

3º Trabalho Computacional

**Solução Numérica do Escoamento Unidimensional: Equação de Convecção-Difusão**

Aluno: Romulo de Aguiar Beninca

link do github : <https://rbeninca.github.io/cfd1/>

link códigos : <https://github.com/rbeninca/cfd1>

## 1. Definição do Problema e Organização

Este relatório foi elaborado em HTML, utilizando dados gerados por um código Fortran e gráficos produzidos com Gnuplot. O objetivo principal foi desenvolver um código computacional para resolver a equação de convecção-difusão linear unidimensional em regime permanente, utilizando o Método de Volumes Finitos (MVF). O problema físico consiste em um escoamento unidimensional com velocidade prescrita nos contornos. Foram realizadas poucas modificações no código original fornecido pelo professor, focando principalmente na adaptação para a geração dos dados e resultados apresentados.

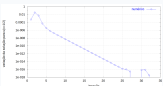
O relatório apresenta a metodologia, os parâmetros de execução interativos, os resultados gráficos (convergência e perfil de velocidade), tabelas comparativas e a análise de erro.

Bibliotecas JavaScript Utilizadas:

- **Tailwind CSS (via CDN):** Utilizado para estilização rápida e responsiva do layout do relatório.
- **MathJax:** Essencial para a renderização de equações matemáticas em formato LaTeX diretamente no navegador.
- **Prism.js:** Usado para o destaque de sintaxe do código-fonte Fortran e JavaScript, proporcionando uma leitura mais clara e organizada.

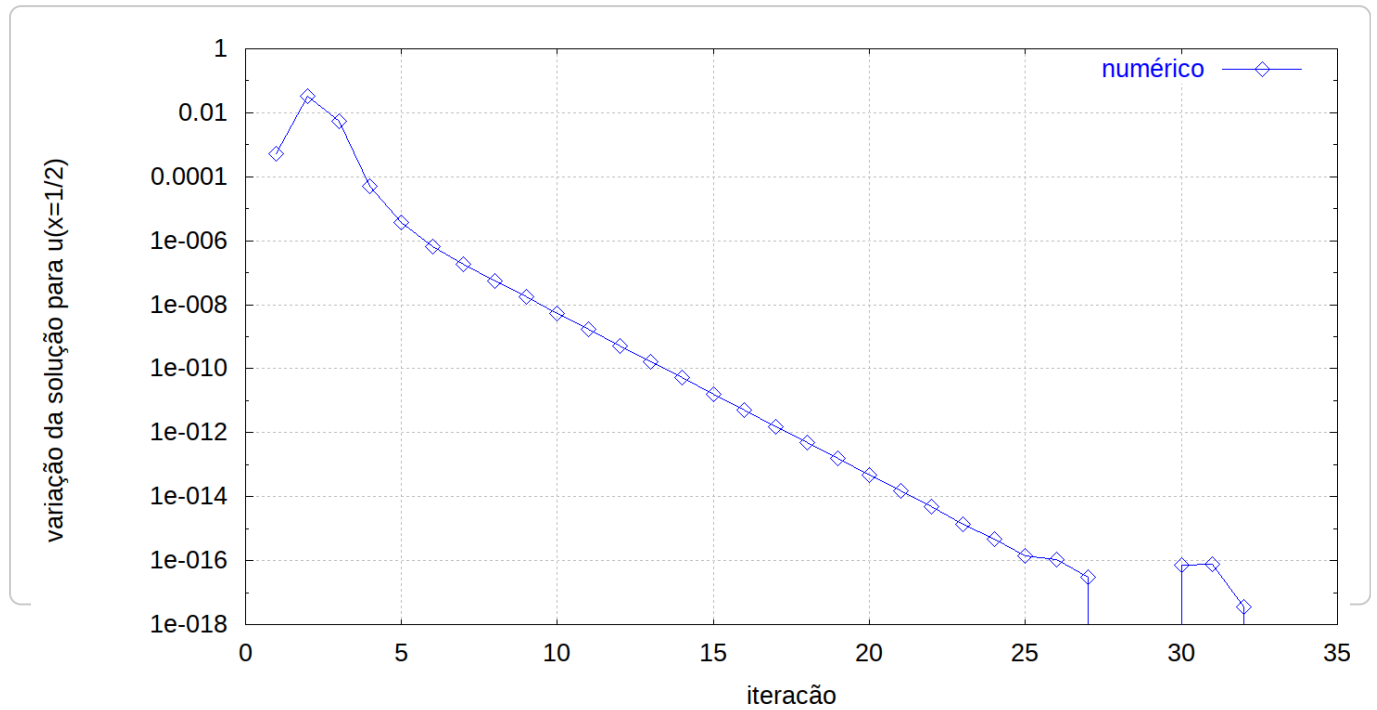
### ITENS SOLICITADOS

- Item 1:** Número de iterações que foram necessárias para atingir o erro de arredondamento de máquina. E gráfico da variação de  $u(1/2)$  em cada iteração (em escala logarítmica) versus número da iteração (em escala decimal).
- Item 2:** Tabela de Coeficientes ( $a_W, a_E, a_P, b_P$ ) para a solução final .
- Item 3:** Tabela comparativa:  $u_{Analítico}$  vs  $u_{Numérico}$  e erro absoluto.
- Item 4:** Gráfico de  $u(x)$  contendo as soluções analítica e numérica.
- Item 5:** Soluções da velocidade média e erro associado (Integração numérica).
- Item 6:** Cálculo da Média da Norma L1 do erro.
- Item 7:** Listagem do programa computacional.



## Item 1: Número de iterações e Gráfico de Convergência

Foram necessárias **28** iterações para convergência até a tolerância especificada. O gráfico abaixo mostra a variação do resíduo máximo em escala logarítmica ao longo das iterações.



2. Tabelas de Coeficientes e Resultados

Para a solução final, tabela contendo em cada linha: número do nó,  $x_P$ ,  $a_W$ ,  $a_P$ ,  $a_E$ ,  $b_P$ , onde  $a_P u_P = a_W u_W + a_E u_E + b$

Item 2: Tabela de Coeficientes Finais

Nota: Linhas em vermelho indicam volumes fictícios.

Vol (i)	$a_W$	$a_E$	$a_P$	$b_P$ (Fonte)
1	0.000000	-1.000000	1.000000	0.000000E+00
2	2.000000	2.000000	3.990898	-1.408196E-04
3	1.990898	2.000000	3.982064	-3.134034E-04
4	1.982064	2.000000	3.973844	-7.412659E-04
5	1.973844	2.000000	3.966967	-1.802073E-03
6	1.966967	2.000000	3.963108	-4.427064E-03
7	1.963108	2.000000	3.966284	-1.089540E-02
8	1.966284	2.000000	3.986308	-2.667710E-02
9	1.986308	2.000000	4.047357	-6.421890E-02
10	2.047357	2.000000	4.209389	-1.475209E-01
11	2.209389	2.000000	4.623351	-2.949012E-01
12	2.623351	2.000000	5.818182	-5.132946E-01
13	-1.000000	0.000000	1.000000	2.000000E+00

Coeficientes e termos-fontes

Item 3: Tabela de Resultados (Numérico vs. Analítico)

Uma tabela contendo em cada linha (incluindo os dois dos contornos): número do volume, xP, uP analítico, uP numérico, e o erro.

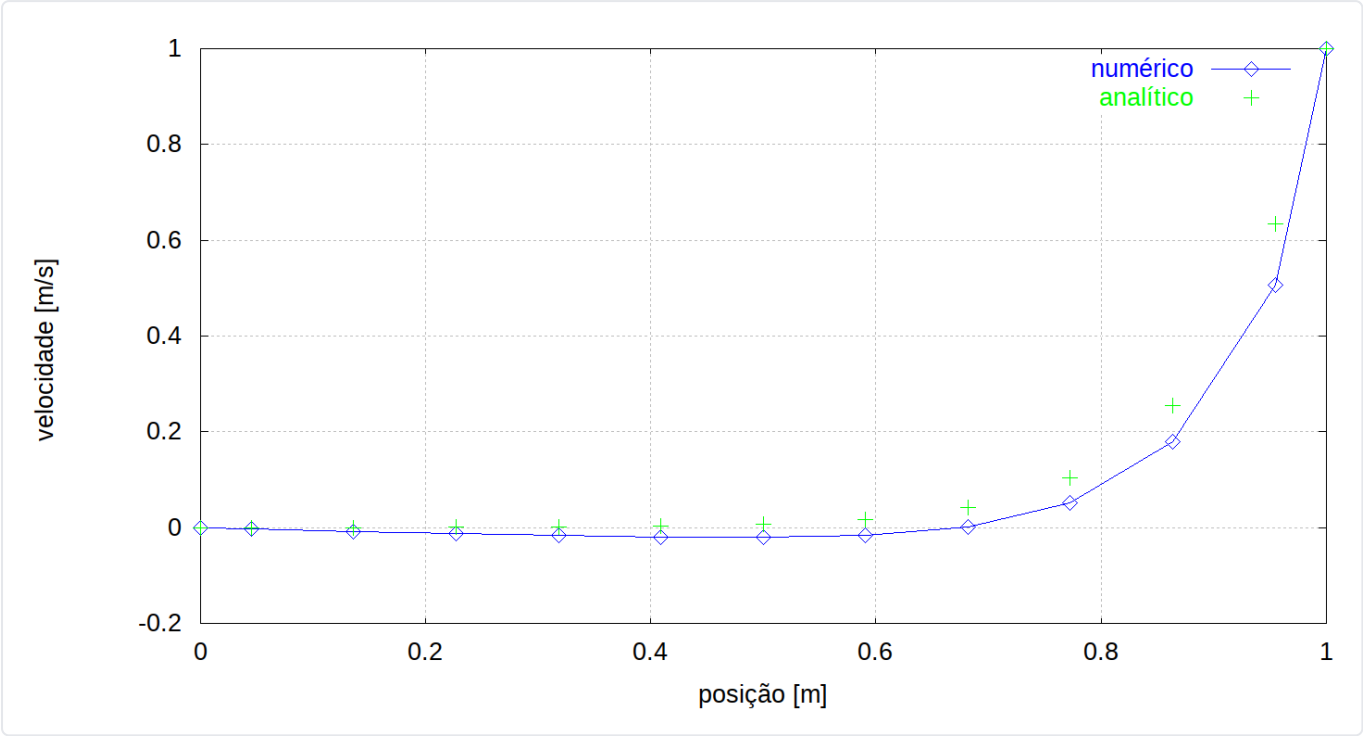
Nota: Linhas em vermelho indicam volumes fictícios.

A Tabela A seguir apresenta, para cada volume de controle, incluindo os volumes fictícios de contorno, a posições volumes de controles calculados pelo código fortran, são comparados com os valores analíticos obtidos a partir da solução exata da equação de Burgers.

Vol (i)	Posição $x$	$u_{Analítico}$	$u_{Numérico}$	Erro Absoluto
1	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	4.545455E-02	2.612690E-05	-2.523513E-03	2.549640E-03
3	1.363636E-01	1.321374E-04	-7.488645E-03	7.620782E-03
4	2.272727E-01	3.952623E-04	-1.224140E-02	1.263666E-02
5	3.181818E-01	1.048355E-03	-1.653059E-02	1.757894E-02
6	4.090909E-01	2.669375E-03	-1.980580E-02	2.247518E-02
7	5.000000E-01	6.692851E-03	-2.077518E-02	2.746803E-02
8	5.909091E-01	1.667938E-02	-1.631197E-02	3.299135E-02
9	6.818182E-01	4.146660E-02	1.251236E-03	4.021536E-02
10	7.727273E-01	1.029901E-01	5.084185E-02	5.214823E-02
11	8.636364E-01	2.556954E-01	1.794862E-01	7.620921E-02
12	9.545455E-01	6.347198E-01	5.061996E-01	1.285202E-01
13	1.000000E+00	1.000000E+00	1.000000E+00	0.000000E+00

Item 4: Gráfico de  $u_P$  versus  $x_P$  (Analítico x Numérico)

Inclui os dois volumes fictícios de contorno ( $i = 1$  e  $i = 13$ ).



Item 5: Velocidade média (regra do retângulo) e erro

A velocidade média no domínio é definida por:  $\bar{u} = (1/L) \int_0^L u(x) \, dx$ . No método numérico, a integral é aproximada pela **regra do retângulo**:  $\bar{u}_{num} \approx (1/L) \sum u_P(i) \cdot \Delta x$ , com  $L = 1$ .

Grandeza	Valor
Velocidade média numérica, $\bar{u}_{num}$ (retângulo)	5.837288626E-02
Velocidade média analítica, $\bar{u}_{ana}$	9.995459801E-02
Erro absoluto, $ \bar{u}_{ana} - \bar{u}_{num} $	4.158171175E-02
Erro relativo, $dfrac{ \bar{u}_{ana} - \bar{u}_{num} }{ \bar{u}_{ana} }$	4.159E-01 ( $\approx 41.6\%$ )

Observação: como  $L = 1$ , a normalização por  $L$  não altera o valor numérico.

Item 6: Média da norma L1 do erro de  $u(x)$

Seja o erro em cada volume:  $e(i) = |uP\_num(i) - uP\_ana(i)|$ . A norma L1 discreta é definida por:  $||e||_1 = \sum e(i)$ . A **média** (norma L1 normalizada por N) é:  $||e||_1 / N$ .

Grandeza	Valor
Norma L1, $  e  _1$	4.204136184E-01
Média da norma L1, $  e  _1/N$	3.821941985E-02
Número de volumes (incluindo contornos), $N$	13

Observação: aqui foi usado **N = 13** (incluindo os dois volumes fictícios), como aparece no relatório do programa. Se o seu professor exigir a média apenas nos volumes reais (N-2), é só ajustar para N=11.



## Item 7: Listagem do Programa Computacional

O programa computacional utilizado neste trabalho foi fornecido pelo professor e já implementava corretamente a solução numérica da equação de Burgers unidimensional em regime permanente por meio do Método dos Volumes Finitos, as alterações foram realizadas diretamente no código fortran. São descritas a seguir as modificações efetuadas em relação ao código original.

### Arquivos modificados

As alterações foram restritas aos seguintes arquivos:

- coef.f90
- dados.f90
- resultados.f90
- dados.txt

### coef.f90

```
module coeficientes

use dados

contains

!-----
! Sub-rotina para definir coeficientes fictícios de contorno
subroutine coef_fict

    ! Coeficientes para o primeiro volume (condição de contorno de entrada)
    aw(1) = 0.0d0
    ae(1) = -1.0d0
    ap(1) = 1.0d0
    bp(1) = 0.0d0

    ! Coeficientes para o último volume (condição de contorno de saída)
    aw(N) = -1.0d0
    ae(N) = 0.0d0
    ap(N) = 1.0d0
    bp(N) = 2.0d0

end subroutine coef_fict

!-----
! Sub-rotina para calcular coeficientes reais para volumes internos
subroutine coef_reais
```

```

integer :: i    ! Variável de iteração
real*8  :: S    ! Variável auxiliar para termo fonte
real*8  :: local_dx ! Cópia local de dx para demonstração

local_dx = dx ! Atribuição de dx a uma variável local

! Loop sobre os volumes internos
do i = 2, N-1
    aw(i) = 2.0d0 + Re*( u(i-1) + u(i) )*local_dx
    ae(i) = 2.0d0
    ap(i) = 4.0d0 + Re*( u(i) + u(i+1) )*local_dx

    S = (Re**2)*dexp( x(i)*Re )*( 2.0d0*dexp( x(i)*Re )-dexp(Re) -
        ( dexp(Re)-1.0d0 )**2
    bp(i) = 2.0d0*S*(local_dx**2) + 0.5d0*beta*Re*( 2.0d0*u(i)**2
end do

end subroutine coef_reais

!-----
! Sub-rotina para definir as posições dos volumes de controle
subroutine posicao

    integer :: i ! Variável de iteração

    x(1) = 0.0d0
    do i = 2, N-1
        x(i) = (i-1.5d0)*dx
    end do
    x(N) = 1.0d0

end subroutine posicao

!-----

end module coeficientes

```

```

module dados

! Propriedades físicas do problema
real*8 :: Re ! Número de Reynolds

! Variáveis do modelo numérico
integer :: N ! Número de volumes de controle
integer :: iter ! Contador de iterações
integer :: itmax ! Número máximo de iterações
character(len=20) :: program_version = "v1.0.0-romulo" ! Versão do pr

real*8 :: dx ! Comprimento de cada volume de controle

real*8,dimension(:),allocatable :: x ! Posição do centro de cada vol

real*8,dimension(:),allocatable :: ae ! Coeficiente do volume a Leste
real*8,dimension(:),allocatable :: aw ! Coeficiente do volume a Oeste
real*8,dimension(:),allocatable :: ap ! Coeficiente do volume central
real*8,dimension(:),allocatable :: bp ! Termo-fonte do volume central

! Variáveis de interesse para análise
real*8 :: um ! Velocidade média numérica
real*8 :: uman ! Velocidade média analítica

real*8,dimension(:),allocatable :: u ! Velocidade no volume P (num
real*8,dimension(:),allocatable :: uanl ! Velocidade no volume P (ana

! Variáveis do processo numérico
real*8 :: beta ! Coeficiente do termo difusivo da função de interpo

! Norma L1 para avaliação de erro
real*8 :: L1 ! Norma L1 do erro

! Variável para execução de comandos do sistema
integer :: dos

```

contains

```
!-----  
! Sub-rotina para leitura dos dados de entrada  
subroutine le_dados  
  
    use portlib  
  
    dos = system("notepad dados.txt") ! Abre o arquivo de dados no notepad  
  
    open(5,file='dados.txt')  
  
    read(5,*) N  
    read(5,*) Re  
    read(5,*) beta  
    read(5,*) itmax  
  
    close(5)  
  
end subroutine le_dados  
  
!-----  
! Sub-rotina para escrita dos dados de saída  
subroutine escreve_dados  
  
    use portlib  
  
    integer :: i ! Contador de loop  
  
    open(5,file='saida.txt')  
    open(7,file='u.dat')  
  
    ! Cabeçalho e dados iniciais  
    write(5,10) program_version  
    write(5,11)  
    write(5,12) N  
    write(5,13) Re  
    write(5,14) beta  
    write(5,15) itmax  
  
    ! Coeficientes e termos-fonte calculados  
    write(5,51)
```

```

do i = 1, N
    write(5,52) i, x(i), aw(i), ae(i), ap(i), bp(i)
end do

! Resultados de velocidade (numérica e analítica)
write(5,21)
write(5,22) 1, 0.0d0, 0.0d0, 0.0d0, 0.0d0
write(7,23) 0.0d0, 0.0d0, 0.0d0
do i = 2, N-1
    write(5,22) i, x(i), u(i), uanl(i), uanl(i)-u(i)
    write(7,23) x(i), u(i), uanl(i)
end do
write(5,22) N, 1.0d0, 1.0d0, 1.0d0, 0.0d0
write(7,23) 1.0d0, 1.0d0, 1.0d0

! Variáveis secundárias (velocidades médias)
write(5,34) um
write(5,35) uman
write(5,36) uman - um

! Normas de erro
write(5,41) L1
write(5,42) L1 / (N-2.0d0)

close(5)
close(7)

! Execução de scripts de plotagem e abertura do arquivo de saída
dos = system ("wgnuplot ucentral.gnu")
dos = system ("wgnuplot u.gnu")
dos = system ("notepad saida.txt")

10 format ( "Versão do Programa: ", A )
11 format (      "Solução Numérica do Escoamento Unidimensional em 1D", &
            /, "Equação de Burgers", &
            2/, "Dados de entrada")
12 format (/ , t10, i9, t20, " = N:      Número de volumes de controle")
13 format ( 1pe18.10, t20, " = Re:      Número de Reynolds")
14 format ( 1pe18.10, t20, " = beta :   Coeficiente do termo difusivo")
15 format (/ , t10, i9, t20, " = itmax:  Número máximo de iterações")

51 format (2/, "Coeficientes e termos-fontes", &

```

```

                2/, t3, "Volume i", t15, "xp(i)", t40, "aw(i)", t60, "
52 format (i10, 4(1pe25.15) )

21 format (2/, "Soluções numéricas", &
            2/, t5, "Volume", t16, "Posição [m]", t35, "Vel. numéri
22 format (i10, 1pe20.9, 3(1pe25.15) )

23 format (1pe20.9, 2(1pe25.15) )

34 format (2/, "Velocidade média numérica [m/s]:      ", 1pe25.15)
35 format (      "Velocidade média analítica [m/s]:      ", 1pe25.15)
36 format (      "Erro numérico para a velocidade média: ", 1pe25.15)

41 format (2/, "Norma L1:      ", 1pe25.15)
42 format (      "Norma L1 / N: ", 1pe25.15)

end subroutine escreve_dados

!-----

end module dados

```

## dados.txt

```

13 ..... N:      número de volumes de controle (reais + dois
10.0d0 ..... Re:      número de Reynolds [adim.]
1.0d+0 .... beta:  coeficiente do termo difusivo (funções de in
35 ..... itmax:  número máximo de iterações

```

## principal.f90

```

program trab02

use resultados
use portlib

call processamento

```

```
end program trab02
```

## resultados.f90

```
module resultados

use dados
use coeficientes
use solvers

contains

!-----
! Sub-rotina principal para o processamento da simulação
subroutine processamento

    real*8 :: ucent ! Variável auxiliar para u(x=1/2) da iteração anterior

    ! Leitura dos dados de entrada
    call le_dados

    ! Alocação dinâmica das variáveis
    allocate ( aw(N), ae(N), ap(N), bp(N), u(N), uanl(N), x(N) )

    ! Definição do comprimento do volume de controle
    dx = 1.0d0/(N-2.0d0)

    ! Definição da posição de cada volume de controle
    call posicao

    ! Cálculo da solução analítica
    call solucao_analitica

    ! Inicialização da solução numérica com a solução analítica
    u = uanl
    ucent = u((N+1)/2)

    ! Cálculo dos coeficientes e termos-fonte fictícios (condições de
    call coef_fict
```

```

open(8,file = 'ucentral.dat')

! Loop principal de iterações
do iter = 1, itmax

    ! Cálculo dos coeficientes e termos-fonte reais para os volumes
    call coef_reais

    ! Solução do sistema de equações lineares usando o método TDM
    call tdma ( N, ap, aw, ae, bp, u )

    ! Escrita do erro de convergência para o ponto central
    write(8,5) iter, dabs( ucent-u((N+1)/2) )
    ucent = u((N+1)/2)

    ! Exemplo de linha de depuração (não altera a lógica principal)
    print *, "Iteração: ", iter, " | u_central: ", u((N+1)/2)

end do

close (8)

! Pós-processamento dos resultados
call pos_processamento

! Escrita dos dados de saída para arquivos
call escreve_dados

5 format ( i9, 1pe25.15 )

end subroutine processamento

!-----
! Sub-rotina para pós-processamento dos resultados
subroutine pos_processamento

    integer :: i ! Contador de loop

    ! Cálculo da velocidade média numérica
    um = 0.0d0
    do i = 2, N-1
        um = um + u(i)
    end do

```



```

end do
um = um*dx

! Cálculo da Norma L1 do erro
L1 = 0.0d0
do i = 2, N-1
    L1 = L1 + dabs( uanl(i) - u(i) )
end do

! Cálculo da velocidade média analítica
uman = (dexp(Re) - Re - 1.0d0) / ( Re*(dexp(Re)-1.0d0) )

end subroutine pos_processamento

!-----
! Sub-rotina para cálculo da solução analítica
subroutine solucao_analitica

    integer :: i ! Contador de loop

    ! Cálculo da velocidade analítica para cada ponto
    do i = 1, N
        uanl(i) = (dexp(x(i)*Re) - 1.0d0) / (dexp(Re) - 1.0d0)
    end do

end subroutine solucao_analitica

!-----

end module resultados

```

## saida.txt

Versão do Programa: v1.0.0-romulo  
 Solução Numérica do Escoamento Unidimensional em Regime Permanente:  
 Equação de Burgers

Dados de entrada

13 = N:            Número de volumes de controle (reais + 0

1.0000000000E+01 = Re: Número de Reynolds  
 1.0000000000E+00 = beta : Coeficiente do termo difusivo (funções de transferência)  
 35 = itmax: Número máximo de iterações

#### Coeficientes e termos-fontes

Volume i	xp(i)	aw(i)	ae(i)
1	0.0000000000000000E+00 0.0000000000000000E+00	0.0000000000000000E+00	-1.0000000000000000E+00
2	4.545454545454546E-02 -1.408196108886616E-04	2.0000000000000000E+00	2.0000000000000000E+00
3	1.363636363636364E-01 -3.134033794493489E-04	1.990898038283405E+00	2.0000000000000000E+00
4	2.272727272727273E-01 -7.412659210599084E-04	1.982063595817331E+00	2.0000000000000000E+00
5	3.181818181818182E-01 -1.802073310922836E-03	1.973843649136886E+00	2.0000000000000000E+00
6	4.090909090909091E-01 -4.427064076976021E-03	1.966966918850314E+00	2.0000000000000000E+00
7	5.000000000000000E-01 -1.089540247774857E-02	1.963108197119905E+00	2.0000000000000000E+00
8	5.909090909090909E-01 -2.667709567798279E-02	1.966284410752332E+00	2.0000000000000000E+00
9	6.818181818181819E-01 -6.421889649185068E-02	1.986308425701091E+00	2.0000000000000000E+00
10	7.727272727272727E-01 -1.475209220694491E-01	2.047357349726212E+00	2.0000000000000000E+00
11	8.636363636363637E-01 -2.949011743974043E-01	2.209389096817946E+00	2.0000000000000000E+00
12	9.545454545454546E-01 -5.132946195365880E-01	2.623350690175596E+00	2.0000000000000000E+00
13	1.000000000000000E+00 2.000000000000000E+00	-1.000000000000000E+00	0.000000000000000E+00

#### Soluções numéricas

Volume	Posição [m]	Vel. numérica [m/s]	Vel. analítica [m/s]
1	0.000000000E+00	0.000000000000000E+00	0.000000000E+00
2	4.545454545E-02	-2.523513038308085E-03	2.61268982E-03

3	1.363636364E-01	-7.488644849946920E-03	1.32137367!
4	2.272727273E-01	-1.224139975098953E-02	3.95262251!
5	3.181818182E-01	-1.653058619843549E-02	1.04835534!
6	4.090909091E-01	-1.980580306621889E-02	2.66937489!
7	5.000000000E-01	-2.077518010188620E-02	6.69285092!
8	5.909090909E-01	-1.631196807054894E-02	1.66793803!
9	6.818181818E-01	1.251236341748639E-03	4.14665961!
10	7.727272727E-01	5.084184835708505E-02	1.02990079!
11	8.636363636E-01	1.794861581426554E-01	2.55695368!
12	9.545454545E-01	5.061996010505006E-01	6.34719835!
13	1.000000000E+00	1.000000000000000E+00	1.00000000!

Velocidade média numérica [m/s]: 5.837288625596868E-02  
 Velocidade média analítica [m/s]: 9.995459800899031E-02  
 Erro numérico para a velocidade média: 4.158171175302163E-02

Norma L1: 4.204136184016680E-01  
 Norma L1 / N: 3.821941985469710E-02

## solver.f90

```

module solvers

contains

!-----

! método direto Tri-Diagonal Matrix Algorithm (TDMA)

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

  implicit none

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

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

```

```
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
```