# Projeto 3 de Física Computacional

Pedro Bicudo e Nuno Cardoso , IST,

18 de Novembro de 2017

**Resumo das instruções**

Este projeto deve ser submetido como um único ficheiro comprimido, na página pessoal de aluno no fénix:

- no menu superior devem selecionar estudante
- no menu lateral devem selecionar submeter, projetos;
- deve surgir o grupo desta disciplina de FC, e lá podem submeter projetos e visualizar os projetos submetidos.

Este ficheiro comprimido, por exemplo .zip, deve incluir os vários ficheiros criados,

- usando as extensões .cpp e .h para os códigos de g++,
- .nb para os códigos de mathematica (a enviar sem figuras),
- resultados na forma de gráficos em formato .pdf com labels,
- a memória descritiva em formato .txt (ver parágrafo seguinte).

para diminuir o número de ficheiros, não é necessário enviar os ficheiros de dados produzidos (ou de texto) .txt. Não devem ser incluídos os ficheiros executáveis.

É importante que enviem uma pequena memória descritiva, em texto simples, chamado `readme_g#p#.txt`, que inclua apenas

- o número e nome dos alunos do grupo
- o número do grupo
- qualquer comentário, observação ou análise pedidos no enunciado,
- a lista de ficheiros submetidos, e para cada ficheiro algumas palavras a descrever o seu objectivo e indicando nos `.cpp` como compilar, executar, etc
- e finalmente, apenas caso hajam vários ficheiros a linkar em conjunto (por exemplo várias funções e um header), a linha de comandos para os linkar, por exemplo algo do tipo g++ -o g09p2c1.o g09p2c1.cpp g09p2c2.cpp g09p2c3.cpp, ou com mais linhas.

Para sua própria organização os alunos/grupos devem designar os códigos que forem construindo por g#p3c#.cpp devendo substituir os # por números, sendo o primeiro # o número do seu grupo, o 3 refere-se ao projeto 3 e o segundo # o número do código (ex: g32p3c2.cpp será o segundo código que o grupo 32 realiza para este projeto. Deve usar uma designação semelhante para os outros ficheiros com outras extensões. O próprio ficheiro zip deve ser denominado g#p3.zip .

Serão avaliados,

- o valor dos resultados (os plots .pdf e os comentários no readme_g#p#.txt),
- a clareza dos resultados (de novo na forma de plots e comentários),
- se o código compila,
- a ausência de erros e de fugas de memória nos executáveis,
- a simplicidade e economia de recursos computacionais do código,
- a indentação e os comentários que ajudem ao entendimento e uso do código.

# 1 Resolver a eq. de Fourier com arrays

Um objetivo deste projeto *(que pode desenvolver* **em paralelo** *com o da secção 2)* é resolver a eq. de Fourier a 2 dimensões espaciais,

$$T_t = k\,(T_{xx} + T_{yy})\ ,$$

onde k é a difusibilidade térmica. Temos uma placa bi-dimensional retangular cuja temperatura $T(t,x,y)$ depende do tempo $t$ e do espaço $x,y$. Para começar, aqui vamos usar a programação mais simples, com `arrays` (ou matrizes) apenas, ainda sem ser obrigatório (apenas facultativo) usar classes ou funções.
Vamos trabalhar com arrays com `nx*ny` elementos, que assim chamamos para não confundir com o tempo `t` . Use memória **dinâmica** para que `nx` e `ny` possam ser definido apenas na execução. A cada passo temos um array dinâmico inicial tt0 e com ele pretendemos calcular para o passo seguinte `tt1` . Este processo é semelhante ao das eq. diferenciais ordinárias, mas agora atualizamos um `array` e não apenas um número.

## 1.a

(2pt)

Usando diferenças finitas substituímos, as derivadas parciais no tempo por (aproximação de Euler retardada)
$T_t \simeq \frac{[T(t+h_t,x,y)-T(h,x,y)]}{h_t} \rightarrow$ `(tt1[i][j]-tt0[i][j])/ht` ,
e as derivadas parciais no espaço por (segunda derivada centrada)
$T_{xx} \simeq \frac{[T(t,x+h_x,y)-2T(t,x,y)+T(t,x-h_x,y)]}{{h_x}^2} \rightarrow$ `(tt0[i+1][j]-2*tt0[i][j]+tt0[i-1][j])/(hx*hx)`
,
$T_{yy} \rightarrow \frac{[T(t,x,y+h_x)-2T(t,x,y)+T(t,x,y-h_x)]}{{h_x}^2} \rightarrow$ `(tt0[i][j+1]-2*tt0[i][j]+tt0[i][j-1])/(hx*hx)`
,
onde usamos o mesmo passo `hx` na direções espaciais $x$ e $y$. Determine então a equação que nos dá cada elemento `tt1[i][j]` a partir de elementos do array inicial `tt0[i][j]`.

Escreva essa equação, no formato próprio do C++ , no ficheiro `readme_g#p3.txt`.

Comente se é semelhante à instrução correspondente da eq. de Poisson ou de Laplace que resolveu nas aulas de laboratório. Em particular explicite numa frase apenas qual a(s) diferença(s) entre a iteração de 1.a) e a iteração da Eq. de Poisson que usou nas aulas.

## 1.b

(6 pt) Crie então o novo código `g#p3c1.cpp`, que vai estudar a evolução da temperatura da barra ao longo do tempo. Pode adaptar um dos códigos, que agora denomina `g#p3c0.cpp`, que escreveu nas aulas de laboratório e práticas para resolver a eq. de Poisson ou de Laplace. A cada novo passo do tempo `ht` deve calcular uma nova distribuição da temperatura `tt1[i][j]` no plano $xOy$.

Considere os seguintes dados (que corresponde a uma placa quadrada, inicialmente fria, colocada num forno quente) a serem lidos dum ficheiro, sendo os parâmetros ,
- condutibilidade térmica bi-dimensional $k = 1.22 \times 10{-}3$ m$^2$ s$^{-1}$ do grafite pirolítico,
- comprimento do retângulo/quadrado $L_x = L_y = 1$ m,

as condições fronteira e iniciais,
- temperaturas na fronteira $T(t,0,y) = T(t,L,y) = T(t,x,0) = T(t,x,L) = 373$ oK, todas iguais e constantes,
- temperatura inicial no interior da barra $T(0,x,y)$= 273 oK,

e a discretização,
- passo no tempo $h_t = 1.0 \times 10^{-2}$ s,
- passo no espaço $h_x = h_y = 0.02$ m, *( por Von Neumann bastaria $h_t = 8.0 \times 10^{-2}$ s)*

- número de passos total no tempo `n_max` à sua escolha
- número de passos no tempo `n_print` ao fim do qual escreve um output no ficheiro, sabendo que pretendemos imprimir os resultados do array `tt0[i][j]` para um ficheiro todos os 30 segundos.

### 1.c

(2 pt) Mostre um gráfico com um 3D plot (instrução `ListPlot3D` do `matematica`) da temperatura nos 3 casos: $t =$ de 30 s, 1 min e 90 s.

Comente que tipo de superfície deverá ser o limite para a temperatura $T(x, y)$, para um tempo suficientemente grande para a temperatura estabilizar?

Estime ainda aproximadamente fim de quanto tempo deverá a temperatura no centro diferir em menos de 1% da temperatura final.

## 2 Classe de vetores ou matrizes com overload

Outro objetivo deste projeto *(que pode desenvolver* **em paralelo** *com o da secção 1)* é realizar o `overload` de operadores, para trabalhar com vetores com $n$ elementos. Disponibilizamos na página da disciplina um código, `VecAnyD.zip` , semelhante aos códigos que os alunos treinaram nas aulas práticas e de problemas, e que permite trabalhar com um vetor de qualquer número $n$ de elementos.

### 2.a

(4pt) Partindo do código disponibilizado, ao qual dará o nome `g\#p3c2.zip`, implemente um novo `overload` relevante para trabalhar com vetores: o produto dum escalar pelo vetor, e ainda calcule o produto interno de dois vetores (que também pode ser usado para calcular o módulo).

Crie também um método, `friend Vector med( const Vector &r2);` que tenha como argumento um vetor e que retorne outro vetor, no qual cada elemento do vetor é substituído pela média dos primeiros vizinhos, ou seja $v_i \to (v_{i+1} + v_{i-1})/2$ ; com a exceção do primeiro elemento $v_0$ e o último $v_{n-1}$ elementos que não são alterados.

Na função `main` , crie exemplos para testar todos os métodos e operadores da classe.

### 2.b

(4pt) Aplique a classe de 2.a) à solução da eq. de Fourier uni-dimensional $T_t = k\, T_{xx}$. Deve usar a soma de vetores, o produto de vetores por uma constante, e o método `med` para simplificar as iterações. Note que deve usar a prescrição de 1.a) nesta alínea.

Aplique aos mesmos material, passos e a condições iniciais e de fronteira semelhantes às de 1.b); para uma barra fria a $T = 373$ oK cujos extremos são aquecidos a $T = 473$ oK, com comprimento $l = 1$m. Produza um plot da temperatura $T(30, x)$ ao fim de 30s.

### 2.c

(2pt) Finalmente esta pontuação está reservada a quem ainda conseguir estender as alíneas 2.a) e 2.b) para resolver a Eq. de Fourier bi-dimensional com uma classe para matrizes. Neste caso deve ser capaz de, ou estender a classe do ficheiro `VecAnyD.zip` para trabalhar com matrizes de dimensão `nx*ny` , ou então utilizar uma classe que o grupo já tenha desenvolvido anteriormente para resolver a série IV. Deve aplicar às condições iniciais e fronteira de 1.b), fazendo o plot ao fim de 30s.

**Importante:** se já tiver usado uma classe para matrizes em 1.b), não necessita de realizar esta alínea, basta referir isso na memória descritiva relativa ao 2.c).