

Uma experiência com o cluster SEARCH

Carlos Alberto Campos
Rui Manuel da Silva Ralha

Centro de Matemática
Universidade do Minho
Workshop – Braga – 21/Abril/2006

ccampos@estg.ipleiria.pt
r_ralha@math.uminho.pt



Contexto

- ▶ Primeiro contacto com o cluster **SEARCH**
- ▶ **Objectivos**
 - Configurar o ambiente de trabalho para a utilização do cluster
 - ◆ Implementações desenvolvidas em Fortran 90
 - ◆ Uso das bibliotecas numéricas LAPACK e ScaLAPACK
 - Comprovar a instalação do software
 - Utilização do cluster para processamento massivo
 - ◆ Obtenção de resultados das implementações desenvolvidas
 - ◆ Comparação de resultados com os obtidos nos clusters da Universidade Politécnica de Valência

Conteúdo (I)

- ▶ Configuração do ambiente de trabalho
 - Ambiente Windows
 - Ambiente Unix
- ▶ Acesso ao cluster **SEARCH**
 - Acesso a search.di.uminho.pt
 - “Visita guiada”
- ▶ Experiências com **BLAS**, **LAPACK** e **ScaLAPACK**
- ▶ Alguns tópicos adicionais

Conteúdo (II)

- ▶ **BLAS**
 - Apresentação da especificação
 - Programação de exemplos
 - ◆ Fase de implementação
 - Produto interno e norma (BLAS-1)
 - Produto matriz-vector (BLAS-2)
 - Produto matriz-matriz (BLAS-3)
 - ◆ Fase de compilação
 - ◆ Fase de execução
 - ◆ Detecção de erros

Conteúdo (III)

► LAPACK

- Apresentação da biblioteca
- Experiências com rotinas *driver*
 - ◆ Resolução de sistemas de equações lineares
 - ◆ Cálculo da decomposição em valores singulares
- Experiências com rotinas computacionais
 - ◆ Cálculo da decomposição QR de uma matriz
 - ◆ Redução de uma matriz à forma bidiagonal superior
- Programação de exemplos

Conteúdo (IV)

► ScaLAPACK

- Apresentação da biblioteca
- Experiências computacionais
 - Redução de uma matriz à forma bidiagonal superior
 - Cálculo da decomposição em valores singulares
- Programação de exemplos

► Alguns tópicos adicionais

- Exemplos com MPI (Message Passing Interface)
- Exemplo de um algoritmo orientado a blocos matriciais

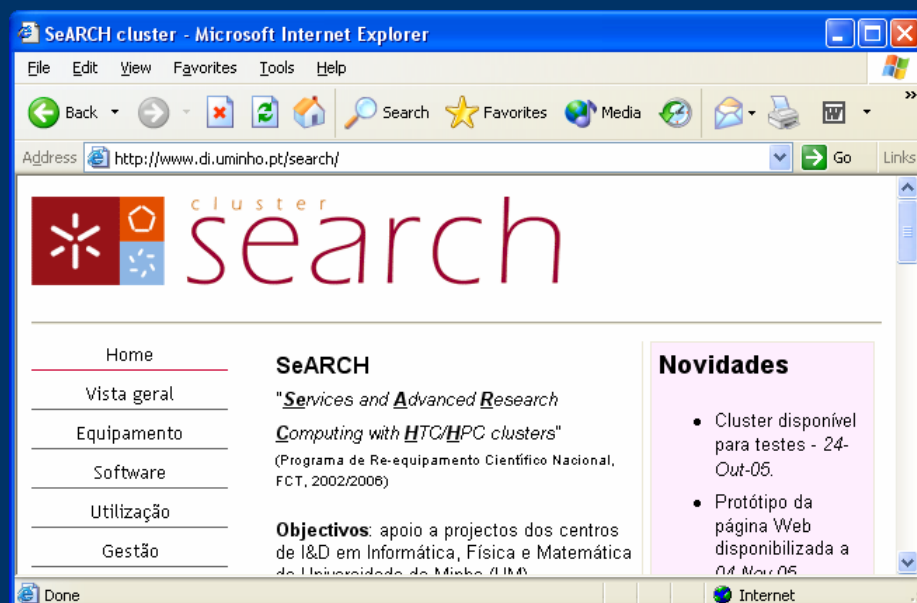
Ambiente de Trabalho

Ambiente Windows
Ambiente Unix

Ambiente Windows (I)

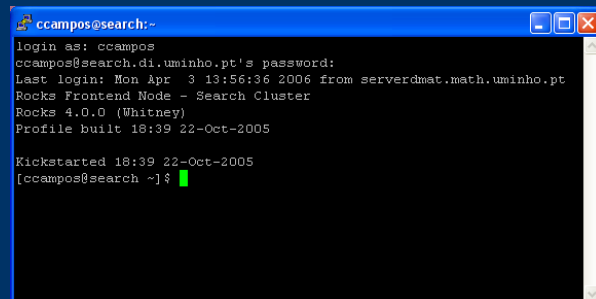
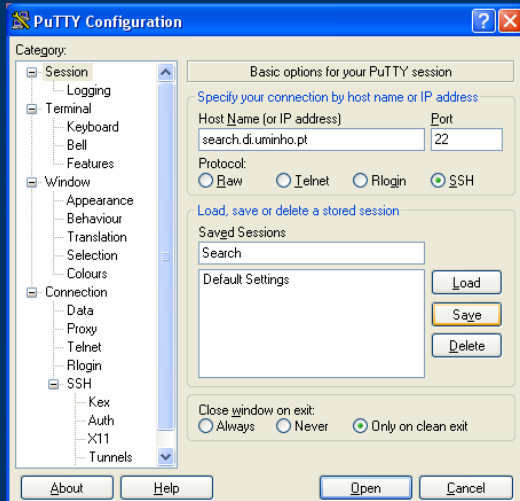
► Endereço Web

■ www.di.uminho.pt/search



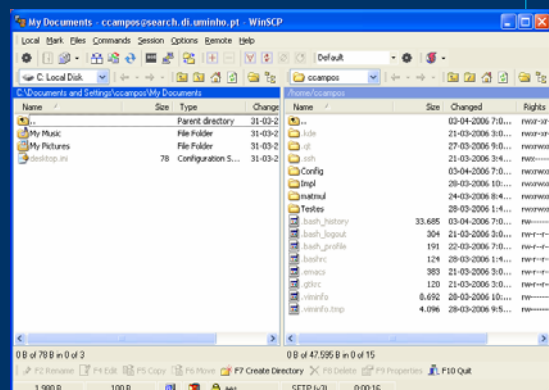
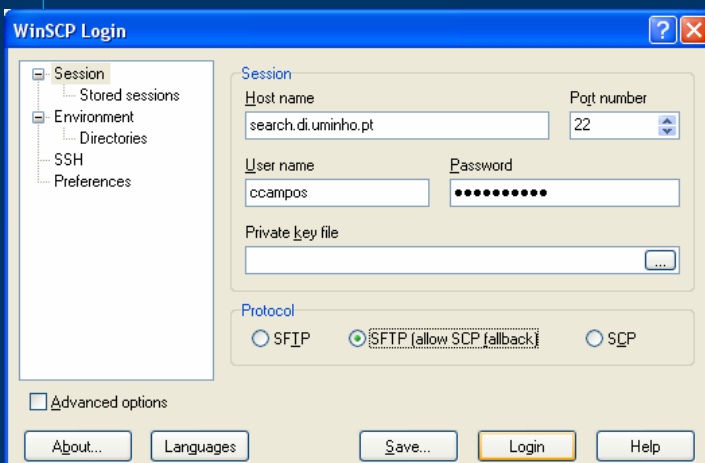
Ambiente Windows (II)

- ▶ Instalar sistema **SSH** / PuTTY
 - <http://www.chiark.greenend.org.uk/~sgtatham/putty>
- ▶ Criar uma sessão com PuTTY



Ambiente Windows (III)

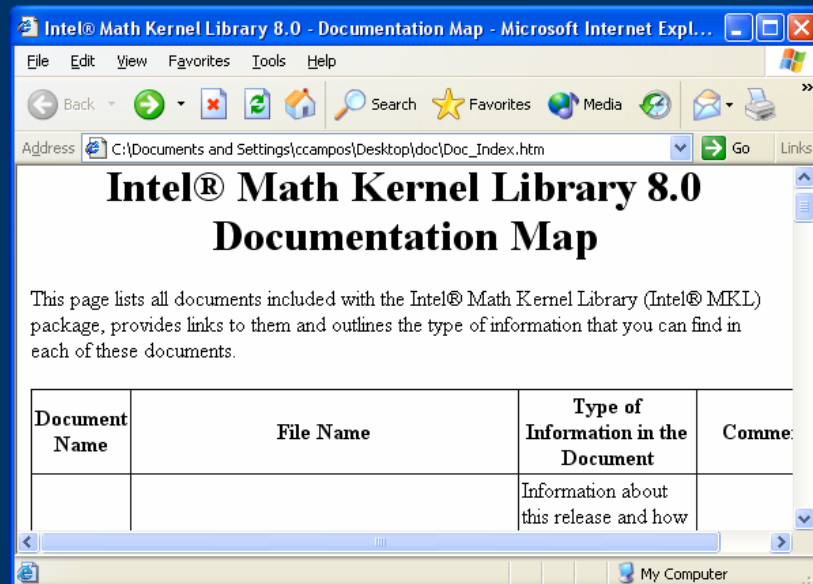
- ▶ Instalar sistema **FTP** / WinSCP
 - <http://winscp.net/>
- ▶ Criar uma sessão com WinSCP



Ambiente Unix (I)

► Documentação inicial

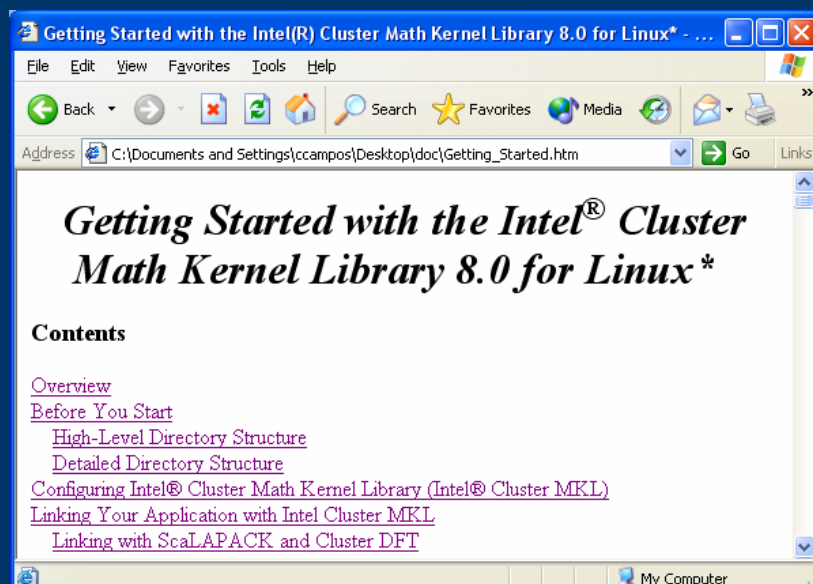
■ `/opt/intel/cmkl/8.0/doc/Doc_Index.htm`



Ambiente Unix (II)

► Documentação inicial

■ `/opt/intel/cmkl/8.0/doc/Getting_Started.htm`



Ambiente Unix (III)

▶ Variáveis de ambiente de CMKL

■ `/opt/intel/cmkl/8.0/tools/environment`

◆ Ficheiro `mklvarsem64t.sh`

● Variável `INCLUDE`

◆ `export INCLUDE=/opt/intel/cmkl/8.0/include:$INCLUDE`

● Variável `LD_LIBRARY_PATH`

◆ `export LD_LIBRARY_PATH=/opt/intel/cmkl/8.0/lib/em64t:$LD_LIBRARY_PATH`

▶ Variável de execução de CMKL

◆ Configuração do número de *threads*

● Variável `OMP_NUM_THREADS`

◆ `export OMP_NUM_THREADS=1`

Ambiente Unix (IV)

▶ Variável de ambiente de MPI

■ `/opt/intel/mpi/1.0.2`

◆ Configuração do acesso

● Variável `PATH`

◆ `export PATH=/opt/intel/mpi/1.0.2/bin64:$PATH`

◆ `export PATH=/opt/intel/mpi/1.0.2/lib64:$PATH`

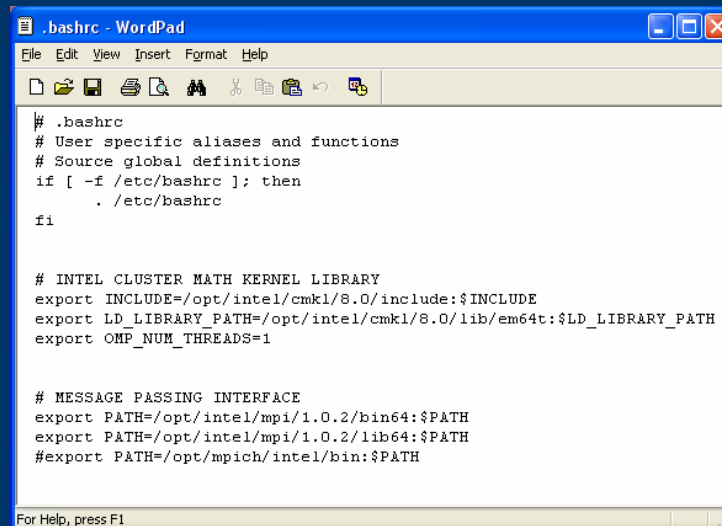
▶ Como configurar?

■ Inserir o código no ficheiro `.bashrc`

Ambiente Unix (v)

► Ficheiro **.bashrc**

- Cluster Math Kernel Library : /opt/intel/cmkl/8.0/
- Message Passing Interface : /opt/intel/mpi/1.0.2



```
# .bashrc
# User specific aliases and functions
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

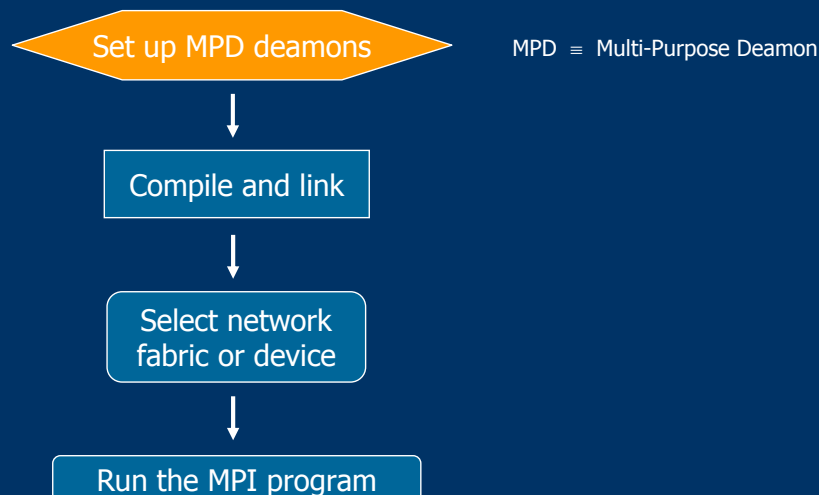
# INTEL CLUSTER MATH KERNEL LIBRARY
export INCLUDE=/opt/intel/cmkl/8.0/include:$INCLUDE
export LD_LIBRARY_PATH=/opt/intel/cmkl/8.0/lib/em64t:$LD_LIBRARY_PATH
export OMP_NUM_THREADS=1

# MESSAGE PASSING INTERFACE
export PATH=/opt/intel/mpi/1.0.2/bin64:$PATH
export PATH=/opt/intel/mpi/1.0.2/lib64:$PATH
#export PATH=/opt/mpich/intel/bin:$PATH
```

Ambiente Unix (VI)

► Uso de Intel® MPI

- /opt/intel/mpi/1.0.2/doc/Getting_Started.pdf
 - ◆ Trabalhar com a biblioteca Intel® MPI



Ambiente Unix (VII)

► Uso de Intel® MPI

■ Set up MPD daemons

- ◆ Para o sistema EM64T a variável PATH deve incluir
 - <installdir>/bin64 (/opt/intel/mpi/1.0.2/bin64) ✓
- ◆ Criar na raiz da área de login o ficheiro **.mpd.conf** que contém a nossa MPD password
 - password=<nossa mpd password> (password=carloscampos)
- ◆ Alterar as permissões do ficheiro **.mpd.conf**
 - Executar o comando **chmod 600 .mpd.conf**

Ambiente Unix (VIII)

► Uso de Intel® MPI

■ Set up MPD daemons

- ◆ Criar os “MPD daemons”
 - Executar o comando **mpdboot**

■ Outros comandos

- ◆ Ver o status dos “MPD daemons”
 - Executar o comando **mpdtrace**
- ◆ Terminar execução dos “MPD daemons”
 - Executar o comando **mpdallexit**

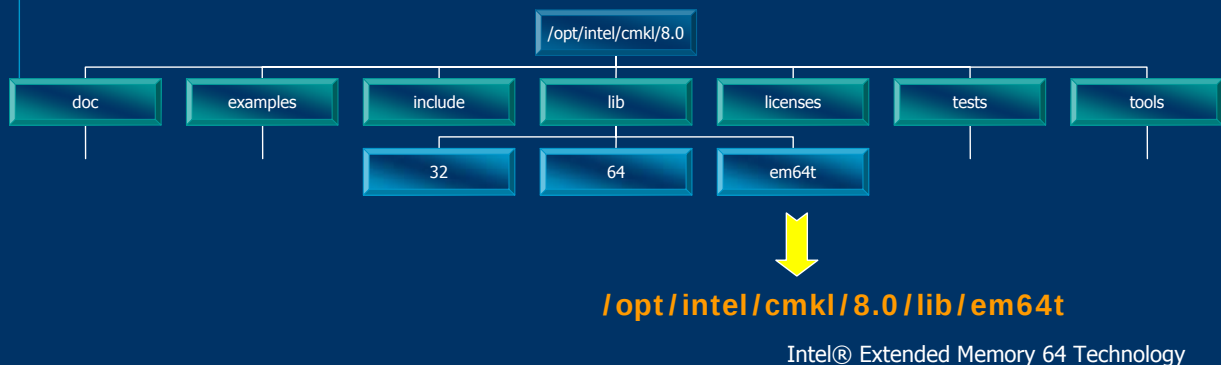
Acesso ao cluster SEARCH

www.di.uminho.pt/search
search.di.uminho.pt

Cluster SEARCH (I)

- ▶ Buscar informação
 - Executar o comando
 - `rpm -qall | grep blas`

- ▶ Instalação em `/opt/intel/cmkl/8.0`



Cluster SEARCH (II)

- ▶ Directoria `/opt/intel/cmkl/8.0/lib/em64t`
 - `libmkl_em64t.a` : núcleos otimizados para Intel® EM64T
 - `libmkl_lapack.a` : rotinas de LAPACK
 - `libmkl_scalapack.a` : rotinas de ScaLAPACK
 - `libmkl_blacs.a` : rotinas de BLACS
 - `libguide.a` : “threading library for static linking”
 - ...
- ▶ Directoria `/opt/intel/fce/9.0`
 - “Intel® Fortran Compiler 9.0 for Linux”

Basic Linear Algebra Subprograms

BLAS

www.netlib.org/blas/index.html

BLAS: especificação (I)

- ▶ Conjunto de rotinas para realizar operações básicas com vectores e matrizes
- ▶ É uma especificação porque para cada rotina define a **interface** e a **funcionalidade**
- ▶ Existem versões optimizadas para diversas arquitecturas
 - Cada fabricante implementa a sua
- ▶ Serve de base para a construção de outras bibliotecas de Álgebra Linear Numérica

BLAS: especificação (II)

- ▶ Segundo o custo computacional, as rotinas de BLAS subdividem-se em 3 níveis
 - BLAS nível 1
 - ◆ Operações entre **vectores**
 - Realiza $\Theta(n)$ operações sobre $\Theta(n)$ dados
 - BLAS nível 2
 - ◆ Operações entre **matrizes** e **vectores**
 - Realiza $\Theta(n^2)$ operações sobre $\Theta(n^2)$ dados
 - BLAS nível 3
 - ◆ Operações entre **matrizes**
 - Realiza $\Theta(n^3)$ operações sobre $\Theta(n^2)$ dados

BLAS: nomenclatura (I)

► Tipo de dados

■ Prefixo (1 letra)

	Precisão Simples	Precisão Dupla
Reais	S	D
Complexos	C	Z

BLAS: nomenclatura (II)

► BLAS-1 : xYYYYY excepto IxAMAX

■ x : tipo de dados

■ YYYYY : tipo de operação

```
SUBROUTINE xROTG (                                A, B, C, S )
SUBROUTINE xROTMG(                                D1, D2, A, B,          PARAM )
SUBROUTINE xROT ( N,          X, INCX, Y, INCY,          C, S )
SUBROUTINE xROTM ( N,          X, INCX, Y, INCY,          PARAM )
SUBROUTINE xSWAP ( N,          X, INCX, Y, INCY )
SUBROUTINE xSCAL ( N,  ALPHA, X, INCX )
SUBROUTINE xCOPY ( N,          X, INCX, Y, INCY )
SUBROUTINE xAXPY ( N,  ALPHA, X, INCX, Y, INCY )
FUNCTION  xDOT ( N,          X, INCX, Y, INCY )
FUNCTION  xDOTU ( N,          X, INCX, Y, INCY )
FUNCTION  xDOTC ( N,          X, INCX, Y, INCY )
FUNCTION  xxDOT ( N,          X, INCX, Y, INCY )
FUNCTION  xNRM2 ( N,          X, INCX )
FUNCTION  xASUM ( N,          X, INCX )
FUNCTION  IxAMAX( N,          X, INCX )
```

BLAS: nomenclatura (III)

► BLAS-2 e BLAS-3 : xYYZZZ

■ x : tipo de dados

◆ S, D, C, Z

■ YY : tipo de matriz

	Geral	Simétrica/ Hermítica	Triangular
Densa	GE	SY/HE	TR
Banda	GB	SB/HB	TB

BLAS: nomenclatura (IV)

► BLAS-2 e BLAS-3 : xYYZZZ

■ ZZZ : tipo de operação

◆ MV : produto Matriz-Vector

◆ MM : produto Matriz-Matriz

◆ SV : resolução de um sistema de equações lineares

◆ SM : resolução de um sistema de equações lineares múltiplo

◆ R : actualizações de dimensão 1

◆ R2 : actualizações de dimensão 2

◆ RK : actualizações de dimensão k

◆ R2K : actualizações de dimensão 2k

BLAS: nomenclatura (v)

► BLAS-2 e BLAS-3 : xYYZZZ

```

xGEMV (      TRANS,      M, N,      ALPHA, A, LDA, X, INCX, BETA, Y, INCY )
xGBMV (      TRANS,      M, N, KL, KU, ALPHA, A, LDA, X, INCX, BETA, Y, INCY )
xHEMV ( UPLO,      N,      ALPHA, A, LDA, X, INCX, BETA, Y, INCY )
xHBMV ( UPLO,      N, K,      ALPHA, A, LDA, X, INCX, BETA, Y, INCY )
xHPMV ( UPLO,      N,      ALPHA, AP,      X, INCX, BETA, Y, INCY )
xSYMV ( UPLO,      N,      ALPHA, A, LDA, X, INCX, BETA, Y, INCY )
xSBMV ( UPLO,      N, K,      ALPHA, A, LDA, X, INCX, BETA, Y, INCY )
xSPMV ( UPLO,      N,      ALPHA, AP,      X, INCX, BETA, Y, INCY )

xTRMV ( UPLO, TRANS, DIAG, N,      A, LDA, X, INCX )
xTBMV ( UPLO, TRANS, DIAG, N, K,      A, LDA, X, INCX )
xTPMV ( UPLO, TRANS, DIAG, N,      AP,      X, INCX )
xTRSV ( UPLO, TRANS, DIAG, N,      A, LDA, X, INCX )
xTBSV ( UPLO, TRANS, DIAG, N, K,      A, LDA, X, INCX )
xTPSV ( UPLO, TRANS, DIAG, N,      AP,      X, INCX )

options      dim      scalar vector      vector      matrix
xGER (      M, N, ALPHA, X, INCX, Y, INCY, A, LDA )
xGERU (      M, N, ALPHA, X, INCX, Y, INCY, A, LDA )
xGERC (      M, N, ALPHA, X, INCX, Y, INCY, A, LDA )
xHER ( UPLO,      N, ALPHA, X, INCX,      A, LDA )
xHPR ( UPLO,      N, ALPHA, X, INCX,      AP )
xHER2 ( UPLO,      N, ALPHA, X, INCX, Y, INCY, A, LDA )
xHPR2 ( UPLO,      N, ALPHA, X, INCX, Y, INCY, AP )
xSYR ( UPLO,      N, ALPHA, X, INCX,      A, LDA )
xSPR ( UPLO,      N, ALPHA, X, INCX,      AP )
xSYR2 ( UPLO,      N, ALPHA, X, INCX, Y, INCY, A, LDA )
xSPR2 ( UPLO,      N, ALPHA, X, INCX, Y, INCY, AP )

```

BLAS: nomenclatura (VI)

► BLAS-2 e BLAS-3 : xYYZZZ

```

xGEMM (      TRANSA, TRANSB,      M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )
xSYMM ( SIDE, UPLO,      M, N,      ALPHA, A, LDA, B, LDB, BETA, C, LDC )
xHEMM ( SIDE, UPLO,      M, N,      ALPHA, A, LDA, B, LDB, BETA, C, LDC )
xSYRK (      UPLO, TRANS,      N, K, ALPHA, A, LDA,      BETA, C, LDC )
xHERK (      UPLO, TRANS,      N, K, ALPHA, A, LDA,      BETA, C, LDC )
xSYR2K(      UPLO, TRANS,      N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )
xHER2K(      UPLO, TRANS,      N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )
xTRMM ( SIDE, UPLO, TRANSA,      DIAG, M, N,      ALPHA, A, LDA, B, LDB )
xTRSM ( SIDE, UPLO, TRANSA,      DIAG, M, N,      ALPHA, A, LDA, B, LDB )

```

BLAS: vectores

► Definidos por 3 parâmetros

- **N** : tamanho do vector
- **X** : primeiro elemento do vector
- **INCX** : distância entre os elementos

4, X(2), 1

1
2
3
4
5
6

3, X(1), 2

1
2
3
4
5
6

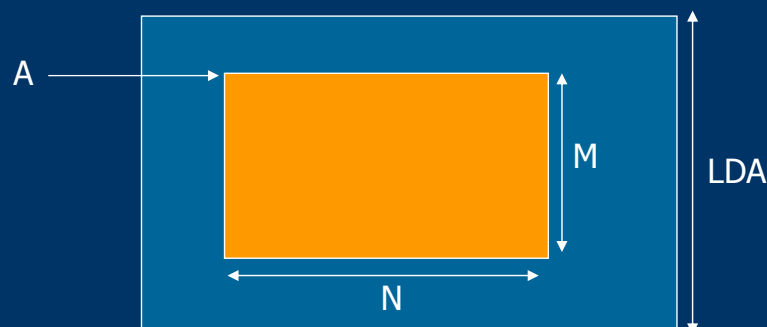
1
2
3
4
5
6
7
8
9
10
11
12

4, X(3), 3

BLAS: matrizes (I)

► Definidas por 4 parâmetros

- **M** : número de linhas
- **N** : número de colunas
- **A** : primeiro elemento da matriz
- **LDA** : distância entre os elementos de uma mesma linha ("Leading Dimension")



BLAS: matrizes (II)

► Definidas por 4 parâmetros

4, 2, A(3,1), 6

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3
4,1	4,2	4,3
5,1	5,2	5,3
6,1	6,2	6,3

3, 2, A(3,2), 6

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3
4,1	4,2	4,3
5,1	5,2	5,3
6,1	6,2	6,3

3, 3, A(1,1), 7

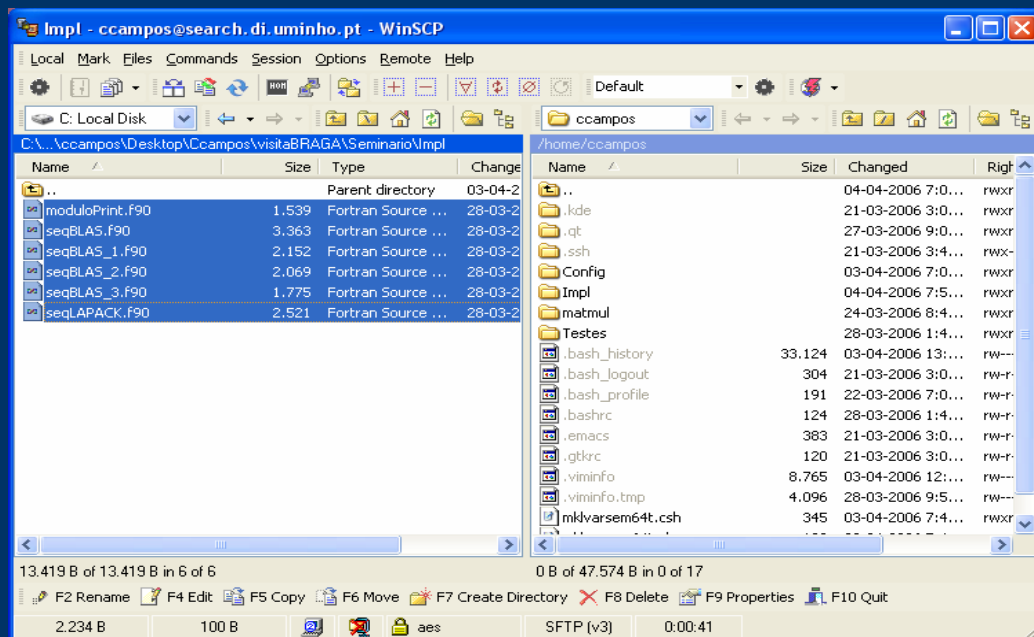
1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3
4,1	4,2	4,3
5,1	5,2	5,3
6,1	6,2	6,3

BLAS

Fortan 90
Experiências computacionais

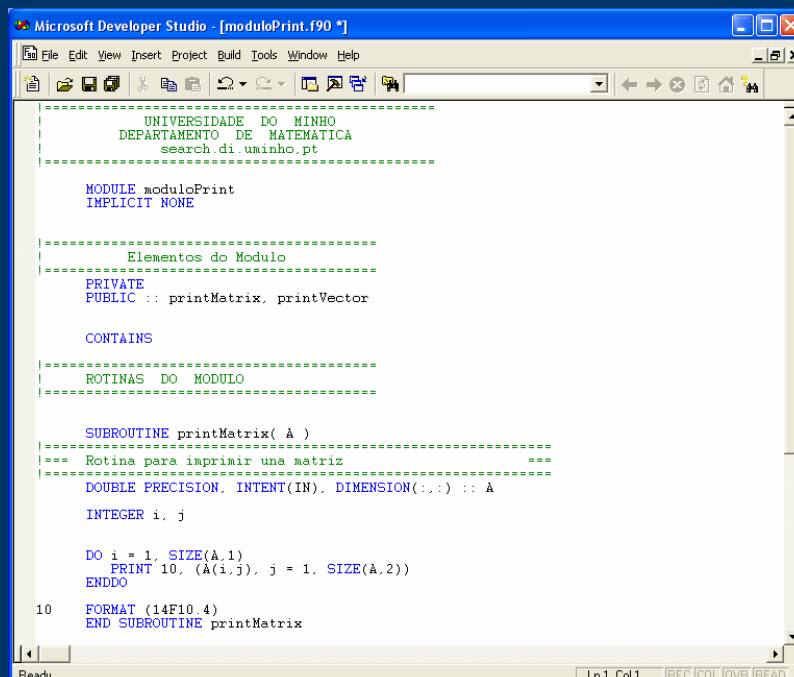
Fortran 90: WinSCP

► Ficheiros de trabalho



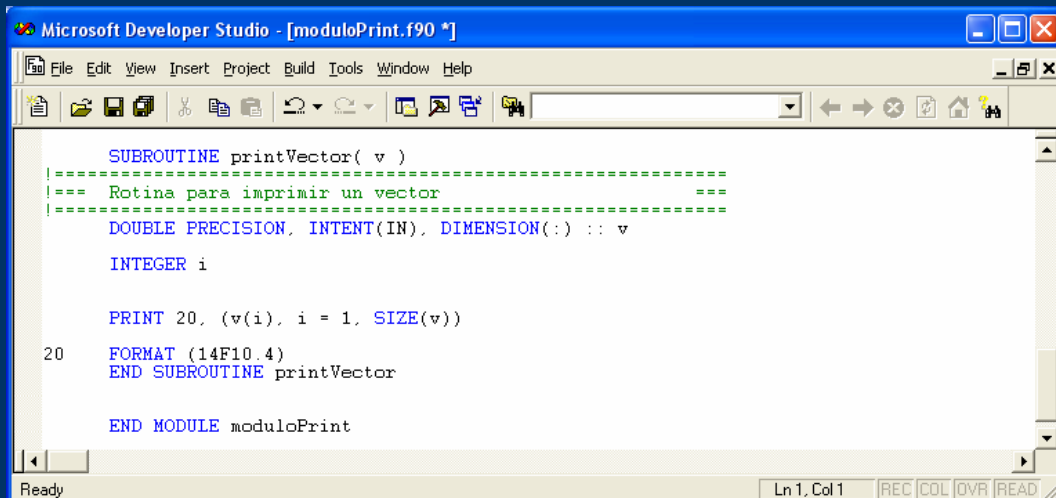
Fortran 90: módulos (I)

► Módulos auxiliares



Fortran 90: módulos (II)

► Módulos auxiliares

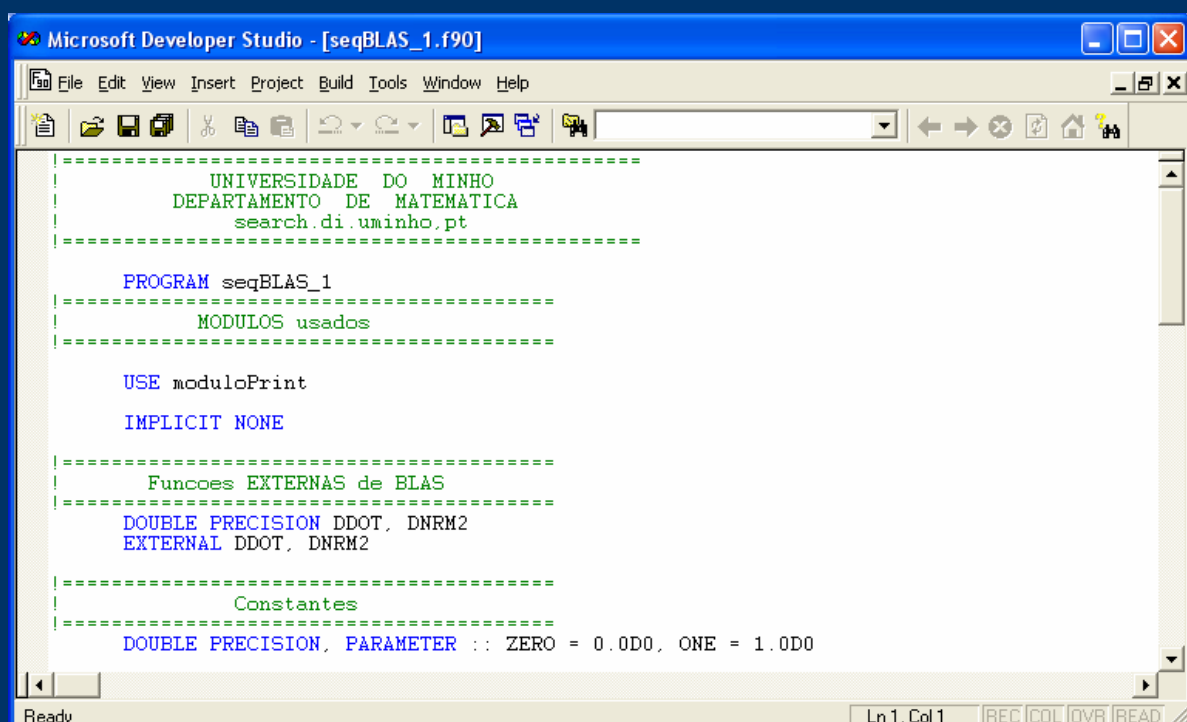


```
Microsoft Developer Studio - [moduloPrint.f90 *]  
File Edit View Insert Project Build Tools Window Help  
SUBROUTINE printVector( v )  
===== Rotina para imprimir un vector =====  
DOUBLE PRECISION, INTENT(IN), DIMENSION(:) :: v  
INTEGER i  
PRINT 20, (v(i), i = 1, SIZE(v))  
20  FORMAT (14F10.4)  
END SUBROUTINE printVector  
END MODULE moduloPrint  
Ready Ln 1, Col 1 REC COL OVR READ
```

► Compilação

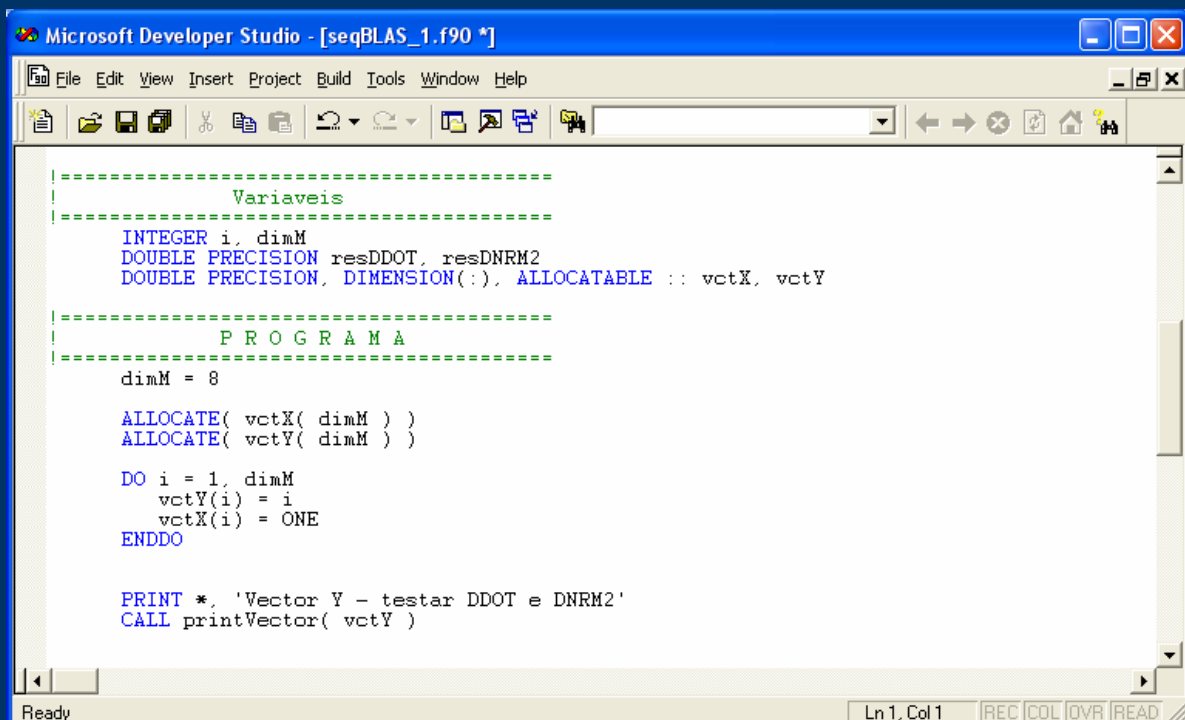
- Executar o comando **ifort -c nomeModulo.f90**

Fortran 90: programa principal (I)



```
Microsoft Developer Studio - [seqBLAS_1.f90]  
File Edit View Insert Project Build Tools Window Help  
===== UNIVERSIDADE DO MINHO =====  
DEPARTAMENTO DE MATEMATICA  
search.di.uminho.pt  
===== PROGRAM seqBLAS_1 =====  
MODULOS usados  
===== USE moduloPrint =====  
IMPLICIT NONE  
===== Funcoes EXTERNAS de BLAS =====  
DOUBLE PRECISION DDOT, DNRM2  
EXTERNAL DDOT, DNRM2  
===== Constantes =====  
DOUBLE PRECISION, PARAMETER :: ZERO = 0.0D0, ONE = 1.0D0  
Ready Ln 1, Col 1 REC COL OVR READ
```

Fortran 90: programa principal (II)



The screenshot shows the Microsoft Developer Studio interface with the file 'seqBLAS_1.f90' open. The code is written in Fortran 90 and includes comments in Portuguese. It defines variables, allocates memory for vectors, and performs some initial calculations.

```
=====
      Variaveis
=====
      INTEGER i, dimM
      DOUBLE PRECISION resDDOT, resDNRM2
      DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctX, vctY

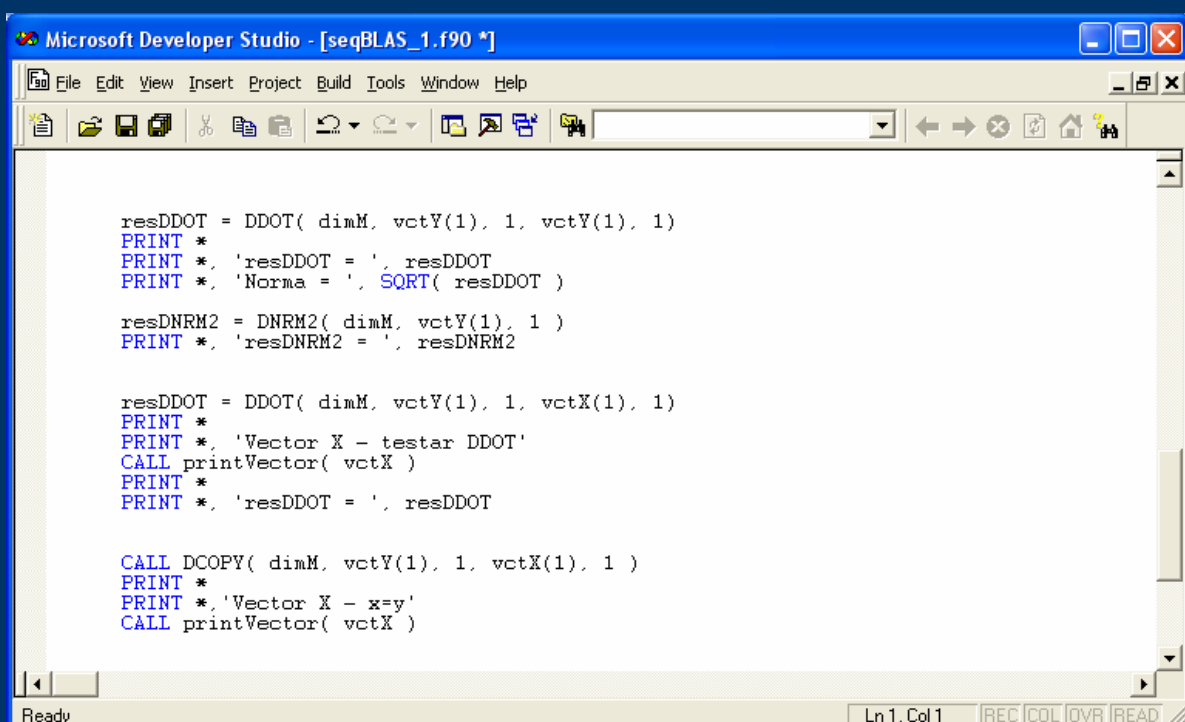
=====
      P R O G R A M A
=====
      dimM = 8

      ALLOCATE( vctX( dimM ) )
      ALLOCATE( vctY( dimM ) )

      DO i = 1, dimM
         vctY(i) = i
         vctX(i) = ONE
      ENDDO

      PRINT *, 'Vector Y - testar DDOT e DNRM2'
      CALL printVector( vctY )
```

Fortran 90: programa principal (III)



The screenshot shows the Microsoft Developer Studio interface with the file 'seqBLAS_1.f90' open. The code continues from the previous slide, performing dot products and norm calculations, and printing the results.

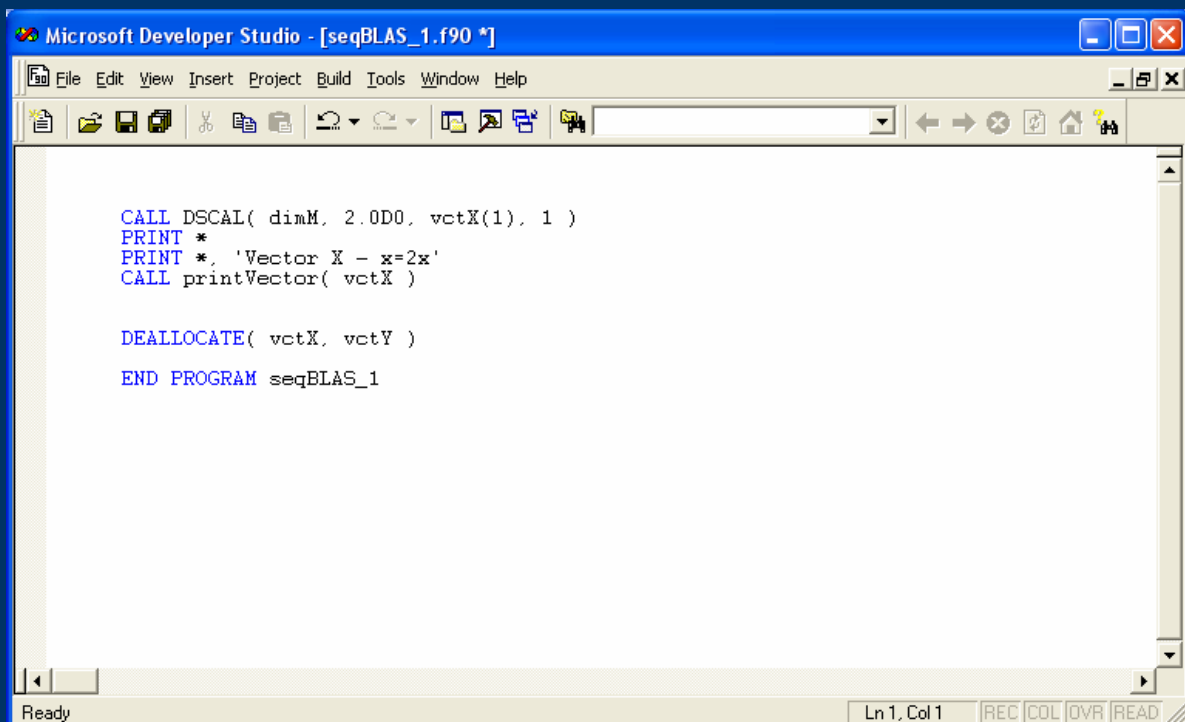
```
      resDDOT = DDOT( dimM, vctY(1), 1, vctY(1), 1 )
      PRINT *
      PRINT *, 'resDDOT = ', resDDOT
      PRINT *, 'Norma = ', SQRT( resDDOT )

      resDNRM2 = DNRM2( dimM, vctY(1), 1 )
      PRINT *, 'resDNRM2 = ', resDNRM2

      resDDOT = DDOT( dimM, vctY(1), 1, vctX(1), 1 )
      PRINT *
      PRINT *, 'Vector X - testar DDOT'
      CALL printVector( vctX )
      PRINT *
      PRINT *, 'resDDOT = ', resDDOT

      CALL DCOPY( dimM, vctY(1), 1, vctX(1), 1 )
      PRINT *
      PRINT *, 'Vector X - x=y'
      CALL printVector( vctX )
```

Fortran 90: programa principal (IV)



The screenshot shows the Microsoft Developer Studio interface with a file named [seqBLAS_1.f90 *]. The code in the editor is as follows:

```
CALL DSCAL( dimM, 2.0D0, vctX(1), 1 )
PRINT *
PRINT *, 'Vector X - x=2x'
CALL printVector( vctX )

DEALLOCATE( vctX, vctY )

END PROGRAM seqBLAS_1
```

The status bar at the bottom indicates 'Ready' and 'Ln 1, Col 1'.

Fortran 90: compilação (I)

► Directiva de compilação

■ `ifort -o nomeExecutável nomeProgramaPrincipal.f90
nomeModulo1.f90 nomeModulo2.f90 ...`

linkagem ao módulo de BLAS (`libmkl_em64t.a`)

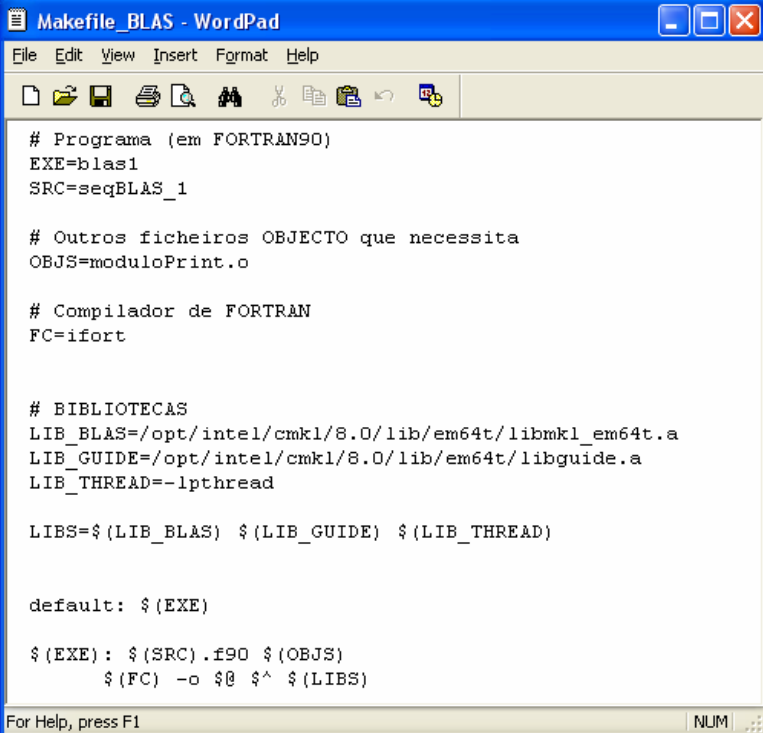
linkagem ao módulo estático (`libguide.a`)

linkagem ao módulo de *threads* (`/usr/lib64/libpthread.a`)

- `ifort -o blas1 seqBLAS_1.f90 moduloPrint.f90
/opt/intel/cmkl/8.0/lib/em64t/libmkl_em64t.a
/opt/intel/cmkl/8.0/lib/em64t/libguide.a -lpthread`

Fortran 90: compilação (II)

- Compilação
 - Fazer **make**



```
# Programa (em FORTRAN90)
EXE=blas1
SRC=seqBLAS_1

# Outros ficheiros OBJECTO que necessita
OBJS=moduloPrint.o

# Compilador de FORTRAN
FC=ifort

# BIBLIOTECAS
LIB_BLAS=/opt/intel/cmk1/8.0/lib/em64t/libmk1_em64t.a
LIB_GUIDE=/opt/intel/cmk1/8.0/lib/em64t/libguide.a
LIB_THREAD=-lpthread

LIBS=${LIB_BLAS} ${LIB_GUIDE} ${LIB_THREAD}

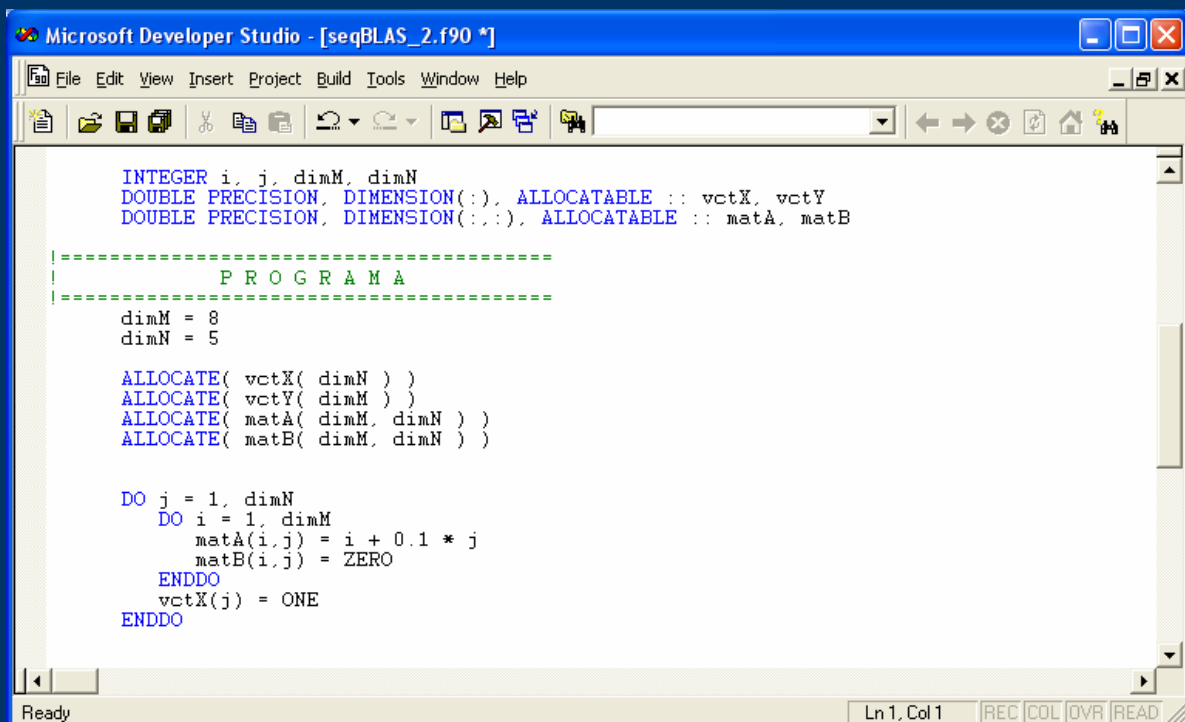
default: ${EXE}

${EXE}: ${SRC}.f90 ${OBJS}
    ${FC} -o $@ $^ ${LIBS}
```

Fortran 90: execução

- Execução
 - Executar o comando **./nomeExecutável**
 - ./blas1
- Exercícios
 - Dado o vector **vctY**
 - ◆ Calcular a norma do vector $7 \cdot \text{vctY}$
 - ◆ Calcular o produto interno do vector que tem as componentes ímpares de **vctY** com o vector que tem as componentes pares de **vctY**

Fortran 90: programa principal (I)



The screenshot shows the Microsoft Developer Studio interface with a file named [seqBLAS_2.f90 *]. The code is as follows:

```
INTEGER i, j, dimM, dimN
DOUBLE PRECISION, DIMENSION(:,), ALLOCATABLE :: vctX, vctY
DOUBLE PRECISION, DIMENSION(:,), ALLOCATABLE :: matA, matB

=====
PROGRAM A
=====

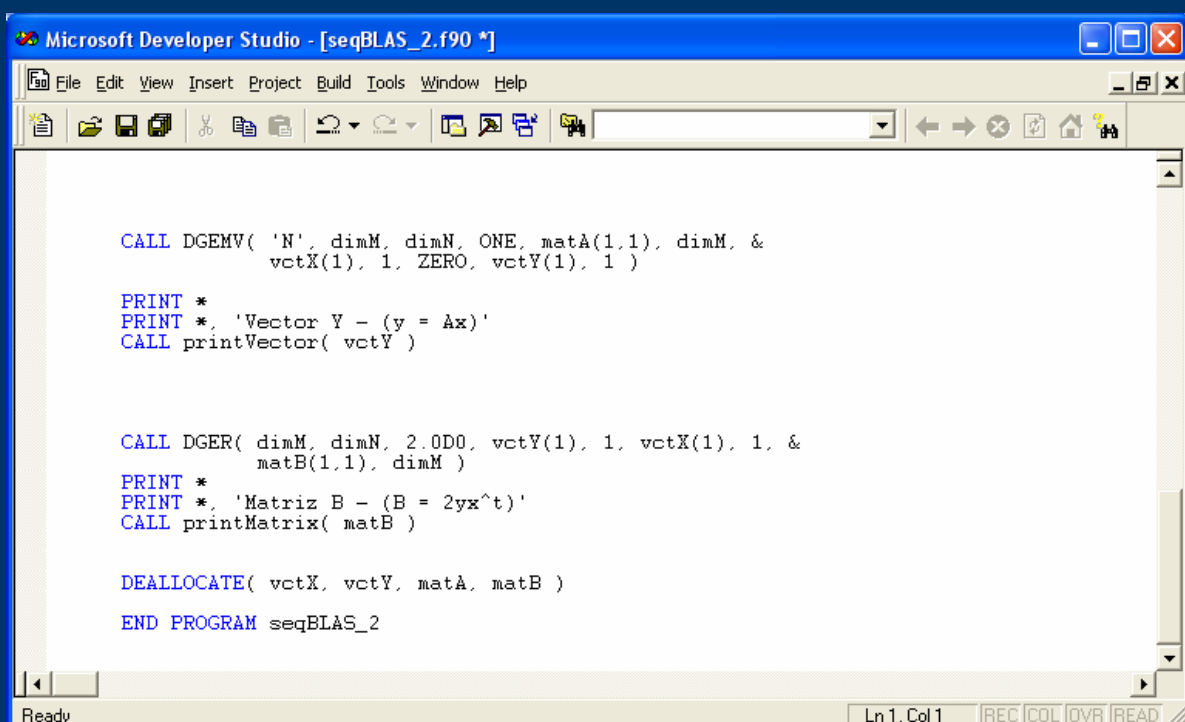
dimM = 8
dimN = 5

ALLOCATE( vctX( dimN ) )
ALLOCATE( vctY( dimM ) )
ALLOCATE( matA( dimM, dimN ) )
ALLOCATE( matB( dimM, dimN ) )

DO j = 1, dimN
  DO i = 1, dimM
    matA(i,j) = i + 0.1 * j
    matB(i,j) = ZERO
  ENDDO
  vctX(j) = ONE
ENDDO
```

The status bar at the bottom indicates 'Ready' and 'Ln 1, Col 1'.

Fortran 90: programa principal (II)



The screenshot shows the Microsoft Developer Studio interface with the same file [seqBLAS_2.f90 *]. The code continues from the previous slide:

```
CALL DGENV( 'N', dimM, dimN, ONE, matA(1,1), dimM, &
           vctX(1), 1, ZERO, vctY(1), 1 )

PRINT *
PRINT *, 'Vector Y - (y = Ax)'
CALL printVector( vctY )

CALL DGER( dimM, dimN, 2.0D0, vctY(1), 1, vctX(1), 1, &
          matB(1,1), dimM )
PRINT *
PRINT *, 'Matriz B - (B = 2yx^t)'
CALL printMatrix( matB )

DEALLOCATE( vctX, vctY, matA, matB )

END PROGRAM seqBLAS_2
```

The status bar at the bottom indicates 'Ready' and 'Ln 1, Col 1'.

Fortran 90: compilação e execução (I)

► Compilação e execução

■ Ficheiro `seqBLAS_2.f90`

- `./blas2`

► Exercícios

■ Dados a matriz `matA` e o vector `vctY`

◆ Calcular a matriz

$$\text{matA} \leftarrow 4 \cdot \text{vctY} \cdot \text{vctY}^T + \text{matA}$$

◆ Com a matriz anterior, calcular o vector

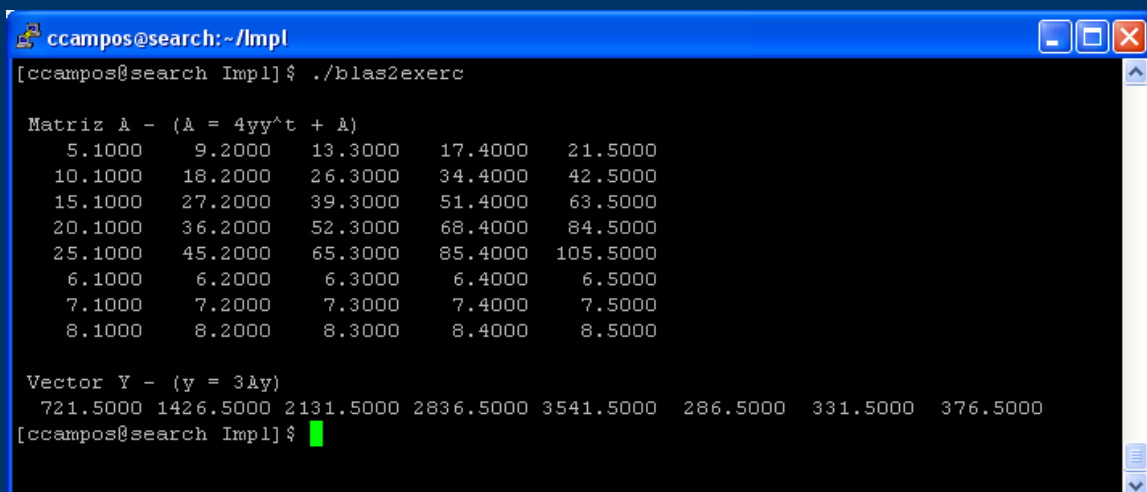
$$\text{vctY} \leftarrow 3 \cdot \text{matA} \cdot \text{vctY}$$

Fortran 90: compilação e execução (II)

► Exercícios

■ `matA` $\leftarrow 4 \cdot \text{vctY} \cdot \text{vctY}^T + \text{matA}$

■ `vctY` $\leftarrow 3 \cdot \text{matA} \cdot \text{vctY}$

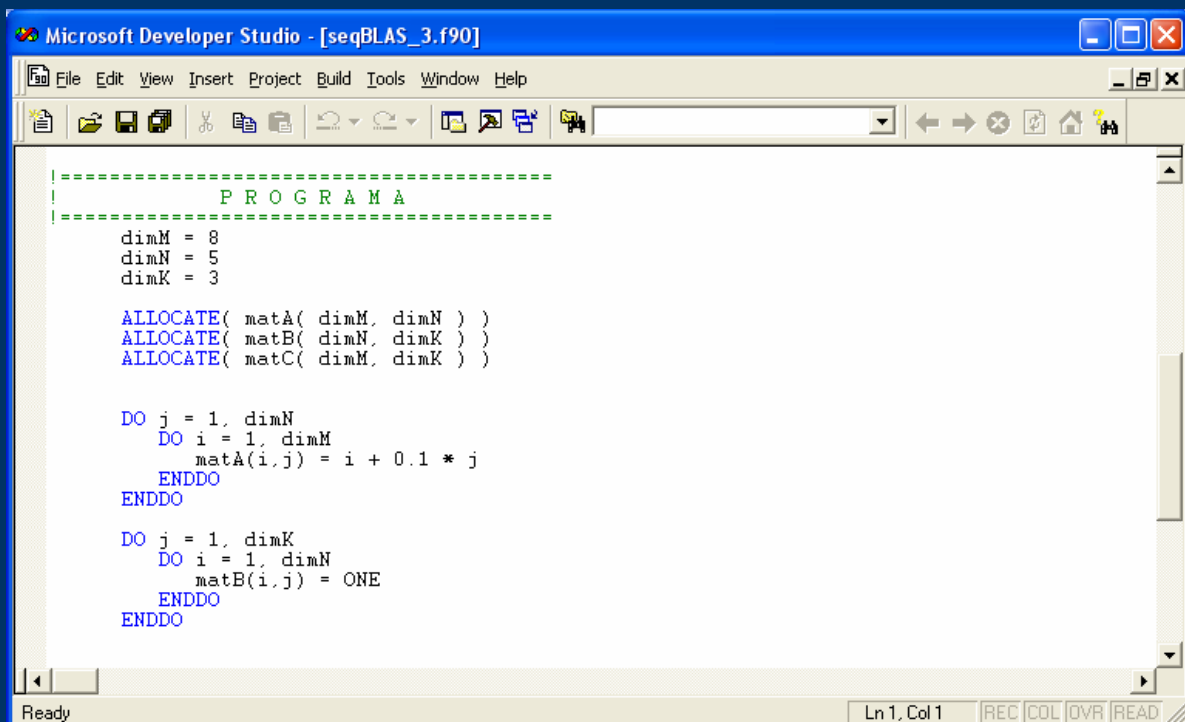


```
ccampos@search:~/Impl
[ccampos@search Impl]$ ./blas2exerc

Matriz A - (A = 4yy^t + A)
  5.1000   9.2000  13.3000  17.4000  21.5000
 10.1000  18.2000  26.3000  34.4000  42.5000
 15.1000  27.2000  39.3000  51.4000  63.5000
 20.1000  36.2000  52.3000  68.4000  84.5000
 25.1000  45.2000  65.3000  85.4000 105.5000
  6.1000   6.2000   6.3000   6.4000   6.5000
  7.1000   7.2000   7.3000   7.4000   7.5000
  8.1000   8.2000   8.3000   8.4000   8.5000

Vector Y - (y = 3Ay)
 721.5000 1426.5000 2131.5000 2836.5000 3541.5000  286.5000  331.5000  376.5000
[ccampos@search Impl]$
```

Fortran 90: programa principal (I)



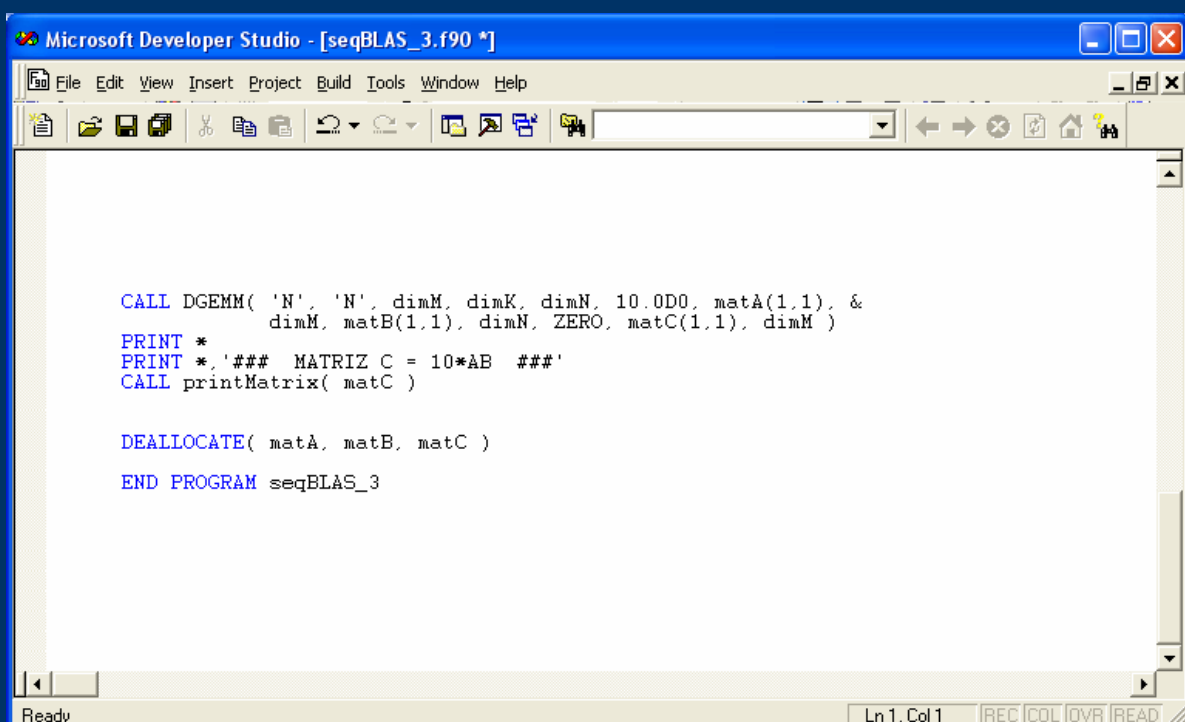
```
=====
PROGRAMA
=====
      dimM = 8
      dimN = 5
      dimK = 3

      ALLOCATE( matA( dimM, dimN ) )
      ALLOCATE( matB( dimN, dimK ) )
      ALLOCATE( matC( dimM, dimK ) )

      DO j = 1, dimN
        DO i = 1, dimM
          matA(i,j) = i + 0.1 * j
        ENDDO
      ENDDO

      DO j = 1, dimK
        DO i = 1, dimN
          matB(i,j) = ONE
        ENDDO
      ENDDO
```

Fortran 90: programa principal (II)



```
CALL DGEMM( 'N', 'N', dimM, dimK, dimN, 10.0D0, matA(1,1), &
           dimM, matB(1,1), dimN, ZERO, matC(1,1), dimM )
PRINT *
PRINT *, '### MATRIZ C = 10*AB ###'
CALL printMatrix( matC )

DEALLOCATE( matA, matB, matC )

END PROGRAM seqBLAS_3
```

Linear Algebra PACKage

LAPACK User's Guide

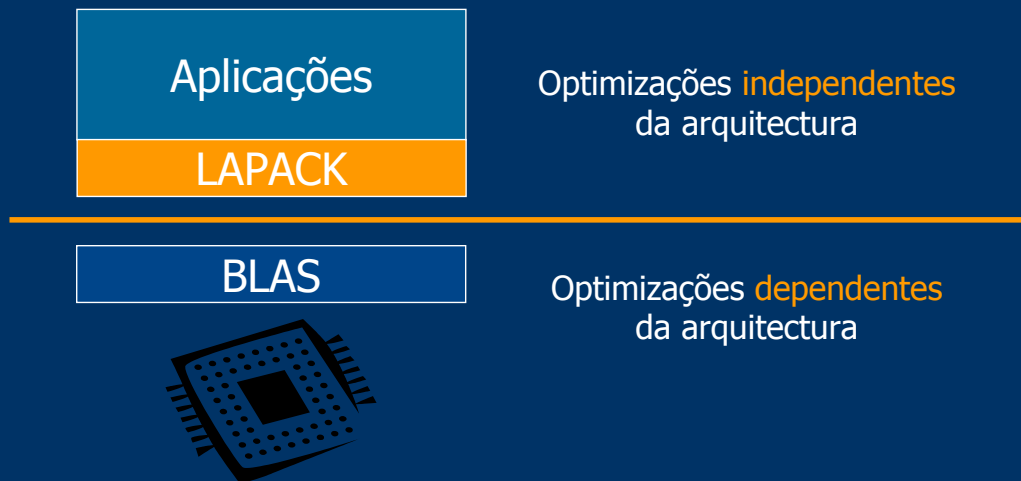
www.netlib.org/lapack/lug/index.html

LAPACK: biblioteca (I)

- ▶ Biblioteca de algoritmos numéricos
- ▶ Conjunto de rotinas destinadas à resolução de problemas numéricos de Álgebra Linear
- ▶ Permite um alto rendimento e um código com um elevado grau de portabilidade
- ▶ Está desenhada para ser eficiente
 - Operações orientadas a blocos matriciais

LAPACK: biblioteca (II)

- ▶ As rotinas de LAPACK realizam a computação através das rotinas de BLAS
 - A eficiência de LAPACK depende da eficiência da BLAS



LAPACK: estrutura

- ▶ LAPACK está estruturada em 3 tipos de rotinas
 - Rotinas **driver** : para a resolução de problemas completos de Álgebra Linear
 - ◆ Resolução de um sistema de equações lineares
 - Rotinas **computacionais** : para a resolução de tarefas concretas de Álgebra Linear
 - ◆ Cálculo da decomposição LU de uma matriz
 - Rotinas **auxiliares** : para a realização de tarefas de menor complexidade
 - ◆ Cálculo da norma de uma matriz

LAPACK: funcionalidade

- ▶ LAPACK resolve numericamente problemas simples e generalizados de Álgebra Linear
 - Sistemas de equações lineares $AX = B$
 - Problemas lineares de mínimos quadrados $\min_x \|b - Ax\|_2$
 - Problemas de valores próprios $Av = \lambda v$
 - Problemas de valores singulares $A = U \Sigma V^t$
 - Outros
 - ◆ Decomposição de matrizes, cálculo de normas matriciais, ...

LAPACK: nomenclatura (I)

- ▶ Igual a BLAS-2 e a BLAS-3 : $xYYZZZ$
 - x : tipo de dados
 - ◆ S, D, C, Z
 - YY : tipo de matriz
 - ◆ BD : bidiagonal
 - ◆ GB : banda geral
 - ◆ GE : densa geral
 - ◆ GG : densa geral generalizada
 - ◆ GT : tridiagonal geral
 - ◆ HB : banda hermítica
 - ◆ HE : hermítica geral
 - ◆ HG : Hessenberg superior generalizada
 - ◆ HP : hermítica compactada
 - ◆ HS : Hessenberg superior
 - ◆ OP : ortogonal compactada
 - ◆ OR : ortogonal geral
 - ◆ PB : simétrica ou banda hermítica positiva definida

LAPACK: nomenclatura (II)

► Igual a BLAS-2 e a BLAS-3 : xYYZZZ

■ YY : tipo de matriz

- ◆ PO : simétrica ou hermítica positiva definida
- ◆ PP : simétrica ou hermítica positiva definida compactada
- ◆ PT : simétrica ou tridiagonal hermítica positiva definida
- ◆ SB : banda simétrica
- ◆ SP : simétrica compactada
- ◆ ST : tridiagonal simétrica
- ◆ SY : simétrica geral
- ◆ TB : banda triangular
- ◆ TG : triangular generalizada
- ◆ TP : triangular compactada
- ◆ TR : triangular geral
- ◆ TZ : trapezoidal geral
- ◆ UN : unitária geral
- ◆ UP : unitária compactada

LAPACK: nomenclatura (III)

► Igual a BLAS-2 e a BLAS-3 : xYYZZZ

■ ZZZ : tipo de operação

- ◆ SV : resolução de sistemas de equações lineares
- ◆ LS : resolução de mínimos quadrados
- ◆ EV : resolução de valores próprios
- ◆ EVD : resolução de valores próprios com o método divide-e-vencerás
- ◆ ES : resolução de valores próprios (factorização de Schur)
- ◆ SVD : resolução da decomposição em valores singulares
- ◆ ESX : resolução de valores próprios generalizados não simétricos com a factorização de Schur
- ◆ ...

LAPACK: gestão de erros (I)

- ▶ O controlo dos erros das rotinas da LAPACK é realizado pela rotina **XERBLA**
- ▶ LAPACK realiza a comprovação dos parâmetros

```
SUBROUTINE DGESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
...
INFO = 0
IF( N.LT.0 ) THEN
    INFO = -1
ELSE IF( NRHS.LT.0 ) THEN
    INFO = -2
ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
    INFO = -4
ELSE IF( LDB.LT.MAX( 1, N ) ) THEN
    INFO = -7
END IF
IF( INFO.NE.0 ) THEN
    CALL XERBLA( 'DGESV ', -INFO )
    RETURN
END IF
...
```

LAPACK: gestão de erros (II)

- ▶ LAPACK devolve os erros dos parâmetros

```
SUBROUTINE DGESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
...
*
*   INFO      (output) INTEGER
*              = 0:  successful exit
*              < 0:  if INFO = -i, the i-th argument had an illegal value
*              > 0:  if INFO = i, U(i,i) is exactly zero.  The factorization
*                   has been completed, but the factor U is exactly
*                   singular, so the solution could not be computed.
*
...
```

- ▶ Conselho

- Durante a fase de implementação deve-se usar esta funcionalidade da LAPACK

LAPACK: espaço de trabalho (I)

- Algumas rotinas de LAPACK necessitam de um **espaço de trabalho** adicional para obter melhores prestações

```
SUBROUTINE DGEQRF( M, N, A, LDA, TAU, WORK, LWORK, INFO )
...
*   WORK      (workspace/output) DOUBLE PRECISION array, dimension (LWORK)
*              On exit, if INFO = 0, WORK(1) returns the optimal LWORK.
*
*   LWORK      (input) INTEGER
*              The dimension of the array WORK.  LWORK >= max(1,N).
*              For optimum performance LWORK >= N*NB, where NB is
*              the optimal blocksize.
...

SUBROUTINE DGESvd( JOBU, JOBVT, M, N, A, LDA, S, U, LDU, VT, LDVT,
$                 WORK, LWORK, INFO )
...
*   LWORK      (input) INTEGER
*              The dimension of the array WORK. LWORK >= 1.
*              LWORK >= MAX(3*MIN(M,N)+MAX(M,N), 5*MIN(M,N)-4).
*              For good performance, LWORK should generally be larger.
...
```

LAPACK: espaço de trabalho (II)

- As rotinas de LAPACK informam sobre a dimensão do **espaço de trabalho óptimo**

```
SUBROUTINE DGESvd( JOBU, JOBVT, M, N, A, LDA, S, U, LDU, VT, LDVT,
$                 WORK, LWORK, INFO )
...
*   WORK      (workspace/output) DOUBLE PRECISION array, dimension (LWORK)
*              On exit, if INFO = 0, WORK(1) returns the optimal LWORK;
*              if INFO > 0, WORK(2:MIN(M,N)) contains the unconverged
*              superdiagonal elements of an upper bidiagonal matrix B
*              whose diagonal is in S (not necessarily sorted). B
*              satisfies A = U * B * VT, so it has the same singular values
*              as A, and singular vectors related by U and VT.
*
*   LWORK      (input) INTEGER
...
*   If LWORK = -1, then a workspace query is assumed; the routine
*   only calculates the optimal size of the WORK array, returns
*   this value as the first entry of the WORK array, and no error
*   message related to LWORK is issued by XERBLA.
...
```

LAPACK: otimização (I)

- ▶ A ÚNICA otimização possível em LAPACK consiste em afinar o **tamanho do bloco óptimo**
- ▶ O tamanho do bloco usado numa rotina depende da rotina **ILAENV**

```
SUBROUTINE DGEBRD( M, N, A, LDA, D, E, TAUQ, TAUP, WORK, LWORK,  
$                INFO )  
...  
    NB = MAX( 1, ILAENV( 1, 'DGEBRD', ' ', M, N, -1, -1 ) )  
...  
    *      Set the crossover point NX.  
    *  
    NX = MAX( NB, ILAENV( 3, 'DGEBRD', ' ', M, N, -1, -1 ) )  
    *  
    *      Determine when to switch from blocked to unblocked code.  
    *  
...
```

LAPACK: otimização (II)

- ▶ ILAENV é definido durante a fase de instalação de LAPACK

- **Pode não ser** o tamanho óptimo

```
FUNCTION ILAENV( ISPEC, NAME, OPTS, N1, N2, N3,  
$              N4 )  
...  
    IF( C2.EQ.'GE' ) THEN  
        IF( C3.EQ.'TRF' ) THEN  
            IF( SNAME ) THEN  
                NB = 64  
            ELSE  
                NB = 64  
            END IF  
...  
    ELSE IF( C3.EQ.'BRD' ) THEN  
        IF( SNAME ) THEN  
            NB = 32  
        ELSE  
            NB = 32  
        END IF  
...  
    RETURN NB  
END FUNCTION
```

LAPACK: otimização (III)

- ▶ Cálculo do tamanho do bloco óptimo a ser usado
 - Começa por definir o tamanho do bloco óptimo invocando a rotina ILAENV
 - Se o valor de LWORK é suficientemente grande, então a rotina usa o tamanho do bloco calculado antes
 - Caso contrário, a rotina calcula o maior tamanho do bloco possível para o valor de LWORK
 - Se o novo valor calculado não é menor que o valor definido por ILAENV, então este é o tamanho do bloco
 - Se não, a rotina usa um algoritmo não orientado a blocos matriciais

LAPACK: parâmetros

- ▶ Os parâmetros das rotinas de LAPACK estão estruturados em 5 grupos
 - Opções da rotina
 - Dimensão do problema
 - Definição das matrizes e vectores
 - Espaço de trabalho
 - Parâmetro INFO

SUBROUTINE DGESVD(
 \$

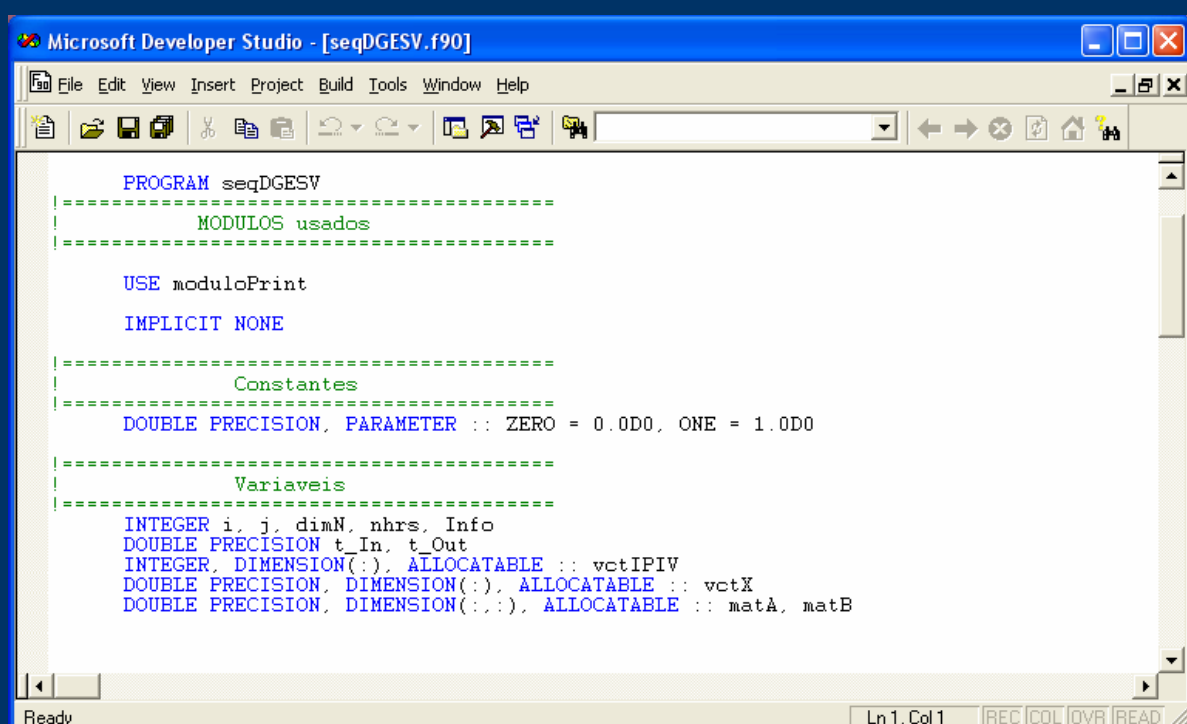
Opções	Dimensão	Matrizes e Vectores
JOBV, JOBV, WORK, LWORK	M, N, A, LDA, S, U, LDU, VT, LDVT, INFO	

Espaço trabalho

LAPACK

Fortran 90 Experiências computacionais

Fortran 90: programa principal (I)



The screenshot shows the Microsoft Developer Studio interface with the file `seqDGESV.f90` open. The code is written in Fortran 90 and includes several sections separated by dashed lines. The code defines constants and variables, and declares arrays for the LAPACK routine `seqDGESV`.

```
PROGRAM seqDGESV
=====
MODULOS usados
=====

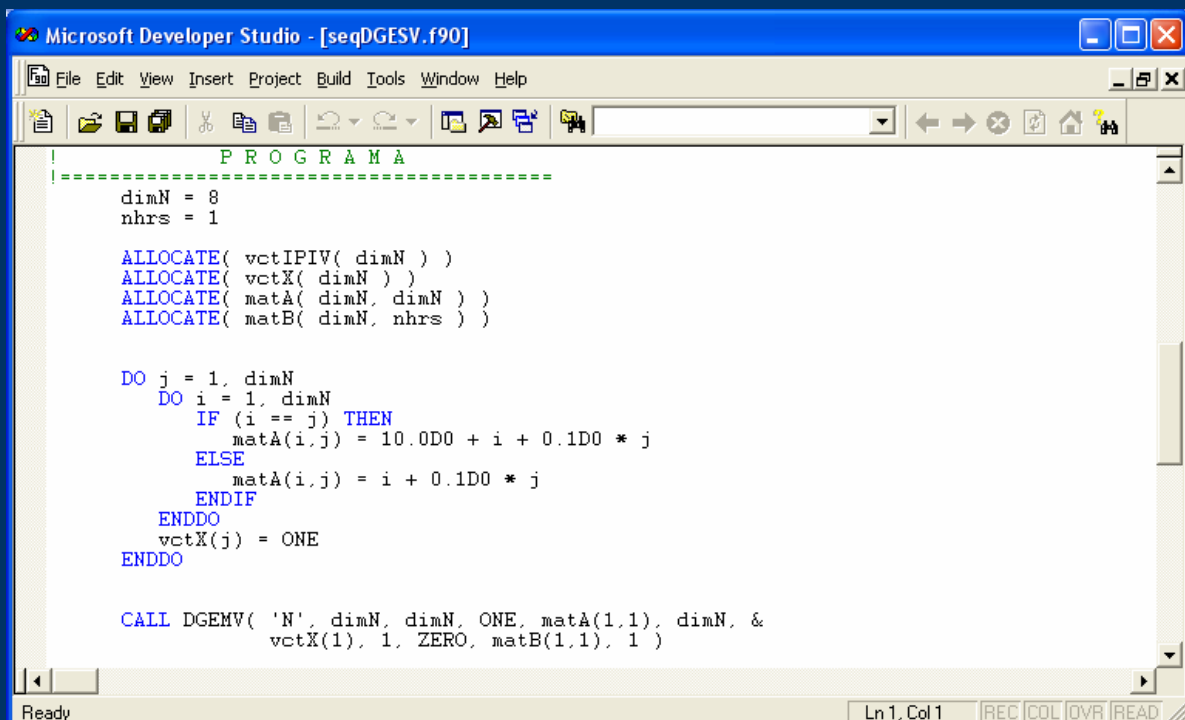
USE moduloPrint

IMPLICIT NONE

=====
Constantes
=====
DOUBLE PRECISION, PARAMETER :: ZERO = 0.0D0, ONE = 1.0D0

=====
Variaveis
=====
INTEGER i, j, dimN, nhrs, Info
DOUBLE PRECISION t_In, t_Out
INTEGER, DIMENSION(:), ALLOCATABLE :: vetIPIV
DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vetX
DOUBLE PRECISION, DIMENSION(:,:), ALLOCATABLE :: matA, matB
```

Fortran 90: programa principal (II)



The screenshot shows the Microsoft Developer Studio interface with the file 'seqDGESV.f90' open. The code defines a program 'PROGRAM A' with the following logic: it sets 'dimN' to 8 and 'nhrs' to 1. It allocates memory for 'vctIPIV', 'vctX', 'matA', and 'matB'. A nested loop over 'j' and 'i' (both from 1 to 'dimN') calculates 'matA(i,j)' based on whether 'i' equals 'j'. After the loop, 'vctX(j)' is set to 'ONE'. Finally, it calls the 'DGEMV' routine with various parameters including 'matA', 'vctX', and 'matB'.

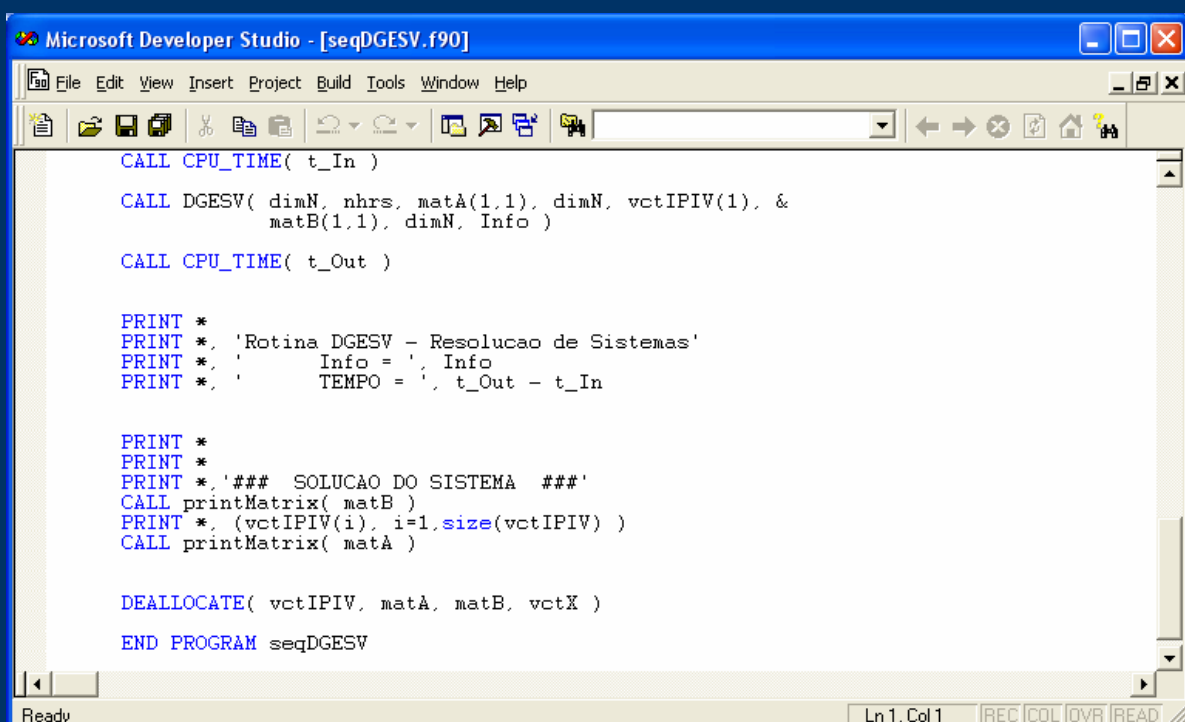
```
PROGRAM A
-----
dimN = 8
nhrs = 1

ALLOCATE( vctIPIV( dimN ) )
ALLOCATE( vctX( dimN ) )
ALLOCATE( matA( dimN, dimN ) )
ALLOCATE( matB( dimN, nhrs ) )

DO j = 1, dimN
  DO i = 1, dimN
    IF ( i == j ) THEN
      matA(i,j) = 10.0D0 + i + 0.1D0 * j
    ELSE
      matA(i,j) = i + 0.1D0 * j
    ENDIF
  ENDDO
  vctX(j) = ONE
ENDDO

CALL DGEMV( 'N', dimN, dimN, ONE, matA(1,1), dimN, &
           vctX(1), 1, ZERO, matB(1,1), 1 )
```

Fortran 90: programa principal (III)



The screenshot shows the continuation of the Fortran 90 code in 'seqDGESV.f90'. It starts by calling 'CPU_TIME' to record the start time. Then it calls 'DGESV' with 'matA', 'vctIPIV', and 'matB' as arguments. Another 'CPU_TIME' call records the end time. The code then prints several lines of information, including the title 'Rotina DGESV - Resolucao de Sistemas', the 'Info' parameter, and the execution time. It also prints the solution of the system using 'printMatrix' for 'matB' and 'matA'. Finally, it deallocates the memory for 'vctIPIV', 'matA', 'matB', and 'vctX', and ends the program.

```
CALL CPU_TIME( t_In )

CALL DGESV( dimN, nhrs, matA(1,1), dimN, vctIPIV(1), &
           matB(1,1), dimN, Info )

CALL CPU_TIME( t_Out )

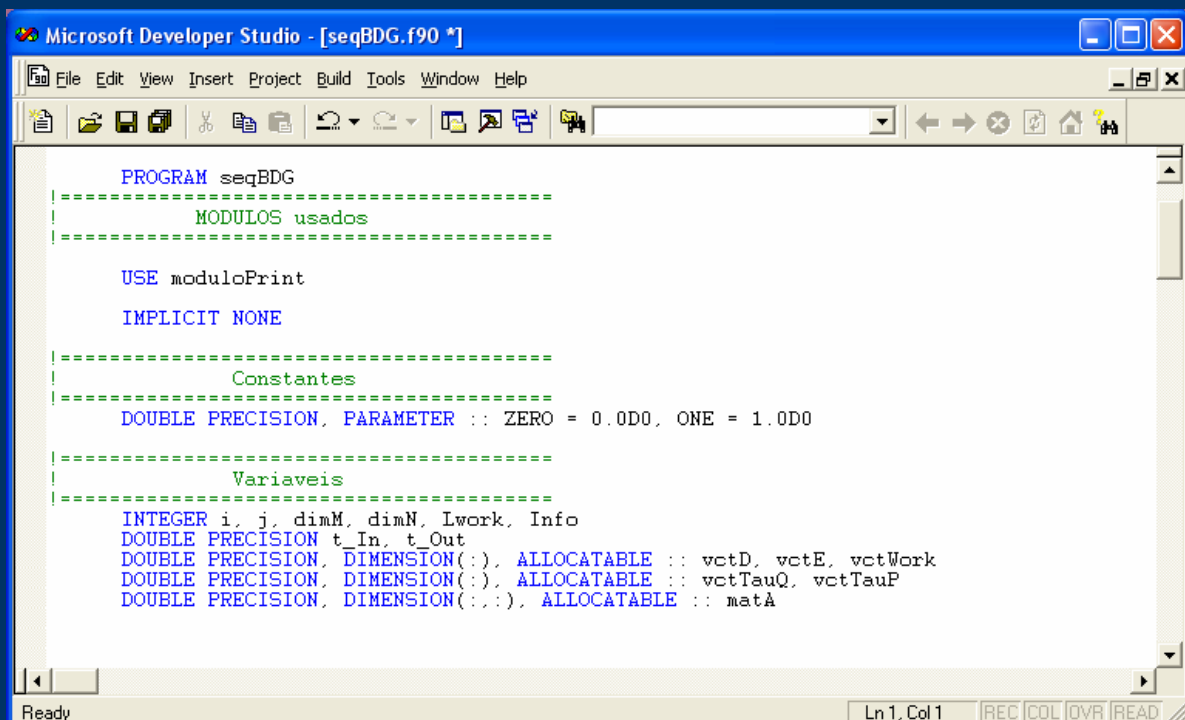
PRINT *
PRINT *, 'Rotina DGESV - Resolucao de Sistemas'
PRINT *, '      Info = ', Info
PRINT *, '      TEMPO = ', t_Out - t_In

PRINT *
PRINT *
PRINT *, '### SOLUCAO DO SISTEMA ###'
CALL printMatrix( matB )
PRINT *, (vctIPIV(i), i=1, size(vctIPIV) )
CALL printMatrix( matA )

DEALLOCATE( vctIPIV, matA, matB, vctX )

END PROGRAM seqDGESV
```

Fortran 90: programa principal (IV)



The screenshot shows the Microsoft Developer Studio interface with the file 'seqBDG.f90' open. The code is written in Fortran 90 and includes several sections separated by dashed lines. The code defines a program 'seqBDG' that uses a module 'moduloPrint' and sets 'IMPLICIT NONE'. It defines constants 'ZERO' and 'ONE' as double precision values. It also declares various variables, including integers 'i', 'j', 'dimM', 'dimN', 'Lwork', and 'Info', and double precision arrays 't_In', 't_Out', 'vctD', 'vctE', 'vctWork', 'vctTauQ', 'vctTauP', and a matrix 'matA'.

```
PROGRAM seqBDG
=====
MODULOS usados
=====

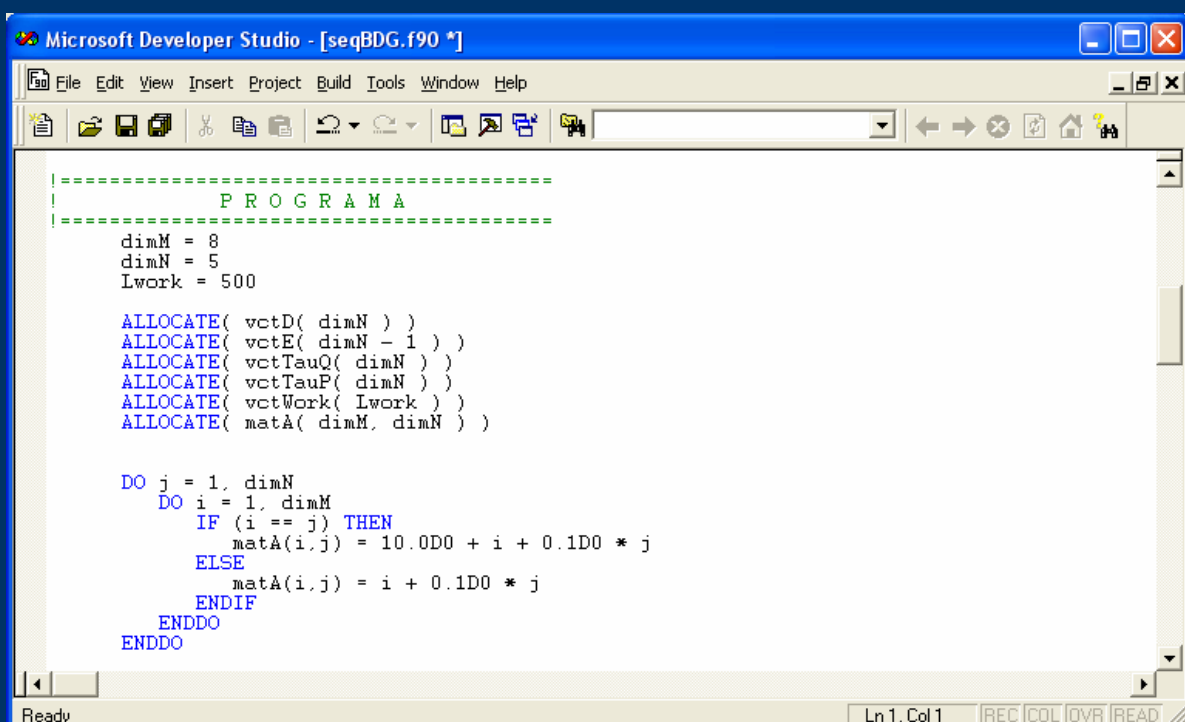
USE moduloPrint

IMPLICIT NONE

=====
Constantes
=====
DOUBLE PRECISION, PARAMETER :: ZERO = 0.0D0, ONE = 1.0D0

=====
Variaveis
=====
INTEGER i, j, dimM, dimN, Lwork, Info
DOUBLE PRECISION t_In, t_Out
DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctD, vctE, vctWork
DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctTauQ, vctTauP
DOUBLE PRECISION, DIMENSION(:,:), ALLOCATABLE :: matA
```

Fortran 90: programa principal (v)



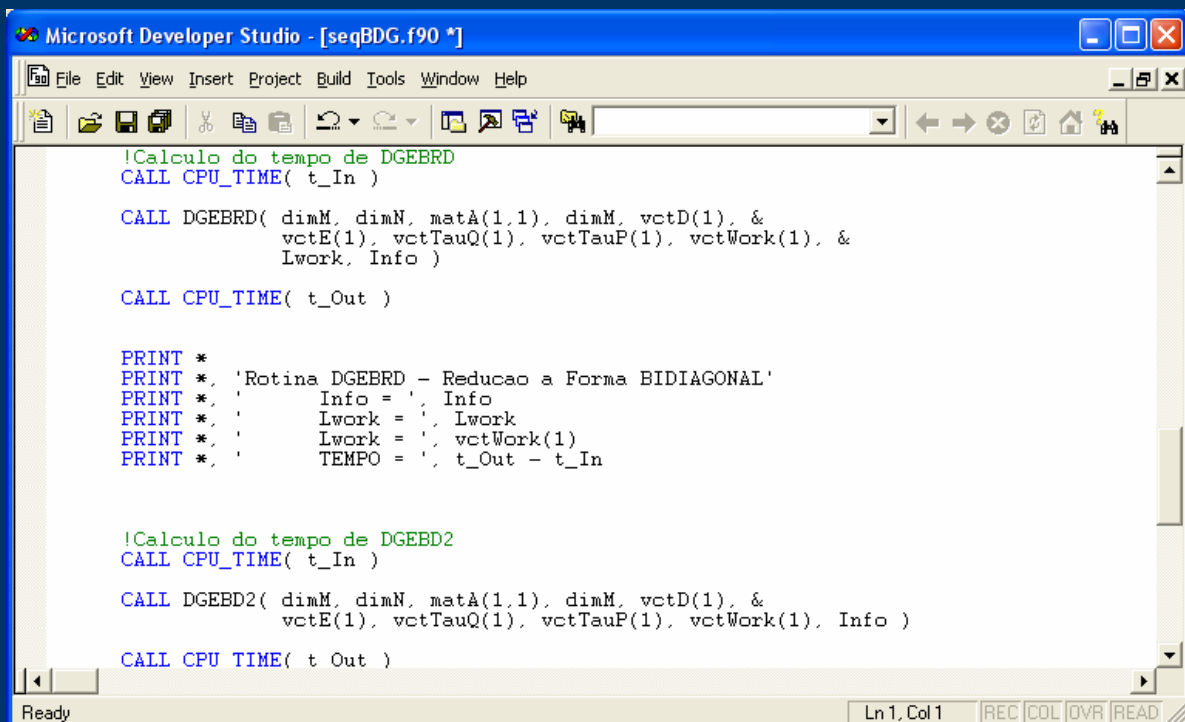
The screenshot shows the Microsoft Developer Studio interface with the file 'seqBDG.f90' open. The code is written in Fortran 90 and includes a section for the main program 'PROGRAMA'. It defines dimensions 'dimM = 8', 'dimN = 5', and 'Lwork = 500'. It then allocates memory for several arrays: 'vctD' (dimension 'dimN'), 'vctE' (dimension 'dimN - 1'), 'vctTauQ' (dimension 'dimN'), 'vctTauP' (dimension 'dimN'), 'vctWork' (dimension 'Lwork'), and 'matA' (dimension 'dimM, dimN'). The main loop is a nested 'DO' loop over 'j' (from 1 to 'dimN') and 'i' (from 1 to 'dimM'). Inside the loop, it checks if 'i == j'. If true, it sets 'matA(i,j) = 10.0D0 + i + 0.1D0 * j'. Otherwise, it sets 'matA(i,j) = i + 0.1D0 * j'.

```
=====
PROGRAMA
=====
dimM = 8
dimN = 5
Lwork = 500

ALLOCATE( vctD( dimN ) )
ALLOCATE( vctE( dimN - 1 ) )
ALLOCATE( vctTauQ( dimN ) )
ALLOCATE( vctTauP( dimN ) )
ALLOCATE( vctWork( Lwork ) )
ALLOCATE( matA( dimM, dimN ) )

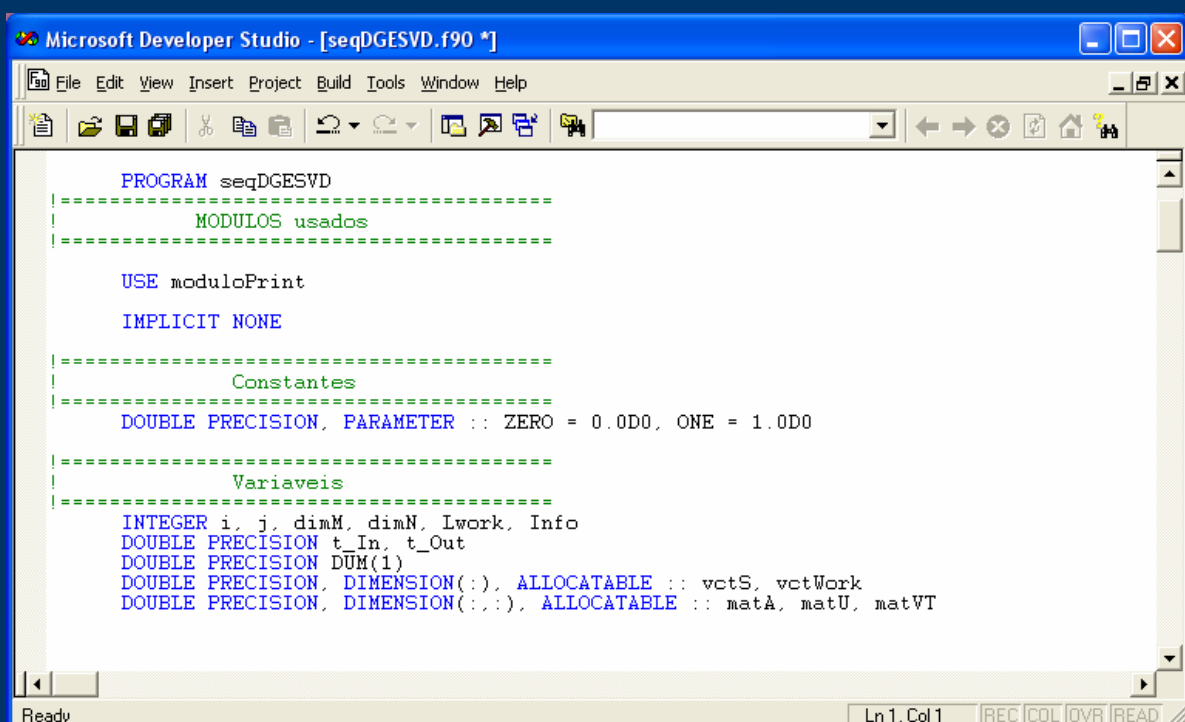
DO j = 1, dimN
  DO i = 1, dimM
    IF (i == j) THEN
      matA(i,j) = 10.0D0 + i + 0.1D0 * j
    ELSE
      matA(i,j) = i + 0.1D0 * j
    ENDIF
  ENDDO
ENDDO
```

Fortran 90: programa principal (VI)



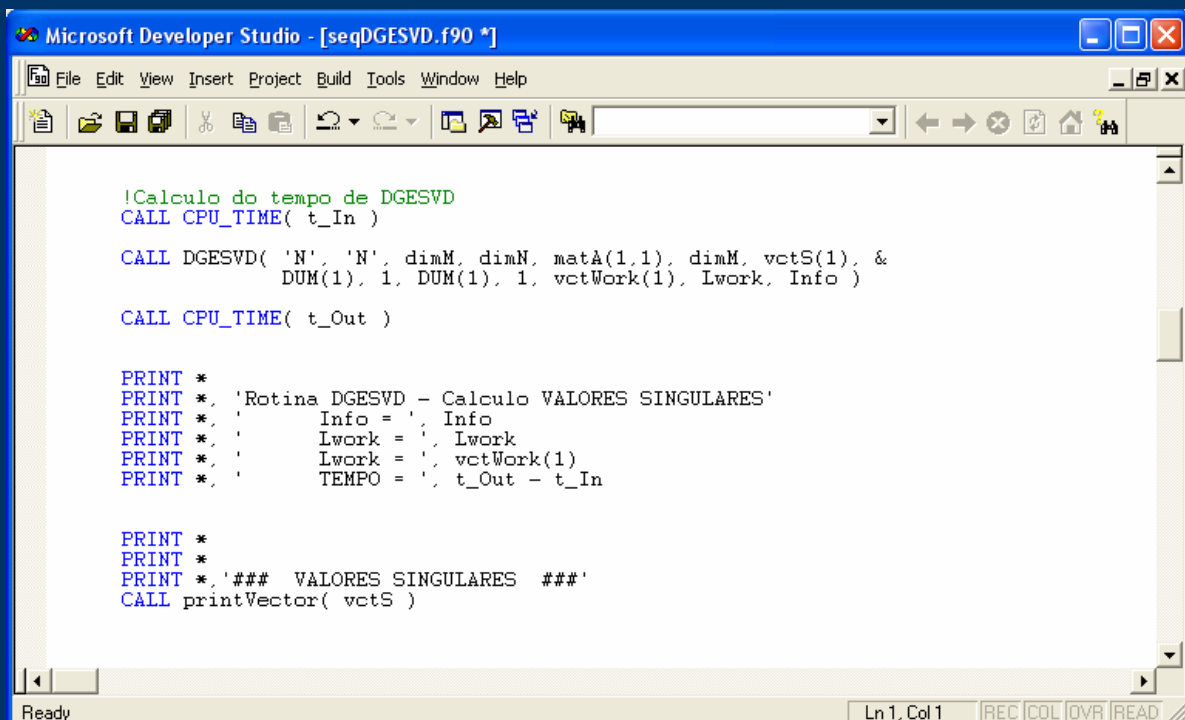
```
Microsoft Developer Studio - [seqBDG.f90 *]  
File Edit View Insert Project Build Tools Window Help  
!Calculo do tempo de DGEGBD  
CALL CPU_TIME( t_In )  
  
CALL DGEGBD( dimM, dimN, matA(1,1), dimM, vctD(1), &  
            vctE(1), vctTauQ(1), vctTauP(1), vctWork(1), &  
            lwork, Info )  
  
CALL CPU_TIME( t_Out )  
  
PRINT *  
PRINT *, 'Rotina DGEGBD - Reducao a Forma BIDIAGONAL'  
PRINT *, '      Info = ', Info  
PRINT *, '      lwork = ', lwork  
PRINT *, '      vctWork(1) = ', vctWork(1)  
PRINT *, '      TEMPO = ', t_Out - t_In  
  
!Calculo do tempo de DGEGBD2  
CALL CPU_TIME( t_In )  
  
CALL DGEGBD2( dimM, dimN, matA(1,1), dimM, vctD(1), &  
             vctE(1), vctTauQ(1), vctTauP(1), vctWork(1), Info )  
  
CALL CPU_TIME( t_Out )  
Ready Ln 1, Col 1 REC COL OVR READ
```

Fortran 90: programa principal (VII)



```
Microsoft Developer Studio - [seqDGESVD.f90 *]  
File Edit View Insert Project Build Tools Window Help  
  
PROGRAM seqDGESVD  
===== MODULOS usados =====  
  
USE moduloPrint  
  
IMPLICIT NONE  
  
===== Constantes =====  
DOUBLE PRECISION, PARAMETER :: ZERO = 0.0D0, ONE = 1.0D0  
  
===== Variaveis =====  
INTEGER i, j, dimM, dimN, lwork, Info  
DOUBLE PRECISION t_In, t_Out  
DOUBLE PRECISION DUM(1)  
DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctS, vctWork  
DOUBLE PRECISION, DIMENSION(:,:), ALLOCATABLE :: matA, matU, matVT  
Ready Ln 1, Col 1 REC COL OVR READ
```

Fortran 90: programa principal (VIII)



The screenshot shows the Microsoft Developer Studio interface with a Fortran 90 program. The code calculates the singular values of a matrix A using the DGESVD routine. It includes timing calls to CPU_TIME and prints the results of the calculation.

```
!Calculo do tempo de DGESVD
CALL CPU_TIME( t_In )

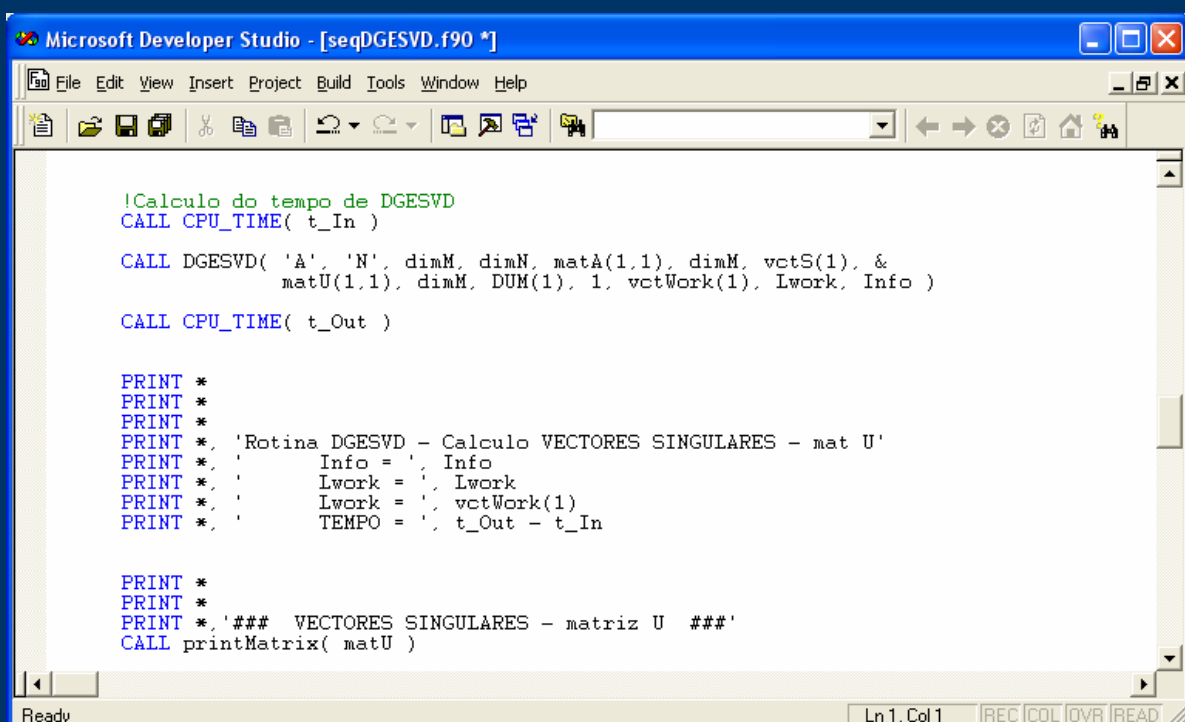
CALL DGESVD( 'N', 'N', dimM, dimN, matA(1,1), dimM, vctS(1), &
            DUM(1), 1, DUM(1), 1, vctWork(1), Lwork, Info )

CALL CPU_TIME( t_Out )

PRINT *
PRINT *, 'Rotina DGESVD - Calculo VALORES SINGULARES'
PRINT *, '      Info = ', Info
PRINT *, '      Lwork = ', Lwork
PRINT *, '      vctWork(1) = ', vctWork(1)
PRINT *, '      TEMPO = ', t_Out - t_In

PRINT *
PRINT *
PRINT *, '### VALORES SINGULARES ###'
CALL printVector( vctS )
```

Fortran 90: programa principal (IX)



The screenshot shows the Microsoft Developer Studio interface with a Fortran 90 program. The code calculates the singular values and the matrix U of a matrix A using the DGESVD routine. It includes timing calls to CPU_TIME and prints the results of the calculation.

```
!Calculo do tempo de DGESVD
CALL CPU_TIME( t_In )

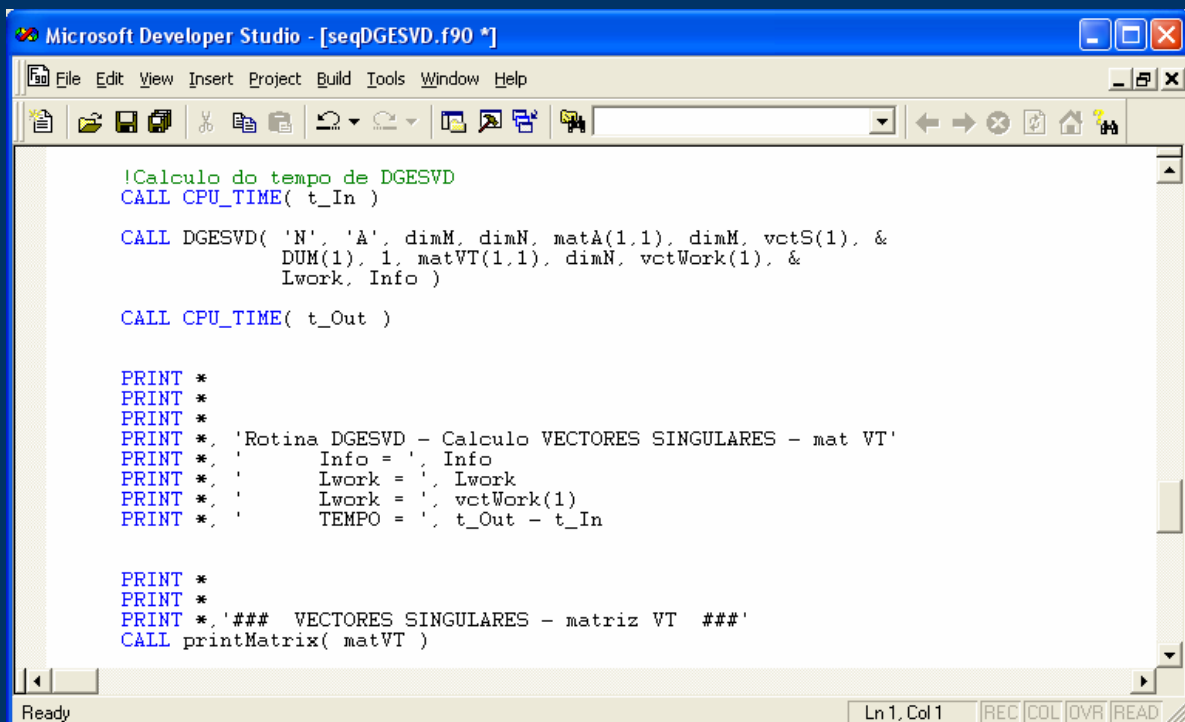
CALL DGESVD( 'A', 'N', dimM, dimN, matA(1,1), dimM, vctS(1), &
            matU(1,1), dimM, DUM(1), 1, vctWork(1), Lwork, Info )

CALL CPU_TIME( t_Out )

PRINT *
PRINT *
PRINT *, 'Rotina DGESVD - Calculo VECTORES SINGULARES - mat U'
PRINT *, '      Info = ', Info
PRINT *, '      Lwork = ', Lwork
PRINT *, '      vctWork(1) = ', vctWork(1)
PRINT *, '      TEMPO = ', t_Out - t_In

PRINT *
PRINT *
PRINT *, '### VECTORES SINGULARES - matriz U ###'
CALL printMatrix( matU )
```

Fortran 90: programa principal (x)



The screenshot shows the Microsoft Developer Studio interface with the file 'seqDGESVD.f90' open. The code is written in Fortran 90 and includes comments in Portuguese. It calls the 'DGESVD' routine from the LAPACK library to perform a singular value decomposition. The code also includes timing calls to 'CPU_TIME' and prints out the results, including the singular values and the matrix 'VT'.

```
!Calculo do tempo de DGESVD
CALL CPU_TIME( t_In )

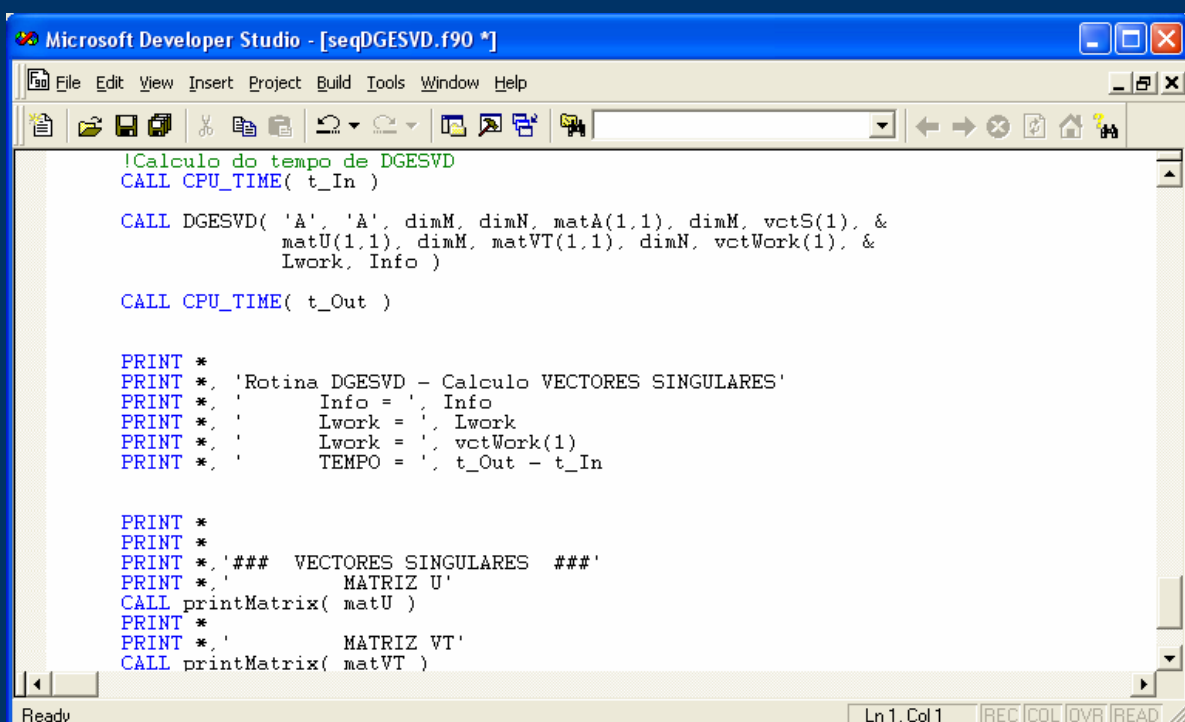
CALL DGESVD( 'N', 'A', dimM, dimN, matA(1,1), dimM, vctS(1), &
             DUM(1), 1, matVT(1,1), dimN, vctWork(1), &
             Lwork, Info )

CALL CPU_TIME( t_Out )

PRINT *
PRINT *
PRINT *
PRINT *, 'Rotina DGESVD - Calculo VECTORES SINGULARES - mat VT'
PRINT *, '      Info = ', Info
PRINT *, '      Lwork = ', Lwork
PRINT *, '      vctWork(1) = ', vctWork(1)
PRINT *, '      TEMPO = ', t_Out - t_In

PRINT *
PRINT *
PRINT *, '### VECTORES SINGULARES - matriz VT ###'
CALL printMatrix( matVT )
```

Fortran 90: programa principal (XI)



This screenshot shows the same Fortran 90 code as the previous slide, but with additional print statements and a call to 'printMatrix' to output the matrix 'U' (labeled as 'MATRIZ U' in the code). The code also prints the singular values and the matrix 'VT'.

```
!Calculo do tempo de DGESVD
CALL CPU_TIME( t_In )

CALL DGESVD( 'A', 'A', dimM, dimN, matA(1,1), dimM, vctS(1), &
             matU(1,1), dimM, matVT(1,1), dimN, vctWork(1), &
             Lwork, Info )

CALL CPU_TIME( t_Out )

PRINT *
PRINT *, 'Rotina DGESVD - Calculo VECTORES SINGULARES'
PRINT *, '      Info = ', Info
PRINT *, '      Lwork = ', Lwork
PRINT *, '      vctWork(1) = ', vctWork(1)
PRINT *, '      TEMPO = ', t_Out - t_In

PRINT *
PRINT *
PRINT *, '### VECTORES SINGULARES ###'
PRINT *, '      MATRIZ U'
CALL printMatrix( matU )
PRINT *
PRINT *, '      MATRIZ VT'
CALL printMatrix( matVT )
```

Fortran 90: compilação (I)

► Directiva de compilação

- `ifort -o nomeExecutável nomeProgramaPrincipal.f90
nomeModulo1.f90 nomeModulo2.f90 ...`

linkagem ao módulo de LAPACK (`libmkl_lapack.a`)

linkagem ao módulo de BLAS (`libmkl_em64t.a`)

linkagem ao módulo estático (`libguide.a`)

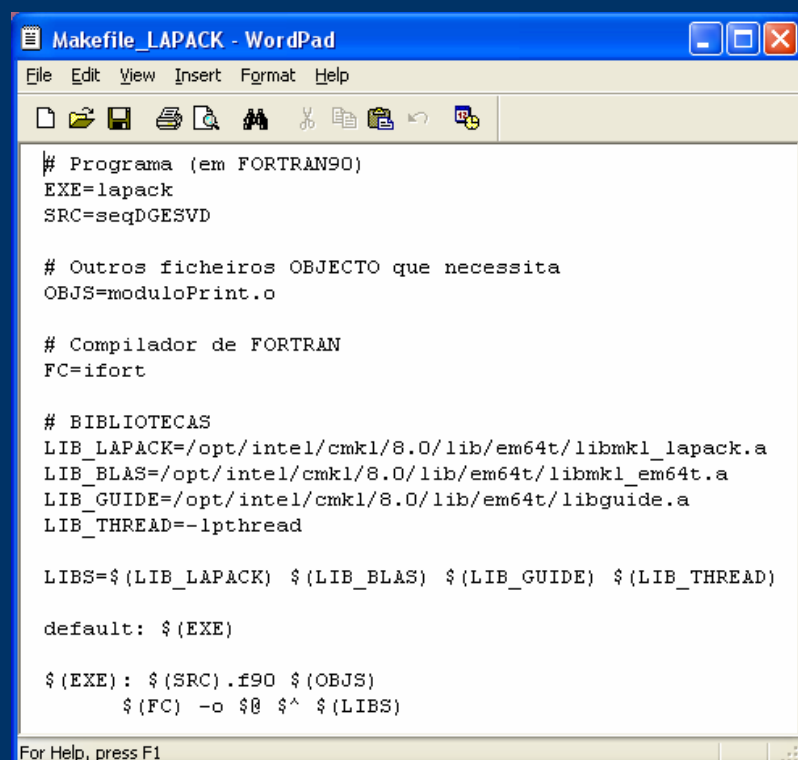
linkagem ao módulo de *threads* (`-lpthread`)

- `ifort -o lapack seqDGESVD.f90 moduloPrint.f90
/opt/intel/cmkl/8.0/lib/em64t/libmkl_lapack.a
/opt/intel/cmkl/8.0/lib/em64t/libmkl_em64t.a
/opt/intel/cmkl/8.0/lib/em64t/libguide.a -lpthread`

Fortran 90: compilação (II)

► Compilação

- Fazer `make`



```
# Programa (em FORTRAN90)
EXE=lapack
SRC=seqDGESVD

# Outros ficheiros OBJECTO que necessita
OBS=moduloPrint.o

# Compilador de FORTRAN
FC=ifort

# BIBLIOTECAS
LIB_LAPACK=/opt/intel/cmkl/8.0/lib/em64t/libmkl_lapack.a
LIB_BLAS=/opt/intel/cmkl/8.0/lib/em64t/libmkl_em64t.a
LIB_GUIDE=/opt/intel/cmkl/8.0/lib/em64t/libguide.a
LIB_THREAD=-lpthread

LIBS=$(LIB_LAPACK) $(LIB_BLAS) $(LIB_GUIDE) $(LIB_THREAD)

default: $(EXE)

$(EXE): $(SRC).f90 $(OBS)
    $(FC) -o $@ $^ $(LIBS)

For Help, press F1
```

LAPACK

Portable Batch System Experiências computacionais

PBS: conceitos gerais

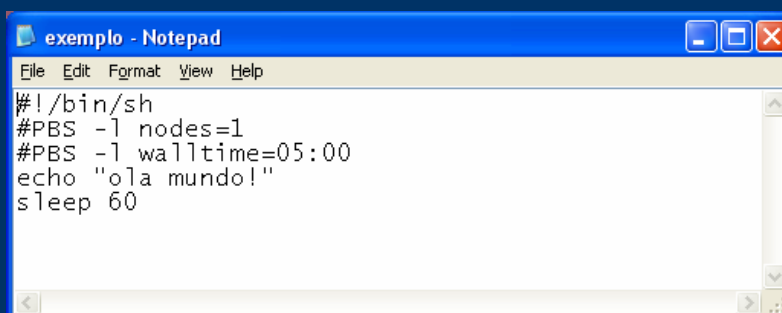
- ▶ Para o processamento massivo os trabalhos devem ser submetidos ao **sistema de gestão de filas**
 - Comandos do **Torque** (baseado no PBS)
 - Comandos do escalonador **Maui**
- ▶ O sistema controla a gestão da lista de trabalhos consoante os recursos disponíveis e de acordo com as políticas definidas e implementadas
 - Maximizar a eficiência da utilização do cluster
 - Justiça no acesso aos recursos

PBS: criação de *scripts* (I)

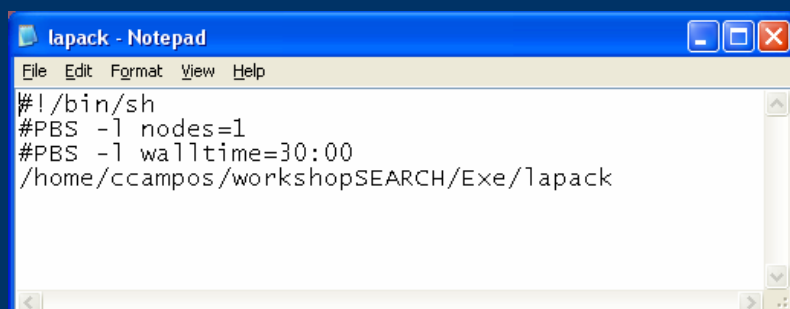
- ▶ Antes de submeter um trabalho para execução deverá criar um *script* com as directivas PBS
 - Número de nós a usar : `-l nodes = #`
 - ◆ Usar os 2 processadores do nó : `-l nodes = # : ppn = 2`
 - Tempo máximo de execução : `-l walltime = hh:mm:ss`
 - Número de CPUS a usar : `-l ncpus = #`
 - Máxima memória requerida : `-l mem = #kb/mb/gb`
 - ...
- ▶ Para obter mais informação
 - Executar o comando `man pbs`

PBS: criação de *scripts* (II)

- ▶ Alguns exemplos de *scripts*



```
#!/bin/sh
#PBS -l nodes=1
#PBS -l walltime=05:00
echo "ola mundo!"
sleep 60
```



```
#!/bin/sh
#PBS -l nodes=1
#PBS -l walltime=30:00
/home/ccampos/workshopSEARCH/Exe/lapack
```

PBS: submeter o trabalho

- ▶ Após a criação do *script* poderá submetê-lo para execução pelo sistema de gestão de filas através do comando **qsub**, utilizando o nome do *script* como parâmetro
 - Executar o comando **qsub nomedascript**
 - ◆ `qsub exemplo.scr`
 - ◆ `qsub lapack.scr`
- ▶ Se o trabalho foi correctamente processado o identificador do novo trabalho será apresentado como resposta do comando
 - ◆ Exemplo : **6990**.search.di.uminho.pt

PBS: apresentação dos resultados

- ▶ Quando o trabalho terminar a directoria onde foi submetido conterá dois ficheiros adicionais
 - Um contendo as mensagens de erro (**stderr**)
 - ◆ `exemplo.scr.e6990`
 - ◆ `lapack.scr.e6990`
 - Outro contendo a saída para a consola (**stdout**)
 - ◆ `exemplo.scr.o6990`
 - ◆ `lapack.scr.o6990`
- ▶ Usando o comando **cat** poderá ver o conteúdo de cada um dos ficheiros

PBS: alguns comandos úteis (I)

► Comandos do Torque

- **qsub** : submeter trabalhos para execução
 - ◆ qsub exemplo.scr
- **qdel** : apagar/remover o trabalho em execução
 - ◆ qdel 6990
- **qstat** : devolve a lista dos trabalhos submetidos para execução
 - ◆ **qstat -a** : lista todos os trabalhos do sistema de gestão
 - ◆ **qstat -r** : lista todos os trabalhos em execução
 - ◆ **qstat -s** : lista todos os trabalhos com comentários de estado
 - ◆ **qstat -q** : lista todas as *queues* do sistema

PBS: alguns comandos úteis (II)

► Comandos do escalonador Maui

- **showq** : lista todos os trabalhos em execução, parados e bloqueados
- **showbf** : lista todos os recursos que actualmente estão disponíveis
- **checkjob** : lista informação sobre os trabalhos
 - ◆ checkjob 6990
- **showstart** : lista o momento em que o trabalho poderá começar a ser executado
 - ◆ showstart 6990
- **showres** : lista a informação de reserva do trabalho

Scalable Linear Algebra PACKage

ScaLAPACK User's Guide

www.netlib.org/scalapack/slug/index.html

ScaLAPACK: biblioteca (I)

- ▶ Biblioteca de rotinas para resolver numericamente problemas de Álgebra Linear em arquiteturas MIMD de memória distribuída e com mecanismos de comunicação
- ▶ **Objectivos**
 - Eficiência
 - Escalabilidade
 - Portabilidade
 - Flexibilidade
 - Facilidade de utilização

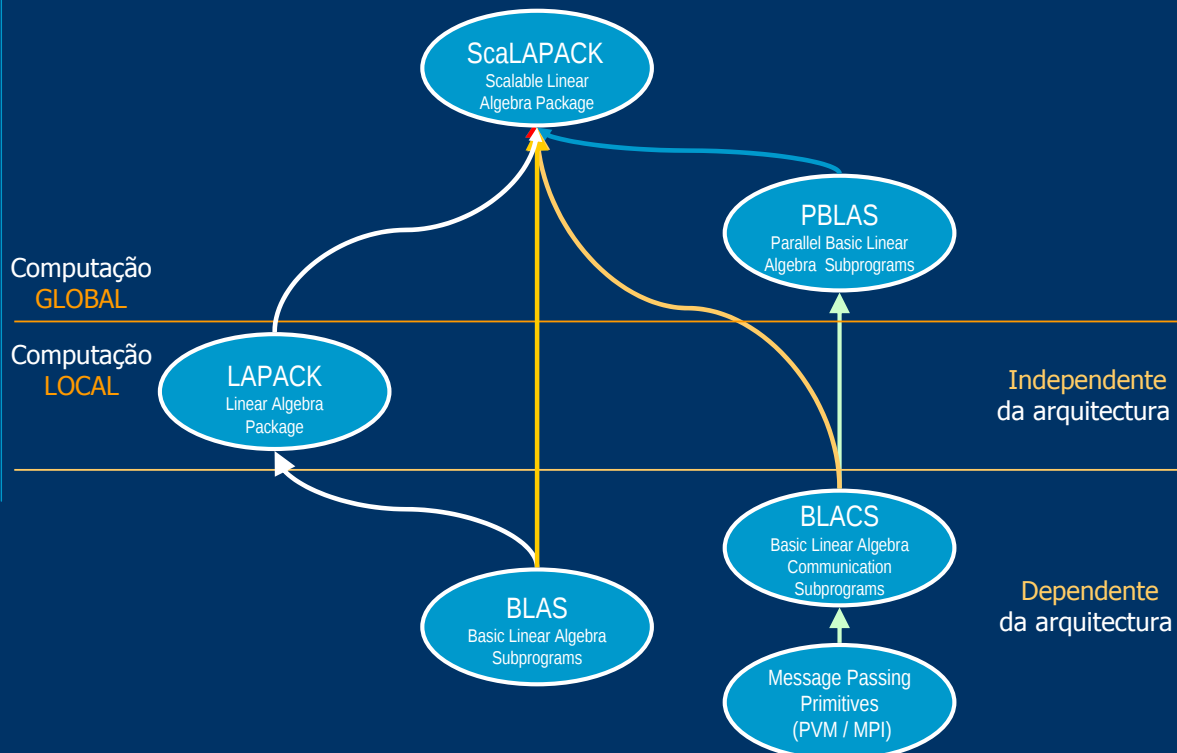
ScaLAPACK: biblioteca (II)

- ▶ Soluciona os mesmos problemas que LAPACK mas em **paralelo**
 - Resolução de sistemas de equações lineares, problemas lineares de mínimos quadrados, problemas de valores próprios, problemas de valores singulares, decomposição de matrizes, cálculo de normas, ...
- ▶ Desenvolvida ao estilo **Single Program Multiple Data**
- ▶ Usa o sistema explícito de troca de mensagens para a comunicação entre processos
 - MPI, PVM, CMMD, MPL, NX

ScaLAPACK: biblioteca (III)

- ▶ Distribuição dos dados de forma **cíclica por blocos**
- ▶ Algoritmos orientados a blocos matriciais
- ▶ Procura ser a extensão da LAPACK para sistemas de memória distribuída
- ▶ Estrutura análoga à da LAPACK
 - Rotinas **driver** : resolvem problemas completos
 - Rotinas **computacionais** : resolvem tarefas concretas
 - Rotinas **auxiliares** : resolvem tarefas de menor complexidade

ScaLAPACK: biblioteca (IV)



ScaLAPACK: biblioteca (V)

► Componentes

- **ScaLAPACK** : Scalable Linear Algebra Package
- **BLAS** : Basic Linear Algebra Subprograms
- **LAPACK** : Linear Algebra Package
- **BLACS** : Basic Linear Algebra Communication Subprograms
 - ◆ Biblioteca com as rotinas dedicadas ao mecanismo de comunicação entre os processos
- **PBLAS** : Parallel Basic Linear Algebra Subprograms
 - ◆ Conjunto de rotinas para memória distribuída análogas às de BLAS

ScaLAPACK: nomenclatura

- ▶ Semelhante à da LAPACK : **PxYYZZZ**
 - P : identifica que as operações são realizadas em paralelo
 - x : tipo de dados
 - YY : tipo de matriz
 - ◆ GB : banda geral
 - ◆ GE : densa geral
 - ◆ GG : densa geral generalizada
 - ◆ HE : hermítica geral
 - ◆ OR : ortogonal geral
 - ◆ SY : simétrica geral
 - ◆ UN : unitária geral
 - ◆ ...
 - ZZZ : tipo de método

ScaLAPACK

BLACS

www.netlib.org/blacs/index.html

BLACS: introdução

► Objectivos

■ Facilidade de implementação

- ◆ Simplifica a troca de mensagens entre os processos, permitindo reduzir os erros de implementação

■ Facilidade de utilização

- ◆ Está a um nível mais elevado, permitindo uma melhor utilização por parte de programadores menos experientes

■ Portabilidade

- ◆ Proporciona uma interface que é suportada por uma ampla gama de arquitecturas de memória distribuída

BLACS: malha de processos (I)

► Mapeamento de P processos em um grid 2D

Vector linear de **ID** de processos

0	1	2	...	P-1
---	---	---	-----	-----

Row-major order

	0	1	2	3
0	0	1	2	3
1	4	5	6	7

8 processos mapeados em um grid de 2x4

	0	1
0	0	1
1	2	3
2	4	5

6 processos mapeados em um grid de 3x2

Column-major order

	0	1	2	3
0	0	2	4	6
1	1	3	5	7

8 processos mapeados em um grid de 2x4

	0	1
0	0	3
1	1	4
2	2	5

6 processos mapeados em um grid de 3x2

BLACS: malha de processos (II)

- ▶ Os P processos são distribuídos em um grid 2D

$$\left. \begin{array}{l} \blacksquare P_r \text{ processos por coluna} \\ \blacksquare P_c \text{ processos por linha} \end{array} \right\} P = P_r * P_c$$

- ▶ Cada processo pode ser referenciado pelas suas coordenadas (pr, pc)

$$\blacksquare 0 \leq pr \leq P_r - 1$$

$$\blacksquare 0 \leq pc \leq P_c - 1$$

	0	1	2	3
0	0	1	2	3
1	4	5	6	7

$P_r = 2$
 $P_c = 4$

8 processos mapeados
em um grid de 2x4

BLACS: malha de processos (III)

- ▶ Definição do grid 2D

$$Grid_{P_r \times P_c} = \{ (pr, pc) : 0 \leq pr \leq P_r - 1 \wedge 0 \leq pc \leq P_c - 1 \}$$

- ▶ Função de localização dos processos

$$\phi : \{0, 1, K, p-1\} \rightarrow \left(\left\lfloor \frac{i}{P_c} \right\rfloor, i \bmod P_c \right)$$

$i \qquad \alpha$

- ▶ Função de identificação dos processos

$$\phi^{-1} : Grid_{P_r \times P_c} \rightarrow \{0, 1, K, p-1\}$$

$(pr, pc) \qquad \alpha \qquad pr * P_c + pc$

BLACS: malha de processos (IV)

Exemplo

Função de localização dos processos

$$\phi : \{0, 1, K, p-1\} \rightarrow \text{Grid}_{Pr \times Pc} \left(\left\lfloor \frac{i}{Pc} \right\rfloor, i \bmod Pc \right)$$

Função de identificação dos processos

$$\phi^{-1} : \text{Grid}_{Pr \times Pc} \rightarrow \{0, 1, K, p-1\}$$

$$(pr, pc) \mapsto pr * Pc + pc$$

$$\phi(3) = \left(\left\lfloor \frac{3}{4} \right\rfloor, 3 \bmod 4 \right) = (0, 3)$$

$$\phi(6) = \left(\left\lfloor \frac{6}{4} \right\rfloor, 6 \bmod 4 \right) = (1, 2)$$

$$\phi^{-1}(0, 2) = 0 * 4 + 2 = 2 \equiv P_2$$

$$\phi^{-1}(1, 2) = 1 * 4 + 2 = 6 \equiv P_6$$

	0	1	2	3
0	0	1	2	3
1	4	5	6	7

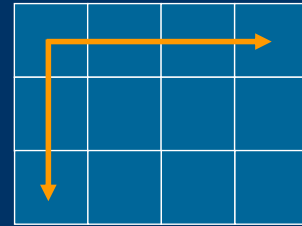
8 processos mapeados em um grid de 2x4

BLACS: contextos

- ▶ Em BLACS, todo o grid de processos está englobado dentro de um **contexto**
- ▶ Um **contexto** é um filtro que permite separar as mensagens de distintos grids de processos
 - Equivalente ao *MPI communicator*
- ▶ Os contextos permitem
 - Criar grupos arbitrários de processos
 - Criar grids disjuntos ou não
 - Isolar grids para que não interfiram com outros grids

BLACS: âmbito das operações

- ▶ Operações em que participam um determinado grupo de processos
- ▶ Em um grid 2D há 3 âmbitos naturais que afectam
 - Toda uma linha de processos
 - Toda uma coluna de processos
 - Todos os processos
- ▶ O “aproveitamento” do hardware deve ser garantido pelas bibliotecas de troca de mensagens sobre as quais BLACS trabalha



ScaLAPACK

PBLAS Quick Reference Guide

www.netlib.org/scalapack/html/pblas_qref.html

PBLAS: introdução

- ▶ PBLAS oferece à ScaLAPACK as rotinas equivalentes que BLAS oferece à LAPACK
 - Conjunto de rotinas para memória distribuída
 - ◆ Consegue-se que as implementações em ScaLAPACK sejam bastante parecidas às implementações em LAPACK
- ▶ PBLAS está estruturada em 3 níveis
 - PBLAS nível 1 : operações entre vetores
 - PBLAS nível 2 : operações entre matrizes e vetores
 - PBLAS nível 3 : operações entre matrizes
- ▶ Nomenclatura das rotinas de PBLAS : **PxYYZZZ**

PBLAS: distribuição dos dados (I)

- ▶ Distribuição **cíclica por blocos**
- ▶ Propriedades
 - Engloba outras distribuições de dados
 - Permite um balanceamento da carga
 - Favorece a performance e a escalabilidade
- ▶ O programador deve distribuir os dados pelos processos antes de chamar as rotinas que sobre eles operam
- ▶ Cada processo tem armazenado na sua memória local o conjunto de dados que lhe corresponde

PBLAS: distribuição dos dados (II)

► Distribuição **cíclica por blocos**

■ Exemplo

- ◆ Matriz de dimensão 5x7
- ◆ Grid de processos 2x2
- ◆ Blocos matriciais de dimensão 2x2

	0	1
0	0	1
1	2	3

grid de 2x2

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}	a_{57}

a_{11}	a_{12}	a_{15}	a_{16}	a_{13}	a_{14}	a_{17}
a_{21}	a_{22}	a_{25}	a_{26}	a_{23}	a_{24}	a_{27}
a_{51}	a_{52}	a_{55}	a_{56}	a_{53}	a_{54}	a_{57}
a_{31}	a_{32}	a_{35}	a_{36}	a_{33}	a_{34}	a_{37}
a_{41}	a_{42}	a_{45}	a_{46}	a_{43}	a_{44}	a_{47}

PBLAS: distribuição dos dados (III)

► Todos os parâmetros que definem uma **matriz distribuída** são encapsulados em um vector de inteiros designado por **descriptor** (array **DESC_**)

► Rotina **DESCINIT** para inicializar o **descriptor**

► Elementos do **descriptor** **DESC_**

- [1] **TYPE_** : tipo de descriptor
- [2] **CTXT_** : contexto de BLACS
- [3] **M_** : número de linhas da matriz
- [4] **N_** : número de colunas da matriz
- [5] **MB_** : tamanho do bloco de linhas
- [6] **NB_** : tamanho do bloco de colunas
- [7] **RSCR_** : linha do processo que contém a primeira linha da matriz
- [8] **CSCR_** : coluna do processo que contém a primeira coluna da matriz
- [9] **LLD_** : dimensão principal da matriz local

PBLAS: distribuição dos dados (IV)

► Distribuição **cíclica por blocos**

■ Exemplo

- ◆ Matriz de dimensão 5x7
- ◆ Grid de processos 2x2
- ◆ Blocos matriciais de dimensão 2x2

	0	1
0	0	1
1	2	3

grid de 2x2

a_{11}	a_{12}	a_{15}	a_{16}	a_{13}	a_{14}	a_{17}
a_{21}	a_{22}	a_{25}	a_{26}	a_{23}	a_{24}	a_{27}
a_{51}	a_{52}	a_{55}	a_{56}	a_{53}	a_{54}	a_{57}
a_{31}	a_{32}	a_{35}	a_{36}	a_{33}	a_{34}	a_{37}
a_{41}	a_{42}	a_{45}	a_{46}	a_{43}	a_{44}	a_{47}

DESC_A

TYPE_A = 1 (densa)

CTXT_A = #

M_A = 5

N_A = 7

MB_A = 2

NB_A = 2

RSRC_A = 0

CSRC_A = 0

LLD_A = 3 ou 2

ScaLAPACK

Fortran 90
Experiências computacionais

Fortran 90: compilação (I)

► Atenção ao compilador

- A compilação de códigos com MPI deve ser realizada com o correcto comando `mpi`
 - ◆ Programas implementados em Fortran 90 : `mpiifort` / `mpif90`

► Linkagem de forma estática

- A linkagem à principal biblioteca de MPI (`libmpi`) é feita de forma estática
 - ◆ Usa-se a biblioteca `libmpi.a` e não a biblioteca `libmpi.so`
 - ◆ A directiva de compilação inclui a opção `-static_mpi`

Fortran 90: compilação (II)

► Directivas de compilação

- `mpiifort -static_mpi -o nomeExecutável
nomeProgramaPrincipal.f90
nomeModulo1.f90 nomeModulo2.f90 ...`

linkagem ao módulo de **ScaLAPACK** (`libmkl_scalapack.a`)

linkagem ao módulo de BLACS (`libmkl_blacs_intelmpi.a`)

linkagem ao módulo de LAPACK (`libmkl_lapack.a`)

linkagem ao módulo de BLAS (`libmkl_em64t.a`)

linkagem ao módulo estático (`libguide.a`)

linkagem ao módulo de *threads* (`-lpthread`)

Fortran 90: compilação (III)

► Módulo de ScaLAPACK

- As rotinas de PBLAS estão incluídas na biblioteca da ScaLAPACK (**libmkl_scalapack.a**)

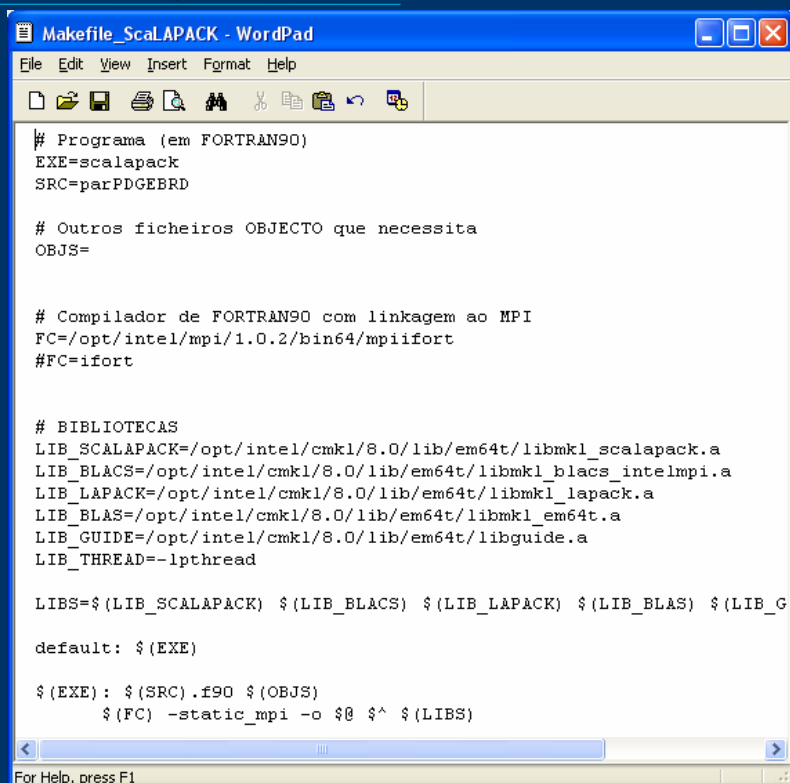
► Pesquisar elementos dentro de uma biblioteca

- Usar o comando **nm** o qual lista os símbolos existentes em ficheiros objecto
- Pesquisa de elementos de PBLAS
 - `nm /opt/intel/cmkl/8.0/lib/em64t/libmkl_scalapack.a | grep pdnmr2`
 - `nm /opt/intel/cmkl/8.0/lib/em64t/libmkl_scalapack.a | grep pddot`

Fortran 90: compilação (IV)

► Compilação

- Fazer **make**



```
# Programa (em FORTRAN90)
EXE=scalapack
SRC=parPDGEBRD

# Outros ficheiros OBJECTO que necessita
OBJ=

# Compilador de FORTRAN90 com linkagem ao MPI
FC=/opt/intel/mpi/1.0.2/bin64/mpiifort
#FC=ifort

# BIBLIOTECAS
LIB_SCALAPACK=/opt/intel/cmkl/8.0/lib/em64t/libmkl_scalapack.a
LIB_BLACS=/opt/intel/cmkl/8.0/lib/em64t/libmkl_blacs_intelmpi.a
LIB_LAPACK=/opt/intel/cmkl/8.0/lib/em64t/libmkl_lapack.a
LIB_BLAS=/opt/intel/cmkl/8.0/lib/em64t/libmkl_em64t.a
LIB_GUIDE=/opt/intel/cmkl/8.0/lib/em64t/libguide.a
LIB_THREAD=-lpthread

LIBS=$(LIB_SCALAPACK) $(LIB_BLACS) $(LIB_LAPACK) $(LIB_BLAS) $(LIB_G

default: $(EXE)

$(EXE): $(SRC).f90 $(OBJ)
    $(FC) -static_mpi -o $@ $^ $(LIBS)
```

Fortran 90: execução (I)

► Processamento "ligeiro"

- Executar o comando **mpirun** com as opções adequadas

- ◆ Executar o comando

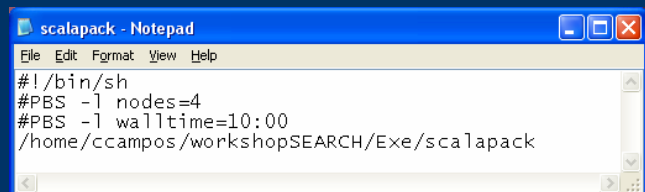
- **mpirun -np # ./nomeExecutavel**

► Processamento massivo

- Criar os ficheiros *script* com as directivas PBS desejadas e submeter o trabalho ao sistema de gestão de filas

- ◆ Executar o comando

- **qsub nomeScript**

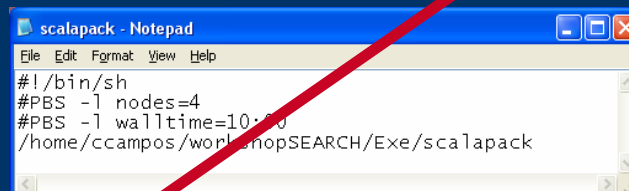


```
#!/bin/sh
#PBS -l nodes=4
#PBS -l walltime=10:00
/home/ccampos/workshopSEARCH/Exe/scalapack
```

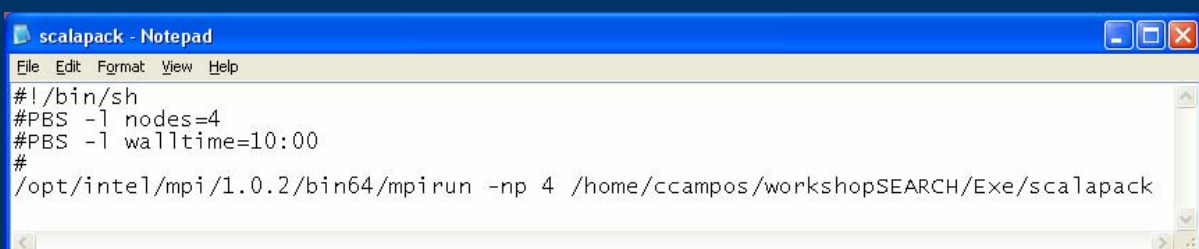
Fortran 90: execução (II)

► Processamento massivo

- O script anterior não é correcto e gera erros de execução devido ao facto do comando **qsub** usar o **mpirun** de **mpich**

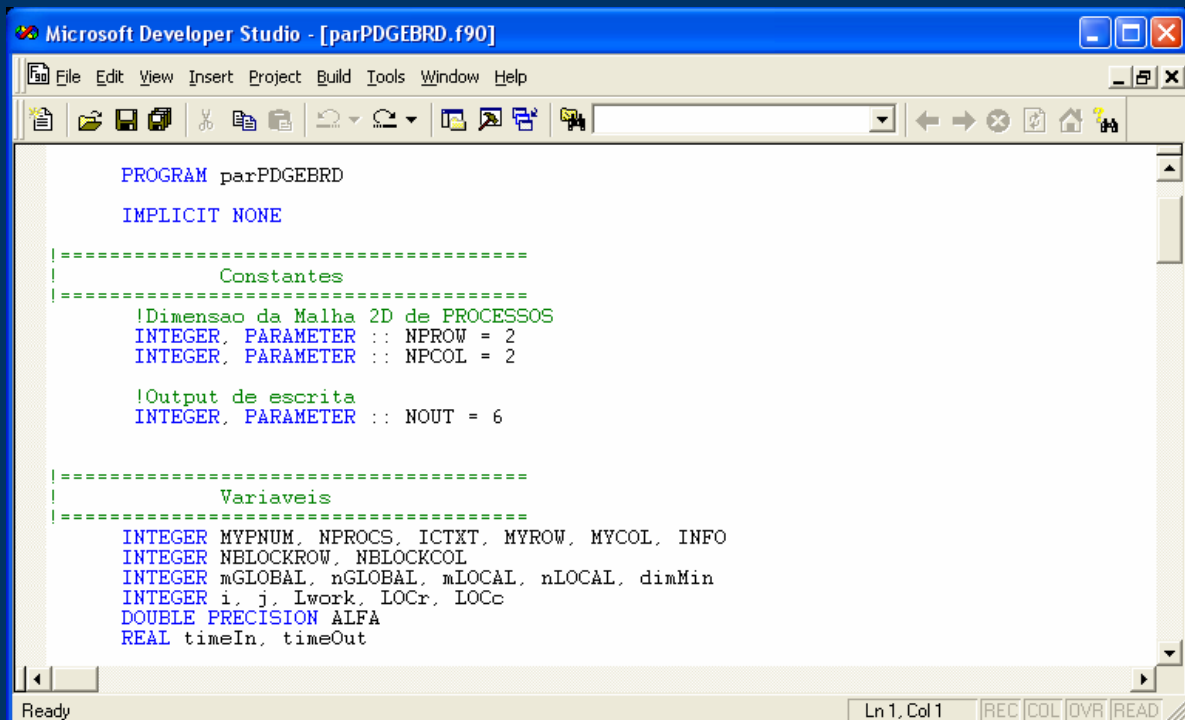


```
#!/bin/sh
#PBS -l nodes=4
#PBS -l walltime=10:00
/home/ccampos/workshopSEARCH/Exe/scalapack
```



```
#!/bin/sh
#PBS -l nodes=4
#PBS -l walltime=10:00
#
/opt/intel/mpi/1.0.2/bin64/mpirun -np 4 /home/ccampos/workshopSEARCH/Exe/scalapack
```

Fortran 90: programa principal (I)



The screenshot shows the Microsoft Developer Studio interface with the file 'parPDGEBRD.f90' open. The code is written in Fortran 90 and includes several sections: a program declaration, implicit none, constants, and variables. The constants section defines NPROW and NPCOL as 2, and NOU as 6. The variables section defines various integer and real variables, including MYPNUM, NPROCS, ICTXT, MYROW, MYCOL, INFO, NBLOCKROW, NBLOCKCOL, mGLOBAL, nGLOBAL, mLOCAL, nLOCAL, dimMin, i, j, lwork, LOCr, LOCc, ALFA, timeIn, and timeOut.

```
PROGRAM parPDGEBRD

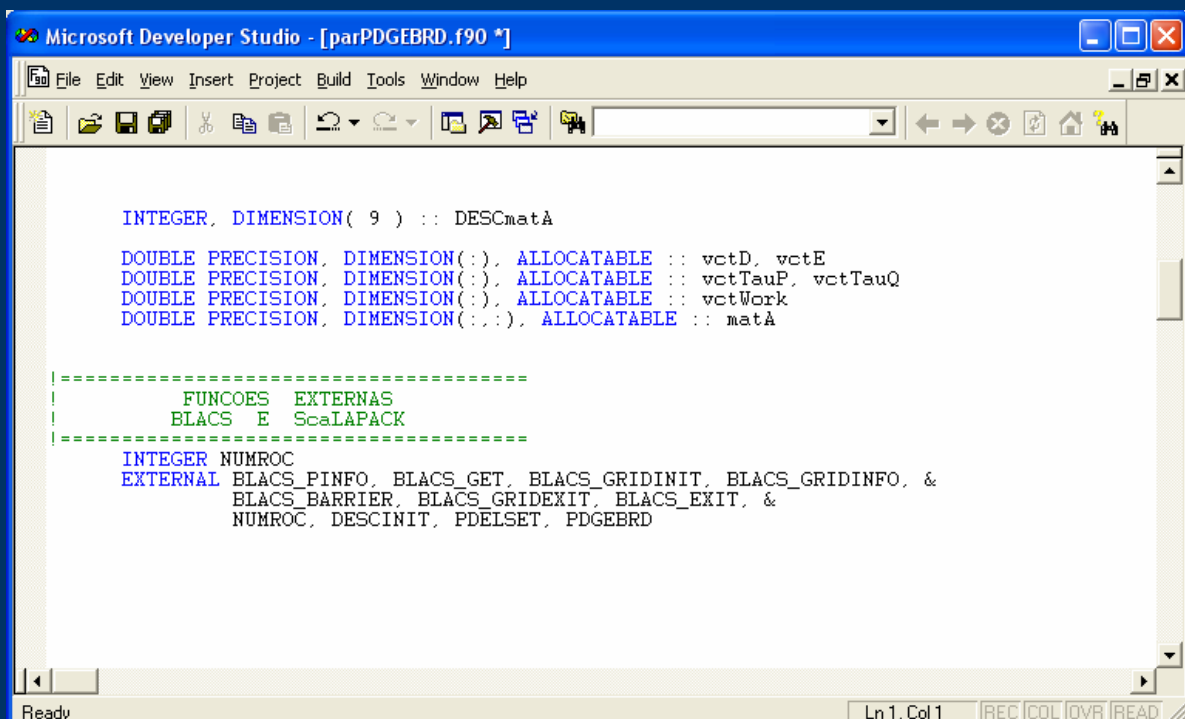
IMPLICIT NONE

=====
!Constantes
=====
!Dimensao da Malha 2D de PROCESSOS
INTEGER, PARAMETER :: NPROW = 2
INTEGER, PARAMETER :: NPCOL = 2

!Output de escrita
INTEGER, PARAMETER :: NOU = 6

=====
!Variaveis
=====
INTEGER MYPNUM, NPROCS, ICTXT, MYROW, MYCOL, INFO
INTEGER NBLOCKROW, NBLOCKCOL
INTEGER mGLOBAL, nGLOBAL, mLOCAL, nLOCAL, dimMin
INTEGER i, j, lwork, LOCr, LOCc
DOUBLE PRECISION ALFA
REAL timeIn, timeOut
```

Fortran 90: programa principal (II)



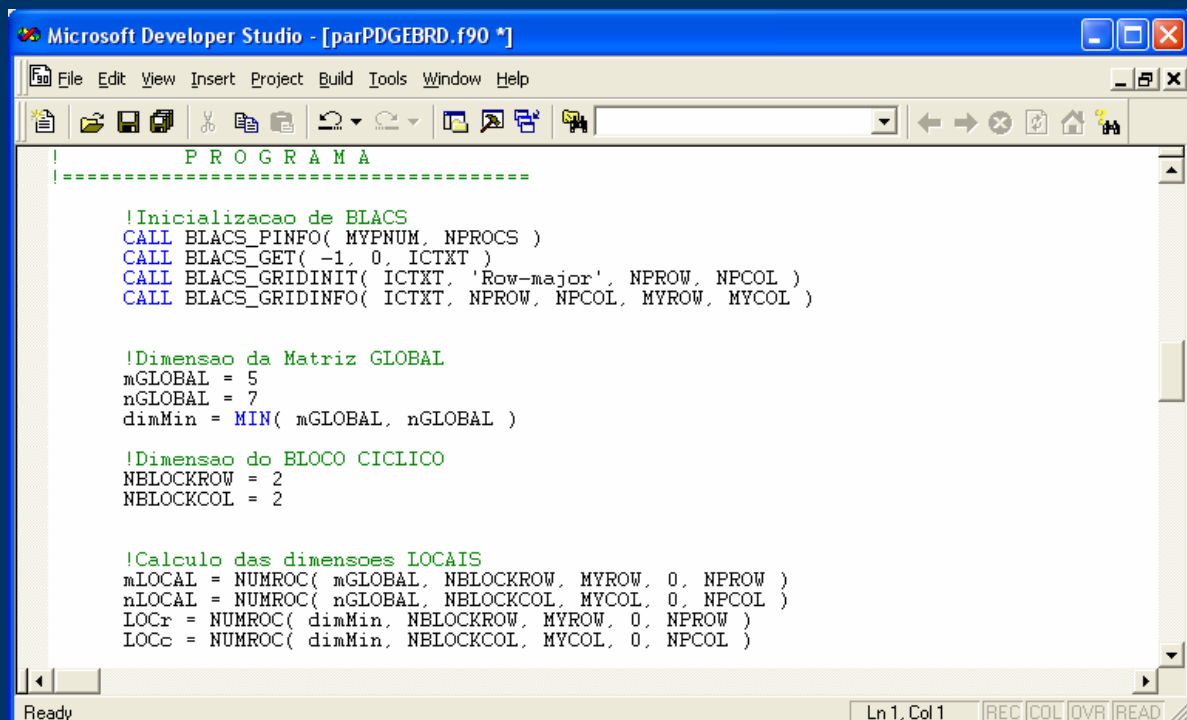
The screenshot shows the Microsoft Developer Studio interface with the file 'parPDGEBRD.f90' open. The code continues from the previous section, defining dimensions and external functions. It includes a dimension declaration for DESCmata, a list of allocatable variables (vctD, vctE, vctTauP, vctTauQ, vctWork, mata), and a list of external functions (FUNCOES EXTERNAS) including BLACS and ScaLAPACK routines.

```
INTEGER, DIMENSION( 9 ) :: DESCmata

DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctD, vctE
DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctTauP, vctTauQ
DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: vctWork
DOUBLE PRECISION, DIMENSION(:,:), ALLOCATABLE :: mata

=====
FUNCOES EXTERNAS
BLACS E ScaLAPACK
=====
INTEGER NUMROC
EXTERNAL BLACS_PINFO, BLACS_GET, BLACS_GRIDINIT, BLACS_GRIDINFO, &
          BLACS_BARRIER, BLACS_GRIDEXIT, BLACS_EXIT, &
          NUMROC, DESCINIT, PDELSET, PDGEBRD
```

Fortran 90: programa principal (III)



The screenshot shows the Microsoft Developer Studio interface with a Fortran 90 program titled 'PROGRAMA'. The code is as follows:

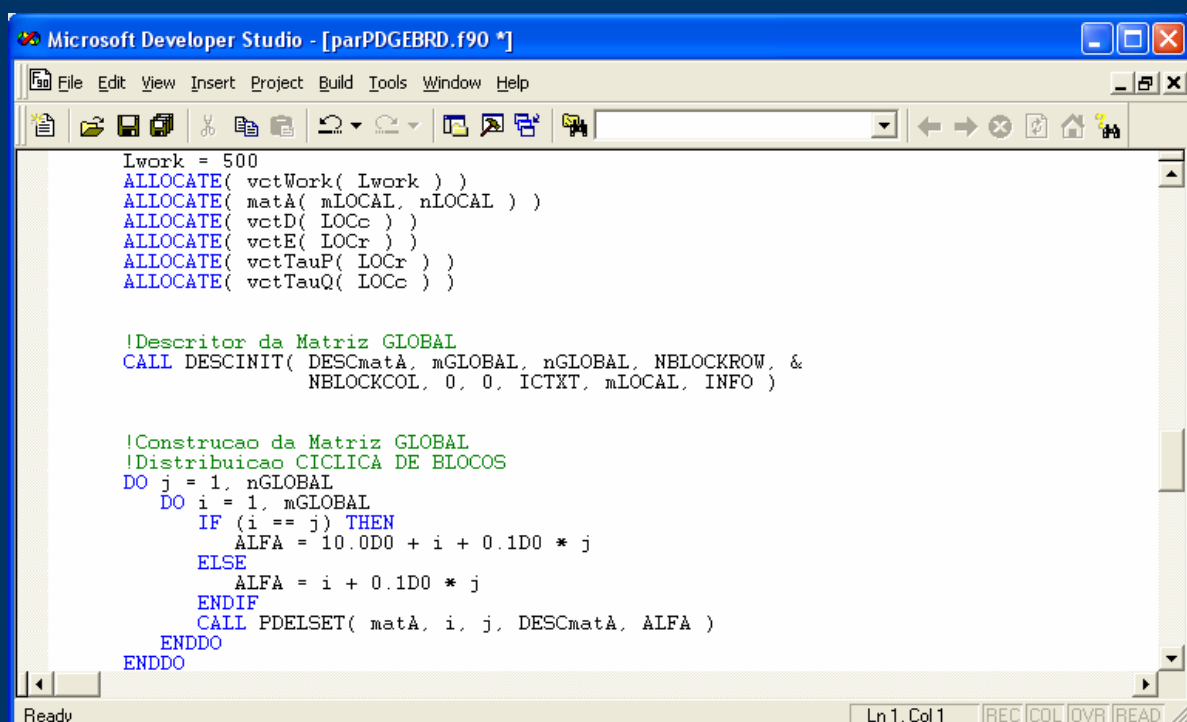
```
PROGRAMA
-----
!Inicializacao de BLACS
CALL BLACS_PINFO( MYPNUM, NPROCS )
CALL BLACS_GET( -1, 0, ICTXT )
CALL BLACS_GRIDINIT( ICTXT, 'Row-major', NPROW, NPCOL )
CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )

!Dimensao da Matriz GLOBAL
mGLOBAL = 5
nGLOBAL = 7
dimMin = MIN( mGLOBAL, nGLOBAL )

!Dimensao do BLOCO CICLICO
NBLOCKROW = 2
NBLOCKCOL = 2

!Calculo das dimensoes LOCAIS
mLOCAL = NUMROC( mGLOBAL, NBLOCKROW, MYROW, 0, NPROW )
nLOCAL = NUMROC( nGLOBAL, NBLOCKCOL, MYCOL, 0, NPCOL )
LOCr = NUMROC( dimMin, NBLOCKROW, MYROW, 0, NPROW )
LOCc = NUMROC( dimMin, NBLOCKCOL, MYCOL, 0, NPCOL )
```

Fortran 90: programa principal (IV)



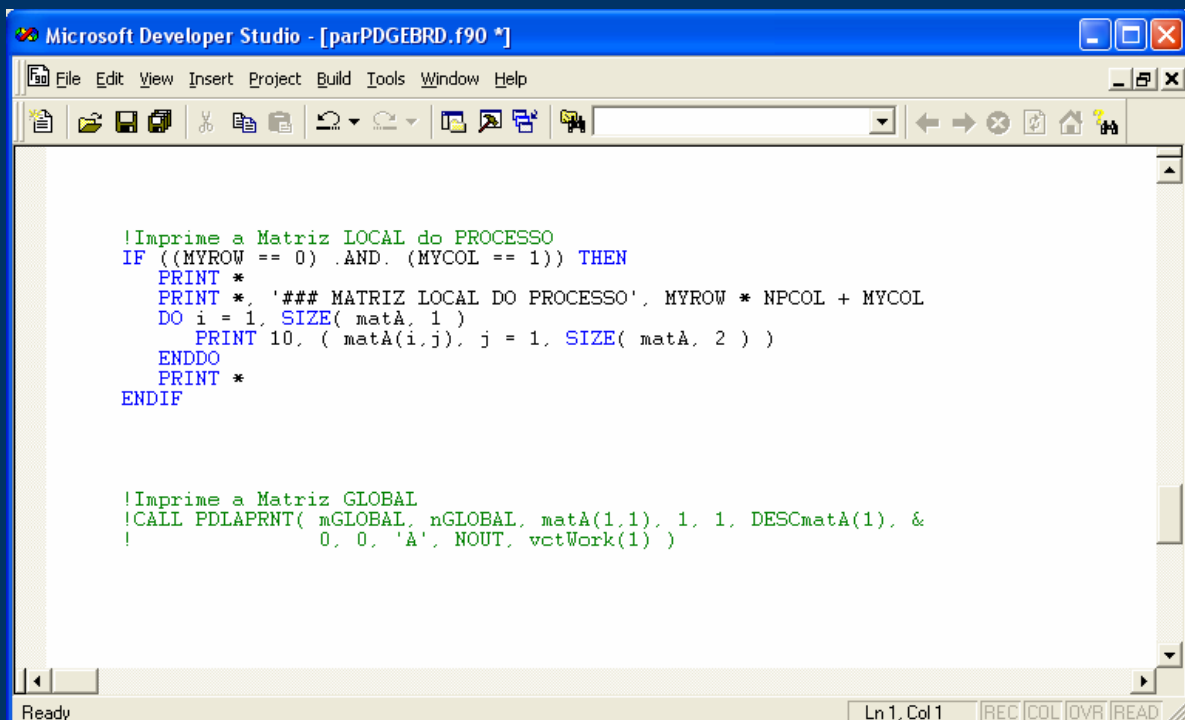
The screenshot shows the Microsoft Developer Studio interface with the same Fortran 90 program. The code continues from the previous slide:

```
Lwork = 500
ALLOCATE( vctWork( Lwork ) )
ALLOCATE( matA( mLOCAL, nLOCAL ) )
ALLOCATE( vctD( LOCc ) )
ALLOCATE( vctE( LOCr ) )
ALLOCATE( vctTauP( LOCr ) )
ALLOCATE( vctTauQ( LOCc ) )

!Descriptor da Matriz GLOBAL
CALL DESCINIT( DESCmatA, mGLOBAL, nGLOBAL, NBLOCKROW, &
              NBLOCKCOL, 0, 0, ICTXT, mLOCAL, INFO )

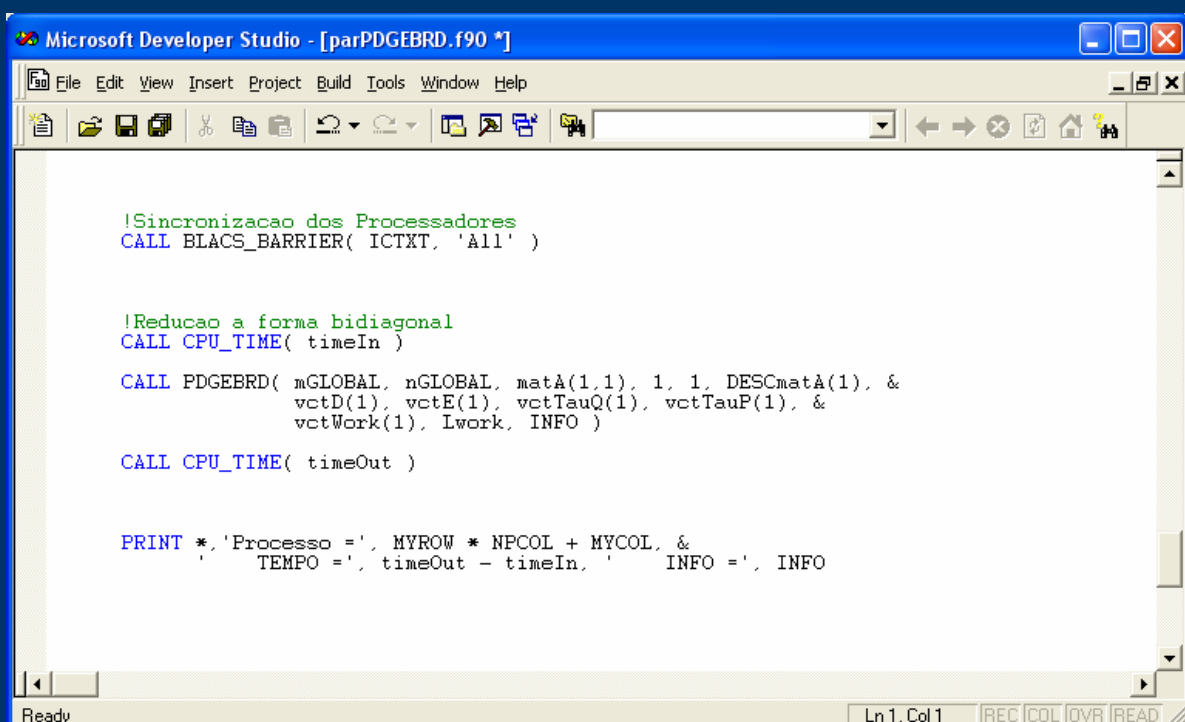
!Construcao da Matriz GLOBAL
!Distribuicao CICLICA DE BLOCOS
DO j = 1, nGLOBAL
  DO i = 1, mGLOBAL
    IF (i == j) THEN
      ALFA = 10.0D0 + i + 0.1D0 * j
    ELSE
      ALFA = i + 0.1D0 * j
    ENDIF
    CALL PDELSET( matA, i, j, DESCmatA, ALFA )
  ENDDO
ENDDO
```

Fortran 90: programa principal (v)



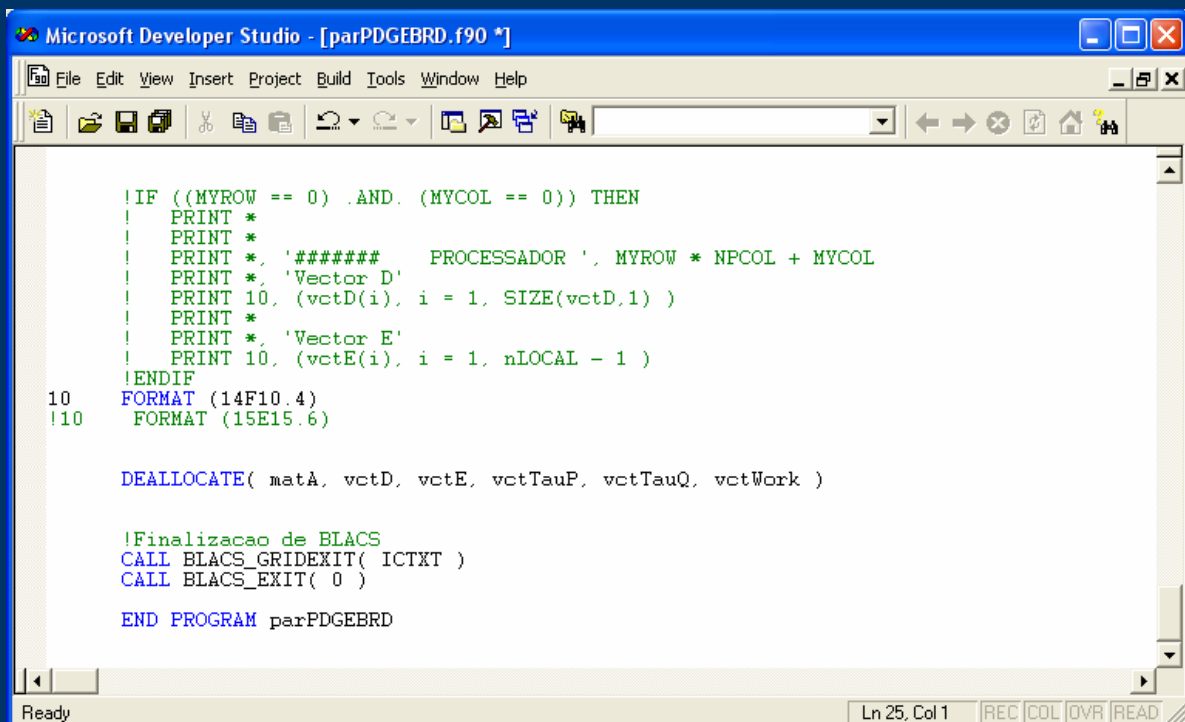
```
Microsoft Developer Studio - [parPDGEBRD.f90 *]  
File Edit View Insert Project Build Tools Window Help  
!Imprime a Matriz LOCAL do PROCESSO  
IF ((MYROW == 0) .AND. (MYCOL == 1)) THEN  
  PRINT *  
  PRINT *, '### MATRIZ LOCAL DO PROCESSO', MYROW * NPCOL + MYCOL  
  DO i = 1, SIZE( matA, 1 )  
    PRINT 10, ( matA(i,j), j = 1, SIZE( matA, 2 ) )  
  ENDDO  
  PRINT *  
ENDIF  
  
!Imprime a Matriz GLOBAL  
!CALL PDLAPRNT( mGLOBAL, nGLOBAL, matA(1,1), 1, 1, DESCmatA(1), &  
!              0, 0, 'A', NOUT, vctWork(1) )
```

Fortran 90: programa principal (vI)



```
Microsoft Developer Studio - [parPDGEBRD.f90 *]  
File Edit View Insert Project Build Tools Window Help  
  
!Sincronizacao dos Processadores  
CALL BLACS_BARRIER( ICTXT, 'All' )  
  
!Reducao a forma bidiagonal  
CALL CPU_TIME( timeIn )  
  
CALL PDGEBRD( mGLOBAL, nGLOBAL, matA(1,1), 1, 1, DESCmatA(1), &  
             vctD(1), vctE(1), vctTauQ(1), vctTauP(1), &  
             vctWork(1), Lwork, INFO )  
  
CALL CPU_TIME( timeOut )  
  
PRINT *, 'Processo =', MYROW * NPCOL + MYCOL, &  
        ' TEMPO =', timeOut - timeIn, ' INFO =', INFO
```

Fortran 90: programa principal (VII)



The screenshot shows the Microsoft Developer Studio interface with the file [parPDGEBRD.f90] open. The code is a Fortran 90 program. It starts with an IF statement checking if MYROW and MYCOL are both zero. If so, it prints several messages and vectors. It then calls DEALLOCATE on several variables. Finally, it calls BLACS_GRIDEXIT and BLACS_EXIT, and ends the program.

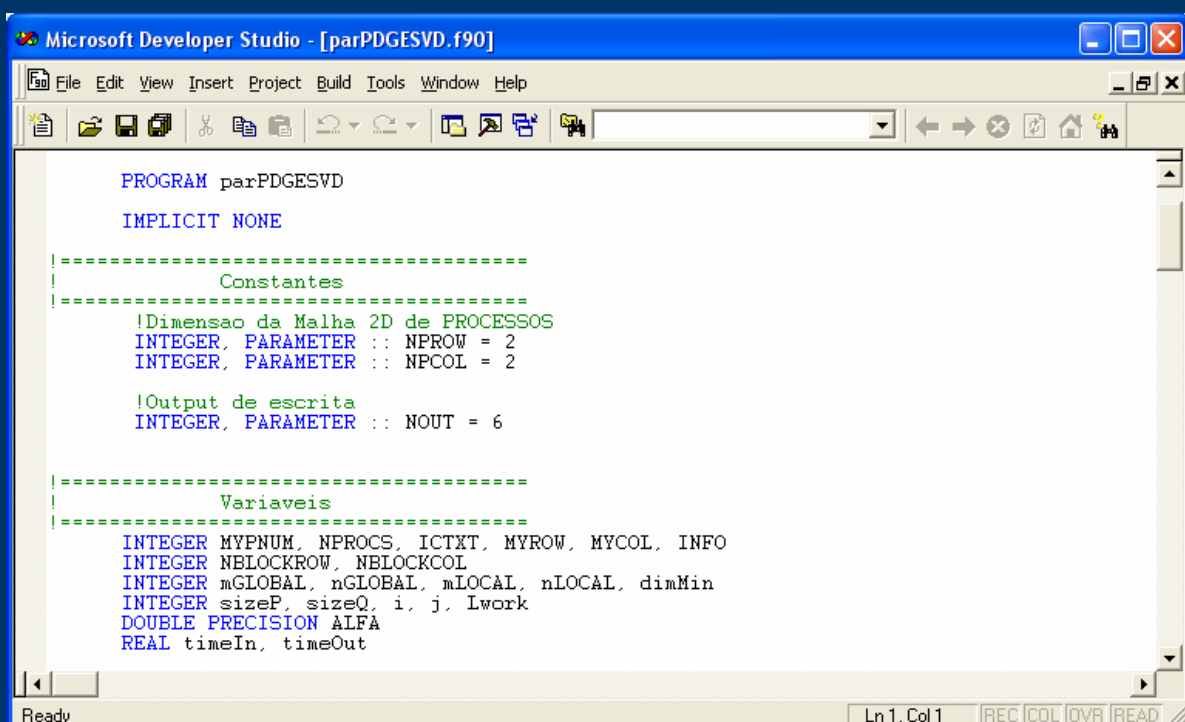
```
! IF ((MYROW == 0) .AND. (MYCOL == 0)) THEN
!   PRINT *
!   PRINT *
!   PRINT *, '#####   PROCESSADOR ', MYROW * NPCOL + MYCOL
!   PRINT *, 'Vector D'
!   PRINT 10, (vctD(i), i = 1, SIZE(vctD,1) )
!   PRINT *
!   PRINT *, 'Vector E'
!   PRINT 10, (vctE(i), i = 1, nLOCAL - 1 )
!ENDIF
10  FORMAT (14F10.4)
110  FORMAT (15E15.6)

DEALLOCATE( matA, vctD, vctE, vctTauP, vctTauQ, vctWork )

!Finalizacao de BLACS
CALL BLACS_GRIDEXIT( ICTXT )
CALL BLACS_EXIT( 0 )

END PROGRAM parPDGEBRD
```

Fortran 90: programa principal (VIII)



The screenshot shows the Microsoft Developer Studio interface with the file [parPDGESVD.f90] open. The code is a Fortran 90 program. It starts with the PROGRAM statement, followed by IMPLICIT NONE. It then defines constants for the grid size and output. Finally, it defines variables for the program, including integers for dimensions and indices, and real variables for timing.

```
PROGRAM parPDGESVD

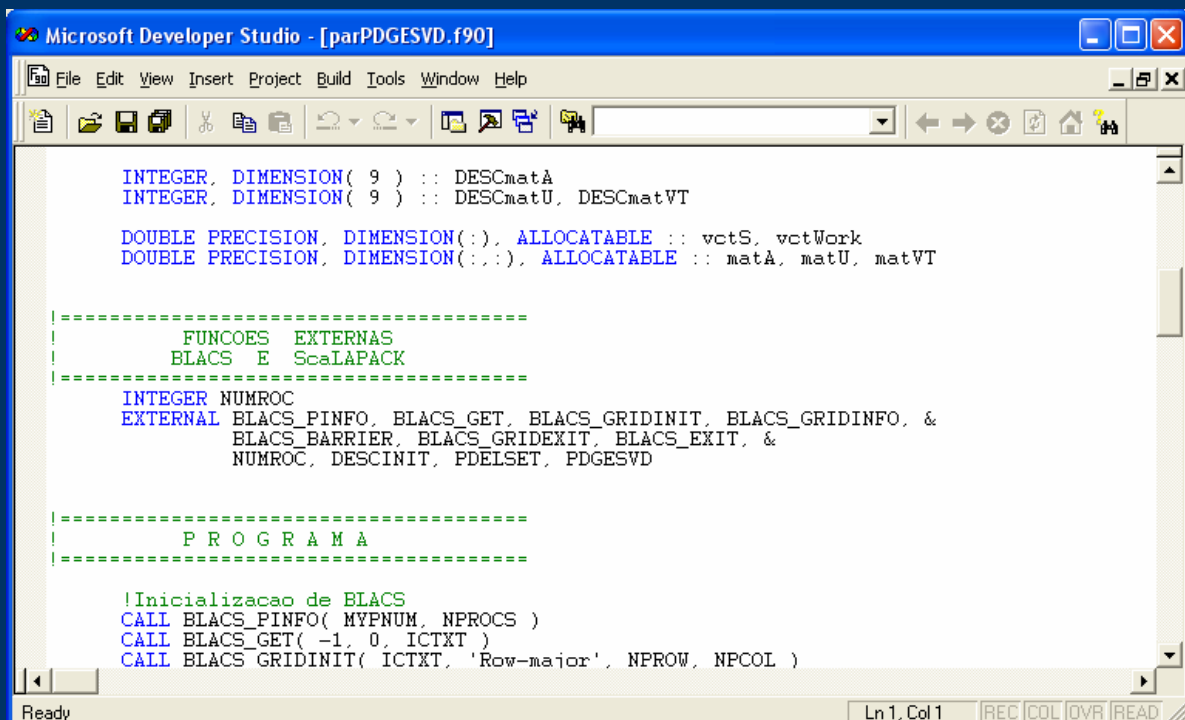
IMPLICIT NONE

!-----
!      Constantes
!-----
!Dimensao da Malha 2D de PROCESSOS
INTEGER, PARAMETER :: NPROW = 2
INTEGER, PARAMETER :: NPCOL = 2

!Output de escrita
INTEGER, PARAMETER :: NOUT = 6

!-----
!      Variaveis
!-----
INTEGER MYPNUM, NPROCS, ICTXT, MYROW, MYCOL, INFO
INTEGER NBLOCKROW, NBLOCKCOL
INTEGER mGLOBAL, nGLOBAL, mLOCAL, nLOCAL, dimMin
INTEGER sizeP, sizeQ, i, j, lwork
DOUBLE PRECISION ALFA
REAL timeIn, timeOut
```

Fortran 90: programa principal (IX)



```
Microsoft Developer Studio - [parPDGESVD.f90]

File Edit View Insert Project Build Tools Window Help

INTEGER, DIMENSION( 9 ) :: DESCmatA
INTEGER, DIMENSION( 9 ) :: DESCmatU, DESCmatVT

DOUBLE PRECISION, DIMENSION( :: ), ALLOCATABLE :: vctS, vctWork
DOUBLE PRECISION, DIMENSION( :: ), ALLOCATABLE :: matA, matU, matVT

=====
FUNCOES EXTERNAS
BLACS E ScalAPACK
=====

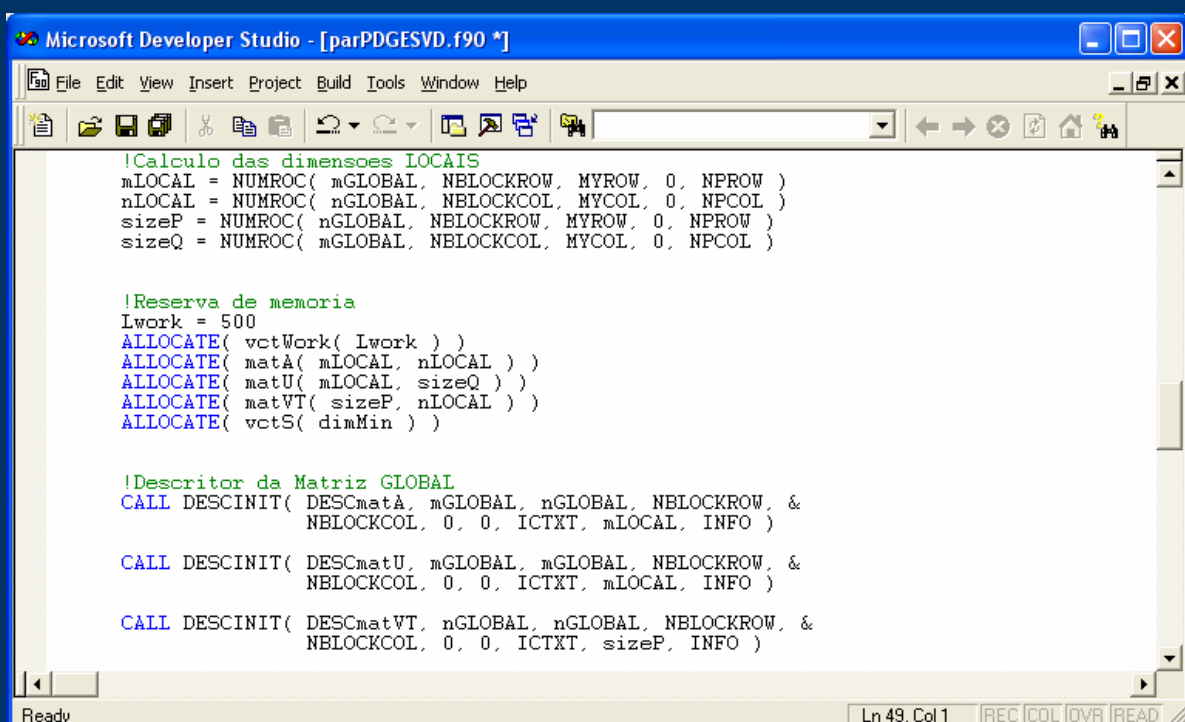
INTEGER NUMROC
EXTERNAL BLACS_PINFO, BLACS_GET, BLACS_GRIDINIT, BLACS_GRIDINFO, &
          BLACS_BARRIER, BLACS_GRIDEXIT, BLACS_EXIT, &
          NUMROC, DESCINIT, PDELSET, PDGESVD

=====
P R O G R A M A
=====

!Inicializacao de BLACS
CALL BLACS_PINFO( MYPNUM, NPROCS )
CALL BLACS_GET( -1, 0, ICTXT )
CALL BLACS_GRIDINIT( ICTXT, 'Row-major', NPROW, NPCOL )

Ready Ln 1, Col 1 REC COL OVR READ
```

Fortran 90: programa principal (X)



```
Microsoft Developer Studio - [parPDGESVD.f90 *]

File Edit View Insert Project Build Tools Window Help

!Calculo das dimensoes LOCAIS
mLOCAL = NUMROC( mGLOBAL, NBLOCKROW, MYROW, 0, NPROW )
nLOCAL = NUMROC( nGLOBAL, NBLOCKCOL, MYCOL, 0, NPCOL )
sizeP = NUMROC( nGLOBAL, NBLOCKROW, MYROW, 0, NPROW )
sizeQ = NUMROC( mGLOBAL, NBLOCKCOL, MYCOL, 0, NPCOL )

!Reserva de memoria
lwork = 500
ALLOCATE( vctWork( lwork ) )
ALLOCATE( matA( mLOCAL, nLOCAL ) )
ALLOCATE( matU( mLOCAL, sizeQ ) )
ALLOCATE( matVT( sizeP, nLOCAL ) )
ALLOCATE( vctS( dimMin ) )

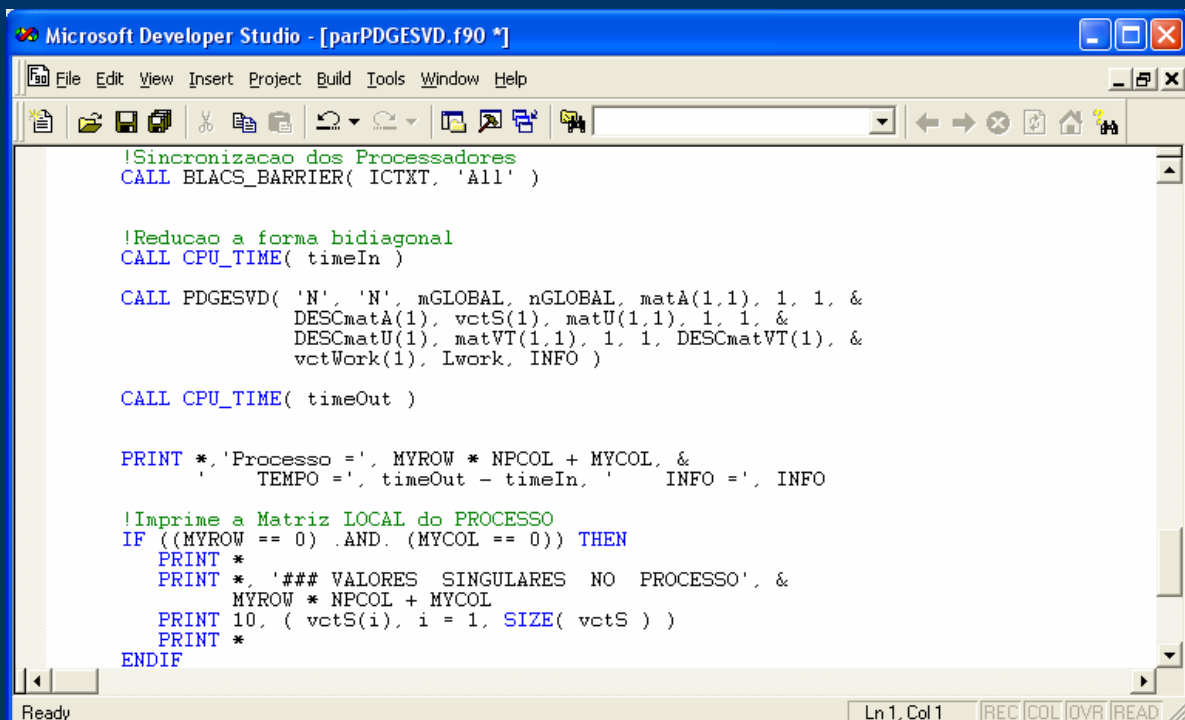
!Descritor da Matriz GLOBAL
CALL DESCINIT( DESCmatA, mGLOBAL, nGLOBAL, NBLOCKROW, &
              NBLOCKCOL, 0, 0, ICTXT, mLOCAL, INFO )

CALL DESCINIT( DESCmatU, mGLOBAL, mGLOBAL, NBLOCKROW, &
              NBLOCKCOL, 0, 0, ICTXT, mLOCAL, INFO )

CALL DESCINIT( DESCmatVT, nGLOBAL, nGLOBAL, NBLOCKROW, &
              NBLOCKCOL, 0, 0, ICTXT, sizeP, INFO )

Ready Ln 49, Col 1 REC COL OVR READ
```

Fortran 90: programa principal (XI)



```
Microsoft Developer Studio - [parPDGESVD.f90 *]  
File Edit View Insert Project Build Tools Window Help  
!Sincronizacao dos Processadores  
CALL BLACS_BARRIER( ICTXT, 'All' )  
  
!Reducao a forma bidiagonal  
CALL CPU_TIME( timeIn )  
  
CALL PDGESVD( 'N', 'N', mGLOBAL, nGLOBAL, matA(1,1), 1, 1, &  
DESCmatA(1), vctS(1), matU(1,1), 1, 1, &  
DESCmatU(1), matVT(1,1), 1, 1, DESCmatVT(1), &  
vctWork(1), Lwork, INFO )  
  
CALL CPU_TIME( timeOut )  
  
PRINT *, 'Processo =', MYROW * NPCOL + MYCOL, &  
' TEMPO =', timeOut - timeIn, ' INFO =', INFO  
  
!Imprime a Matriz LOCAL do PROCESSO  
IF ((MYROW == 0) .AND. (MYCOL == 0)) THEN  
PRINT *  
PRINT *, '### VALORES SINGULARES NO PROCESSO', &  
MYROW * NPCOL + MYCOL  
PRINT 10, ( vctS(i), i = 1, SIZE( vctS ) )  
PRINT *  
ENDIF  
Ready Ln 1, Col 1 REC COL OVR READ
```

Alguns tópicos adicionais

Exemplos com MPI
Exemplo de um algoritmo orientado a blocos

Exemplos com MPI: intel/mpi

- ▶ Exemplo em `/opt/intel/mpi/1.0.2/test`
 - Ficheiro `test.f90`
 - Exemplo típico de "Hello world"
- ▶ Como o ambiente de trabalho está configurado para usar esta versão do MPI, podemos compilar e executar o programa exemplo usando os comandos habituais
 - Compilação : `mpiifort -static_mpi -o tst test.f90`
 - Execução : `mpirun -np # ./tst`

Exemplos com MPI: mpich/intel

- ▶ Exemplo em `/opt/mpich/intel/examples`
 - Ficheiro `pi3f90.f90`
 - Cálculo de uma aproximação para o valor de π

$$\int_0^1 \frac{4}{1+x^2} dx = \pi \approx \frac{1}{n} \sum_{k=1}^n \frac{4}{1+\left(\frac{1}{n}(k-0.5)\right)^2}$$

- Dado o número de processos `np`, o utilizador deve definir o número de subintervalos (`n`) em que se subdivide o intervalo `[0, 1]`



Um algoritmo: bidiagonalização (I)

► Redução de uma matriz à forma bidiagonal superior

■ Dada a matriz $A \in \mathbb{R}^{m \times n}$ ($m > n$)

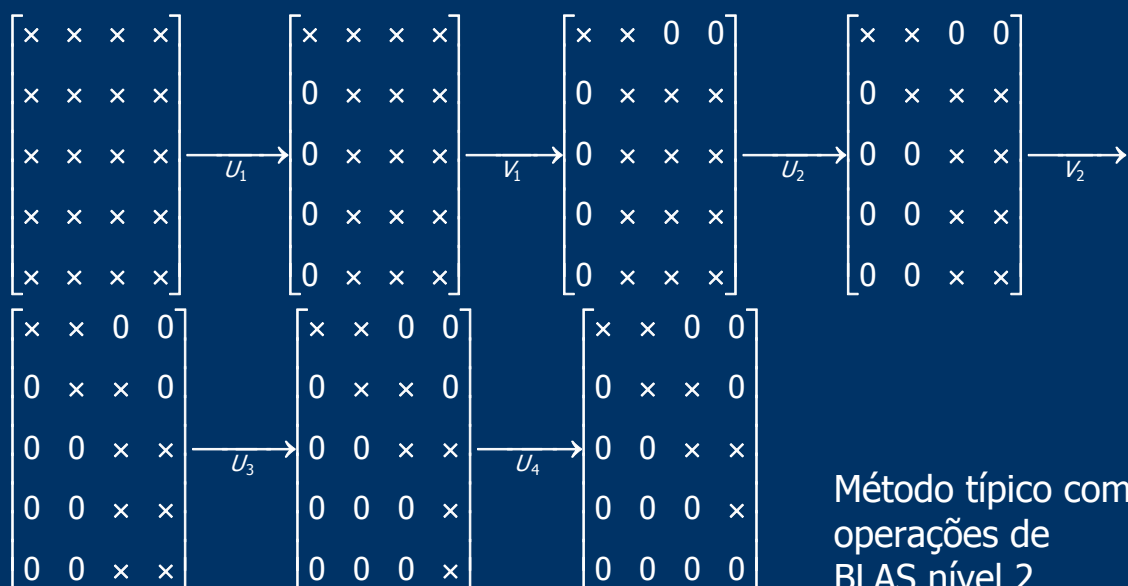
$$A = U_B \begin{bmatrix} B \\ 0 \end{bmatrix} V_B^t \quad \text{com} \quad B = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ & 0 & 0 & \\ & & 0 & \beta_n \\ & & & \alpha_n \end{bmatrix}$$

onde $U_B = U_1 U_2 \dots U_n \in \mathbb{R}^{m \times m}$ e

$V_B = V_1 V_2 \dots V_{n-2} \in \mathbb{R}^{n \times n}$ são matrizes ortogonais

Um algoritmo: bidiagonalização (II)

► Método de Golub-Kahan

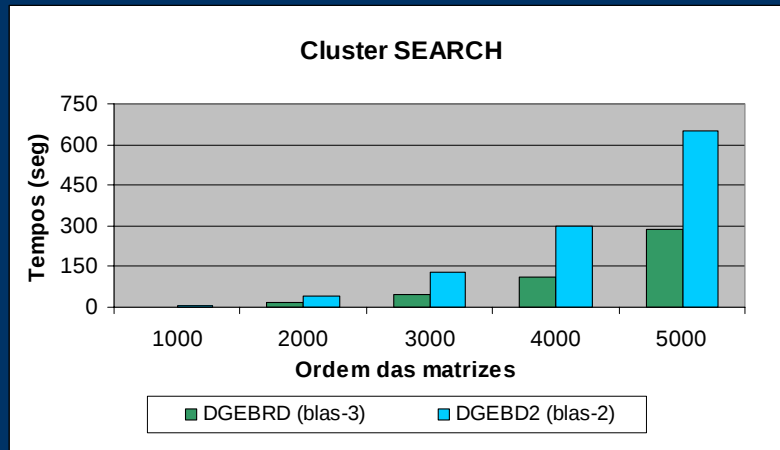


Um algoritmo: bidiagonalização (III)

► Método de Golub-Kahan

■ Versão sequencial orientada a blocos matriciais

- ◆ Realiza-se a acumulação das transformações ortogonais e só depois de reduzir o bloco matricial à forma bidiagonal superior é que se actualiza a restante parte da matriz



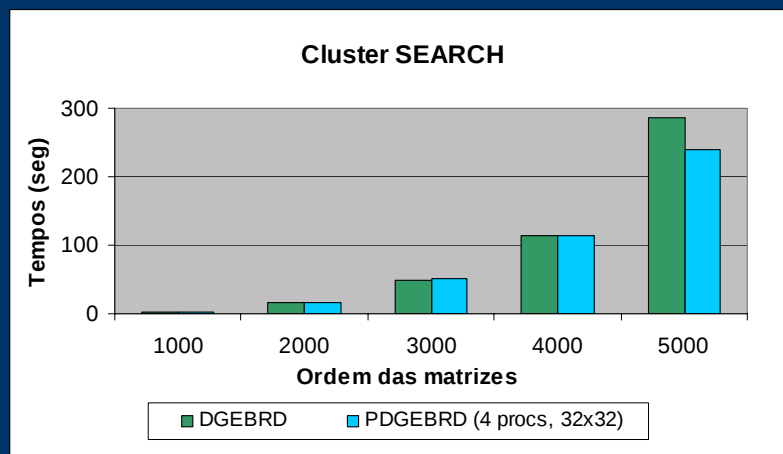
Cluster SEARCH — Centro de Matemática — Universidade do Minho

133

Um algoritmo: bidiagonalização (IV)

► Método de Golub-Kahan

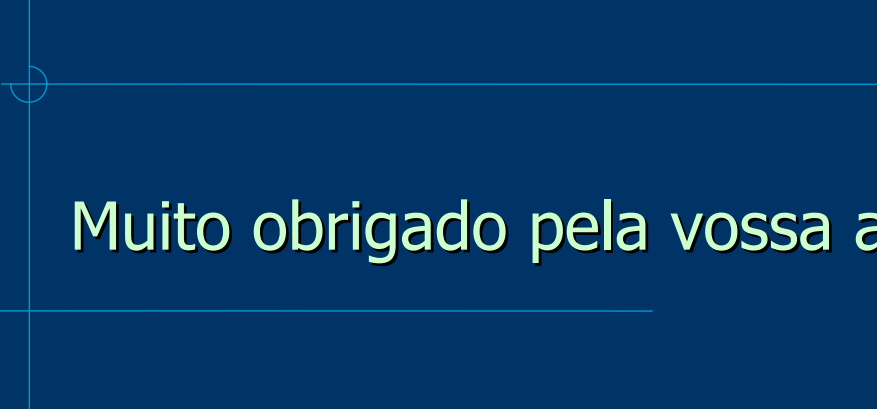
■ Versão paralela com 4 processadores



- ◆ Necessidade de encontrar a melhor performance...

Cluster SEARCH — Centro de Matemática — Universidade do Minho

134



Muito obrigado pela vossa atenção!

Perguntas?

