Capítulo 20. **Exemplos: SISTEMAS de EQUAÇÕES**

11 Nov 2024

OBJETIVOS DO CAPÍTULO

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

20.1 projeto programa20

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 do sistema de equações lineares do problema é obtida com métodos diretos e iterativo.
2) No Fortran, **criar um projeto** com o nome **projeto20**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **variaveis.f90** e **copiar** da **Tabela 20.1**.
4) No Fortran, **criar e inserir** no projeto o programa-fonte **Solvers_1D.f90** e **copiar** da **Tabela 20.2**.
5) No Fortran, **criar e inserir** no projeto o programa-fonte **dados.f90** e **copiar** da **Tabela 20.3**.
6) No Fortran, **criar e inserir** no projeto o programa-fonte **coeficientes.f90** e **copiar** da **Tabela 20.4**.
7) No Fortran, **criar e inserir** no projeto o programa-fonte **resultados.f90** e **copiar** da **Tabela 20.5**.
8) No Fortran, **criar e inserir** no projeto o programa-fonte **programa20.f90** e **copiar** da **Tabela 20.6**.
9) Comentários sobre o programa:
   (a) O projeto programa20 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

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

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 NxN, e de T e B, Nx1, 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

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

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 programa20.ent mostrado na Figura 20.1. A outra é usada para escrever os dados lidos no 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 no 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 comandos20.gnu; e (4) chamar o aplicativo Wgnuplot para fazer o gráfico de *T* com os comandos mostrados na Figura 20.2.

(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 freqüência (*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 = fonte nulo; 2 = fonte não nulo

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

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

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

  !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
         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

    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 programa20.ent') ! lista dados

    open(7,file='programa20.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

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

    1 format(/,5x,'DADOS',//,  &
                a,' = 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

    ! pontos 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 CPU (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='comandos20.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 comandos20.gnu')

  end subroutine escreve_T

!-----------------------------------------------

end module resultados
```

Tabela 20.6 Programa20.f90

```
program programa20a

!     Difusão de calor unidimensional permanente

!     Versão original 1.0 (25 Mai 07)
!     Versão atual    1.0 (25 Mai 07)
!     última alteração =   10 Nov 24

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

!     MODELO MATEMÁTICO (resumo)
!       Equação diferencial: d2T/dx2 = 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, ... N (número de nós)

!     ARQUIVOS envolvidos no programa:
!       programa20.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.f9    = Solvers Eliminação de Gauss, Gauss-Seidel e TDMA
!       variaveis.f90    = define todas as variáveis globais do programa
!       programa20.ent   = arquivo de dados do programa
!       "caso"           = listagem dos resultados
!       T.dat            = arquivo de dados para fazer gráfico
!       comandos20.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

!-----------------------------------------------

  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)

  write(10,18) trim(adjustl(title)), dia, hora
  18 format(/,'Título = ', a, &
            //,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
```

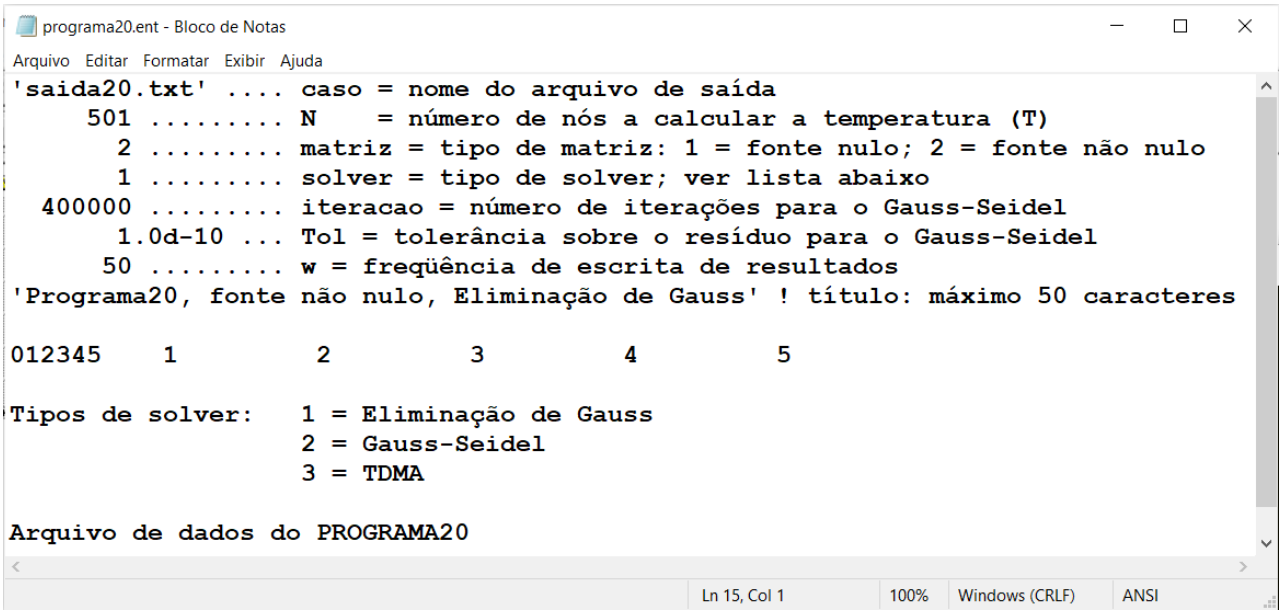


Figura 20.1 Arquivo programa20.ent do projeto programa20.

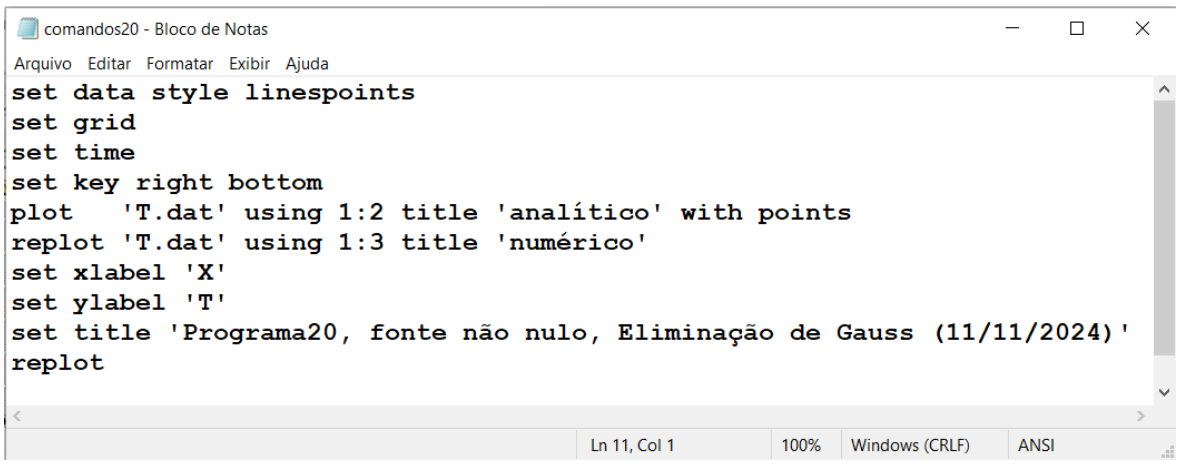


Figura 20.2 Arquivo comandos20.gnu para o aplicativo Wgnuplot do projeto programa20.

10) **Estudar o programa-principal** considerando os comentários do item 9, acima.
11) 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 **programa20.f90**.
12) Gerar o programa-executável fazendo **Build, Build**.
13) Executar o programa através de **Build, Execute**. **Usar os dados mostrados na Figura 20.1**.

14) **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. Notar que o tempo de CPU foi de 0.422 segundo. O computador usado nos cálculos tem processador i3 de 2.30 GHz e sistema operacional Windows 10.

15) **Executar** novamente o programa usando os mesmos dados da Figura 20.1 mas com **solver = 2** (Gauss-Seidel) e **analisar** os novos resultados. Observar que o tempo de CPU foi de 4.968 segundos; isso representa um aumento significativo do tempo de CPU em relação ao método de Eliminação de Gauss.

16) **Executar** novamente o programa usando os mesmos dados da Figura 20.1 mas com **solver = 3** (TDMA) e **analisar** os novos resultados. Ver que o tempo de CPU foi de 0.000 segundo, ou seja, a execução foi tão rápida que não conseguiu ser medida pelo comando TIMEF. Portanto, houve uma redução significativa no tempo de CPU ao se usar o método TDMA em relação ao método de Eliminação de Gauss (solver = 1).

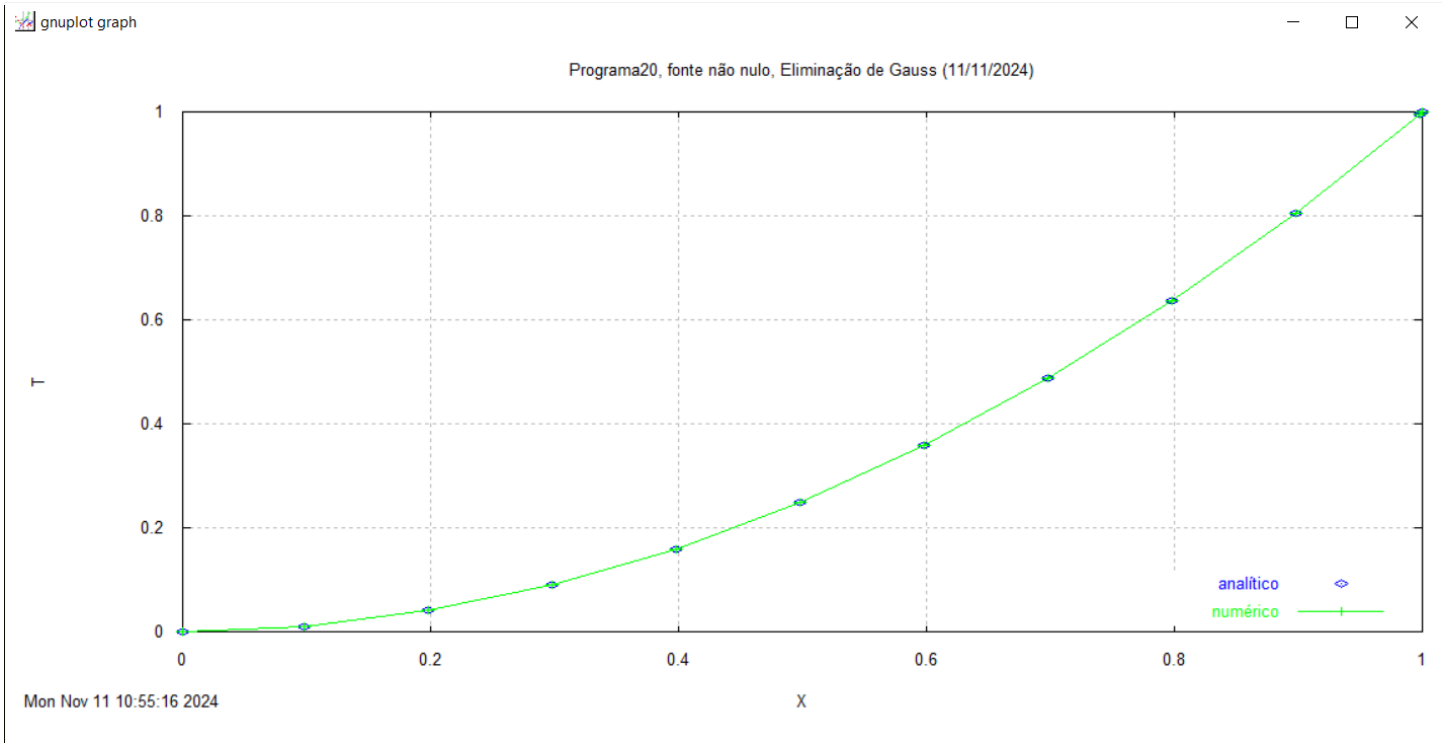


Figura 20.3. Gráfico com resultados do programa20 para os dados da Figura 20.1.

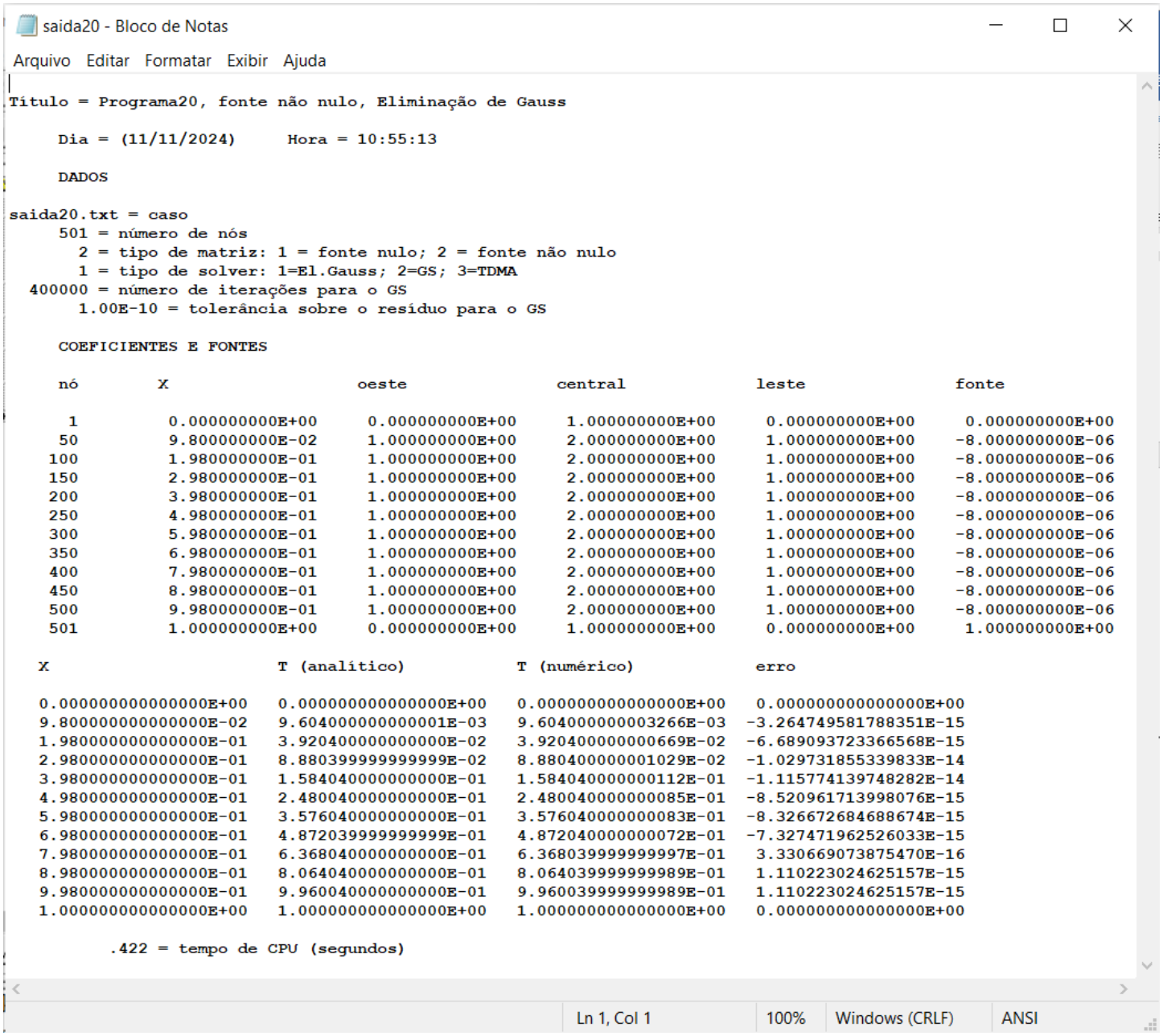
20.2 EXERCÍCIOS

Exercício 20.1

Executar novamente o programa20 usando os mesmos dados da Figura 20.1 mas com matriz = 1 e solver = 2 (método de Gauss-Seidel). Anotar o tempo de CPU.

Exercício 20.2

Executar novamente o programa20 usando os mesmos dados da Figura 20.1 mas com matriz = 1 e solver = 1 (método de Eliminação de Gauss). Anotar o tempo de CPU.



saida20 - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Título = Programa20, fonte não nulo, Eliminação de Gauss

     Dia = (11/11/2024)     Hora = 10:55:13

     DADOS

saida20.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
  400000 = número de iterações para o GS
       1.00E-10 = tolerância sobre o resíduo para o GS

     COEFICIENTES E FONTES

     nó        X                   oeste               central             leste               fonte

      1         0.000000000E+00     0.000000000E+00     1.000000000E+00     0.000000000E+00     0.000000000E+00
     50         9.800000000E-02     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    100         1.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    150         2.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    200         3.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    250         4.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    300         5.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    350         6.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    400         7.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    450         8.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    500         9.980000000E-01     1.000000000E+00     2.000000000E+00     1.000000000E+00    -8.000000000E-06
    501         1.000000000E+00     0.000000000E+00     1.000000000E+00     0.000000000E+00     1.000000000E+00

   X                      T (analítico)           T (numérico)            erro

   0.000000000000000E+00  0.000000000000000E+00   0.000000000000000E+00   0.000000000000000E+00
   9.800000000000000E-02  9.604000000000001E-03   9.604000000003266E-03  -3.264749581788351E-15
   1.980000000000000E-01  3.920400000000000E-02   3.920400000000669E-02  -6.689093723366568E-15
   2.980000000000000E-01  8.880399999999999E-02   8.880400000001029E-02  -1.029731855339833E-14
   3.980000000000000E-01  1.584040000000000E-01   1.584040000000112E-01  -1.115774139748282E-14
   4.980000000000000E-01  2.480040000000000E-01   2.480040000000085E-01  -8.520961713998076E-15
   5.980000000000000E-01  3.576040000000000E-01   3.576040000000083E-01  -8.326672684688674E-15
   6.980000000000000E-01  4.872039999999999E-01   4.872040000000072E-01  -7.327471962526033E-15
   7.980000000000000E-01  6.368040000000000E-01   6.368039999999997E-01   3.330669073875470E-16
   8.980000000000000E-01  8.064040000000000E-01   8.064039999999989E-01   1.110223024625157E-15
   9.980000000000000E-01  9.960040000000000E-01   9.960039999999989E-01   1.110223024625157E-15
   1.000000000000000E+00  1.000000000000000E+00   1.000000000000000E+00   0.000000000000000E+00

          .422 = tempo de CPU (segundos)
```

Ln 1, Col 1 | 100% | Windows (CRLF) | ANSI

Figura 20.4 Arquivo de resultados do programa20 para os dados da Figura 20.1.

Exercício 20.3

Executar novamente o programa20 usando os mesmos dados da Figura 20.1 mas com matriz = 1 e solver = 3 (método TDMA). Anotar o tempo de CPU.

Exercício 20.4

Fazer uma tabela para comparar o tempo de CPU mostrado nos itens 14 a 16 da seção 20.1, acima, com aqueles obtidos nos Exercícios 20.1 a 20.3. Observar os efeitos da matriz (1 ou 2) e do solver (1, 2 ou 3). Deve-se perceber que há um grande efeito do tipo de solver usado para resolver o mesmo problema.

Exercício 20.5

Repetir os Exercícios 20.1 a 20.4 para N = 1001. Quando usar solver = 2, utilizar iteracao = 1400000.

Exercício 20.6

Executar novamente o programa20 usando os mesmos dados da Figura 20.1 mas com N = 1000001, matriz = 1, solver = 3 (método TDMA) e w = 10000. Verificar o tempo de CPU.