UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Ciência da Computação

Conceitos de Linguagens de Programação: Fortran & Python

Implementação com Visualização Gráfica e Duas Linguagens de Programação

Pedro Henrique dos Santos Pinto
Fabrício Barbosa Viegas

Pelotas
15 de agosto de 2025

SUMÁRIO

Sumário 2
1 Introdução 3
2 Descrição da Aplicação 3
3 Integração entre as Linguagens 3
4 Implementação 4
4.1 Código Fortran 4
4.2 Interface Python 5
5 Execução 7
6 Considerações Finais 7

REFERÊNCIAS 8

1 INTRODUÇÃO

Este trabalho implementa a integração entre um solver numérico para equação do calor em regime permanente, desenvolvido em Fortran, e uma interface gráfica em Python. A interoperabilidade entre as linguagens foi realizada utilizando `f2py` [The NumPy Developers], ferramenta do ecossistema NumPy para integração com Fortran.

2 DESCRIÇÃO DA APLICAÇÃO

O problema resolvido consiste na simulação da distribuição estacionária de temperatura em um material homogêneo bidimensional, sujeito a condições de contorno de Dirichlet. Cada borda (superior, inferior, esquerda e direita) possui temperatura fixa, e o módulo Fortran `heated_plate_mod.f90` implementa um método iterativo para solução do sistema até a convergência ($\Delta T < \text{tol}$).

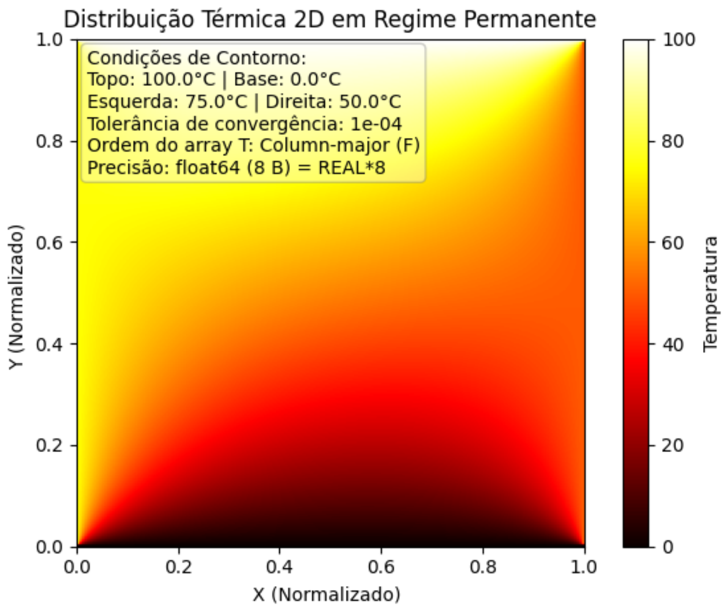

Figura 1 – Distribuição de temperatura em regime permanente. As condições de contorno e detalhes técnicos da implementação são exibidos na anotação.

3 INTEGRAÇÃO ENTRE AS LINGUAGENS

A interface entre Python e Fortran foi estabelecida através do `f2py`, que gera automaticamente wrappers para chamadas Fortran a partir de Python. O processo de compilação produz um módulo Python importável (`.so`), conforme evidenciado no corte abaixo da saída do `f2py`:

```
Building module "heated_plate"...
    Constructing wrapper function "heated_plate_mod.heated_plate_solver"...
      t = heated_plate_solver(m,n,top,bottom,left,right,tol)
```

Notavelmente, o `f2py` realiza transformações semânticas significativas:

- Converte a subrotina Fortran (com parâmetro `intent(out)`) para uma função Python que retorna valores
- Preserva a ordem column-major dos arrays Fortran (`order='F'`) sem conversão implícita
- Mantém a precisão numérica equivalente (`REAL*8` →`float64`)

4 IMPLEMENTAÇÃO

4.1 CÓDIGO FORTRAN

O solver implementa um método iterativo baseado em diferenças finitas:

Listing 1 – Implementação do solver térmico em `heated_plate_mod.f90`

```
subroutine heated_plate_solver(m, n, top, bottom, left, right, tol,
    T)

  integer ( kind = 4 ), intent(in) :: m, n
  real( kind = 8 ), intent(in) :: top, bottom, left, right
  real( kind = 8 ), intent(in) :: tol
  real( kind = 8 ), intent(out) :: T(m, n)

  real( kind = 8 ) :: diff
  real( kind = 8 ) :: mean
  real( kind = 8 ), allocatable :: u(:,:)
  integer ( kind = 4 ) :: i, j

  allocate(u(m, n))

  !  Set the boundary values, which don't change.
  T(2:m-1,1) = left
  T(2:m-1,n) = right
  T(1,1:n) = top
  T(m,1:n) = bottom

  !  Average the boundary values, to come up with a reasonable
```

```
  !  initial value for the interior.
  mean = ( &
      sum ( T(2:m-1,1) ) &
    + sum ( T(2:m-1,n) ) &
    + sum ( T(1,1:n)   ) &
    + sum ( T(m,1:n)   ) ) &
    / real ( 2 * m + 2 * n - 4, kind = 8 )

  !  Initialize the interior solution to the mean value.
  T(2:m-1,2:n-1) = mean

  !  Iterate until the new solution T differs from the old solution
      U
  !  by no more than TOL.
  diff = tol + 1.0
  do while (diff > tol)
    u = T  ! Save previous solution

    ! Update solution (serial)
    do j = 2, n-1
      do i = 2, m-1
        T(i,j) = 0.25 * (u(i-1,j) + u(i+1,j) + u(i,j-1) + u(i,j+1))
      end do
    end do

    ! Check convergence
    diff = maxval(abs(T - u))
  end do

  deallocate(u)
end subroutine heated_plate_solver
```

Principais características:

- Alocação dinâmica do array de trabalho `u(m,n)`
- Critério de convergência baseado na tolerância `tol`
- Uso de `intent(out)` para o array de resultados `T`

4.2 INTERFACE PYTHON

O código Python demonstra a integração:

Listing 2 – Caso de estudo em study_case.py

```
import numpy as np
import matplotlib.pyplot as plt
from heated_plate import heated_plate_mod

# Parâmetros da simulação:
m, n = 200, 200           # Número de pontos na grade (x, y)
top = 100.0               # Temperatura no topo (condição de contorno
    )
bottom = 0.0              # Temperatura na base
left = 75.0               # Temperatura à esquerda
right = 50.0              # Temperatura à direita
tol = 1e-4                # Tolerância para convergência do solver

# A rotina `heated_plate_solver`, implementada em Fortran, retorna a
    solução convergida.
T = heated_plate_mod.heated_plate_solver(m, n, top, bottom, left,
    right, tol)

plt.imshow(T, cmap='hot', extent=[0,1,0,1], origin='upper')
plt.colorbar(label='Temperatura')
plt.title('Distribuição Térmica 2D em Regime Permanente')
plt.xlabel('X (Normalizado) ')
plt.ylabel('Y (Normalizado) ')

annotation_text = (
    f"Condições de Contorno:\n"
    f"Topo: {top}°C | Base: {bottom}°C\n"
    f"Esquerda: {left}°C | Direita: {right}°C\n"
    f"Tolerância de convergência: {tol:.0e}\n"
    f"Ordem do array T: {'Column-major (F)' if T.flags.f_contiguous
        else 'Row-major (C)'}\n"
    f"Precisão: {T.dtype} ({T.dtype.itemsize} B) = REAL*8"
)

plt.annotate(
    annotation_text,
    xy=(0.02, 0.98),
    xycoords='axes fraction',
    ha='left',
    va='top',
    bbox=dict(
```

```
            boxstyle='round',
            alpha=0.4,
            facecolor='white',
            edgecolor='gray'
        )
)

plt.tight_layout()
plt.show()
```

Aspectos relevantes:

- Chamada direta ao solver Fortran sem pré-alocação explícita
- Verificação explícita da ordem de memória (`.flags.f_contiguous`)
- Visualização com anotações técnicas incorporadas

5 EXECUÇÃO

As instruções completas de compilação e execução estão disponíveis no arquivo `README.md` do repositório [Pinto e Viega 2025]. Para conveniência, os comandos básicos são:

```
git clone https://github.com/santosphp/2d-steady-state-thermal-solver.git
cd 2d-steady-state-thermal-solver
make       # Compila o módulo Fortran
make run   # Executa a interface gráfica
make study # Roda o caso de estudo pré-configurado
```

6 CONSIDERAÇÕES FINAIS

A abordagem adotada demonstra eficientemente a complementaridade entre linguagens:

- Fortran para computação numérica de alto desempenho
- Python para interface gráfica e pós-processamento

A solução elimina a necessidade de arquivos intermediários (como em abordagens baseadas em gnuplots [Janert 2016]), proporcionando integração direta entre os componentes.

REFERÊNCIAS

[Burkardt 2010]BURKARDT, J. *heated_plate_workshare.f90: Solution of the heat equation on a rectangular plate using OpenMP*. 2010. Disponível em: `https://people.math.sc.edu/Burkardt/f_src/heated_plate_workshare/heated_plate_workshare.html`. Acessado em: 9 ago. 2025.

[Erica]ERICA. *Iterative Solution of the Poisson Equation*. Acessado em: 8 ago. 2025. Disponível em: <https://bluehound2.circ.rochester.edu/astrobear/raw-attachment/wiki/u/erica/PoissonSolver/prjpoisson.pdf>.

[Janert 2016]JANERT, P. K. *Gnuplot in action: understanding data with graphs*. [S.l.]: Simon and Schuster, 2016.

[Pinto e Viega 2025]PINTO, P. H. S.; VIEGA, F. B. *2D Steady-State Thermal Solver*. 2025. `https://github.com/santosphp/2d-steady-state-thermal-solver`. Acesso em: 10 ago. 2025.

[Rickman 2024]RICKMAN, S. L. *Introduction to Numerical Methods in Heat Transfer*. [S.l.], 2024. Acessado em: 8 ago. 2025. Disponível em: <https://ntrs.nasa.gov/citations/20200006182>.

[The NumPy Developers]The NumPy Developers. *f2py - Fortran to Python interface generator*. [S.l.]. Acessado em: 6 ago. 2025. Disponível em: <https://numpy.org/doc/stable/f2py/>.