

OBJETIVOS DO CAPÍTULO

- Aplicar, num único programa, diversos recursos e comandos do FORTRAN vistos nos capítulos anteriores.
- Resolver sistemas de equações lineares com métodos diretos e iterativos.

Para inicializar as atividades deste capítulo, deve-se acessar o programa Fortran, no Windows, através de: **Start, Programs, Fortran PowerStation 4.0, Microsoft Developer Studio**

20.1 programa20a.f90

- 1) **Objetivo do programa:** resolver um problema de condução de calor unidimensional permanente através de simulação numérica com o método de diferenças finitas. A solução dos sistemas de equações lineares do problema é obtida com métodos diretos e iterativos.
- 2) No Fortran, seguindo o **procedimento-padrão**, **criar um projeto** com o nome **projeto_20a**
- 3) **Acessar** o site ftp://ftp.demec.ufpr.br/Disciplinas/Tm784/projeto_20a
- 4) **Clicar** com o botão do lado direito do mouse sobre o arquivo programa20a.f90
- 5) **Escolher** a opção Copiar para pasta... (Save Target As)
- 6) **Localizar** a pasta do projeto
- 7) **Clicar** no botão OK
- 8) **Repetir** os itens 3 a 7, acima, para os arquivos variaveis.f90, solvers_1D.f90, dados.f90, coeficientes.f90, resultados.f90, programa20a.ent, comandos20a.gnu e Wgnuplot.exe
- 9) No Fortran, seguindo o **procedimento-padrão**, **inserir** no projeto o programa-fonte **variaveis.f90**, mostrado na **Tabela 20.1**.
- 10) Repetir o item 9 para os programas-fonte: **solvers_1D.f90** (Tabela 20.2), **dados.f90** (Tabela 20.3), **coeficientes.f90** (Tabela 20.4), **resultados.f90** (Tabela 20.5) e **programa20a.f90** (Tabela 20.6).
- 11) **Estudar o programa-principal** considerando os comentários do item 12, abaixo.
- 12) Comentários sobre o programa:
 - (a) O programa20a.f90 é composto pelo programa-principal e cinco módulos, editados em seis programas-fonte diferentes.
 - (b) No módulo VARIAVEIS, contido no programa-fonte variaveis.f90, são definidas as variáveis globais do programa. Este módulo não contém nenhuma sub-rotina.

- (c) O módulo SOLVERS_1D, contido no programa-fonte solvers_1D.f90, é dedicado à solução de sistemas de equações lineares do tipo

$$AT = B \quad (20.1)$$

originados de problemas unidimensionais. A é a matriz de coeficientes, T é o vetor incógnita e B é o vetor dos termos independentes. A dimensão de A é $N \times N$, e de T e B , $N \times 1$, onde N é o número de incógnitas (T) do problema. Este módulo contém quatro sub-rotinas. Três delas são métodos para resolver sistemas de equações. O método de eliminação de Gauss resolve uma matriz A do tipo cheia de forma direta. O método de Gauss-Seidel resolve uma matriz A do tipo cheia mas de forma iterativa. Neste programa-exemplo, os métodos eliminação de Gauss e Gauss-Seidel estão adaptados para resolver uma matriz A do tipo tridiagonal, isto é, apenas três diagonais da matriz A têm valores não-nulos. Finalmente o método TDMA (Tri-Diagonal Matrix Algorithm) resolve uma matriz A do tipo tridiagonal de forma direta. Além disso, existe uma rotina que calcula a norma l_1 média do resíduo das equações do sistema, definido por

$$R = B - AT \quad (20.2)$$

O valor da norma é usado para comparar com uma tolerância especificada pelo usuário visando interromper o processo iterativo do método de Gauss-Seidel.

- (d) O módulo DADOS, contido no programa-fonte dados.f90, tem duas sub-rotinas. Uma é usada para ler os dados do programa, do arquivo programa20a.ent. A outra é usada para escrever os dados lidos num arquivo de saída de nome genérico.
- (e) O módulo COEFICIENTES, contido no programa-fonte coeficientes.f90, tem duas sub-rotinas. Uma é usada para definir os valores dos coeficientes da matriz A e do termo independente B do sistema de equações das N variáveis T . São definidos os valores dos coeficientes apenas em três diagonais da matriz A . Isso ocorre porque estes coeficientes são originados da discretização de uma equação diferencial que modela a condução de calor unidimensional em regime permanente; e a aproximação usada, do método de diferenças finitas, é a diferença central de 2ª ordem, CDS-2, vista no capítulo 19. O valor de B depende da definição do usuário no arquivo de dados, gerando valores nulos ou não-nulos. A outra sub-rotina é usada para escrever os valores dos coeficientes e do termo independente num arquivo de saída de nome genérico.
- (f) O módulo RESULTADOS, contido no programa-fonte resultados.f90, também tem duas sub-rotinas. A primeira é usada para: (1) chamar a sub-rotina de cálculo de coeficientes e termos independentes; (2) escrever os coeficientes; (3) resolver a Eq. (20.1) com um dos três métodos

disponíveis, de acordo com a escolha do usuário no arquivo de dados; (4) cronometrar o tempo de CPU; (5) chamar a segunda sub-rotina do módulo; e (6) escrever o tempo de CPU num arquivo de saída de nome genérico. A segunda sub-rotina deste módulo é usada para: (1) criar o arquivo T.dat; (2) escrever neste arquivo a solução analítica e numérica de T , e seu erro; (3) escrever um título no arquivo comandos20a.gnu; e (4) chamar o aplicativo Wgnuplot para fazer o gráfico de T com os comandos mostrados na Figura 20.1.

- (g) O programa-principal: (1) apresenta comentários descrevendo um resumo das características do programa; (2) obtém a data e a hora do sistema operacional; (3) chama a sub-rotina de leitura dos dados do programa; (4) cria o arquivo de saída de nome genérico; (5) escreve nele o título da simulação, a data e hora; (6) faz a alocação de memória; (7) calcula a coordenada X correspondente a T em N pontos; (8) chama a sub-rotina que escreve os dados; (9) chama a sub-rotina que resolve a Eq. (20.1); (10) com o aplicativo Notepad, mostra o conteúdo do arquivo de saída.
- (h) Os campos de coeficientes, T e gráfico são escritos com uma frequência (w) definida pelo usuário.

Tabela 20.1 Variaveis.f90

```

module variáveis

use portlib

implicit none

integer :: N          ! número total de nós

integer :: i          ! número do nó
                    ! i = 1, nó no contorno esquerdo
                    ! i = N, nó no contorno direito
                    ! 2 <= i <= N-1, nós internos

integer :: matriz     ! tipo de matriz: 1 = sem fonte; 2 = com fonte

integer :: solver     ! tipo de solver: 1 = Eliminação de Gauss (EG)
                    !                2 = Gauss-Seidel (GS)
                    !                3 = TDMA

integer :: iteracao   ! número de iterações para o GS

integer :: w          ! frequência de escrita de resultados

```

```

real*8  :: Tol      ! tolerância sobre resíduo para o GS

real*8  :: tcpu     ! tempo de CPU em segundos

integer :: ver      ! auxílio do comando System

real*8,dimension(:),allocatable :: T ! solução numérica

real*8,dimension(:),allocatable :: x ! coordenada espacial nodal

real*8,dimension(:),allocatable :: aP ! coeficiente central de u
real*8,dimension(:),allocatable :: aW ! coeficiente esquerdo de u
real*8,dimension(:),allocatable :: aE ! coeficiente direito de u

real*8,dimension(:),allocatable :: bP ! termo fonte de u

character*20 :: caso      ! nome do arquivo de saída

character*50 :: title     ! título do gráfico
character*62 :: head      ! título do gráfico + dia

character*12 :: dia       ! data da simulação
character*8  :: hora      ! horário da simulação
integer*4    :: var(8)    ! data e hora
character*20 :: vardate   ! data e hora
character*20 :: vartime   ! data e hora
character*20 :: varzone   ! data e hora
character*70 :: note_caso ! notepad + caso
character*2  :: aux1,aux2
character*4  :: aux3
character*50 :: aux

end

```

Tabela 20.2 Solvers_1D.f90

```

module solvers_1D

! objetivo: resolver sistema linear de equações algébricas
!           originado de problemas unidimensionais

use variaveis

```

```

implicit none

contains

!-----

! Método direto eliminação de Gauss

subroutine EG (N,ap,b,c,d,T)

  implicit none

  integer :: i      ! número do nó
    integer :: ii, j
    real*8  :: mji, S

  integer,intent(in) :: N ! número de nós

  real*8,dimension(:,,:),allocatable :: A ! matriz de coeficientes

  real*8,intent(in), dimension(N) :: ap ! coeficiente aP
  real*8,intent(in), dimension(N) :: b  ! coeficiente aW
  real*8,intent(in), dimension(N) :: c  ! coeficiente aE
  real*8,intent(in), dimension(N) :: d  ! termo fonte bP

  real*8,intent(out),dimension(N) :: T ! incógnita

  allocate ( A(N,N+1) )

  A = 0.0d0

  ! gera a matriz de coeficientes A com o termo independente

  do i = 1,N

    if (i > 1) A(i,i-1) = -b(i)

    A(i,i) = ap(i)

    if (i < N) A(i,i+1) = -c(i)

    A(i,N+1) = d(i)

```

```

end do

! Escalonamento

do i = 1,N-1

    do ii = i+1,N

        mji= A(ii,i) / A(i,i)

        do j = i,N+1

            A(ii,j) = A(ii,j) - mji*A(i,j)

        end do

    end do

end do

end do

!Substituicao retroativa

T(N) = A(N,N+1) / A(N,N)

do i = N-1,1,-1

    S = 0

    do j = i+1,N

        S = S + A(i,j)*T(j)

    end do

    T(i) = (A(i,N+1) - S) / A(i,i)

end do

deallocate ( A )

end subroutine EG

```

```

!-----
! método iterativo de Gauss-Seidel

subroutine GS (N,ite,tol,a,b,c,d,T)

  implicit none

  integer :: i    ! número do nó
  integer :: it   ! número da iteração
  integer :: ite  ! número de iterações

  integer,intent(in) :: N    ! número de nós

  real*8, intent(in) :: Tol ! tolerância sobre R

  real*8,intent(in), dimension(N) :: a ! coeficiente aP
  real*8,intent(in), dimension(N) :: b ! coeficiente aW
  real*8,intent(in), dimension(N) :: c ! coeficiente aE
  real*8,intent(in), dimension(N) :: d ! termo fonte bP

  real*8,intent(out),dimension(N) :: T ! incógnita

  real*8 :: R ! norma l1 média dos resíduos

  T = 0.0d0

  do it = 1, ite

    T(1) = ( c(1)*T(2) + d(1) ) / a(1)

    do i = 2, N-1

      T(i) = ( b(i)*T(i-1) + c(i)*T(i+1) + d(i) ) / a(i)

    end do

    T(N) = ( b(N)*T(N-1) + d(N) ) / a(N)

    call norma (N,a,b,c,d,T,R)

    if ( R <= Tol ) then
      write(10,1) it, Tol, R
    end if
  end do
end GS

```

```

1 format(//, 'O processo iterativo convergiu em', I8, ' iterações', &
/, 'para a tolerância de', 1pe10.2, &
/, 'A norma l1 média dos resíduos é', 1pe10.2, /)

exit
end if

end do

if ( R > Tol ) then
write(10,2) ite, Tol, R
2 format(//, 'O processo iterativo NÃO convergiu em', I8, ' iterações', &
/, 'para a tolerância de', 1pe10.2, &
/, 'A norma l1 média dos resíduos é', 1pe10.2, /)
end if

end subroutine GS

!-----

! calcula a norma l1 média do resíduo das equações

subroutine norma (N,a,b,c,d,T,R)

implicit none

integer :: i ! número do nó

integer,intent(in) :: N ! número de nós

real*8,intent(in), dimension(N) :: a ! coeficiente aP
real*8,intent(in), dimension(N) :: b ! coeficiente aW
real*8,intent(in), dimension(N) :: c ! coeficiente aE
real*8,intent(in), dimension(N) :: d ! termo fonte bP

real*8,intent(out),dimension(N) :: T ! incógnita

real*8,intent(inout) :: R ! norma l1 média dos resíduos

R = 0.0d0

R = R + dabs ( c(1)*T(2) + d(1) - T(1)*a(1) )

do i = 2, N-1

```



```

    R = R + dabs ( b(i)*T(i-1) + c(i)*T(i+1) + d(i) - T(i)*a(i) )

end do

R = R + dabs ( b(N)*T(N-1) + d(N) - T(N)*a(N) )

R = R / N

end subroutine norma

!-----

! método direto Tri-Diagonal Matrix Algorithm (TDMA)

subroutine TDMA (N,a,b,c,d,T)

  implicit none

  integer :: i ! número do nó
  real*8  :: div ! variável auxiliar

  integer,intent(in) :: N ! número de nós

  real*8,dimension(:),allocatable :: P ! coeficiente do tdma
  real*8,dimension(:),allocatable :: Q ! coeficiente do tdma

  real*8,intent(in), dimension(N) :: a ! coeficiente aP
  real*8,intent(in), dimension(N) :: b ! coeficiente aW
  real*8,intent(in), dimension(N) :: c ! coeficiente aE
  real*8,intent(in), dimension(N) :: d ! termo fonte bP

  real*8,intent(out),dimension(N) :: T ! incógnita

  allocate(P(N),Q(N))

  P(1) = c(1) / a(1)
  Q(1) = d(1) / a(1)

  do i = 2, N
    div = a(i) - b(i)*P(i-1)
    P(i) = c(i) / div
    Q(i) = (d(i) + b(i)*Q(i-1))/div
  end do

```

```

end do

T(N) = Q(N)

do i = N-1, 1, -1
  T(i) = P(i)*T(i+1) + Q(i)
end do

deallocate(P,Q)

end subroutine tdma

!-----

end module solvers_1D

```

Tabela 20.3 Dados.f90

```

module dados

! objetivo: ler e escrever os dados

use variaveis

!-----

implicit none

contains

!-----

subroutine le_dados

  ver = system('notepad programa20a.ent') ! lista dados

  open(7,file='programa20a.ent')

  read(7,*) caso
  read(7,*) N
  read(7,*) matriz
  read(7,*) solver
  read(7,*) iteracao

```

```

read(7,*) Tol
read(7,*) w
read(7,*) title

close(7)

end subroutine le_dados

!-----

subroutine mostra_dados

integer :: comp

comp = len(trim(adjustl(caso)))

write(10,1) trim(adjustl(caso)), N, matriz, solver, iteracao, Tol

1 format(/,5x,'DADOS',/,/, &
         a<comp>,' = caso',/,/, &
         i6,' = número de nós',/,/, &
         i6,' = tipo de matriz: 1 = fonte nulo; 2 = fonte não-nulo',/,/, &
         i6,' = tipo de solver: 1=El.Gauss; 2=GS; 3=TDMA',/,/, &
         i6,' = número de iterações para o GS',/,/, &
         1pe10.2,' = tolerância sobre o resíduo para o GS')

end subroutine mostra_dados

!-----

end module dados

```

Tabela 20.4 Coeficientes.f90

```

module coeficientes

! objetivo: calcular os coeficientes e termos fontes
!           das equações discretizadas

use dados

implicit none

```

```
contains
```

```
!-----
```

```
subroutine lista_coeficientes
```

```
write(10,4)
```

```
4 format(/,5x,'COEFICIENTES E FONTES',//, &  
        t6,'nó',t16,'X',t36,'oeste',t56,'central', &  
        t76,'leste',t96,'fonte',/)
```

```
do i = 1, N
```

```
if ( i==1 .or. i==n .or. mod(i,w)==0 ) &
```

```
write(10,2) i, X(i), aw(i), aP(i), ae(i), bP(i)
```

```
end do
```

```
2 format(i6,4x,5(1pe20.9))
```

```
end subroutine lista_coeficientes
```

```
!-----
```

```
subroutine coeficientes_e_fontes
```

```
! volumes internos
```

```
do i = 2, N-1
```

```
aw(i) = 1.0d0
```

```
ae(i) = 1.0d0
```

```
aP(i) = aw(i) + ae(i)
```

```
end do
```

```
select case ( matriz )
```

```
case ( 1 ) ! fonte nulo
```

```
bP = 0.0d0
```

```
case ( 2 ) ! fonte não-nulo
```

```
bP = - 2.0d0 / ( (N-1)**2 )
```

```
end select
```

```
! contorno esquerdo
```

```
aw(1) = 0.0d0
```

```
ae(1) = 0.0d0
```

```
aP(1) = 1.0d0
```

```
bP(1) = 0.0d0
```

```

! contorno direito
aw(N) = 0.0d0
ae(N) = 0.0d0
aP(N) = 1.0d0
bP(N) = 1.0d0

end subroutine coeficientes_e_fontes

!-----

end module coeficientes

```

Tabela 20.5 Resultados.f90

```

module resultados

! objetivo: calcular solução numérica e
!           apresentar gráficos dos resultados

!-----

use coeficientes
use solvers_1D

implicit none

contains

! -----

subroutine solucao_numerica

! cálculo dos coeficientes e termos fontes
  call coeficientes_e_fontes

! escrita dos coeficientes e termos fontes
  call lista_coeficientes

  tcpu = timef() ! zera cronômetro

! solução do sistema de equações

  select case ( solver )

```

```

    case ( 1 ) ! Eliminação de Gauss

        call EG (N,aP,aw,ae,bP,T)

    case ( 2 ) ! Gauss-Seidel

        call GS (N,iteracao,Tol,aP,aw,ae,bP,T)

    case ( 3 ) ! TDMA

        call tdma (N,aP,aw,ae,bP,T)

end select

tcpu = timef()

! escrita da variável primária e sua visualização
call escreve_T

write(10,1) tcpu
1 format(/, f14.3, ' = tempo de processamento (segundos)')

end subroutine solucao_numerica

!-----

subroutine escreve_T

    real*8  :: T_exato ! auxiliar
    integer :: j      ! auxiliar

    ! abertura de arquivo para gravar resultados de T (analítico e numérico)
    open(7,file='T.dat')

    write(10,1)
    1 format(/,t4,'X',t28,'T (analítico)',t52,'T (numérico)',t76,'erro',/)

    do i = 1, N

        select case ( matriz )
            case ( 1 ) ! fonte nulo
                T_exato = (i-1.0d0) / (N-1)

```

```

        case ( 2 ) ! fonte não-nulo
            T_exato = ( (i-1.0d0) / (N-1) ) ** 2
        end select

        if ( i==1 .or. i==n .or. mod(i,w)==0 ) then
            write( 7,2) X(i), T_exato, T(i), T_exato - T(i)
            write(10,2) X(i), T_exato, T(i), T_exato - T(i)
            2 format(4(1pe24.15))
        end if

    end do

    close(7)

    ! adapta arquivo de comandos para fazer gráfico
    open(7,file='comandos20a.gnu')
    do j = 1, 8
        read(7,*)
    end do
    write(7,3) head
    3 format("set title '",a62,/, "replot")
    close(7)

    ! mostra o gráfico de T
    ver = system('wgnuplot comandos20a.gnu')

end subroutine escreve_T

!-----

end module resultados

```

Tabela 20.6 Programa20a.f90

```

program programa20a

!     Difusão de calor unidimensional permanente

!     Versão original 1.0 (25 Mai 07)
!     Versão atual     1.0 (25 Mai 07)
!     última alteração = 26 Mai 07

```

```

! autor: Carlos Henrique Marchi (Curitiba, DEMEC/UFPR)

!
! MODELO MATEMÁTICO (resumo)
! Equação diferencial:  $d^2T/dx^2 = S$ 
! Condição de contorno de Dirichlet em  $x = 0$ :  $T(0) = 0$ 
! Condição de contorno de Dirichlet em  $x = 1$ :  $T(1) = 1$ 
!  $x$  = coordenada espacial (variável independente)
!  $T$  = temperatura (variável dependente)
!  $S$  = termo fonte
! Solução analítica conhecida da equação diferencial
! Solução analítica conhecida da equação discretizada

!
! MODELO NUMÉRICO (resumo)
! Incógnita (variável primária, dependente):  $T$ 
! Método numérico: diferenças finitas
! Função de interpolação: CDS (variável primária  $T$ )
! As condições de contorno são aplicadas forçando os
! coeficientes e fontes a satisfazer a C.C.
! Malha uniforme
! Solvers: Eliminação de Gauss, Gauss-Seidel e TDMA
! Precisão: dupla
! Linguagem FORTRAN 95
! Aplicativo usado: Fortran 4.0 Microsoft
! Tipo de projeto: Console
! Expressão genérica do sistema de equações discretizado:
!  $aP(i)*T(i) = aw(i)*T(i-1) + ae(i)*T(i+1) + bP(i)$ 
! onde  $i = 1, 2, \dots, N$  (número de nós)

!
! ARQUIVOS envolvidos no programa:
! programa20a.f90 = programa principal
! coeficientes.f90 = calcula coeficientes e fontes do sistema linear
! dados.f90 = lê e lista os dados do programa
! resultados.f90 = resolve equações e gera listagens dos resultados
! solvers_1D.f90 = Solvers Eliminação de Gauss, Gauss-Seidel e TDMA
! variaveis.f90 = define todas as variáveis globais do programa
! programa20a.ent = arquivo de dados do programa
! "caso" = listagem dos resultados
! T.dat = arquivo de dados para fazer gráfico
! comandos20a.gnu = arquivo de comandos para gerar gráfico
! notepad.exe = aplicativo editor dos arquivos tipo texto
! Wgnuplot.exe = aplicativo gerador de gráfico

! -----

```



```

use dados

use resultados

! -----

implicit none

integer :: comp

!-----

call date_and_time(vardate,vartime,varzone,var)

write(aux,*) var(3)
aux1 = trim(adjustl(aux))
write(aux,*) var(2)
aux2 = trim(adjustl(aux))
write(aux,*) var(1)
aux3 = trim(adjustl(aux))
dia = '('//trim(aux1)//' '//trim(aux2)//' '//aux3//')'

write(aux,*) var(5)
aux1 = trim(adjustl(aux))
write(aux,*) var(6)
aux2 = trim(adjustl(aux))
write(aux,*) var(7)
aux3 = trim(adjustl(aux))
hora = trim(aux1)//':'//trim(aux2)//':'//aux3

call le_dados

head = trim(title)//" "//trim(dia)//""

open(10,file=caso)

comp = len(trim(adjustl(title)))

write(10,18) trim(adjustl(title)), dia, hora
18 format(//,'Título = ', a<comp>, &
//,5x,'Dia = ',a12,5x,'Hora = ',a8)

```

```

! alocação de memória
allocate ( X(N), T(N) )
allocate ( aw(N), aP(N), ae(N), bP(N) )

do i = 1, N
    X(i) = (i-1.0d0) / (N-1)
end do

call mostra_dados

call solucao_numerica

close (10)

note_caso = 'notepad '//caso
ver = system(note_caso) ! lista arquivo de resultados

! -----
End

```

```

comandos20a - Notepad
File Edit Format View Help
set data style linespoints
set grid
set time
set key right bottom
plot 'T.dat' using 1:2 title 'analitico' with points
replot 'T.dat' using 1:3 title 'numérico'
set xlabel 'X'
set ylabel 'T'
set title 'Programa20a, fonte não-nulo, EG (27/5/2007)'
replot

```

Figura 20.1 Arquivo de comandos para o aplicativo Wgnuplot do programa20a.f90.

- 13) Executar **Build, Compile** para compilar o programa-fonte **variaveis.f90**. Em seguida, executar **Build, Compile** para compilar os demais programas-fonte na seguinte ordem: **solvers_1D.f90**, **dados.f90**, **coeficientes.f90**, **resultados.f90** e **programa20a.f90**.
- 14) Gerar o programa-executável fazendo **Build, Build**.
- 15) Executar o programa através de **Build, Execute**. Usar os dados mostrados na Figura 20.2.
- 16) **Analisar os resultados** mostrados nas Figuras 20.3 e 20.4. O erro apresentado pela solução na Figura 20.4 deve-se aos erros de arredondamento.
- 17) **Executar** novamente o programa usando os mesmos dados da Figura 20.2, exceto, **matriz = 1** e **analisar** os novos resultados.

- 18) **Executar** novamente o programa usando os mesmos dados da Figura 20.2, exceto, **matriz = 1** e **solver = 3**, e **analisar** os novos resultados. Notar a redução significativa no tempo de CPU ao se usar o método TDMA em relação ao método de Eliminação de Gauss.
- 19) **Executar** novamente o programa usando os mesmos dados da Figura 20.2, exceto, **matriz = 2** e **solver = 3**, e **analisar** os novos resultados.

```

programa20a - Notepad
File Edit Format View Help
'saida20a.txt' ... caso = nome do arquivo de saída
501 ..... N      = número de nós a calcular a temperatura (T)
2 ..... matriz = tipo de matriz: 1 = fonte nulo; 2 = fonte não-nulo
1 ..... solver = tipo de solver: ver lista abaixo
0 ..... iteracao = número de iterações para o Gauss-Seidel
1.0d-10 ... Tol = tolerância sobre o resíduo para o Gauss-Seidel
50 ..... w = frequência de escrita de resultados
'Programa20a, fonte não-nulo, EG'          ! título: máximo 50 caracteres

012345  1      2      3      4      5

Tipos de solver:  1 = Eliminação de Gauss
                  2 = Gauss-Seidel
                  3 = TDMA

Arquivo de dados do PROGRAMA20A
|

```

Figura 20.2 Arquivo de dados do programa20a.f90.

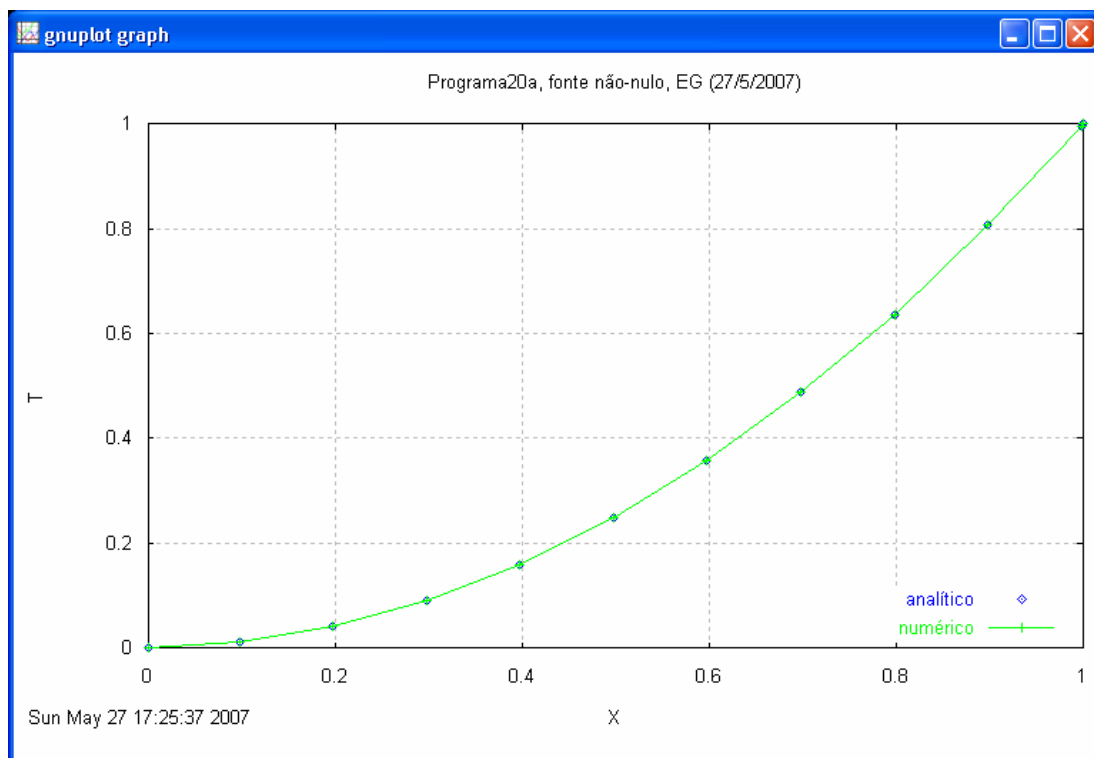
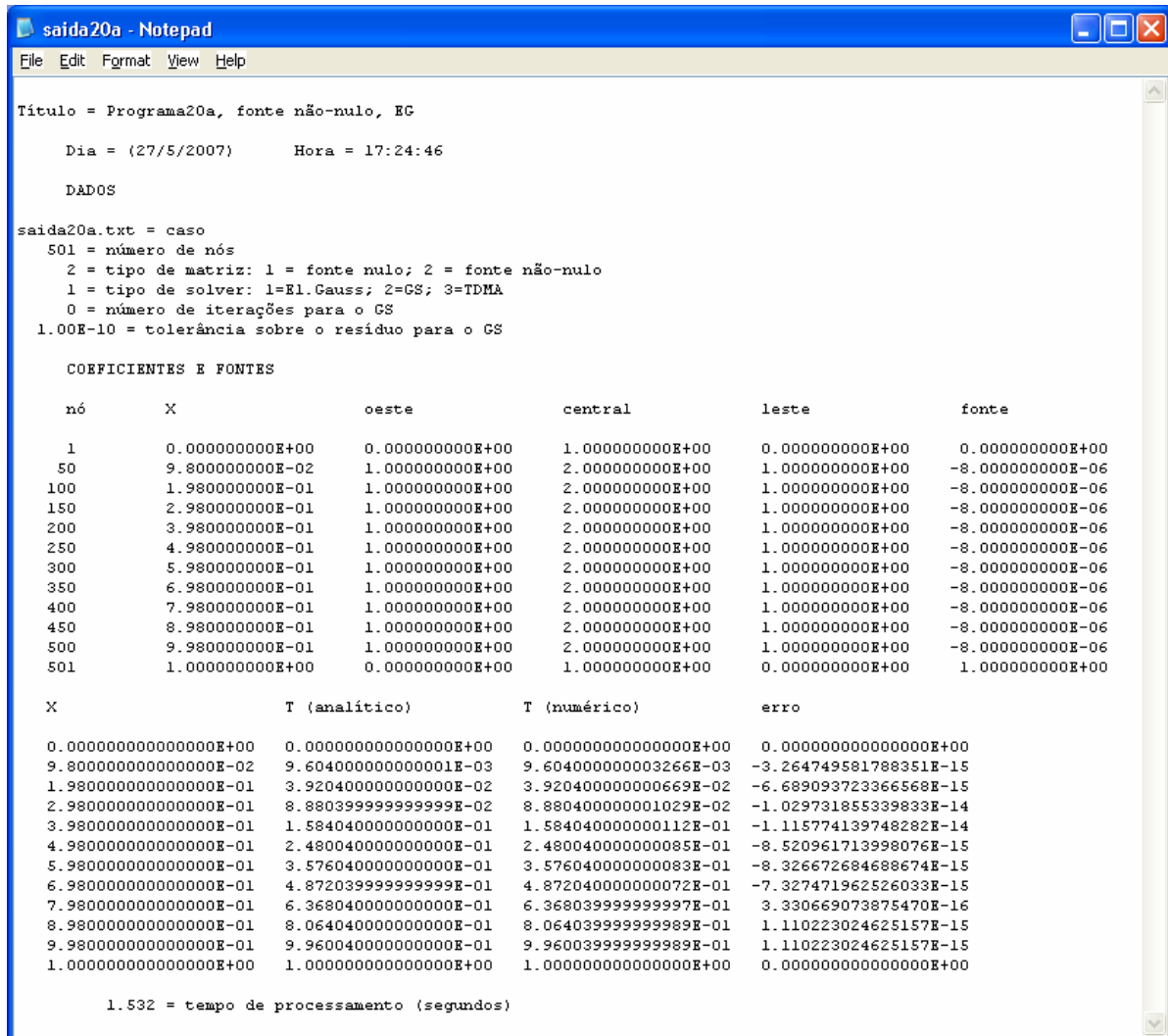


Figura 20.3. Gráfico com resultados do programa20a.f90 para os dados da Fig. 20.2.

20.2 EXERCÍCIOS

Exercício 20.1

Executar novamente o programa20a.f90 usando os mesmos dados da Figura 20.2, exceto, matriz = 1 e solver = 2. Por tentativa e erro, usar um valor para a variável “iteracao” que seja suficiente para satisfazer a tolerância (Tol) estipulada na Figura 20.2. Comparar o tempo de CPU resultante neste caso, para o método de Gauss-Seidel, àquele do método TDMA (item 18) e do método de Eliminação de Gauss (item 17).



```
saida20a - Notepad
File Edit Format View Help

Titulo = Programa20a, fonte não-nulo, EG

Dia = (27/5/2007) Hora = 17:24:46

DADOS

saida20a.txt = caso
501 = número de nós
2 = tipo de matriz: 1 = fonte nulo; 2 = fonte não-nulo
1 = tipo de solver: 1=El.Gauss; 2=GS; 3=TDMA
0 = número de iterações para o GS
1.00E-10 = tolerância sobre o residuo para o GS

COEFICIENTES E FONTES

nó      X          oeste          central          leste          fonte
1       0.00000000E+00  0.00000000E+00  1.00000000E+00  0.00000000E+00  0.00000000E+00
50      9.80000000E-02  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
100     1.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
150     2.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
200     3.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
250     4.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
300     5.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
350     6.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
400     7.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
450     8.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
500     9.98000000E-01  1.00000000E+00  2.00000000E+00  1.00000000E+00  -8.00000000E-06
501     1.00000000E+00  0.00000000E+00  1.00000000E+00  0.00000000E+00  1.00000000E+00

X          T (analítico)      T (numérico)      erro
0.00000000E+00  0.00000000E+00    0.00000000E+00    0.00000000E+00
9.80000000E-02  9.60400000E-03    9.60400000E-03    -3.264749581788351E-15
1.98000000E-01  3.92040000E-02    3.92040000E-02    -6.689093723366568E-15
2.98000000E-01  8.88039999999999E-02  8.880400000001029E-02  -1.029731855339833E-14
3.98000000E-01  1.58404000000000E-01  1.58404000000112E-01  -1.115774139748282E-14
4.98000000E-01  2.48004000000000E-01  2.48004000000085E-01  -8.520961713998076E-15
5.98000000E-01  3.57604000000000E-01  3.57604000000083E-01  -8.326672684688674E-15
6.98000000E-01  4.87203999999999E-01  4.87204000000072E-01  -7.327471962526033E-15
7.98000000E-01  6.36804000000000E-01  6.36803999999997E-01  3.330669073875470E-16
8.98000000E-01  8.06404000000000E-01  8.06403999999989E-01  1.110223024625157E-15
9.98000000E-01  9.96004000000000E-01  9.96003999999989E-01  1.110223024625157E-15
1.00000000E+00  1.00000000E+00    1.00000000E+00    0.00000000E+00

1.532 = tempo de processamento (segundos)
```

Figura 20.4 Arquivo de resultados do programa20a.f90 para os dados da Fig. 20.2.

Exercício 20.2

Executar novamente o programa20a.f90 usando os mesmos dados da Figura 20.2, exceto, matriz = 2 e solver = 2. Por tentativa e erro, usar um valor para a variável “iteracao” que seja suficiente para satisfazer a tolerância (Tol) estipulada na Figura 20.2. Comparar o tempo de CPU resultante neste caso, para o método de Gauss-Seidel, àquele do método TDMA (item 19) e do método de Eliminação de Gauss (item 16).