TRABALHO DE FIM DE CURSO

fvMatrix

Curso:

L2C - Soluções em Computação Científica

Do Cálculo à Simulação Computacional – Fundamentos de Métodos Numéricos com Aplicações – Turma 2025 / 2

Professor:

Rafael Gabler Gontijo

Aluno:

Edgard Wiggers

Resumo

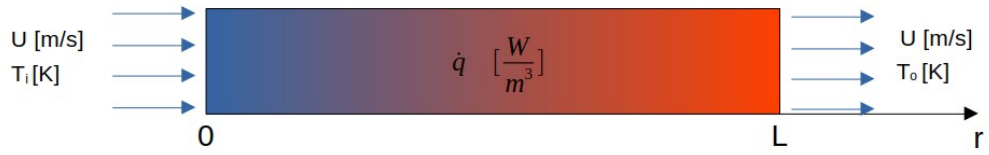
O objetivo deste trabalho foi a implementação em C++ dos algorítmos para solução iterativa de sistemas de equações algébricas, usando os seguintes métodos:

- Gauss-Siedel;
- Gradiente Conjugado (GC);
- Gradiente Conjugado pré-condicionado (PGC);
- Gradiente Bi-Conjugado (GBiC);
- Gradiente Bi-Conjugado pré-condicionado (PGBiC);
- Gradiente Bi-Conjugado pré-condicionado estabilizado (PGBiC-Stab).

Com o objetivo de aplicar e validar os algoritmos implementados, foi desenvolvido um programa para gerar um sistema de equações, na forma A.X = B, baseado na discretização de um problema físico, utilizando o Método dos Volumes Finitos (FVM). Para se entender o funcionamento do programa, será feito primeiramente uma revisão do FVM, já no contexto do enunciado do problema físico que se pretende resolver.

1 – Enunciado do problema físico

Considere o escoamento de ar através de um duto unidimensional (1D), de comprimento L [m], em regime estacionário, com velocidade U [m/s], temperaturas fixadas na entrada e saída do duto, Ti [K] e To [K], respectivamente, com geração interna de calor $\dot{q}$ [W/m^3], conforme ilustrado abaixo. Determinar a termperatura do ar ao longo do duto.



$$\nabla\cdot(\rho.c_p.\vec{U}.T)=\nabla\cdot(k\vec{\nabla}T)+F_\phi \qquad (1)$$

$$\nabla\cdot(\vec{U}.T)=\nabla\cdot(\alpha\vec{\nabla}T)+\frac{1}{\rho.c_p}F_\phi \qquad (2)$$

$$\int_V \nabla\cdot(\vec{U}.T)-\nabla\cdot(\alpha\vec{\nabla}T)dV=\frac{1}{\rho.c_p}\int_V \dot{q}\,dV \qquad (3)$$

$$T(x)=T_i+\frac{x}{L}(T_o-T_i)+\frac{\dot{q}}{2k}x(L-x) \qquad (4.a)$$

$(para \quad u=0)$

$$\frac{T(x)-T_i}{T_o-T_i}=\frac{e^{(\frac{u.x}{\alpha})}-1}{e^{(\frac{u.L}{\alpha})}-1} \qquad (4.b)$$

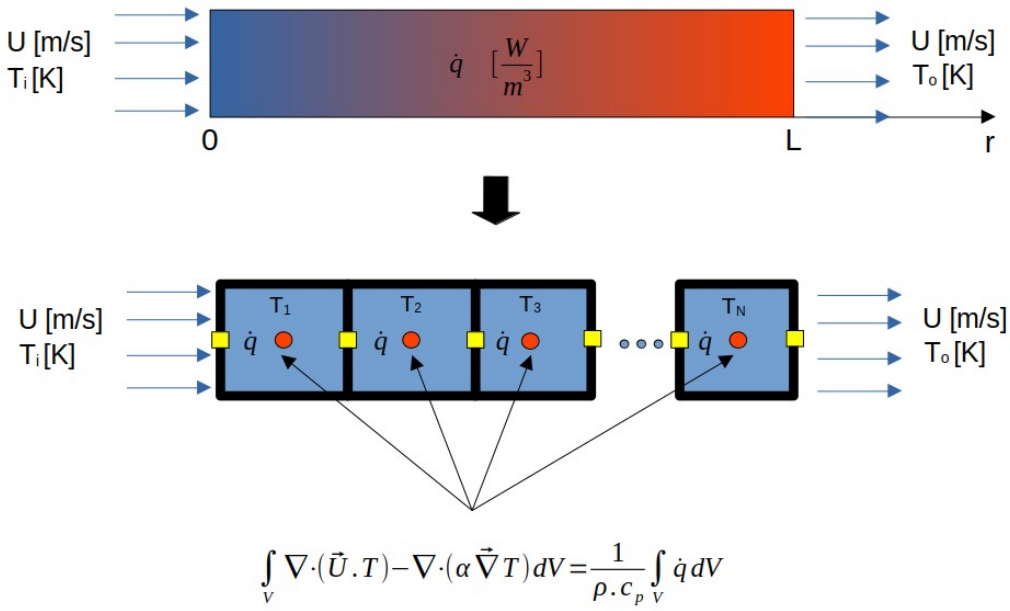
$(para \quad \dot{q}=0)$

α [m^2/s] = $\frac{k}{\rho.c_p}$, onde: k [W/mK], c_p [J/kgK], ρ [kg/m^3]

O perfil de temperatura ao longo do duto pode ser determinado analiticamente usando-se as equações 4.a ou 4.b, atentando-se para as restrições de aplicação de cada uma delas.

Pode-se também obter uma solução numérica, usando-se o método dos volumes finitos (FVM), através da discretização da geometria do duto em volumes de controle, onde se resolve a equação 3 em cada um dos volumes discretizados.

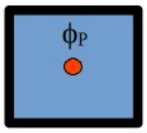
2 – Revisão do Método dos Volumes Finitos (FVM)



$$\int_V \nabla\cdot(\vec{U}.T) - \nabla\cdot(\alpha\vec{\nabla}T)\,dV = \frac{1}{\rho.c_p}\int_V \dot{q}\,dV$$

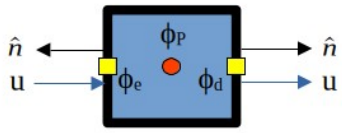
Para obter a solução numérica do problema, primeiramente aplica-se o Teorema de Gauss na equação 3 (ou equação 5, já escrita utilizando a variável genérica phi), que converte uma integral de volume em uma integral de superfície (a superfície em torno do volume, que delimita o volume). Como neste caso a malha é 1D, a integral de superfície deve ser resolvida considerando-se a superfície esquerda (subíndice "e") e a superfície direita (subíndice "d"), de cada volume de controle. A integração pelo método da quadratura de Gauss permite expressar a integral de superfície (superfície de cada face) através de um valor situado no centróide de cada face, possibilitando que a integral seja convertida em um somatório das faces. Observa-se, no entanto, que a operação de integração do volume de controle resultou em uma expressão para o cálculo dos valores da variável de interesse nas faces dos volumes de controle. Porém, necessita-se de uma expressão para cálculo da variável de interesse que utilize os valores de centróide do volume de controle. Obtem-se a expressão na forma desejada, equação 6, utilizando esquemas de interpolação convenientes para escrever os valores nas faces dos volumes de controle, em função dos valores de centróide dos volumes de controle. Neste caso, utilizou-se a interpolação linear (diferenças centrais), tanto para o termo advectivo, quanto para o termo difusivo. Convém ressaltar que a forma de interpolação não é única, podendo ser adotadas várias formas, em função, principalmente, do tipo de equação diferencial sendo resolvida.

$$\int_V \nabla\cdot(\vec{U}.\phi)-\nabla\cdot(\alpha\vec{\nabla}\phi)\,dV=\boxed{\frac{1}{\rho.c_p}\int_V \dot{q}\,dV} \rightarrow F_\phi \qquad (5)$$



Teorema de Gauss e Integração por Quadratura de Gauss

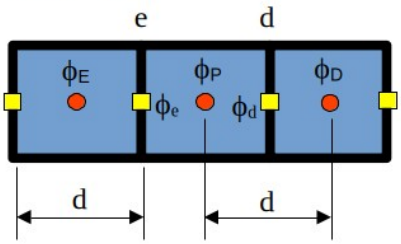
$$\sum_{faces}(\vec{U}\cdot\hat{n}.A.\phi)-\sum_{faces}(\alpha.A.\vec{\nabla}\phi\cdot\hat{n})=\frac{\dot{q}.V}{\rho.c_p}$$



$$(u.A.\phi)\big|_d-(u.A.\phi)\big|_e-[(\alpha.A.\frac{\partial\phi}{\partial x})\Big|_d-(\alpha.A.\frac{\partial\phi}{\partial x})\Big|_e]=\frac{\dot{q}.V}{\rho.c_p}$$

Interpolação linear para escrever valores nas faces em função de valores de centro

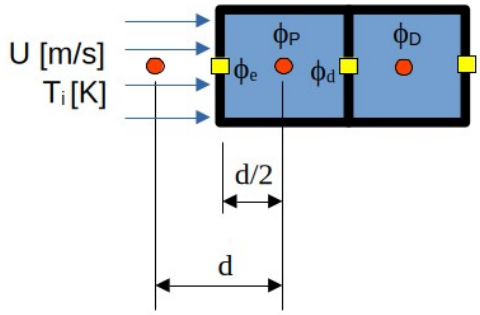
Considerando malha uniforme, temos:



$$(u.A\frac{\phi_D+\phi_P}{2})-(u.A\frac{\phi_P+\phi_E}{2})-[(\alpha.A.\frac{\phi_D-\phi_P}{d})-(\alpha.A.\frac{\phi_P-\phi_E}{d})]=\frac{\dot{q}.V}{\rho.c_p}$$

$$\boxed{(\frac{\alpha}{d}+\frac{\alpha}{d})\phi_P=(\frac{\alpha}{d}-\frac{u}{2})\phi_D+(\frac{\alpha}{d}+\frac{u}{2})\phi_E+\frac{\dot{q}.d}{\rho.c_p}} \qquad (6)$$

A dedução acima vale para volumes de controle no interior do domínio. No caso de volumes de controle na fronteira do dominio, ou seja, volumes de controle cuja uma, ou mais faces, formam o contorno do domínio, a equação 6 toma formas diferentes. No caso do volume de controle do lado esquerdo do duto, os valores das condições de contorno são aplicados diretamente à face esquerda do volume, cujo afastamento em relação ao centróide do volume de controle é igual a metade do comprimento do volume de controle. Analogamente, no volume de controle do lado direito do duto, os valores das condições de contorno são aplicados diretamente à face direita do volume de controle.

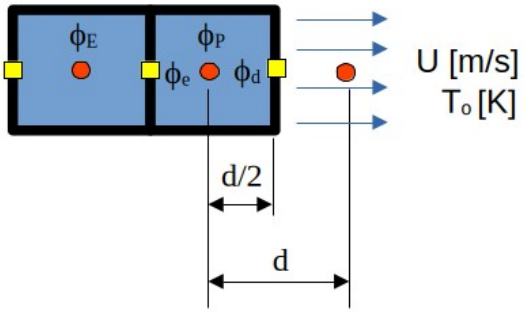


$$(u.A.\phi)\big|_d-(u.A.\phi)\big|_e-[(\alpha.A.\frac{\partial\phi}{\partial x})\Big|_d-(\alpha.A.\frac{\partial\phi}{\partial x})\Big|_e]=\frac{\dot{q}.V}{\rho.c_p}$$

$$u.A(\frac{\phi_D+\phi_P}{2})-(u.A.T_i)-[(\alpha.A.\frac{\phi_D-\phi_P}{d})-(\alpha.A.\frac{\phi_P-T_i}{\frac{d}{2}})]=\frac{\dot{q}.V}{\rho.c_p}$$

$$\frac{u}{2}(\phi_D+\phi_P)-[\frac{\alpha}{d}(\phi_D-\phi_P)-\frac{2.\alpha}{d}\phi_P]=\frac{\dot{q}.V}{\rho.c_p.A}+(u+\frac{2.\alpha}{d})T_i$$

$$\boxed{(\frac{u}{2}+\frac{\alpha}{d}+\frac{2.\alpha}{d})\phi_P=(\frac{\alpha}{d}-\frac{u}{2})\phi_D+0.\phi_E+\frac{\dot{q}.d}{\rho.c_p}+(u+\frac{2.\alpha}{d})T_i} \qquad (7)$$



$$\boxed{(\frac{\alpha}{d}+\frac{2.\alpha}{d}-\frac{u}{2})\phi_P=0.\phi_D+(\frac{\alpha}{d}+\frac{u}{2})\phi_E+\frac{\dot{q}.d}{\rho.c_p}+(\frac{2.\alpha}{d}-u)T_o} \qquad (8)$$

Basicamente, cada volume de controle tem uma equação algébrica associada com subindice P, que o relaciona com os volumes vizinhos, correspondendo a uma linha de um sistema de equações algébricas lineares. O volume de controle em questão tem o subindice "P, o volume à esquerda, subindice "E" e o volume a direita, subindice "D". As equações 6, 7 e 8 foram deduzidas e já colocadas na forma padrão (equação 9), o que facilita o reconhecimento dos coeficientes para montagem das matrizes.

$$a_P.\phi_P = a_D.\phi_D + a_E.\phi_E + b_P \qquad (9)$$

$$b_P = F_\phi + c_P.F_P \qquad (10)$$

$$a_P = a_D + a_E + F_P \qquad (11)$$

Substituindo (6) em (9), temos a expressão para elementos internos

$$(\underbrace{\frac{\alpha}{d}+\frac{\alpha}{d}}_{a_P})\phi_P = (\underbrace{\frac{\alpha}{d}-\frac{u}{2}}_{a_D})\phi_D + (\underbrace{\frac{\alpha}{d}+\frac{u}{2}}_{a_E})\phi_E + \boxed{\underbrace{\frac{\dot{q}.d}{\rho.c_p}}_{F_\phi}}^{b_p}$$

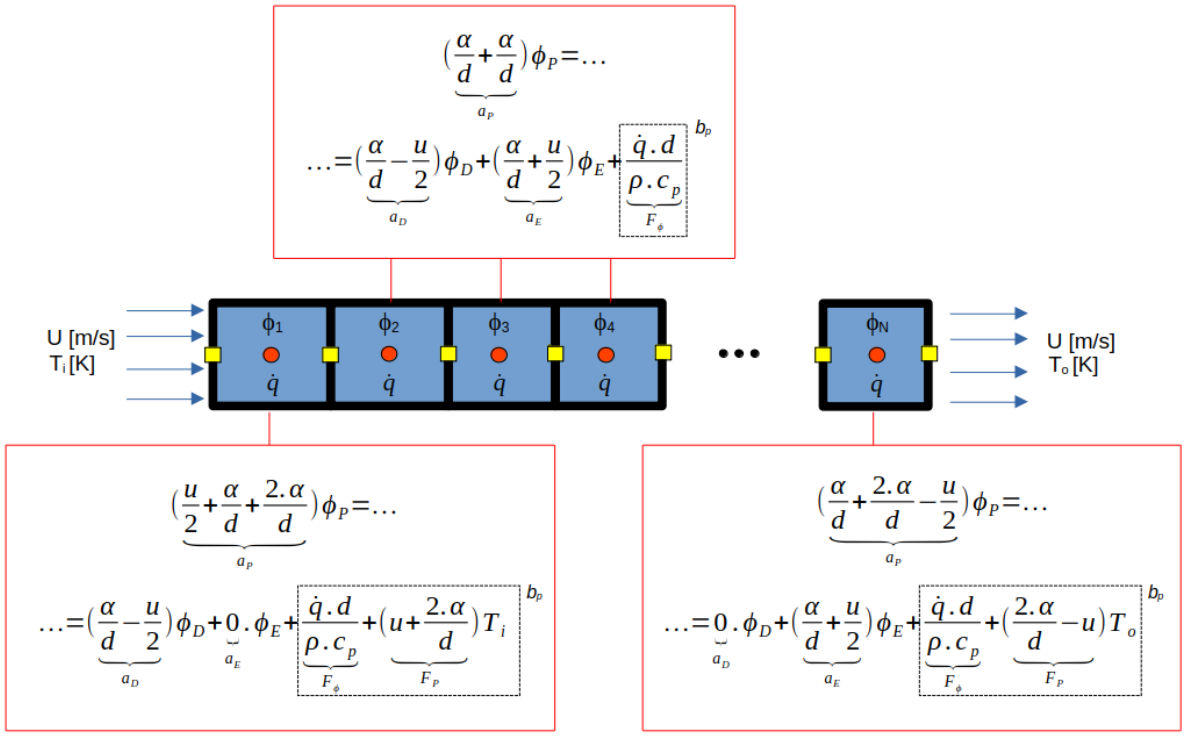
Substituindo (7) em (9), temos a expressão para elemento de contorno esquerdo

$$(\underbrace{\frac{u}{2}+\frac{\alpha}{d}+\frac{2.\alpha}{d}}_{a_P})\phi_P = (\underbrace{\frac{\alpha}{d}-\frac{u}{2}}_{a_D})\phi_D + \underbrace{0}_{a_E}.\phi_E + \boxed{\underbrace{\frac{\dot{q}.d}{\rho.c_p}}_{F_\phi} + \underbrace{(u+\frac{2.\alpha}{d})}_{F_P}T_i}^{b_p}$$

Substituindo (8) em (9), temos a expressão para elemento de contorno direito

$$(\underbrace{\frac{\alpha}{d}+\frac{2.\alpha}{d}-\frac{u}{2}}_{a_P})\phi_P = \underbrace{0}_{a_D}.\phi_D + (\underbrace{\frac{\alpha}{d}+\frac{u}{2}}_{a_E})\phi_E + \boxed{\underbrace{\frac{\dot{q}.d}{\rho.c_p}}_{F_\phi} + \underbrace{(\frac{2.\alpha}{d}-u)}_{F_P}T_o}^{b_p}$$

Depois de obter as equações algébricas para o cálculo da variável de interesse do escoamento, é necessário arranjar as equações na forma de um sistema linear, no formato A.X = B, montar os vetores A e B, para que então o sistema possa ser resolvido utilizando algum método numérico implementado neste repositório.



Para facilitar a montagem dos vetores A e B na equação matricial A.X = B, convém rearranjar os termos da equação 9 da seguinte forma:

$$-a_E.\phi_E+a_P.\phi_P-a_D.\phi_D=b_P$$

Mudando os indices locais (E, P, D) das variáveis phi, para indices globais (1, 2, n, … ,N) adequados ao sistema de equações, temos:

$$-a_{E,n}.\phi_{n-1}+a_{P,n}.\phi_n-a_{D,n}.\phi_{n+1}=b_n$$

$$\begin{bmatrix} a_{P,1} & -a_{D,1} & 0 & 0 & 0 & 0 & 0 \\ -a_{E,2} & a_{P,2} & -a_{D,2} & 0 & 0 & 0 & 0 \\ 0 & -a_{E,3} & a_{P,3} & -a_{D,3} & 0 & 0 & 0 \\ 0 & 0 & -a_{E,4} & a_{P,4} & -a_{D,4} & 0 & 0 \\ 0 & 0 & 0 & -a_{E,5} & a_{P,5} & -a_{D,5} & 0 \\ 0 & 0 & 0 & 0 & -a_{E,6} & a_{P,6} & -a_{D,6} \\ 0 & 0 & 0 & 0 & 0 & -a_{E,N} & a_{P,N} \end{bmatrix} . \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_N \end{bmatrix}$$

3 – Componentes de software

fvTestCase.cpp

Sintaxe:

```
~/fvMatrix/bin$
./fvTestCase [quantidade de volumes de controle] [velocidade do escoamento] [taxa de geração calor interno]
```

Exemplo:

```
~/fvMatrix/bin$
./fvTestCase 100 0.1 0
```

Resultado: 100 elementos de malha (volumes de controle), velocidade do escoamento de 0.1 m/s e sem geração de calor.

Este programa gera os vetores "A" e "B", do sistema A.X = B e o vetor "X0" de solução inicial, a partir das equações discretizadas do problema físico descrito neste trabalho. Esses vetores são usados pelos programas de solução de sistemas lineares deste repositório. Além disso, é gerada a solução analítica, selecionada automaticamente em função dos dados de entrada fornecidos pelo usuário.

Propriedades do fluido e condições de contorno são informadas no código fonte. Quando há alteração, é necessário compilar novamente.

Cada vez que o fvTestCase é executado, são gerados/editados os seguintes arquivos:

```
~/fvMatrix/bin$
```

- A.dat - Vetor A (matriz de coeficientes).
- B.dat - Vetor B (termos fonte).
- X0.dat - Vetor X0 (solução inicial).
- r.dat - Coordenadas (afastamentos) dos centróides dos volumes de controle, ao longo do duto.
- solAnalitica.dat - Solução analítica do escoamento através do duto (pares r,T). Está no formato apropriado para ser usado no GNUPlot.

gaussSiedel.cpp

Sintaxe:

```
~/fvMatrix/bin$
./gaussSiedel [coeficiente de relaxação lambda] - default: lambda = 1
```

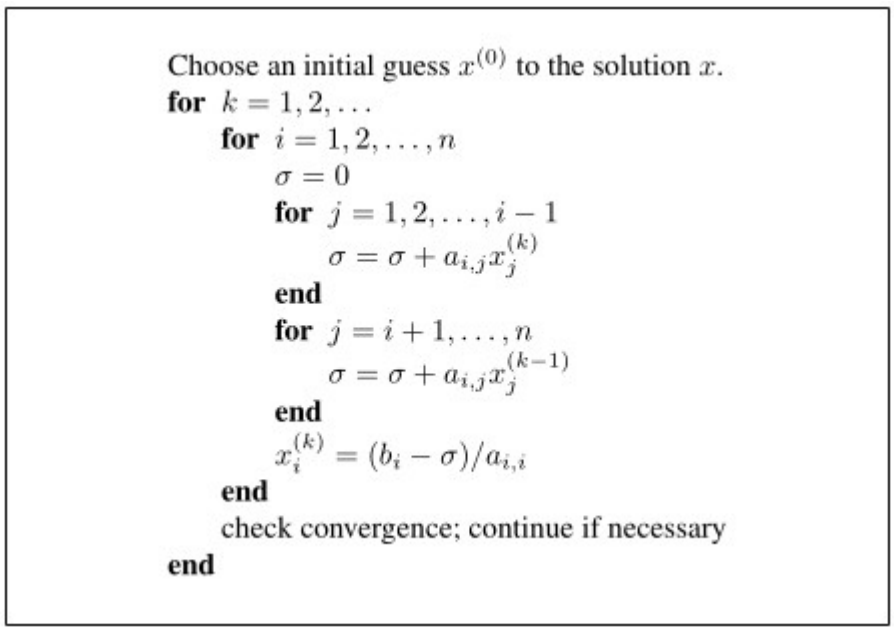
Exemplos:

```
~/fvMatrix/bin$
./gaussSiedel
./gaussSiedel 0.5
```

Resultados:

```
lambda = 1.0
lambda = 0.5
```

Este programa resolve o sistema A.X = B, usando o método Gauss-Siedel. A convergência do método pode ocorrer se, e somente se, a matriz "A" for diagonal dominante. A matriz "A" não precisa ser simétrica. Os dados de entrada são os vetores "A.dat", "B.dat" e X0.dat, gerados pelo usuário, ou pelo programa fvTestCase.cpp.



$$x_i^{(k)} = (b_i - \sum_{j<i} a_{i,j} x_j^{(k)} - \sum_{j>i} a_{i,j} x_j^{(k-1)})/a_{i,i}.$$

No método de Gauss-Siedel com relaxação, após o cálculo do valor atual $x_i^{(k)}$, aplicamos uma correção do tipo:

$$x_i^{(k)} = lambda \,.\, x_i^{(k)} + (1 - lambda) \,.\, x_i^{(k-1)}$$

Para: lambda = 1 --> Gauss-Siedel sem relaxação

Para: 0 < lambda < 1 --> Sub-relaxação. Ajuda alguns sistemas não-convergentes a convergirem.

Para: 1 < lambda < 2 --> Sobre-relaxação. Acelera convergência de sistemas que convergem lentamente.

Cada vez que o gaussSiedel é executado, são gerados/editados os seguintes arquivos:

```
~/fvMatrix/bin$
```

- `X.dat - Vetor X (solução do sistema).`
- `solNumerica.dat - Solução numérica do escoamento através do duto (pares r,X). Está no formato apropriado para ser usado no GNUPlot.`

gradConjugado.cpp

Sintaxe:

```
~/fvMatrix/bin$
./gradConjugado
```

Este programa resolve o sistema A.X = B, usando o método do gradiente conjungado. Para tanto, é necessário que a matriz "A" seja simétrica e positivo definida. Os dados de entrada são os vetores "A.dat", "B.dat" e X0.dat, gerados pelo usuário, ou pelo programa fvTestCase.cpp. O programa fornece o vetor solução no arquivo "X.dat".

$\mathbf{d}^{(0)} = \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\boldsymbol{\phi}^{(0)}$ (choose residual as starting direction)

iterate starting at (n) until convergence

$\alpha^{(n)} = \frac{\mathbf{d}^{(n)T}\mathbf{r}^{(n)}}{\mathbf{d}^{(n)T}\mathbf{A}\mathbf{d}^{(n)}}$ (Choose factor in $\mathbf{d}$ direction)

$\boldsymbol{\phi}^{(n+1)} = \boldsymbol{\phi}^{(n)} + \alpha^{(n)}\mathbf{d}^{(n)}$ (Obtain new $\boldsymbol{\phi}$)

$\mathbf{r}^{(n+1)} = \mathbf{r}^{(n)} - \alpha^{(n)}\mathbf{A}\mathbf{d}^{(n)}$ (calculate new residual)

$\beta^{(n)} = \frac{\mathbf{r}^{(n+1)T}\mathbf{r}^{(n+1)}}{\mathbf{r}^{(n)T}\mathbf{r}^{(n)}}$ (Calculate coefficient to conjugate residual)

$\mathbf{d}^{(n+1)} = \mathbf{r}^{(n+1)} + \beta^{(n)}\mathbf{d}^{(n)}$ (obtain new conjugated search direction)

Cada vez que o gradConjugado é executado, são gerados/editados os seguintes arquivos:

```
~/fvMatrix/bin$
```

- X.dat - Vetor X (solução do sistema).
- solNumerica.dat - Solução numérica do escoamento através do duto (pares r,X). Está no formato apropriado para ser usado no GNUPlot.

gradBiConjugado.cpp

Sintaxe:

```
~/fvMatrix/bin$
./gradBiConjugado
```

Este programa resolve o sistema A.X = B, usando o método do gradiente bi-conjungado. Para tanto, é necessário que a matriz "A" seja positivo definida, porém não há necessidade de ser simétrica. Os dados de entrada são os vetores "A.dat", "B.dat" e X0.dat, gerados pelo usuário, ou pelo programa fvTestCase.cpp. O programa fornece o vetor solução no arquivo "X.dat".

$$\mathbf{d}^{(0)} = \mathbf{r}^{(0)} = \tilde{\mathbf{d}}^{(0)} = \tilde{\mathbf{r}}^{(0)} = \mathbf{b} - \mathbf{A}\boldsymbol{\phi}^{(0)} \quad \text{(choose starting directions)}$$

iterate starting at (n) until convergence

$$\alpha^{(n)} = \frac{\left(\tilde{\mathbf{r}}^{(n)}\right)^{\mathrm{T}} \mathbf{r}^{(n)}}{\left(\tilde{\mathbf{d}}^{(n)}\right)^{\mathrm{T}} \mathbf{A}\mathbf{d}^{(n)}} \quad \text{(Choose factor in } \mathbf{d} \text{ direction)}$$

$$\boldsymbol{\phi}^{(n+1)} = \boldsymbol{\phi}^{(n)} + \alpha^{(n)}\mathbf{d}^{(n)} \quad \text{(Obtain new } \boldsymbol{\phi}\text{)}$$

$$\mathbf{r}^{(n+1)} = \mathbf{r}^{(n)} - \alpha^{(n)}\mathbf{A}\mathbf{d}^{(n)} \quad \text{(calculate new } \mathbf{r} \text{ residual)}$$

$$\tilde{\mathbf{r}}^{(n+1)} = \tilde{\mathbf{r}}^{(n)} - \alpha^{(n)}\mathbf{A}^{\mathrm{T}}\tilde{\mathbf{d}}^{(n)} \quad \text{(calculate new } \tilde{\mathbf{r}} \text{ residual)}$$

$$\beta^{(n+1)} = \frac{\left(\tilde{\mathbf{r}}^{(n+1)}\right)^{\mathrm{T}} \mathbf{r}^{(n+1)}}{\left(\tilde{\mathbf{r}}^{(n)}\right)^{\mathrm{T}} \mathbf{r}^{(n)}} \quad \text{(Calculate coefficient to conjugate residual)}$$

$$\mathbf{d}^{(n+1)} = \mathbf{r}^{(n+1)} + \beta^{(n+1)}\mathbf{d}^{(n)} \quad \text{(obtain new search } \mathbf{d} \text{ direction)}$$

$$\tilde{\mathbf{d}}^{(n+1)} = \tilde{\mathbf{r}}^{(n+1)} + \beta^{(n+1)}\tilde{\mathbf{d}}^{(n)} \quad \text{(obtain new search } \tilde{\mathbf{d}} \text{ direction)}$$

Cada vez que o gradBiConjugado é executado, são gerados/editados os seguintes arquivos:

```
~/fvMatrix/bin$
```

- `X.dat - Vetor X (solução do sistema).`
- `solNumerica.dat - Solução numérica do escoamento através do duto (pares r,X). Está no formato apropriado para ser usado no GNUPlot.`

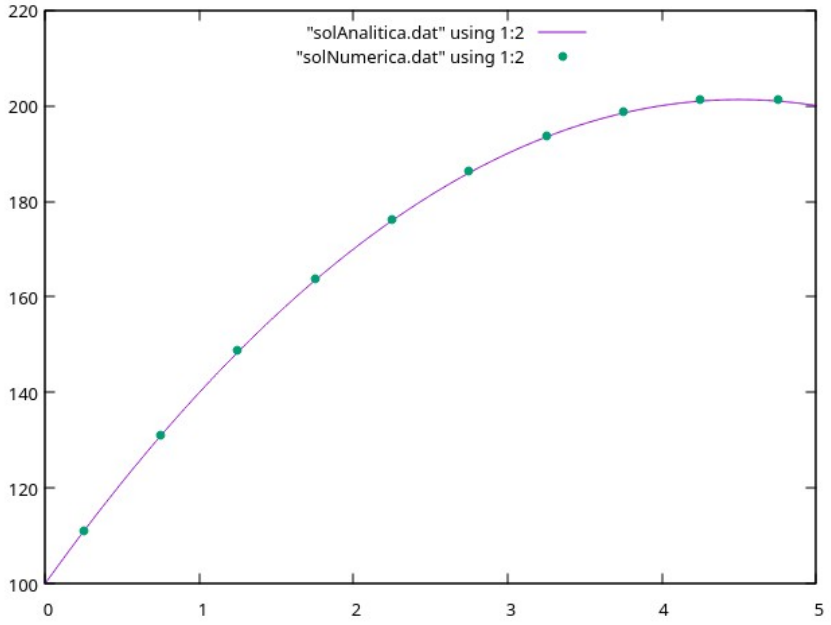
4 – Conclusão

A melhor maneira de validar o funcionamento dos códigos implementados é a comparação entre a solução gerada numericamente e a solução analítica do problema físico proposto. Lembrando que quando o “fvTestCase” é executado, é gerado automaticamente o arquivo “solAnalitica.dat”, cujos dados podem ser plotados em forma de curva, utilizando o aplicativo GNUPlot. Da mesma forma, quando obtem-se a solução numérica do problema físico proposto, através de algum dos programas disponíveis no repositório, é gerado o arquivo “solNumerica.dat”. O desempenho pode ser imediatamente avaliado, simplesmente plotando os dois arquivos no mesmo gráfico. Para facilitar essa tarefa, foi disponibilizado um script chamado simplesmente de “plot”, que chama o aplicativo GNUPlot, passando os parâmetros necessários para a correta geração do gráfico. Para isso, basta executar o scritp utilizando a sintaxe: `./plot` na linha de comando do terminal, dentro do diretório `~/fvMatrix/bin$`. Segue abaixo alguns prints demonstrando o procedimento.

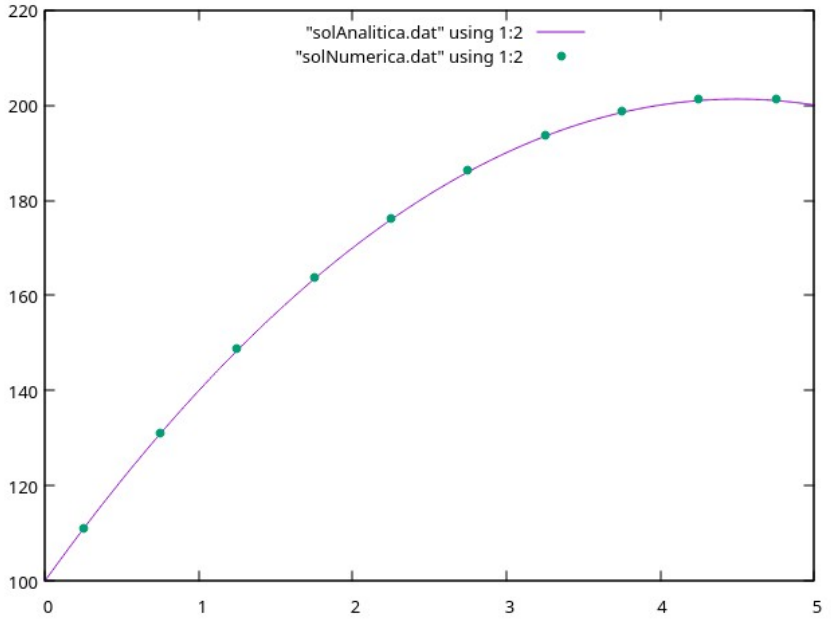
No exemplo abaixo, o `fvTestCase` foi usado para gerar um caso discretizado com 10 elementos (volumes de controle), com velocidade u = 0 m/s e geração interna de calor de 1000 W/m^2. Como trata-se de um caso de difusão de calor, sem componente advectiva, a matriz de coeficientes será simétrica. Nesse caso, pode-se utilizar os métodos de solução de Gauss-Siedel e Gradiente Conjugado.

```
edw@81:~/Documents/fvMatrix/bin$ ./fvTestCase 10 0 1000

---------------------------------- fvMatrix ----------------------------------
-------------------- Gerador de Sistemas de Equações FVM --------------------
---------------------- Duto de ar / Regime estacinário ----------------------

N = 10 elementos
u = 0 m/s
qdot = 1000 W/m3

Arquivo r.dat (coordenadas dos centróides) criado/atualizado com sucesso.
Arquivo A.dat (coeficientes) criado/atualizado com sucesso.
Arquivo B.dat (termos fonte) criado/atualizado com sucesso.
Arquivo X0.dat (estimativa inicial) criado/atualizado com sucesso.
Arquivo solAnalitica.dat (solução analítica) criado/atualizado com sucesso.
edw@81:~/Documents/fvMatrix/bin$ ./gaussSiedel

------------------------------ Método Gauss-Siedel ------------------------------

lambda (default) = 1
Dados carregados......[OK]
Dados consistentes....[OK]

Solução do sistema:
Norma do resíduo = 0.000955146
Número de iterações = 78
Arquivo X.dat (solução do sistema) criado/atualizado com sucesso.
Arquivo solNumerica.dat (r:posição, X:solução do sistema) criado/atualizado com sucesso.

edw@81:~/Documents/fvMatrix/bin$ ./plot
edw@81:~/Documents/fvMatrix/bin$ ./gradConjugado

---------------------------- Método Gradiente Conjugado  ----------------------------
Dados carregados......[OK]
Dados consistentes....[OK]

Solução do sistema de equações:
Norma do resíduo = 3.17764e-14
Número de iterações = 10
Arquivo X.dat (solução do sistema) criado/atualizado com sucesso.
Arquivo solNumerica.dat (r:posição, X:solução do sistema) criado/atualizado com sucesso.

edw@81:~/Documents/fvMatrix/bin$ ./plot
edw@81:~/Documents/fvMatrix/bin$
```

Solução obtida pelo método de Gauss-Siedel, usando o programa `gaussSiedel.`



Solução obtida pelo método do gradiente conjugado, usando o programa `gradConjugado.`



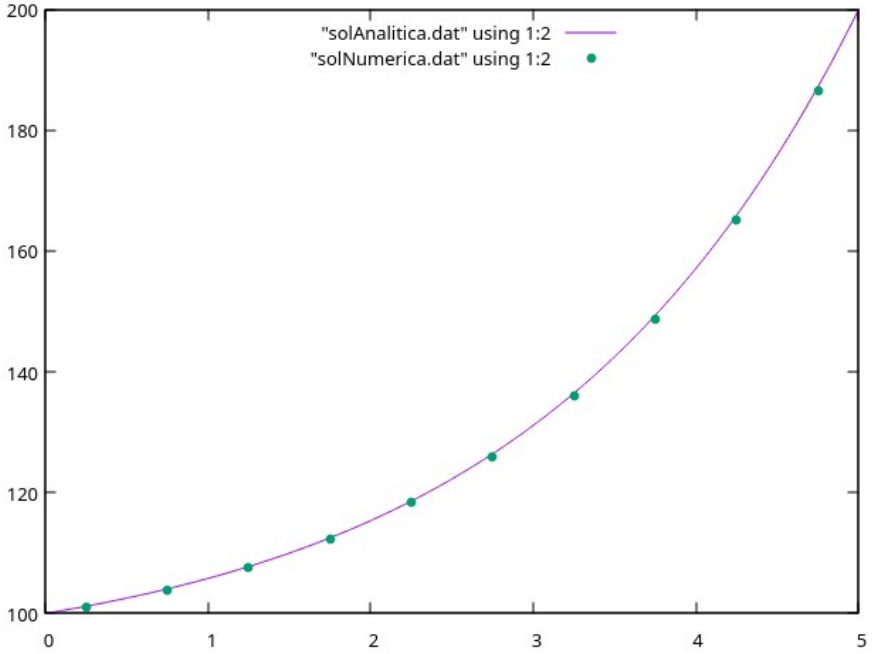
Nota-se primeiramente que há perfeita aderência entre a solução numérica e a solução analítica. Observa-se também que os gráficos são exatamente iguais, porém gerados por

métodos numéricos diferentes. O método Gauss-Siedel levou 78 iterações para obter a solução, enquanto o método do gradiente conjugado levou 10 iterações.

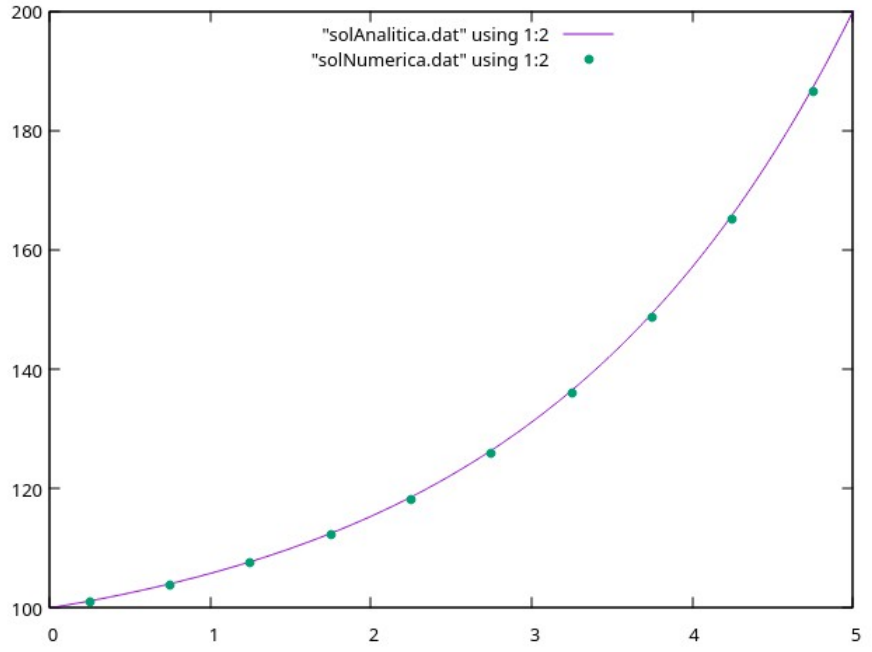
Repete-se o experimento, porém agora com novos dados de entrada. Desta vez,o `fvTestCase` foi usado para gerar um caso discretizado com 10 elementos (volumes de controle), com velocidade u = 0,05 m/s e sem geração interna de calor. Como trata-se de um caso com adveção, a matriz de coeficientes será assimétrica. Nesse caso, pode-se utilizar os métodos de solução de Gauss-Siedel e Gradiente Bi-conjugado. (Lembrando que gradiente conjugado não funciona para matrizes assimétricas).

```
edw@81:~/Documents/fvMatrix/bin$ ./fvTestCase 10 0.05 0

---------------------------------- fvMatrix ----------------------------------
-------------------- Gerador de Sistemas de Equações FVM --------------------
---------------------- Duto de ar / Regime estacinário ----------------------

N = 10 elementos
u = 0.05 m/s
qdot = 0 W/m3

Arquivo r.dat (coordenadas dos centróides) criado/atualizado com sucesso.
Arquivo A.dat (coeficientes) criado/atualizado com sucesso.
Arquivo B.dat (termos fonte) criado/atualizado com sucesso.
Arquivo X0.dat (estimativa inicial) criado/atualizado com sucesso.
Arquivo solAnalitica.dat (solução analítica) criado/atualizado com sucesso.
edw@81:~/Documents/fvMatrix/bin$ ./gaussSiedel

------------------------------ Método Gauss-Siedel ------------------------------

lambda (default) = 1
Dados carregados......[OK]
Dados consistentes....[OK]

Solução do sistema:
Norma do resíduo = 0.000928905
Número de iterações = 61
Arquivo X.dat (solução do sistema) criado/atualizado com sucesso.
Arquivo solNumerica.dat (r:posição, X:solução do sistema) criado/atualizado com sucesso.

edw@81:~/Documents/fvMatrix/bin$ ./gradBiConjugado

------------------------------ Método Gradiente Biconjugado  ------------------------------
Dados carregados......[OK]
Dados consistentes....[OK]

Solução do sistema de equações:
Norma do resíduo = 1.90424e-15
Número de iterações = 10
Arquivo X.dat (solução do sistema) criado/atualizado com sucesso.
Arquivo solNumerica.dat (r:posição, X:solução do sistema) criado/atualizado com sucesso.

edw@81:~/Documents/fvMatrix/bin$
```

Solução obtida pelo método de Gauss-Siedel, usando o programa `gaussSiedel.`



Solução obtida pelo método do gradiente Bi-conjugado, usando o programa `gradBiConjugado.`



Novamente, nota-se que há perfeita aderência entre a solução numérica e a solução analítica, e que os gráficos são exatamente iguais, porém gerados por métodos numéricos diferentes. O método Gauss-Siedel levou 61 iterações para obter a solução, enquanto o método do gradiente Bi-conjugado levou 10 iterações.

Por fim, vale mencionar que a diferença de performace entre os métodos de Gauss-Siedel e Gradientes fica bem mais evidente, quanto maior forem os sistemas de equações envolvidos. O print abaixo mostra um caso em que o `fvTestCase` foi usado para gerar uma discretização com 100 elementos (volumes de controle), com velocidade u = 0 m/s e geração interna de calor de 1.000 W/m^2. O método Gauss-Siedel levou 6.239 iterações para obter a solução, enquanto o método do gradiente conjugado levou somente 99 iterações.

```
edw@81:~/Documents/fvMatrix/bin$ ./fvTestCase 100 0 1000

---------------------------------- fvMatrix ----------------------------------
------------------- Gerador de Sistemas de Equações FVM --------------------
---------------------- Duto de ar / Regime estacinário -----------------------

N = 100 elementos
u = 0 m/s
qdot = 1000 W/m3

Arquivo r.dat (coordenadas dos centróides) criado/atualizado com sucesso.
Arquivo A.dat (coeficientes) criado/atualizado com sucesso.
Arquivo B.dat (termos fonte) criado/atualizado com sucesso.
Arquivo X0.dat (estimativa inicial) criado/atualizado com sucesso.
Arquivo solAnalitica.dat (solução analítica) criado/atualizado com sucesso.
edw@81:~/Documents/fvMatrix/bin$ ./gaussSiedel

------------------------------ Método Gauss-Siedel -----------------------------

lambda (default) = 1
Dados carregados......[OK]
Dados consistentes....[OK]

Solução do sistema:
Norma do resíduo = 0.000999237
Número de iterações = 6239
Arquivo X.dat (solução do sistema) criado/atualizado com sucesso.
Arquivo solNumerica.dat (r:posição, X:solução do sistema) criado/atualizado com sucesso.

edw@81:~/Documents/fvMatrix/bin$ ./gradConjugado

------------------------------ Método Gradiente Conjugado  -----------------------------
Dados carregados......[OK]
Dados consistentes....[OK]

Solução do sistema de equações:
Norma do resíduo = 0.000745356
Número de iterações = 99
Arquivo X.dat (solução do sistema) criado/atualizado com sucesso.
Arquivo solNumerica.dat (r:posição, X:solução do sistema) criado/atualizado com sucesso.

edw@81:~/Documents/fvMatrix/bin$
```

Outras conclusões podem ser tiradas, ampliando-se a analise dos resultados obtidos variando-se os dados de entrada. Encontra-se, inclusive, os limites dos métodos gradientes. A principal conclusão que se chega é que não existe um método numérico perfeito, que resolve todos os problemas. O que existe é o método numérico mais adequado para resolver cada um dos diversos tipos de problemas existentes. Na maior parte dos casos, descobre-se o método certo somente por tentativa e erro. Motivo pelo qual é tão importante dominar vários métodos numéricos.

Referências bibliográficas

[1] GONTIJO Rafael G. / Apostilas e lousas do curso: Do Cálculo à Simulação Computacional – Fundamentos de Métodos Numéricos com Aplicações – L2C, Turma 2025/2

[2] GREENSHIELDS Christopher; WELLER Henry / Notes on Computational Fluid Dynamics: General Principles – CFD Direct Ltd, 2022, 291p.
Disponível também em https://doc.cfd.direct/notes/cfd-general-principles/

[3] MOUKALLED F.; MANGANI L.; DARWISH M. / The Finite Volume Method in Computational Fluid Dynamics – Ed. Springer, 2016, 791p.

[4] BARRETT Richard *et al*. / Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods – 2ª ed – http://www.netlib.org, 107p.

[5] SHEWCHUK Jonathan R. / An Introduction to the Conjugate Gradient Method Without the Agonizing Pain – 1.1/4ª ed – Carnegie Mellon University, 1994, 58p.