Universidade Federal Rural do Semiárido – UFERSA
Centro de Engenharias – CE
Departamento de Engenharia e Tecnologia – DET
Curso de Engenharia Mecânica

Artigo

Aplicação da PETSc em um código computacional de Dinâmica dos Fluidos Computacional

Lucas de Oliveira Gomes [1], Victor Wagner Freire de Azevedo [2]

[1] Universidade Federal Rural do Semi-Árido - UFERSA, Engenharia Mecânica; Lucas de Oliveira Gomes; lucasoliveira_gomes@hotmail.com
[2] Universidade Federal Rural do Semi-Árido - UFERSA, Engenharia Mecânica; Victor Wagner Freire de Azevedo; victorwfreire@ufersa.edu.br

Data da defesa: 18/04/2024;

Resumo: Com o avanço da tecnologia e também da indústria 4.0, a utilização de sistemas computacionais para resolução de problemas de engenharia se torna cada vez mais necessária. Nesse cenário, a Dinâmica dos Fluidos Computacional figura como uma forma rápida e prática de resolver problemas ligados ao escoamento de fluidos. A utilização de métodos que tornem a solução numérica eficiente é sempre útil e está em constante desenvolvimento. A biblioteca PETSc é uma ferramenta disponível que vem sendo utilizada por muitos pacotes comerciais e códigos “in-house”. Dessa forma, o presente trabalho irá desenvolveu uma aplicação para a PETSc em um código capaz de resolver problemas de dinâmica dos fluidos computacional, mais especificadamente um problema de transferência de calor simples, cujo resultado foi comparado ao resultado obtido pelo método S.O.R, sendo possível visualizar a diferença no tempo que a PETSc leva para executar as iterações. O código desenvolvido será utilizado como base para o desenvolvimento de futuros códigos analisando problemas mais complexos.

Palavras-chave: Python; PETSc; Método dos Volumes Finitos

.

1. INTRODUÇÃO

As ferramentas computacionais para resolução de problemas de engenharia vêm em constante evolução. Códigos computacionais tanto "in-house" quanto comerciais são utilizados em diferentes soluções na indústria. Pelo volume e facilidade de acesso às informações, os códigos "in-house" e até os abertos estão ganhando bastante espaço na solução de problemas, uma vez que as licenças de softwares comerciais possuem custo bastante elevado, dificultando o acesso para pequenas ou médias empresas na indústria, por exemplo. Várias bibliotecas foram implementadas de forma que é possível aplicá-las em diferentes códigos computacionais.

Nesse contexto, destaca-se a biblioteca "Portable Extensible Toolkit for Scientific Computation", a PETSc [1], que foi desenvolvida pela “Argonne National Laboratory”. A PETSc foi feita com o propósito de auxiliar na resolução de problemas de simulação ao ser aplicada em projetos científicos em larga escala, portanto, inclui em suas bibliotecas uma diversidade de ferramentas para resolução de problemas lineares e não lineares que podem ser facilmente implementadas em várias linguagens de programação, tais como C, C++, Fortran e Python.

Dentre os solucionadores disponíveis pode ser citado os baseados no método Krylow e também o SNES (“Scalable Nonlinear Equations Solver”) que é um solucionador não-linear. Além disso, a PETSc segue em constante desenvolvimento e tornando possível sua implementação em novas linguagens de programação, em que se destaca o Python. A PETSc também está sendo utilizada em diversos códigos computacionais aliada

aos conceitos de paralelização [3], que não será abordada na implementação proposta no presente artigo mas é um tema bastante em voga na computação de alta performance, o que torna a solução de casos complexos mais eficiente.

Através da "Message Passing Interface", a MPI [2,3], os códigos computacionais dividem os processos em diferentes processadores, de modo que há comunicação entre eles, permitindo a troca de informação e a execução de funções. Após executar todas as funções, a MPI permite a união das soluções em um único conjunto, através de entidades comunicadoras, deste modo facilitando e agilizando a execução de diversas tarefas em paralelo.

Dessa forma, um código computacional baseado no método dos volumes finitos foi implementado no presente trabalho utilizando a PETSc na linguagem de programação Python, com o intuito de testar sua eficiência em relação à métodos convencionais na solução de um problema de transferência de calor em 2D simples. A eficiência na solução desta biblioteca foi analisada em relação a um código similar também implementado na mesma linguagem de programação, permitindo visualizar a grande contribuição da biblioteca na solução de problemas de dinâmica dos fluidos computacional.

Adicionalmente, o desenvolvimento do presente trabalho servirá como primeiro passo para construir a base de implementação da PETSc em um código numérico em linguagem de programação Python para futuras aplicações em processos mais complexos, como por exemplo o estudo da transferência de calor em um objeto levando em consideração variáveis como a difusividade e a condutividade térmica.

2. MATERIAIS E MÉTODOS

Mesmo não sendo a linguagem de programação com maior velocidade, o Python possui vasta aplicação no dia-a-dia e possui espaço para muitas implementações [3,4], o que inclui o MPI e a PETSc, além de ser relativamente mais simples que as demais linguagens de programação, facilitando seu entendimento.

A utilização da PETSc envolverá a aplicação da sintaxe da biblioteca para a escrita de matrizes e vetores, utilização de solucionadores de sistemas lineares e não-lineares e de pré-condicionadores para otimizar a solução.

A "petsc4py" [4] e a "mpi4py" [3] são bibliotecas que já estão implementadas no Python e que podem ser aplicadas em diferentes soluções. Dessa forma, um código computacional que aplica o Método dos Volumes Finitos em uma placa plana bidimensional será o objeto de estudo para aplicação dessas ferramentas.

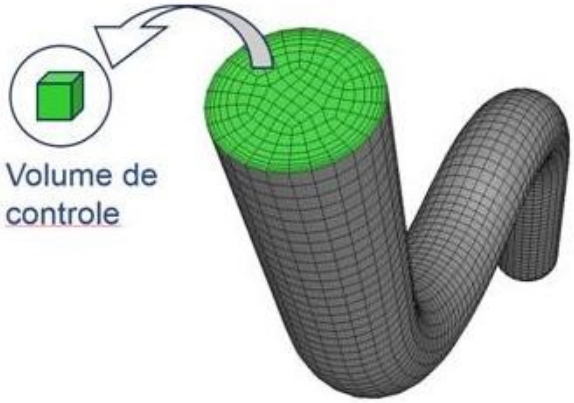
2.1 O Método dos Volumes Finitos

Quando se trata de simulações computacionais, o Método dos Volumes Finitos (MVF) [5] é bastante utilizado para verificar se algum objeto ou projeto foi devidamente projetado, esse método é eficaz, pois permite que os responsáveis por um determinado projeto possam realizar simulações de forma rápida, que consumiria muito mais tempo caso fosse feita de forma manual, por isso, reduz os custos dos projetos, tornando-os mais viáveis economicamente.

Em uma simulação de algum fenômeno real tal como a transferência de calor que ocorre em uma placa metálica ou o escoamento de um fluido em uma tubulação se torna difícil analisa-lo como um todo, pois muitas vezes a geometria de um determinado corpo é muito complexa fazendo com que este apresente diferentes comportamentos, portanto, é mais viável fazer o uso de modelos numéricos, dos quais permitem a aplicação de métodos como o MVF.

A forma como o MVF auxilia a resolver um determinado problema é de certa forma simples, basicamente este método realiza uma "quebra" do objeto que se deseja analisar, ou seja, um objeto de formato complexo é dividido em vários pequenos volumes que possuem geometria conhecida da seguinte forma:

Figura 1: Corpo subdividido em diversos volumes

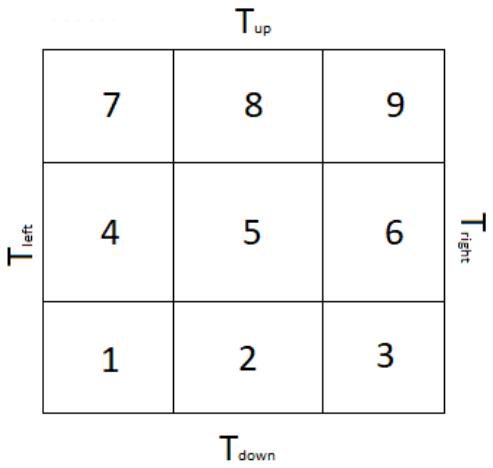


Fonte: Adaptado de Zanin et al. [6]

Observando a Figura 1, é possível visualizar que desta forma fica mais fácil fazer os cálculos, uma vez que a partir dessa "quebra" o próximo passo é o de analisar cada um desses pequenos volumes gerados e realizar os cálculos em cada um deles, verificando como cada um se comporta sob determinadas condições de contorno, após isso, para ter um resultado mais concreto de modo a visualizar o objeto como um todo, é possível juntar todos os resultados das simulações desses pequenos volumes e assim ter um resultado que consiga traduzir o que está acontecendo no objeto como um todo.

No presente artigo, um caso mais simples será abordado, de transferência de calor em uma placa bidimensional, portanto, utilizando o Método dos Volumes Finitos, a placa bidimensional pode ser dividida em diferentes zonas de interesse (ver Fig. 2). Note que temperaturas prescritas estão sendo consideradas nas faces do domínio computacional.

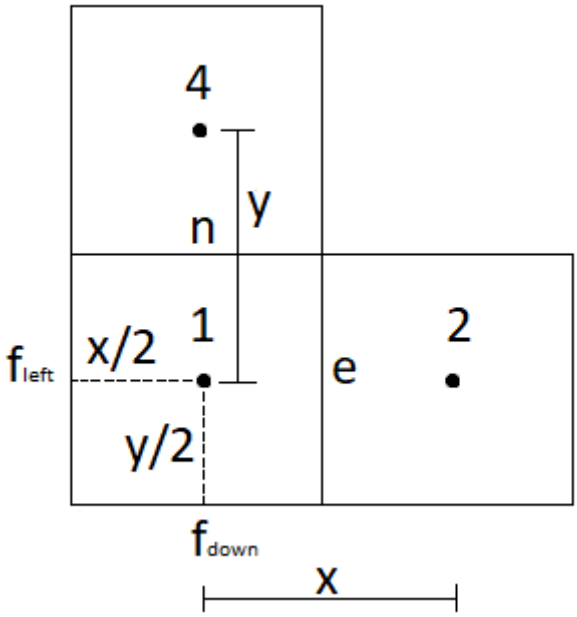
Figura 2: Placa bidimensional



Fonte: Autoria própria

Com o volume de controle apresentado na Fig. 2, pode ser verificado que a placa está sujeita a diferentes condições de contorno, isto é, terá temperaturas diferentes para cada lado, fazendo com que cada pequeno volume de controle gerado pelo método dos Volumes Finitos esteja sujeito a condições diferentes, porém todos podem ser determinados da mesma forma. Tomando como exemplo o Volume 1, a análise pode ser realizada da seguinte forma:

Figura 3: Análise do ponto 1 da placa



Fonte: Autoria própria

Primeiramente, sabendo que a equação da condução de calor 2D em regime permanente e puramente difusiva é dada por:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

e também que a equação que rege as temperaturas em todas as direções (conforme mostra a Fig. 3) é dada por:

$$A_p T_p = A_e T_e + A_w T_w + A_s T_s + A_n T_n + B \tag{2}$$

podemos resolver a equação para obter quais funções definem os A_p, A_e, o vetor B e todos os demais termos através da integração da Eq. 1 nos eixos X e Y:

$$\iint_{f_{down} f_{left}}^{n\ e} \frac{\partial^2 T}{\partial x^2} + \iint_{y\ x} \frac{\partial^2 T}{\partial y^2} = 0 \tag{3}$$

obtendo:

$$\left(\frac{T_e - T_p}{\Delta x} - \frac{T_p - T_{left}}{\frac{\Delta x}{2}}\right)\Delta y + \left(\frac{T_n - T_p}{\Delta y} - \frac{T_p - T_{down}}{\frac{\Delta y}{2}}\right)\Delta x = 0 \tag{4}$$

resolvendo a equação acima temos:

$$-3T_p\frac{\Delta y}{\Delta x} + T_e\frac{\Delta y}{\Delta x} + 2T_{left}\frac{\Delta y}{\Delta x} + T_n\frac{\Delta x}{\Delta y} - 3T_p\frac{\Delta x}{\Delta y} + 2T_{down}\frac{\Delta x}{\Delta y} = 0 \tag{5}$$

agora, podemos isolar todos os termos que multiplicam T_p para o outro lado da igualdade e assim, obtermos o resultado final, da seguinte forma:

$$\left(3\frac{\Delta y}{\Delta x}+3\frac{\Delta x}{\Delta y}\right)T_p=\frac{\Delta y}{\Delta x}T_e+\frac{\Delta x}{\Delta y}T_n+2\frac{\Delta y}{\Delta x}T_{left}+2\frac{\Delta x}{\Delta y}T_{down} \quad (6)$$

Ao comparar a equação (6) com a equação (2), podemos ver as funções que representam A_p, A_e, A_n e o vetor B, que são dados por:

$$A_p=\left(3\frac{\Delta y}{\Delta x}+3\frac{\Delta x}{\Delta y}\right) \quad (7)$$

$$A_e=\frac{\Delta y}{\Delta x} \quad (8)$$

$$A_n=\frac{\Delta x}{\Delta y} \quad (9)$$

$$B=2\frac{\Delta y}{\Delta x}T_{left}+2\frac{\Delta x}{\Delta y}T_{down} \quad (10)$$

A operação que foi realizada acima foi a discretização da equação governante que rege a transferência de calor por difusão em cada ponto do volume de controle, todos os outros volumes podem ser modelados de forma análoga ao que foi feito com o Volume 1 (para maiores detalhes sobre o processo de discretização em placas bidimensionais o leitor é convidado a consultar os trabalhos de Júnior & de Azevedo [7] e da Silva & de Azevedo [8]). Com os resultados de todos os pontos obtidos é possível adicionar a matriz de coeficientes ao código computacional, tal matriz terá o seguinte formato:

$$\begin{bmatrix}
A_p & A_e & 0 & A_n & 0 & 0 & 0 & 0 & 0 \\
A_w & A_p & A_e & 0 & A_n & 0 & 0 & 0 & 0 \\
0 & A_w & A_p & 0 & 0 & A_n & 0 & 0 & 0 \\
A_s & 0 & 0 & A_p & A_e & 0 & A_n & 0 & 0 \\
0 & A_s & 0 & A_w & A_p & A_e & 0 & A_n & 0 \\
0 & 0 & A_s & 0 & A_w & A_p & 0 & 0 & A_n \\
0 & 0 & 0 & A_s & 0 & 0 & A_p & A_e & 0 \\
0 & 0 & 0 & 0 & A_s & 0 & A_w & A_p & A_e \\
0 & 0 & 0 & 0 & 0 & A_s & 0 & A_w & A_p
\end{bmatrix}$$

Os valores dos A_p, A_e, A_w, A_n, A_s vão depender das condições de contorno fornecidas, que no presente caso, são temperaturas prescritas nas paredes do domínio. Com o código em Python juntamente com as funções da PETSc sendo capazes de fornecer a matriz com seus devidos valores, o próximo passo é o de escolher algum dos solucionadores fornecidos pela PETSc e obter o resultado.´

2.2 Métodos Numéricos baseados no Subespaço de Krylov

A aplicação do MVF gera um sistema de equações descrito por matrizes esparsas, ou seja, com muitos elementos zeros. Tradicionalmente, é comum resolver tais sistemas utilizando os chamados “solucionadores esparsos diretos”, onde se enquadram as variações dos métodos baseados na Eliminação de Gauss [7,9]. Atualmente, métodos iterativos baseados no subespaço de Krylov são utilizados na solução do sistema de equações, o que deixou inclusive métodos consolidados, como o S.O.R (Método das Sobrerrelaxações Sucessivvas), “ultrapassados” [9].

O subespaço de Krylov é definido como [10]

$$K_k(A, r_0) \equiv \{r_0, Ar_0, A^2r_0, A^3r_0, \dots, A^{k-1}r_0\} \tag{11}$$

Onde $K_k(A, r_0)$ é o subespaço de Krylov de dimensão k e é gerado por A e r_0, onde A é a matriz de coeficientes obtida através do MVF e r_0 o resíduo da solução

$$r_0 = b - A_{x_{(0)}} \tag{12}$$

Nesse aspecto, o objetivo desse método é encontrar uma aproximação x_k dentro do seu respectivo subespaço. O ideal é que a aproximação x_k seja ortogonal ao subespaço de Krylov $K_k(A, r_0)$ na qual x_k é escolhida. Para garantir que isso aconteça existe a condição de Galerkin, que procura fazer com que o resultado seja sempre ortogonal ao subespaço e é dada por:

$$b - A_{x_k} \perp K_k \tag{13}$$

Deste modo, note que em algum momento durante o processo iterativo o resultado será zero, resultando em $0 = b - A_{x_k}$, e esta é a solução desejada. A vantagem de utilizar o método do subespaço de Krylov é que o espaço de busca se torna menor do que um espaço de busca completo, deste modo, torna-se mais fácil determinar um vetor x_k satisfatório, ou seja, a obtenção da solução do problema [10].

2.3 Aspectos da Implementação do Código Computacional

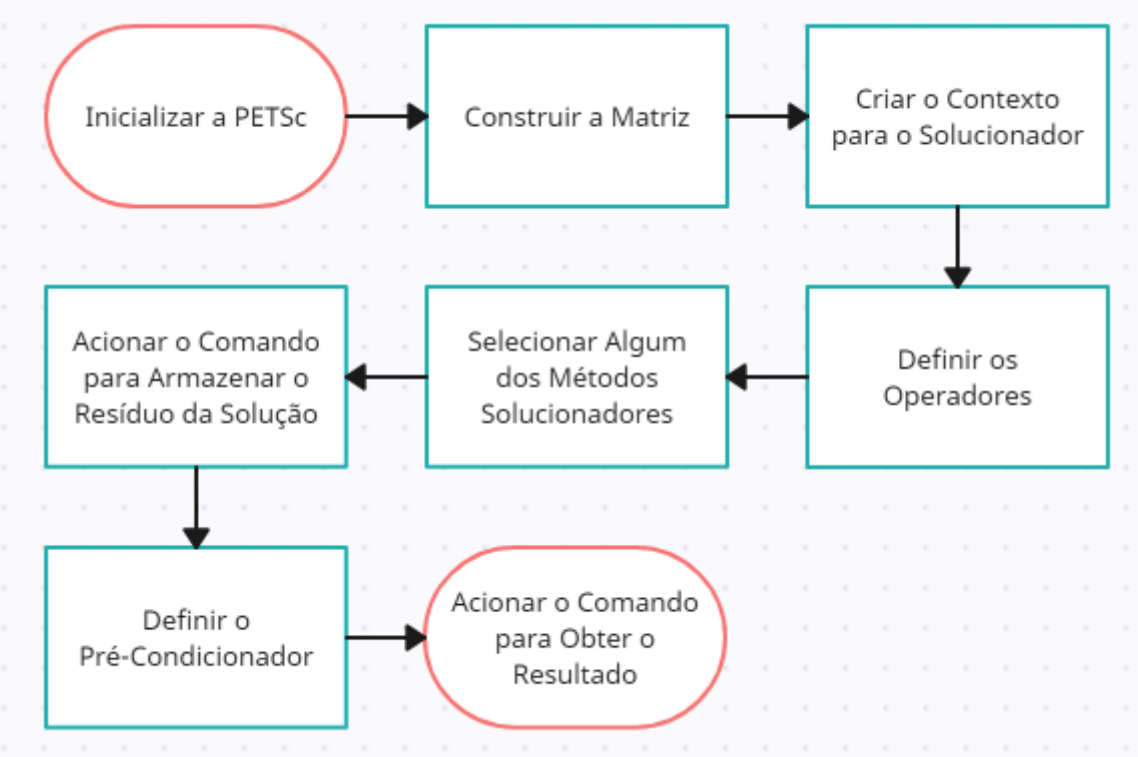
A presente seção tem por objetivo comentar os principais aspectos da implementação da PETSc em Python, de forma que um melhor entendimento do código possa ser alcançado em implementações futuras. Primeiramente, a biblioteca deve ser importada no código através do "import petsc4py" e logo em seguida inicializada através do "petsc4py.init(sys.argv)". Este último comando informa para o código que as funções relacionadas à PETSc devem ser carregadas no momento da execução.

O solucionador baseado no método do subespaço de Krylov deve ser carregado através de uma instância "ksp = PETSc.KSP().create()", que cria o objeto para utilização dentro do código. Cada termo mostrado na Eq. 11 é chamado de "operador", dessa forma, operadores devem ser criados também no código para a matriz de coeficientes através do "ksp.setOperators(A)", note que "A" é o nome da variável dada à matriz. Diversos solucionadores estão disponíveis dentro da PETSc baseados nos subespaço de Krylov [1], dos quais o BiCGSTAB ("Biconjugate Gradient Stabilized Method") é um dos mais utilizados atualmente. O solucionador pode ser especificado pelo comando "ksp.setType('bcgs')", para utilização do solucionador mencionado anteriormente.

Com essas definições, é possível resolver o sistema de equações lineares de maneira bem simples com o comando "ksp.solve(b, x)", em que "b" é o vetor contendo as condições de contorno e "x" o vetor de solução do problema.

De forma mais simplificada, o passo-a-passo para a resolução do problema proposto neste artigo e para a implementação da PETSc no código bem como o seu uso pode ser mais facilmente compreendido pelo fluxograma apresentado na Fig. 4.

Figura 4: Fluxograma de uso da PETSc para o problema proposto



Fonte: Autoria Própria

3. RESULTADOS

Para a análise do problema, foi considerada uma placa bidimensional com temperaturas prescritas nas faces do domínio, conforme descrito na Sec. 2.1. Foram consideradas temperaturas nulas nas faces esquerda, direita e inferior e, uma temperatura de 1.0 na face superior. A placa bidimensional possui comprimento igual a 1.0 nas direções x e y (horizontal e vertical, respectivamente). Todo o caso foi estudado de maneira unitária, cujos resultados foram comparados com um código similar cuja solução foi obtida através de métodos iterativos convencionais baseados na eliminação de Gauss , neste caso, o método S.O.R foi analisado [8]. Para verificar a eficácia de utilização da PETSc, foi adotado como parâmetro o tempo que o código levou para obtenção do resultado. Diferentes malhas computacionais foram analisadas e o código foi executado, tanto com a PETSc quanto com o método S.O.R, utilizando um único processador.

3.1. Resultados dos Códigos

Através da Tabela 1 pode ser visto que para matrizes pequenas o tempo que a PETSc leva para resolver o caso já é superior ao S.O.R, o que fica ainda mais evidente quando o tamanho do domínio computacional é aumentado.

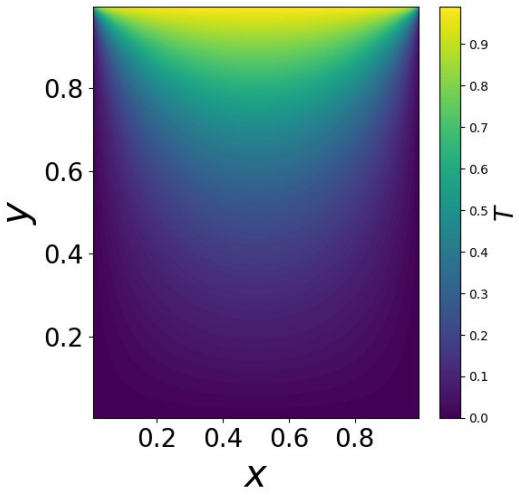
TABELA 1. Tempos de solução de matrizes de diferentes tamanhos com a utilização da PETSc e do S.O.R

Tamanho da Malha Computacional	**Tempo de Solução da PETSc (s)**	**Tempo de Solução do S.O.R (s)**	**Percentual de diferença do tempo de solução da S.O.R para a PETSc**
16	0,001097	0,0029	~200%
1600	0,01875	5,5998	~30.000%
10000	0,16086	202,57	~120.000%

3.1.1. Distribuição de Temperatura nas Placas

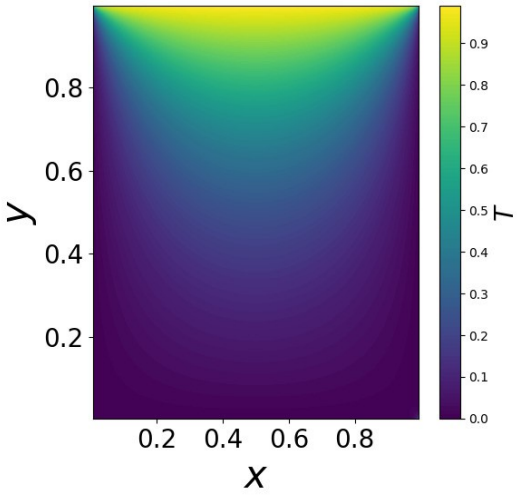
As Figs. 5 e 6 mostram a distribuição de temperatura obtida nas placas bidimensionais utilizando a PETSc (ver Fig. 5) e o S.O.R (ver Fig. 6) considerando a malha computacional formada por 10.000 volumes. Pode ser visto que ambos os casos forneceram soluções similares para a solução do problema.

Figura 5: Resultado obtido com a PETSc para uma malha computacional de tamanho 10000



Fonte: Autoria Própria

Figura 6: Resultado obtido pelo S.O.R para uma malha computacional de tamanho 10000



Fonte: da Silva & de Azevedo [8].

É claramente visível que ambos os códigos foram capazes de fornecer o mesmo resultado, a diferença estará no tempo em que cada um levou para fornecer o resultado, conforme apresentado na Tab. 1.

4. CONCLUSÃO

O presente artigo mostrou que a solução de um problema de transferência de calor bidimensional, considerando o MVF, pode ser otimizado através do uso de bibliotecas de computação de alta performance. A PETSc acelerou o processo de convergência da solução em relação aos métodos convencionais baseados na Eliminação de Gauss. Diante disso, apesar da dificuldade da implementação da PETSc na linguagem de programação Python, através da Tabela 1 é possível ver que os resultados se mostraram muito satisfatórios, visto que a PETSc mostrou ser muito superior aos métodos convencionais no que diz respeito ao tempo de

solução do problema, vale ressaltar que o problema proposto no presente artigo era de certo modo simples, apenas a transferência de calor em uma placa bidimensional, mas mesmo assim foi possível visualizar a diferença que a PETSc fez na redução do tempo de solução, além disso, a PETSc pode fazer grande diferença na melhoria de códigos computacionais para realização de simulações e com isso facilitar o desenvolvimento de projetos de engenharia.

5. REFERÊNCIAS

[1] BALAY, S. et al. **PETSc/TAO Users Manual (Rev. 3.20)**. Argonne National Laboratory (ANL), Argonne, IL (United States), 2023.

[2] WALKER, David W.; DONGARRA, Jack J. MPI: a standard message passing interface. Supercomputer, v. 12, p. 56-68, 1996.

[3] DALCÍN, Lisandro; PAZ, Rodrigo; STORTI, Mario. MPI for Python. Journal of Parallel and Distributed Computing, v. 65, n. 9, p. 1108-1115, 2005.

[4]HAM, D. et al. firedrakeproject/petsc4py: The Python interface to PETSc.

[5] MALISKA, Clovis R. **Fundamentals of computational fluid dynamics: the finite volume method**. Springer Nature, 2023.

[6] ZANIN, Massimiliano et al. An early stage researcher's primer on systems medicine terminology. **Network and systems medicine**, v. 4, n. 1, p. 2-50, 2021.

[7] Júnior & de Azevedo. Análise da transferência de calor em uma placa utilizando o Método dos Volumes Finitos. UFERSA, 2023.

[8] da Silva & de Azevedo. Implementação de gerador de malha computacional para o caso de difusão pura de calor em uma placa bidimensional. UFERSA, 2023.

[9] GUTKNECHT, Martin H. A brief introduction to Krylov space methods for solving linear systems. In: **Frontiers of Computational Science: Proceedings of the International Symposium on Frontiers of Computational Science 2005**. Springer Berlin Heidelberg, 2007. p. 53-62.

[10] SAAD, Youcef. **Overview of Krylov subspace methods with applications to control problems**. 1989.