');
    DIAGRAMA (24,17,'4 - K-esima potencia da matriz');
    QUADRO (22,19,66,23);
    DIAGRAMA (24,20,'Determinante -> ');
    write (DET:16);
    DIAGRAMA (24,21,'Traco -----> ');
    write (TRACO:16);
    DIAGRAMA (24,22,'Posto -----> ');
    write (POSTO,' Com precisao de 0.00001');
    gotoxy (55,17);
    read (kbd,OPC);
  until OPC in ['1'..'4'];
  case OPC of

```



```

      '1' : PAVET;
      '2' : PAVAL;
      '3' : begin
              PAVET;
              LIMPATELA (2,6,78,23);
              DIAGRAMA (34,10,'Autovetores em :');
              GUARDAR;
              PAVAL;
            end;
      '4' : POTENCIA;
    end; { case OPC }
    release (LIMPAUTO);
  end { if SIM = true }
else
  begin { else }
    QUADRO (22,11,45,13);
    DIAGRAMA (24,12,'Matriz nao simetrica');
    TECLENTER (24,15);
  end; { else }
end { if NULA = false }
else
  begin { else }
    QUADRO (22,11,72,13);
    DIAGRAMA (24,12,'Matriz nula, nao possui autovalores/
                  autovetores');
    TECLENTER (24,15);
  end; { else }
end; { ESPECTRO }

```

```

{... Arquivo ALGEDIAG.PAS ...}
overlay procedure DIAGONALIZAR;
{... DIAGONALIZA MATRIZ SIMETRICA E DEFINIDA POSITIVA POR
... CONGRUENCIA ...}
const
{... FOLGA = TOLERANCIA ...}
  FOLGA = 1E-8;
type
  POICE = ^VETCE;
  VETCE = array [1..NO] of real;
var
{... I,J,K,IJ,IK,KI,KJ,JK,II,KK = VARIAVEIS DE CONTROLE ...}
{... OPDIAG = OPÇÃO QUE DEFINE A MATRIZ RESULTADO,
... SE C' OU D ...}
{... ACHEI = VARIÁVEL QUE INFORMA PIVO NULO ...}
{... CE = MATRIZ IDENTIDADE ...}
{... U = VARIÁVEL AUXILIAR ...}
{... LIBERA = VARIÁVEL AUXILIAR PARA LIBERAÇÃO DE VETORES DO
... TIPO PONTEIRO ...}
  I,
  J,
  K,
  IJ,
  IK,
  KI,
  KJ,
  JK,
  II,
  KK      : integer;
  OPDIAG  : char;
  ACHEI   : boolean;
  CE      : POICE;
  CTE,
  U       : real;
  LIBERA  : ^integer;

procedure TROCA;
{... TROCA LINHA QUANDO PIVO É NULO ...}
var
{... I,J,II,KJ,JK,IND,INDJ,JIND = VARIAVEIS DE CONTROLE ...}
  I,
  J,
  II,
  KJ,
  JK,
  IND,
  INDJ,
  JIND   : integer;
begin { TROCA }
  ACHEI := false;
  I := K;

```

```

repeat
  I := I + 1;
  II := (I - 1) * L + I;
  IND := I;
  if abs (A^[II]) > FOLGA
    then
      ACHEI := true;
until (I = L) or (abs(A^[II]) > FOLGA);
if ACHEI = true
  then
    begin { if ACHEI = true }
      J := 1;
      while J <= L do
        begin { while J <= L }
          INDJ := (IND - 1) * L + J;
          KJ := (K - 1) * L + J;
          U := A^[INDJ];
          A^[INDJ] := A^[KJ];
          A^[KJ] := U;
          U := CE^[INDJ];
          CE^[INDJ] := CE^[KJ];
          CE^[KJ] := U;
          J := J + 1;
        end; { while J <= L }
      J := 1;
      while J <= L do
        begin { while J <= L }
          JIND := (J - 1) * L + IND;
          JK := (J - 1) * L + K;
          U := A^[JIND];
          A^[JIND] := A^[JK];
          A^[JK] := U;
          J := J + 1;
        end; { while J <= L }
      end; { if ACHEI = true }
    end; { TROCA }

```

```

procedure CALCULO;
{... CALCULA AS MATRIZES C' E D ...}
begin { CALCULA }
  I := 1;
  while I <= L do
    begin { while I <= L }
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I - 1) * C + J;
          if I = J
            then
              CE^[IJ] := 1
            else

```

```

        CE^[IJ] := 0;
        J := J + 1;
    end; { while J <= C }
    I := I + 1;
end; { while I <= L }
K := 1;
while K <= L - 1 do
    begin { while K <= L - 1 }
        ACHEI := true;
        KK := (K - 1) * L + K;
        if abs (A^[KK]) <= FOLGA
            then
                TROCA;
        if ACHEI = true
            then
                begin { if ACHEI = true }
                    I := K + 1;
                    while I <= L do
                        begin { while I <= L }
                            IK := (I - 1) * L + K;
                            CTE := - A^[IK] / A^[KK];
                            J := 1;
                            while J <= L do
                                begin { while J <= L }
                                    KJ := (K - 1) * C + J;
                                    IJ := (I - 1) * C + J;
                                    A^[IJ] := CTE * A^[KJ] + A^[IJ];
                                    CE^[IJ] := CTE * CE^[KJ] + CE^[IJ];
                                    J := J + 1;
                                end; { while J <= L }
                            I := I + 1;
                        end; { while I <= L }
                    I := K + 1;
                    while I <= L do
                        begin { while I <= L }
                            KI := (K - 1) * L + I;
                            A^[KI] := 0;
                            I := I + 1;
                        end; { while I <= L }
                    end { if ACHEI = true }
                else
                    K := L + 1;
                K := K + 1;
            end; { while K <= L - 1 }
    end; { CALCULO }

begin { DIAGONALIZAR }
    SIMETRIA;
    if (SIM = true)
        then
            begin { if (SIM = true) }

```

```

TESTAHEAP (int(NO));
if HEAPCHEIO
  then
    exit;
  mark (LIBERA);
  new (CE);
  QUADRO (13,7,65,11);
  DIAGRAMA (15,8,'Dada A, simetrica, existem C e D tais
              que C''AC=D');
  DIAGRAMA (20,10,'com C nao singular e D diagonal');
  QUADRO (32,13,47,21);
  DIAGRAMA (34,14,'Desejas ...:');
  DIAGRAMA (34,16,'1 - C''');
  DIAGRAMA (34,18,'2 - D');
  DIAGRAMA (34,20,'3 - Retorno ');
  repeat
    read (kbd,OPDIAG);
  until OPDIAG in ['1'..'3'];
  LIMPATELA (2,6,78,23);
  if OPDIAG = '3'
    then
      exit;
    QUADRO (29,12,46,15);
    DIAGRAMA (31,13,'Diagonalizacao');
    DIAGRAMA (31,14,' congruencia ');
    CALCULO;
    if OPDIAG = '1'
      then
        begin { if OPDIAG = '1' }
          I := 1;
          while I <= L do
            begin { while I <= L }
              J := 1;
              while J <= C do
                begin { while J <= C }
                  IJ := (I - 1) * L + J;
                  A^[IJ] := CE^[IJ];
                  J := J + 1;
                end; { while J <= C }
              I := I + 1;
            end; { while I <= L }
          end; { if OPDIAG = '1' }
        release (LIBERA);
      end { if SIM = true }
    else
      begin { else }
        QUADRO (26,11,51,13);
        DIAGRAMA (28,12,'Matriz assimetrica !!!');
        TECLENTER (28,15);
      end; { else }
    end; { DIAGONALIZAR }

```

```

(...Arquivo ALGENORM.PAS ...)
overlay procedure NORMA;
(... CALCULA NORMAS DE MATRIZ ...)
type
(... DECLARAÇÃO DE TIPOS ...)
  POISOLIN = ^VETSOLIN;
  POISOCOL = ^VETSOCOL;
  VETSOLIN = array [1..NO] of real;
  VETSOCOL = array [1..NO] of real;
var
(... J,I,IJ,JI = VARIÁVEIS DE CONTROLE ...)
(... SOLIN = VETOR DE SOMA DE LINHA ...)
(... SOCOL = VETOR DE SOMA DE COLUNA ...)
(... NORFROBENIUS = NORMA DE FROBENIUS ...)
(... LIMPNORMA = VARIÁVEL AUXILIAR PARA LIBERAÇÃO DE VETORES
... DO TIPO PONTEIRO ...)
  J,
  I,
  IJ,
  JI      : integer;
  SOLIN   : POISOLIN;
  SOCOL   : POISOCOL;
  NORFROBENIUS : real;
  LIMPNORMA : ^integer;

procedure MAIORES;
(... CALCULA A MAIOR SOMA DE LINHA E A MAIOR SOMA DE COLUNA ...)
var
(... VARIÁVEIS DE CONTROLE ...)
  J,
  I,
  IJ,
  JI      : integer;
begin { MAIORES }
  J := 1;
  while J <= C-1 do
    begin { while J <= C-1 }
      if SOLIN^[1] < SOLIN^[J + 1]
      then
        SOLIN^[1] := SOLIN^[J + 1];
        J := J + 1;
      end; { while J <= C-1 }
  I := 1;
  while I <= L-1 do
    begin { while I <= L-1 }
      if SOCOL^[1] < SOCOL^[I + 1]
      then
        SOCOL^[1] := SOCOL^[I + 1];
        I := I + 1;
      end; { while I <= L-1 }
  end; { MAIORES }

```

```

begin { NORMA }
  HEAPCHEIO := false;
  TESTAHEAP (2 * int (NO) + sqrt(NO) + 5);
  if HEAPCHEIO
  then
    exit;
  mark (LIMPNORMA);
  new (SOLIN);
  new (SOCOL);
  NORFROBENIUS := 0;
  I := 1;
  while I <= L do
    begin { while I <= L }
      SOLIN^[I] := 0;
      SOCOL^[I] := 0;
      J := 1;
      while J <= C do
        begin { while J <= C }
          IJ := (I - 1) * C + J;
          JI := (J - 1) * C + I;
          SOLIN^[I] := SOLIN^[I] + abs(A^[IJ]);
          SOCOL^[I] := SOCOL^[I] + abs(A^[JI]);
          NORFROBENIUS := NORFROBENIUS + sqr (A^[IJ]);
          J := J + 1;
        end; { while J <= C }
      I := I + 1;
    end; { while I <= L }
  MAIORES;
  QUADRO (10,9,71,21);
  DIAGRAMA (12,10,'Norma(A)c : maximo das somas dos modulos
    dentro de colunas');
  DIAGRAMA (12,12,'Norma(A)l : maximo das somas dos modulos
    dentro de linhas');
  DIAGRAMA (12,14,'Norma(A)f : soma de quadrados dos elementos
    (Frobenius)');
  DIAGRAMA (12,16,'Norma(A)c = ');
  write (SOCOL^[1]:11);
  DIAGRAMA (12,18,'Norma(A)l = ');
  write (SOLIN^[1]:11);
  DIAGRAMA (12,20,'Norma(A)f = ');
  write (NORFROBENIUS:11);
  TECLENTER (12,23);
  release (LIMPNORMA);
end; { NORMA }

```

```

{... Arquivo ALGEORTO.PAS ...}
overlay procedure GRAM_SCHMIDT;
{... ORTONORMALIZA A MATRIZ A POR GRAM-SCHMIDT ...}
type
{... DECLARAÇÃO DE TIPOS LOCAIS ...}
  POIP = ^VETP;
  VETP = array [1..NO] of real;
  POIT = ^VETT;
  VETT = array [1..NO] of real;
  POIE = ^VETE;
  VETE = array [1..NO] of real;
var
{... OPCAO = DEFINE QUAL MATRIZ RESULTANTE RETORNARA NA
... MEMORIA DE TRABALHO ...}
{... LIMPORTO = VARIABEL AUXILIAR PARA LIBERAÇÃO DE VETORES DO
... TIPO PONTEIRO ...}
{... P,T,E = VETORES AUXILIARES DO TIPO PONTEIRO ...}
  OPCAO      : char;
  LIMPORTO   : ^integer;
  P          : POIP;
  T          : POIT;
  E          : POIE;

procedure PASSAGEM;
{... PASSA MATRIZ U OU T PARA A (MEMORIA DE TRABALHO) ...}
var
{... I = VARIABEL DE CONTROLE ...}
{... AUX = VARIABEL AUXILIAR ...}
  I : integer;
  AUX : real;
begin { PASSAGEM }
  DIAGRAMA (3,22,'Passando dados');
  I := 1;
  while I <= L * C do
    begin { while I <= L * C }
      if OPCAO = '1'
      then
        A^[I] := P^[I]
      else
        if OPCAO = '2'
        then
          A^[I] := T^[I]
        else
          begin { else }
            AUX := A^[I];
            A^[I] := P^[I];
            P^[I] := AUX;
          end; { else }
      I := I + 1;
    end; { while I <= L * C }
  LIMPALINHA (3,22,76);

```



```

end; { PASSAGEM }
procedure CALTRI;
{... CALCULA A MATRIZ T TRIANGULAR ...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  J,
  K,
  IJ,
  KI,
  KJ
    : integer;
begin { CALTRI }
  DIAGRAMA (3,22,'Triangulando');
  I := 1;
  while I <= C do
    begin { while I <= C }
      J := 1;
      while J <= L do
        begin { while I <= L }
          IJ := (I - 1) * C + J;
          T^[IJ] := 0;
          K := 1;
          while K <= C do
            begin { while K <= C }
              KI := (K - 1) * C + I;
              KJ := (K - 1) * C + J;
              T^[IJ] := T^[IJ] + A^[KI] * P^[KJ];
              K := K + 1;
            end; { while K <= C }
            J := J + 1;
          end;
          I := I + 1;
        end; { while I <= C }
        LIMPALINHA (3,22,76);
      end; { CALTRI }

```

```

procedure CALCULO;
{... CALCULA MATRIZ U ORTOGONAL ...}
var
{... VARIÁVEIS DE CONTROLE ...}
  I,
  I1,
  IJ,
  IK,
  K,
  M,
  J
    : integer;
{... SOM = RESULTADO DO PRODUTO INTERNO DO VETOR COLUNA ORTOGONAL
... PELO VETOR COLUNA DA MATRIZ A ...}
{... SOM1 = RESULTADO DO PRODUTO INTERNO DO VETORES COLUNA
... ORTOGONAL ...}

```

```

{... SOMA = k (SOM/SOM1), TAL QUE K TORNA VETORES ORTOGONAIS ...}
SOM,
SOM1,
SOMA : real;
begin { CALCULO }
  DIAGRAMA (3,22,'Ortogonalizando');
  I := 1;
  while I <= L do
    begin { while I <= L }
      I1 := (I - 1) * C + 1;
      P^[I1] := A^[I1];
      I := I + 1;
    end; { while I <= L }
  J := 2;
  while J <= C do
    begin { while J <= C }
      M := J - 1;
      K := 1;
      while K <= M do
        begin { while K <= M }
          E^[K] := 0;
          SOM := 0;
          SOM1 := 0;
          I := 1;
          while I <= L do
            begin { while I <= L }
              IJ := (I - 1) * C + J;
              IK := (I - 1) * C + K;
              SOM := SOM + A^[IJ] * P^[IK];
              SOM1 := SOM1 + sqr (P^[IK]);
              I := I + 1;
            end; { while I <= L }
            if abs (SOM1) > 1E-6
            then
              E^[K] := SOM / SOM1;
          K := K + 1;
        end; { while K <= M }
      I := 1;
      while I <= L do
        begin { while I <= L }
          SOMA := 0;
          K := 1;
          while K <= M do
            begin { while K <= M }
              IK := (I - 1) * C + K;
              SOMA := SOMA + E^[K] * P^[IK];
              K := K + 1;
            end; { while K <= M }
            IJ := (I - 1) * C + J;
            P^[IJ] := A^[IJ] - SOMA;
            I := I + 1;
          end; { while I <= L }
        J := J + 1;
      end; { while J <= C }
    end; { while I <= L }
  end; { while J <= C }
end; { CALCULO }

```

```

        end; { while I <= L }
        J := J + 1;
        end; { while J <= C }
        LIMPALINHA (3,22,76);
    end; { CALCULO }

begin { GRAM_SCHMIDT }
    HEAPCHEIO := false;
    TESTAHEAP (2 * int (NO) + sqrt (NO) + 5);
    if HEAPCHEIO
    then
        exit;
    mark (LIMPORTO);
    new (P);
    new (T);
    new (E);
    CALCULO;
    LIMPATELA (2,6,78,23);
    QUADRO (5,7,76,9);
    DIAGRAMA (7,8,'Dada a matriz A, existem T, triangular e U,');
    DIAGRAMA (51,8,'tais que A=UT e U''U = I');
    QUADRO (29,11,42,17);
    DIAGRAMA (31,12,'Desejas...');
    DIAGRAMA (31,14,'1 - U');
    DIAGRAMA (31,16,'2 - T ');
    repeat
        read (kbd,OPCAO);
    until OPCAO in ['1'..'2'];
    if OPCAO = '2'
    then
        begin { if OPCAO = '2' }
            OPCAO := '3';
            PASSAGEM;
            NORMALIZA;
            CALTRI;
            OPCAO := '2'
        end; { if OPCAO = '2' }
    PASSAGEM;
    if OPCAO = '1'
    then
        NORMALIZA;
        LIMPATELA (2,6,78,23);
        release (LIMPORTO);
    end; { GRAM_SCHMIDT }

```

```

{...Arquivo ALGESWEE.PAS ...}
overlay procedure SWEEP;
{... OPERA BLOCOS DE MATRIZES OBTENDO INVERSA CONDICIONAL E
... OUTROS RESULTADOS ESTATISTICOS ...}
var
{... VARIAVEIS DE CONTROLE ...}
I,
J,
K,
IJ,
JI,
KJ,
IK,
J1,
J2;
KK      : integer;
{... RES = OPÇÃO DE MAIS OPERAÇÃO ...}
RES      : char;

procedure CALCULO;
{... OPERAÇÃO "SWEEP" ...}
const
  EPS = 0.000000001;
begin { CALCULO }
  for K := J1 to J2 do
    begin { for K := J1 to J2 }
      KK := (K - 1) * C + K;
      for I := 1 to C do
        for J := I to C do
          begin { for J := I to C }
            IJ := (I - 1) * C + J;
            JI := (J - 1) * C + I;
            if abs (A^[IJ]) < 1E-20
              then
                begin { if abs (A^[IJ]) < 1E-20 }
                  A^[IJ] := 0.0;
                  A^[JI] := 0.0
                end; { if abs (A^[IJ]) < 1E-20 }
          end; { for J := I to C }
        gotoxy (32,12);
        write ('Sweep ',K:3);
        if abs(A^[KK]) >= EPS
          then
            begin { if abs(A^[KK]) >= EPS }
              for I := 1 to C do
                begin { for I := 1 to C }
                  IK := (I - 1) * C + K;
                  if (I <> K) And (A^[IK] <> 0)
                    then
                      for J := 1 to C do
                        begin { for J := 1 to C }

```

```

      KJ := (K - 1) * C + J;
      IJ := (I - 1) * C + J;
      if (J <> K) And (A^[KJ] <> 0)
      then
        A^[IJ] := A^[IJ] - A^[IK] / A^[KK] *
          A^[KJ];
      end; { for J := 1 to C }
    end; { for I := 1 to C }
  for J := 1 to C do
    begin { for J := 1 to C }
      KJ := (K - 1) * C + J;
      if J <> K
      then
        A^[KJ] := A^[KJ] / A^[KK];
      end; { for J := 1 to C }
    for I := 1 to C do
      begin { for I := 1 to C }
        IK := (I - 1) * C + K;
        if I <> K
        then
          A^[IK] := -A^[IK] / A^[KK];
        end; { for I := 1 to C }
      A^[KK] := 1/A^[KK];
    end { abs(A^[KK]) >= EPS }
  else
    begin { else }
      for J:=1 to C do
        begin { for J:=1 to C }
          KJ := (K - 1) * C + J;
          A^[KJ] := 0;
        end; { for J:=1 to C }
      for I:=1 to C do
        begin { for I:=1 to C }
          IK := (I - 1) * C + K;
          A^[IK] := 0;
        end; { for I:=1 to C }
      end; { else }
    end; { for K := J1 to J2 }
end; { CALCULO }

begin { SWEEP }
  repeat
    LIMPATELA (2,6,78,23);
    QUADRO (18,11,60,17);
    repeat
      repeat
        {$I-}
        DIAGRAMA (20,12,'Primeira coluna a ser operada : ');
        J1 := trunc(NUMERO (54,12,59));
        {$I+}
      until ioresult = 0;
    end;
  end;
end;

```

```

repeat
{$I-}
  DIAGRAMA (20,14,'  Ultima coluna a ser operada   : ');
  J2 := trunc (NUMERO (54,14,59));
{$I+}
  until ioresult = 0;
until ([J1,J2] <= [1..C]) and (J1 <= J2);
LIMPATELA (19,12,59,16);
DIAGRAMA (20,13,'Processando sweep ');
CALCULO;
LIMPALINHA (20,13,59);
DIAGRAMA (20,16,'Mais operacoes ? (S/N)  ');
repeat
  read (Kbd,RES);
  RES := upcase (RES);
  until RES in ['S','N'];
until (RES='N');
LIMPATELA (2,6,78,23);
end; { SWEEP }

```