ICMC-USP

Disciplina SME0602

Motores Numéricos para Simulação em Engenharia

Do Modelo Matemático ao Gêmeo Digital

Roberto F. Ausas

May 20, 2026

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

Organização da Disciplina e Metodologia

Este documento propõe uma evolução do ensino clássico de Métodos Numéricos, transformando-o numa disciplina que traduz metodologias complexas para uma linguagem atual e apelativa. O foco reside na criação de Gêmeos Digitais (*Digital Twins*) para sistemas físicos encontrados na Engenharia e Ciências Exatas, sintetizando as inovações pedagógicas que tenho desenvolvido nos últimos anos.

As técnicas envolvidas na criação e montagem de um gêmeo digital, que aqui poderemos chamar de Motores Numéricos e Algoritmos, fazem parte do instrumental que qualquer profissional de engenharia ou matemática aplicada precisa dominar. Na prática, esses métodos já se encontram em várias bibliotecas de cálculo científico e Scientific Machine Learning; portanto, o foco do curso não está na implementação desses algoritmos, mas sim:

(i) no seu uso informado, buscando compreender os mecanismos por trás deles;
(ii) e, principalmente, em exercitar o processo de transpor para o computador um problema real, formulado por meio de equações matemáticas.

Por esse motivo, decidi estruturar a disciplina em torno do objetivo concreto de criar o gêmeo digital de um dispositivo microfluídico, que descreverei a seguir. Embora as questões de modelagem envolvidas na descrição desse dispositivo sejam específicas da minha área de pesquisa, acredito que a metodologia pode ser facilmente estendida a outros sistemas por meio de analogias (ex: sistemas elétricos, mecânicos, estruturais, etc.). Aliás, os conceitos físicos e de modelagem envolvidos são relativamente simples e intuitivos, e serão explicados no momento oportuno.

Na sequência, descreverei a organização da disciplina, a metodologia de trabalho e avaliação, bem como algumas ideias iniciais sobre os temas que serão abordados no curso.

Tópicos principais do curso

Este curso abordará questões relacionadas aos seguintes temas:

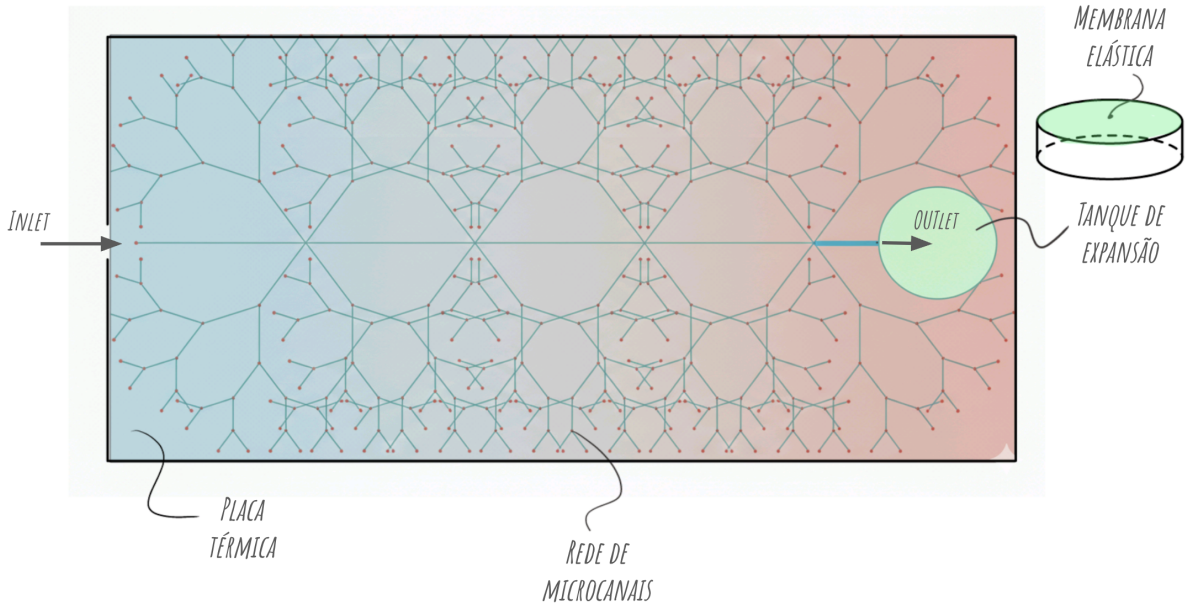
- Álgebra Linear Computacional aplicada a modelos em grafos: (**ALC**);
- Aprendizado por refinamento numérico e Computação iterativa: (**ITE**);
- Amostragem e Integração de modelos: (**INT**);
- Diferenciação de modelos: (**DIF**);
- Otimização de Hiperparâmetros em modelos: (**OTI**);
- Generalização e Regressão de modelos: (**REG**);
- Operadores Dinâmicos e Evolução de modelos: (**DIN**);

Esses temas, por sua vez, incluem diversas técnicas que serão utilizadas ao longo do curso e que detalhamos na tabela a seguir:

ALC	ITE	INT	DIF	OTI	REG	DIN
Sistemas lineares em grafos Matrizes esparsas Fatoração, Analise espectral	Processos iterativos Relaxação de sistemas Iterações de Jacobi, Gauss-Seidel e SOR	Amostragem Monte Carlo Fórmulas de Quadraturas	Diferenciação numérica Aproximação de derivadas Análise de Sensibilidade	Métodos de descida Método de Newton Função de perda e Critérios de parada	Aproximação de funções Interpolação Mínimos quadrados	Métodos de integração para ODEs Acurácia e Estabilidade

Sobre o dispositivo Microfluídico

Na Figura embaixo, apresenta-se um desenho esquemático conceitual do dispositivo cujo Gêmeo Digital pretende-se construir. O sistema é integrado principalmente por uma rede de microcanais embutida entre duas placas e que descarrega num tanque de expansão.



Primeiramente, estudaremos a modelagem matemática e computacional de cada subsistema isoladamente, a saber:

(1) A rede hidráulica:
Consiste em uma rede de microcanais interconectados de seção transversal conhecida, pelos quais circula um fluido incompressível. Em princípio, o fluido ingressa na rede pelo ponto de entrada (*inlet*) e descarrega no reservatório pelo ponto de saída (*outlet*);

(2) O sistema térmico:
Os canais da rede estão embutidos entre duas placas sujeitas a um gradiente de temperatura, onde os efeitos de troca térmica são relevantes;

(3) O tanque de expansão:
Consiste de reservatório cuja tampa superior é uma membrana elástica tensionada, a qual se deforma conforme o fluido preenche o tanque.

Uma vez compreendido o comportamento detalhado de cada subsistema e implementadas as `classes` em `Python` responsáveis por sua representação computacional, com identificação clara de parâmetros, variáveis de estado e entradas/saídas, poderemos definir interfaces consistentes entre os modelos, incorporar os mecanismos de acoplamento e investigar as seguintes questões:

(I) Térmico–Fluido:
De que maneira a distribuição de temperatura modifica as propriedades e o comportamento do fluido nos microcanais? E, reciprocamente, como a sua presença altera a condução térmica no material das placas?

(II) Fluido–Estrutura:
Como a pressão e a vazão do fluido nos pontos de *inlet* e *outlet* influenciam a deformação da membrana? E como essa deformação retroage, a montante, modificando o comportamento hidráulico da rede?

Organização deste Material

Nos próximos capítulos, detalharemos cada subsistema e os respectivos acoplamentos. Serão apresentados os conceitos físicos fundamentais, a sua modelagem por meio de equações matemáticas e as conexões com os conceitos de algebra linear e cálculo diferencial e integral relevantes, junto com uma série de exercícios de programação em `Python`, que permitirão a implementação incremental dos componentes necessários para a construção do Gêmeo Digital (doravante denominado GD):

Capítulo |01| A REDE HIDRÁULICA

↪ Neste capítulo modelamos a rede de distribuição como um grafo, cujos nós e arestas representam, respectivamente, os pontos de conexão e os microcanais do dispositivo. A partir dessa representação, construiremos os operadores e tensores que descrevem a física do sistema. Serão analisadas diferentes condições de operação, e como estas se traduzem em restrições algébricas. Por fim, introduziremos os motores numéricos responsáveis pela predição do comportamento hidráulico da rede.

Capítulo |02| A PLACA TÉRMICA

↪ Neste capítulo modelamos a matriz sólida na qual a rede hidráulica está embutida, considerando os fenômenos de condução de calor que governam a distribuição de temperatura no dispositivo. A partir do modelo contínuo, construiremos a formulação discreta, enfatizando a estrutura algébrica do problema e a montagem dos tensores esparsos correspondentes. Também estudaremos algoritmos de relaxação para a determinação eficiente dos estados de equilíbrio térmico.

Capítulo |03| A MEMBRANA ELÁSTICA

↪ Neste capítulo modelamos o comportamento mecânico da tampa do reservatório de expansão alimentado pela rede hidráulica. Construiremos o operador dinâmico discreto que descreve a deformação de uma membrana elástica sob tensão, incorporando os efeitos de inércia e rigidez. A partir dessa formulação, analisaremos a evolução temporal do sistema e os modos naturais de vibração.

Capítulo |04| ACOPLAMENTO HIDRÁULICO-TÉRMICO

↪ Neste capítulo modelamos o acoplamento entre a rede de microcanais e a placa térmica. Por um lado, analisamos a influência da temperatura local nas propriedades do fluido; por outro, examinamos como a presença e a operação dos microcanais modificam o comportamento térmico efetivo e a condutividade do material da placa.

Capítulo |05| ACOPLAMENTO HIDRÁULICO-MECÂNICO

↪ Neste capítulo investigamos estratégias numéricas iterativas para o estudo da interação entre a rede hidráulica, responsável pela alimentação do reservatório, e a deformação da membrana elástica que atua como sua tampa flexível.

Capítulo |06| O GÊMEO DIGITAL

↪ Finalmente, reunimos todos os componentes desenvolvidos nos capítulos anteriores em uma formulação integrada, permitindo a análise do comportamento global do dispositivo e a construção de seu modelo computacional completo.

Metodologia e Avaliação

Como mencionado anteriormente, a disciplina está estruturada em torno da montagem do Gêmeo Digital descrito acima. Para esse fim, a turma será dividida em grupos, que deverão montar, de maneira incremental, os componentes individuais do GD e as interfaces que permitirão o acoplamento entre eles.

A presença dos alunos é fundamental, pois os trabalhos serão desenvolvidos predominantemente em sala de aula, com acompanhamento direto do professor, que estará disponível para orientar os desenvolvimentos, esclarecer dúvidas conceituais e oferecer suporte nas estratégias de implementação. Espera-se, adicionalmente, que cada estudante dedique entre 3 e 4 horas semanais de estudo extraclasse para a consolidação e finalização das atividades propostas.

Para o desenvolvimento de cada componente, o grupo deverá se aprofundar no estudo do material fornecido. Em cada capítulo, haverá um referencial teórico e uma série de atividades de desenvolvimento que permitirão atingir os objetivos intermediários, culminando na montagem completa do GD ao término do curso.

Avaliação

- Nas datas estabelecidas no cronograma, cada grupo apresentará, apenas ao professor, os avanços realizados em cada etapa;
- Espera-se que todos os membros do grupo participem da apresentação oral de uma parte do trabalho;
- Eventuais faltas deverão ser justificadas mediante a apresentação de atestado/certificado;
- O professor fará arguições e poderá solicitar explicações adicionais a qualquer membro do grupo; por isso, é fundamental que todos dominem o conteúdo exposto;
- A mesma nota será atribuída a todos os membros do grupo em cada instância. Por este motivo, a participação ativa de todos no processo de desenvolvimento é essencial;
- Será calculada uma média das apresentações, denominada N_A;
- Ao final do processo, o grupo entregará um relatório técnico completo do projeto, que receberá uma nota N_R.

A nota final (N) da disciplina será calculada por:

$$N = \frac{N_A + N_R}{2}$$

Para aprovação, a nota N deverá ser superior a $4,9$ e a frequência igual ou superior a 70%.

↪ **Uso de Ferramentas de IA**
O uso de ferramentas de Inteligência Artificial (IA) é permitido nesta disciplina e pode ser útil para apoiar o desenvolvimento de códigos e a organização das soluções. No entanto, seu emprego deve ser criterioso, uma vez que tais ferramentas podem gerar respostas imprecisas ou inconsistentes ("alucinações"). Durante a avaliação das atividades, será verificado se o estudante compreende plenamente as soluções apresentadas, sendo capaz de explicar as decisões de implementação adotadas, e não apenas reproduzir automaticamente saídas geradas por IA. *Sempre que houver utilização dessas ferramentas, deverá ser fornecido o link para o histórico completo da interação que resultou na solução submetida.*

↪ **Critérios de Avaliação**
Por se tratar de uma disciplina de Cálculo Numérico, o processo avaliativo priorizará a correta formulação algébrica, a implementação computacional e a análise dos métodos e algoritmos empregados. Aspectos relacionados à modelagem física terão caráter complementar, servindo como contexto para a aplicação dos métodos numéricos.

Cronograma de trabalho

Data	Observações
02/03/2026	Redes hidráulicas
04/03/2026	Redes hidráulicas
09/03/2026	Redes hidráulicas
11/03/2026	Redes hidráulicas
16/03/2026	Redes hidráulicas
18/03/2026	Redes hidráulicas
23/03/2026	Grupos apresentam A REDE HIDRÁULICA
25/03/2026	Placa térmica
30/03/2026	**Semana Santa**
01/04/2026	**Semana Santa**
06/04/2026	Placa térmica
08/04/2026	Placa térmica
13/04/2026	Placa térmica
15/04/2026	Placa térmica
20/04/2026	**Tiradentes**
22/04/2026	Grupos apresentam A PLACA TÉRMICA
27/04/2026	Membrana elástica
29/04/2026	Membrana elástica
04/05/2026	Membrana elástica
06/05/2026	Membrana elástica
11/05/2026	Grupos apresentam A MEMBRANA ELÁSTICA
18/05/2026	Acoplamento hidráulico-térmico
20/05/2026	Acoplamento hidráulico-térmico
25/05/2026	Acoplamento hidráulico-térmico
27/05/2026	Grupos apresentam ACOPLAMENTO HIDRÁULICO-TÉRMICO
01/06/2026	Acoplamento hidráulico-mecânico
03/06/2026	Acoplamento hidráulico-mecânico
08/06/2026	Acoplamento hidráulico-mecânico
10/06/2026	Grupos apresentam ACOPLAMENTO HIDRÁULICO-MECÂNICO
15/06/2026	Finalizar o Gêmeo Digital
17/06/2026	Finalizar o Gêmeo Digital
22/06/2026	Finalizar o Gêmeo Digital
24/06/2026	Finalizar o Gêmeo Digital
29/06/2026	Grupos apresentam O GÊMEO DIGITAL
01/07/2026	Grupos apresentam O GÊMEO DIGITAL

Contents

A REDE HIDRÁULICA 1

1.1 Preludio

No presente capítulo mostraremos como se formula o problema de distribuição numa rede elétrica ou hidráulica, e em geral, qualquer quantidade ou informação que se propaga por uma rede de maneira conservativa (p.e., em redes de informação ou inclusive treliças).

Objetivos

↪ Introduzir os aspectos de modelagem para uma rede hidráulica;
↪ Entender os aspectos de Algebra Linear envolvidos;
↪ Aprender como resolver o problema de maneira sistemática no computador.

Há muitos sistemas físicos que podem ser modelizados como uma rede na qual se propaga uma certa quantidade física. Na tabela da sequência mostramos três exemplos clássicos que aparecem na engenharia:

Sistema	Nó	Variável nodal	Aresta	Variável de aresta	Lei de conservação
Elétrico	Conexão	Voltagem	Resistencia	Corrente	Carga elétrica
Hidráulico	União	Pressão	Cano	Vazão	Massa
Estrutural	Articulação	Deslocamento	Barra	Tensão	Momento

1.2 Exemplo de resolução de uma rede hidráulica

Nesta parte vamos trabalhar com redes hidráulicas. Para isto, vamos considerar o exemplo de uma rede simples mostrado na figura. Posteriormente, conseguiremos generalizar isto para resolver qualquer rede.

Para o exemplo particular temos: $n_c = 7$ canos e $n_v = 5$ nós:

O objetivo é determinar:

- As pressões em cada nó: $\mathbf{p} = [p_1 \dots p_5]^\intercal$
- As vazões em cada cano: $\mathbf{Q} = [Q_1 \dots Q_7]^\intercal$.

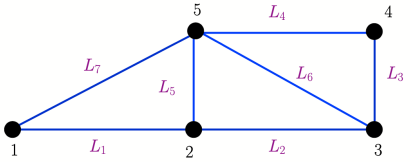


Figure 1.1: Exemplo de uma rede simples.

Conceitos necessários:

- Comportamento de um cano → **Lei constitutiva**
- Condições de acoplamento → **Conservação de massa e Continuidade**

1.2.1 Lei constitutiva: Comportamento de um cano

Para cada cano, $k = 1, \ldots, n_c$, que conecta os nós i e j, vamos considerar a lei de perda de pressão em função da vazão:

$$\Delta p_k = p_i^k - p_j^k = \frac{L_k}{\kappa_k} Q_k$$

$$\Rightarrow \quad Q_k = \frac{\kappa_k}{L_k} \Delta p_k \; = \; C_k \, (p_i^k - p_j^k)$$

- $C_k = \dfrac{\kappa_k}{L_k}$
- κ_k é uma constante[1]
- L_k é o comprimento de cada cano.
- A vazão é positiva quando vai de i a j.

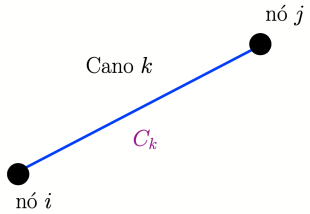


Figure 1.2: Um só cano.

1: Isto é válido quando a velocidade do fluido é baixa, como acontece nos dispositivos microfluidicos, o que se conhece como o regime laminar. No regime chamado de turbulento, a situação é mais complicada.

1.2.2 Condições de acoplamento

↪ **Unicidade da pressão**: Para qualquer nó $i = 1, \ldots, n_v$,

$$p_i^k = p_i$$

para todo nó k que possui o nó i.

↪ **Conservação da massa**:

$$\sum_{\forall\, k \text{ que possui o nó } i} Q_k^{(i)} \; = \; 0, \quad i = 1, 2, \ldots, n_v$$

Desta forma, teremos uma equação por cada nó da rede. Para o caso particular do exemplo:

$$\begin{aligned}
\text{Nó 1} &: \sum_k Q_k^{(1)} = C_1(p_1 - p_2) + C_7(p_1 - p_5) = 0 \\
\text{Nó 2} &: \sum_k Q_k^{(2)} = C_1(p_2 - p_1) + C_5(p_2 - p_5) + C_2(p_2 - p_3) = 0 \\
\text{Nó 3} &: \sum_k Q_k^{(3)} = C_2(p_3 - p_2) + C_6(p_3 - p_5) + C_3(p_3 - p_4) = 0 \\
\text{Nó 4} &: \sum_k Q_k^{(4)} = C_3(p_4 - p_3) + C_4(p_4 - p_5) = 0 \\
\text{Nó 5} &: \sum_k Q_k^{(5)} = C_7(p_5 - p_1) + C_5(p_5 - p_2) + C_6(p_5 - p_3) + C_4(p_5 - p_4) = 0
\end{aligned}$$

i.e., **5 equações e 5 incógnitas**. Agora vamos ordenar o sistema:

$$
\begin{aligned}
(C_1 + C_7)\,p_1 &- C_1\,p_2 + 0\,p_3 + 0\,p_4 - C_7\,p_5 = 0\\
-C_1\,p_1 &+ (C_1 + C_2 + C_5)\,p_2 - C_2\,p_3 + 0\,p_4 - C_5\,p_5 = 0\\
0\,p_1 &- C_2\,p_2 + (C_2 + C_3 + C_6)\,p_3 - C_3\,p_4 - C_6\,p_5 = 0\\
0\,p_1 &+ 0\,p_2 - C_3\,p_3 + (C_3 + C_4)\,p_4 - C_4\,p_5 = 0\\
-C_7\,p_1 &- C_5\,p_2 - C_6\,p_3 - C_4\,p_4 + (C_4 + C_5 + C_6 + C_7)\,p_5 = 0
\end{aligned}
$$

ou em forma matricial:

$$
\begin{pmatrix}
C_1 + C_7 & -C_1 & 0 & 0 & -C_7\\
-C_1 & C_1 + C_2 + C_5 & -C_2 & 0 & -C_5\\
0 & -C_2 & C_2 + C_3 + C_6 & -C_3 & -C_6\\
0 & 0 & -C_3 & C_3 + C_4 & -C_4\\
-C_7 & -C_5 & -C_6 & -C_4 & C_4 + C_5 + C_6 + C_7
\end{pmatrix}
\cdot
\begin{pmatrix} p_1\\ p_2\\ p_3\\ p_4\\ p_5 \end{pmatrix}
=
\begin{pmatrix} 0\\ 0\\ 0\\ 0\\ 0 \end{pmatrix}
$$

De forma geral, a matriz da rede descreve-se como

$$
A_{ij} = \begin{cases}
\sum\limits_{k \text{ conectados com } i} C_k^{(i)} & \text{se } i = j\\
-C_r & \text{se } i \neq j, \text{ com } i \text{ conectado a } j \text{ por } C_r\\
0 & \text{se não há conexão entre } i \text{ e } j
\end{cases}
$$

- A matriz é simétrica
- A soma dos elementos de qualquer linha/coluna é zero
- A matriz **não é invertível** → **A pressão está indeterminada**. Isto significa que não podemos resolver o sistema de equações até não eliminar essa indeterminação.
- Em geral, um nó estará conectado a um número relativamente pequeno de nós. Isto fará com que a matriz A possua muitos zeros

1.2.3 Conexão da rede

Agora, precisamos conectar a rede para resolver um problema que tenha algum sentido físico. Considerar a figura 1.3.

- Para remover a indeterminação se fixa o valor da pressão em algum ponto. Para o exemplo, vamos conectar o nó 3 à atmosfera, do qual podemos deduzir que:

  $$p_3 = 0$$

  Porém, agora teremos uma vazão Q_R para determinar.

- Também, vamos injetar uma vazão Q_B no nó 1, conectando uma bomba, e assim resolver um problema que tenha uma solução não trivial.

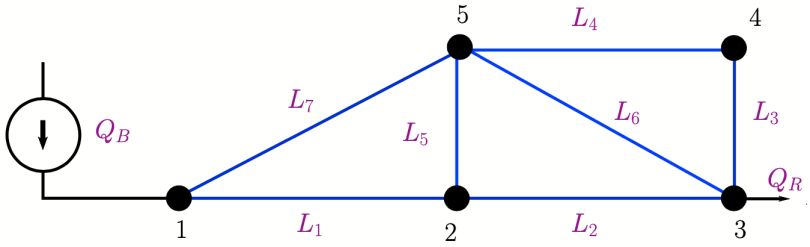


Figure 1.3: Rede conectada a uma bomba e descarregando na atmosfera.

1.2.4 Conexão da rede no nível algébrico

Precisamos incorporar no nível do sistema de equações a informação sobre a conexão da rede que acabamos de mencionar:

- Se estamos injetando no nó 1, então, a sua equação agora é:

$$\sum Q_r^{(1)} = (C_1 + C_7)\, p_1 - C_1\, p_2 + 0\, p_3 + 0\, p_4 - C_7\, p_5 = Q_B$$

- Similarmente, a equação para o nó 3 agora é:

$$\sum Q_r^{(3)} = 0\, p_1 - C_2\, p_2 + (C_2 + C_3 + C_6)\, p_3 - C_3\, p_4 - C_6\, p_5 = Q_R$$

  em que Q_R é a **nova** incógnita. i.e., seria melhor colocar-la no lado esquerdo:

$$0\, p_1 - C_2\, p_2 + (C_2 + C_3 + C_6)\, p_3 - C_3\, p_4 - C_6\, p_5 - 1\, Q_R = 0$$

- mas, se já sabemos que $p_3 = 0$, então, temos uma equação adicional:

$$0\, p_1 + 0\, p_2 + 1\, p_3 + 0\, p_4 + 0\, p_5 = 0$$

No entanto, estamos interessados em achar as pressões, então, podemos **ignorar** a equação de balanço do nó 3 e ficar apenas com essa última equação, resultando o novo sistema de equações (ainda de 5×5):

$$\begin{pmatrix} C_1 + C_7 & -C_1 & 0 & 0 & -C_7 \\ -C_1 & C_1 + C_2 + C_5 & -C_2 & 0 & -C_5 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -C_3 & C_3 + C_4 & -C_4 \\ -C_7 & -C_5 & -C_6 & -C_4 & C_4 + C_5 + C_6 + C_7 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix} = \begin{pmatrix} Q_B \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Esse sistema matriz vetor, sim pode ser resolvido.

De forma geral podemos descrever a matriz $\widetilde{A}$ e o vetor **b** do sistema como: Seja n_{atm} é o índice do nó em que a pressão é fixada ($p_{n_{atm}} = 0$) e n_B o índice do nó em que a bomba é conectada.

$$\widetilde{A}_{ij} = \begin{cases} A_{ij} & \text{se } i \neq n_{atm} \\ 0 & \text{se } i = n_{atm} \text{ e } j \neq n_{atm} \\ 1 & \text{se } i = j = n_{atm} \end{cases} \qquad b_i = \begin{cases} 0 & \text{se } i \neq n_{atm} \\ Q_B & \text{se } i = n_B \\ 0 & \text{se } i = n_{atm} \end{cases}$$

Ficando um sistema matriz-vetor do seguinte tipo:

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n_{atm}} & \dots & A_{1n_B} & \dots & A_{1n_p} \\ A_{21} & A_{22} & \dots & A_{1n_{atm}} & \dots & A_{1n_B} & \dots & A_{2n_p} \\ . & . & \dots & . & \dots & . & \dots & . \\ . & . & \dots & . & \dots & . & \dots & . \\ . & . & \dots & . & \dots & . & \dots & . \\ 0 & 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ . & . & \dots & . & \dots & . & \dots & . \\ . & . & \dots & . & \dots & . & \dots & . \\ . & . & \dots & . & \dots & . & \dots & . \\ A_{n_B1} & A_{n_B2} & \dots & A_{n_Bn_{atm}} & \dots & A_{n_Bn_B} & \dots & A_{n_Bn_p} \\ . & . & \dots & . & \dots & . & \dots & . \\ . & . & \dots & . & \dots & . & \dots & . \\ . & . & \dots & . & \dots & . & \dots & . \\ A_{n_p1} & A_{n_p2} & \dots & A_{n_pn_{atm}} & \dots & A_{n_pn_B} & \dots & A_{n_pn_p} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ . \\ . \\ . \\ p_{n_{atm}} \\ . \\ . \\ . \\ p_{n_B} \\ . \\ . \\ . \\ p_{n_p} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ . \\ . \\ . \\ Q_B \\ . \\ . \\ . \\ 0 \end{pmatrix}$$

1.2.5 Exemplo numérico

Para colocar números concretos, consideremos uma vazão $Q_B = 3\,m^3/s$ e as seguintes propriedades para os canos da rede: Inserindo estes valores

Cano	ρ_k [bar s / m^4]	L_k [m]	C_k [bar s / m^3]$^{-1}$
1	0.1	5	2
2	0.1	5	2
3	0.1	10	1
4	0.1	5	2
5	0.1	10	1
6	0.1	5	2
7	0.1	5	2

chegamos no sistema matriz-vetor (conferir):

$$\begin{pmatrix} 4 & -2 & 0 & 0 & -2 \\ -2 & 5 & -2 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -2 \\ -2 & -1 & -2 & -2 & 7 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

E agora o que precisamos fazer é a resolução do sistema, o qual pode ser feito pelo chamado escalonamento da matriz:

$$\left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ -2 & 5 & -2 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 & -2 & 0 \\ -2 & -1 & -2 & -2 & 7 & 0 \end{array}\right) \xrightarrow{\ell_2 \leftarrow \ell_2 + \frac{1}{2}\ell_1} \left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ 0 & 4 & -2 & 0 & -2 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 & -2 & 0 \\ -2 & -1 & -2 & -2 & 7 & 0 \end{array}\right) \xrightarrow{\ell_5 \leftarrow \ell_5 + \frac{1}{2}\ell_1}$$

$$\left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ 0 & 4 & -2 & 0 & -2 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 & -2 & 0 \\ 0 & -2 & -2 & -2 & 6 & \frac{3}{2} \end{array}\right) \xrightarrow{\ell_5 \leftarrow \ell_5 + \frac{1}{2}\ell_2} \left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ 0 & 4 & -2 & 0 & -2 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 & -2 & 0 \\ 0 & 0 & -3 & -2 & 5 & \frac{9}{4} \end{array}\right) \xrightarrow{\ell_4 \leftarrow \ell_4 + \ell_3}$$

$$\left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ 0 & 4 & -2 & 0 & -2 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & -2 & 0 \\ 0 & 0 & -3 & -2 & 5 & \frac{9}{4} \end{array}\right) \xrightarrow{\ell_5 \leftarrow \ell_5 + 3\ell_3} \left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ 0 & 4 & -2 & 0 & -2 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & -2 & 0 \\ 0 & 0 & 0 & -2 & 5 & \frac{9}{4} \end{array}\right) \xrightarrow{\ell_5 \leftarrow \ell_5 + \frac{2}{3}\ell_4}$$

$$\left(\begin{array}{ccccc|c} 4 & -2 & 0 & 0 & -2 & 3 \\ 0 & 4 & -2 & 0 & -2 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & -2 & 0 \\ 0 & 0 & 0 & 0 & \frac{11}{3} & \frac{9}{4} \end{array}\right) \xrightarrow{\ell_i \leftarrow \ell_i / a_{ii}} \left(\begin{array}{ccccc|c} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & \frac{3}{4} \\ 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} & \frac{3}{8} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{2}{3} & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{27}{44} \end{array}\right)$$

$$\begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{2}{3} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix} = \begin{pmatrix} \frac{3}{4} \\ \frac{3}{8} \\ 0 \\ 0 \\ \frac{27}{44} \end{pmatrix}$$

o que finalmente resulta:

$$\begin{aligned} 1\,p_1 - \tfrac{1}{2}\,p_2 + 0\,p_3 + 0\,p_4 - \tfrac{1}{2}\,p_5 &= \tfrac{3}{4} \rightarrow p_1 = \tfrac{123}{88} \\ 1\,p_2 - \tfrac{1}{2}\,p_3 + 0\,p_4 - \tfrac{1}{2}\,p_5 &= \tfrac{3}{8} \rightarrow p_2 = \tfrac{60}{88} \\ + 1\,p_3 + 0\,p_4 + 0\,p_5 &= 0 \rightarrow p_3 = 0 \\ + 1\,p_4 - \tfrac{2}{3}\,p_5 &= 0 \rightarrow p_4 = \tfrac{54}{132} \\ 1\,p_5 &= \tfrac{27}{44} \rightarrow p_5 = \tfrac{27}{44} \end{aligned}$$

Verificação dos resultados

Lembremos que a equação para o nó 3 era:

$$\sum Q_r^{(3)} = 0\,p_1 - C_2\,p_2 + (C_2 + C_3 + C_6)\,p_3 - C_3\,p_4 - C_6\,p_5 = Q_R$$

ou, colocando os valores da tabela

$$\sum Q_r^{(3)} = 0 - 2\,p_2 + 5\,p_3 - 1\,p_4 - 2\,p_5 = Q_R$$

e inserindo as pressões que acabamos de achar:

$$Q_R = \sum Q_r^{(3)} = 0 - 2\,\tfrac{60}{88} + 0 - 1\,\tfrac{54}{132} - 2\,\tfrac{27}{44} = -3$$

ou seja, pelo nó 3 está saindo exatamente o que injectamos no nó 1.

1.3 Redes em `python`

A matriz de conectividades e montagem da matriz A

Uma forma "prática" de montar a matriz A, utiliza a chamada matriz de conectividades $\texttt{conec} \in \mathbb{N}^{n_c \times 2}$. Esta é uma estrutura muito conveniente

para descrever a rede e é um ingrediente chave no método dos Elementos Finitos:

$$\texttt{conec} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 2 \\ 5 & 3 \\ 5 & 1 \end{pmatrix}$$

Olhando para a matriz **global** da rede, a ideia é ir acumulando nela, a matriz **local** associada a cada cano. Para o cano k, de conductividade C_k, teremos uma matriz de 2×2, que chamaremos a matriz local do cano.

$$\mathbf{C}_{loc_k} = \begin{pmatrix} C_k & -C_k \\ -C_k & C_k \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Cano 1}} \begin{pmatrix} C_1 & -C_1 & 0 & 0 & 0 \\ -C_1 & C_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Cano 2}}$$

$$\begin{pmatrix} C_1 & -C_1 & 0 & 0 & 0 \\ -C_1 & C_1 + C_2 & -C_2 & 0 & 0 \\ 0 & -C_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\text{Cano 3}} \begin{pmatrix} C_1 & -C_1 & 0 & 0 & 0 \\ -C_1 & C_1 + C_2 & -C_2 & 0 & 0 \\ 0 & -C_2 & C_2 + C_3 & -C_3 & 0 \\ 0 & 0 & -C_3 & C_3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \ldots$$

$$\ldots \xrightarrow{\text{Cano 7}} \begin{pmatrix} C_1 + C_7 & -C_1 & 0 & 0 & -C_7 \\ -C_1 & C_1 + C_2 + C_5 & -C_2 & 0 & -C_5 \\ 0 & -C_2 & C_2 + C_3 + C_6 & -C_3 & -C_6 \\ 0 & 0 & -C_3 & C_3 + C_4 & -C_4 \\ -C_7 & -C_5 & -C_6 & -C_4 & C_4 + C_5 + C_6 + C_7 \end{pmatrix}$$

Para fazer tal montagem precisaremos programar uma função de `python` que automatize esse processo de montagem de matrizes locais, o que deveria ser algo do tipo:

```
def Assembly(conec, C):
    nv = ...  # numero de nos
    nc = ...  # numero de canos
    A = np.zeros(shape=(nv,nv))
    for k in range(nc):
        n1 = conec[k,0]
        n2 = conec[k,1]
        .
        .

    return A
```

em que `C` é o vetor que possui os valores das condutâncias dos canos da rede. Os detalhes ficarão como exercício para a lista 2.

1.3.1 Montagem do sistema final e Resolução

Uma vez que temos a matriz A do sistema, precisamos montar a matriz $\tilde{A}$ e o vetor de lado direito que é forma o sistema de equações definitivo que precisa ser resolvido para determinar as pressões nos nós da rede, que é o que nos interessa em concreto. Para isto, precisaremos programar uma função de `python` que modifique a linha da matriz na posição `natm` daquele nó que foi conectado à atmosfera e também cria um vetor de lado direito que incorpora a informação da vazão injetada pela bomba `QB` na posição `nB`. A função deveria ser algo do tipo[2]

2: A função `np.linalg.solve` essencialmente faz o escalonamento da matriz para encontrar a solução.

```
def SolveNetwork(conec, C, natm, nB, QB):
    Atilde = Assembly( ... )
    Atilde[natm, :] = ...
    .
    .
    b = np.zeros( ... )
    .
    pressure = np.linalg.solve(Atilde, ... )
    return pressure
```

Os detalhes ficarão como exercício para a lista 2.

1.3.2 Post-processo da solução

Uma vez resolvido o problema, estamos interessados em analizar a solução, para extrair informações de interesse do cálculo:

- Cálculo das vazões nos canos;
- Cálculo da potência consumida pela bomba;
- Visualização e plotagem da solução.

Para o primeiro ponto, uma forma elegante de fazer o cálculo é usar a seguinte fórmula:

$$\mathbf{Q} = \mathbf{K}\,\mathbf{D}\,\mathbf{p}$$

em que $\mathbf{K} \in \mathbb{R}^{n_c \times n_c}$ é a matriz diagonal com as condutâncias dos canos definida por

$$\mathbf{K}_{ij} = \begin{cases} C_i & \text{se } i = j \\ 0 & \text{no resto} \end{cases}$$

e a matriz $\mathbf{D} \in \mathbb{R}^{n_c \times n_v}$ é a matriz definida por:

$$\mathbf{D}_{kj} = \begin{cases} 1 & \text{se } j = \texttt{conec[k,0]} \\ -1 & \text{se } j = \texttt{conec[k,1]} \\ 0 & \text{no resto} \end{cases}$$

Para o segundo ponto, pode-se provar que a potência consumida pela bomba para fazer circular o fluido é dada por:

$$W = \mathbf{p}^\intercal \ (\mathbf{D}^\intercal \, \mathbf{K} \, \mathbf{D}) \ \mathbf{p}$$

isto surge do fato de que a potência consumida pela bomba tem que ser igual a energia dissipada nos canos da rede, i.e.,

$$W = \sum_{k=1}^{n_c} Q_k \Delta p_k$$

em que Q_k é a vazão pelo cano k e Δp_k é a diferença de pressão nos extremos do cano k. Mas, se colocarmos as vazões e as diferenças de pressão em vetores, podemos escrever

$$W = \mathbf{Q} \cdot \mathbf{\Delta} p = \mathbf{Q}^\intercal \, \mathbf{\Delta} p$$

i.e., W é o igual ao produto escalar desses vetores. Agora, notemos que $\mathbf{\Delta} p = \mathbf{D}\, p$ e $\mathbf{Q} = \mathbf{K}\,\mathbf{D}\,\mathbf{p}$, então

$$W = \mathbf{Q}^\intercal \, \mathbf{\Delta} p = (\mathbf{K}\,\mathbf{D}\,\mathbf{p})^\intercal \, \mathbf{\Delta} p = (\mathbf{p}^\intercal \mathbf{D}^\intercal \mathbf{K}^\intercal)(\mathbf{D}\,\mathbf{p}) = \mathbf{p}^\intercal (\mathbf{D}^\intercal \mathbf{K} \mathbf{D})\, \mathbf{p}$$

pois o produto de matrizes é **associativo** e $\mathbf{K} = \mathbf{K}^\intercal$, por ser ela uma matriz diagonal[3].

3: Notar que o resultado é um número, i.e., uma matriz de 1×1, pois a potência que tem unidades de Energia por unidade de tempo (p.e., [J/s]) é de fato uma quantidade escalar.

1.4 A rede de microcanais do GD

1.4.1 Geração do grafo da rede

Uma vez que aprendemos a implementar e resolver uma rede simples, podemos avançar para a análise de redes mais complexas.

Para isso, será fornecida a função `gera_grafo` (disponível no sistema Tidia), que permite gerar o grafo da rede:

```
Xno, conec = GeraGrafo(levels=meu_valor)
mm_to_m = 0.001
Xno = Xno * mm_to_m
PlotaRede(conec, Xno, p, q, mm_to_m)
```

Nessa chamada, `Xno` é uma matriz que contém as coordenadas x e y de cada nó. As coordenadas são fornecidas em milímetros [mm]. Portanto, após o carregamento, é necessário multiplicar a matriz por um fator de conversão para metros [m] [4]. No momento da plotagem, esse mesmo fator também precisa ser informado.

4: Caso contrário, não haverá consistência de unidades.

Por sua vez, `conec` é a matriz de conectividades, que indica o nó inicial e o nó final de cada aresta, utilizando a numeração do `python` (isto é, iniciando em `0`).

É possível gerar redes com diferentes níveis de complexidade por meio do parâmetro `levels`. Na Figura 1.4, apresentam-se redes obtidas com `levels=1`, `levels=2` e `levels=3`. Nesse tipo de rede, o ponto de entrada (*Inlet*) é sempre o nó `0`, enquanto o ponto de saída (*Outlet*) pode ser considerado o nó `5`, correspondendo, respectivamente, ao nó mais à esquerda e ao nó mais à direita da espinha central da rede.

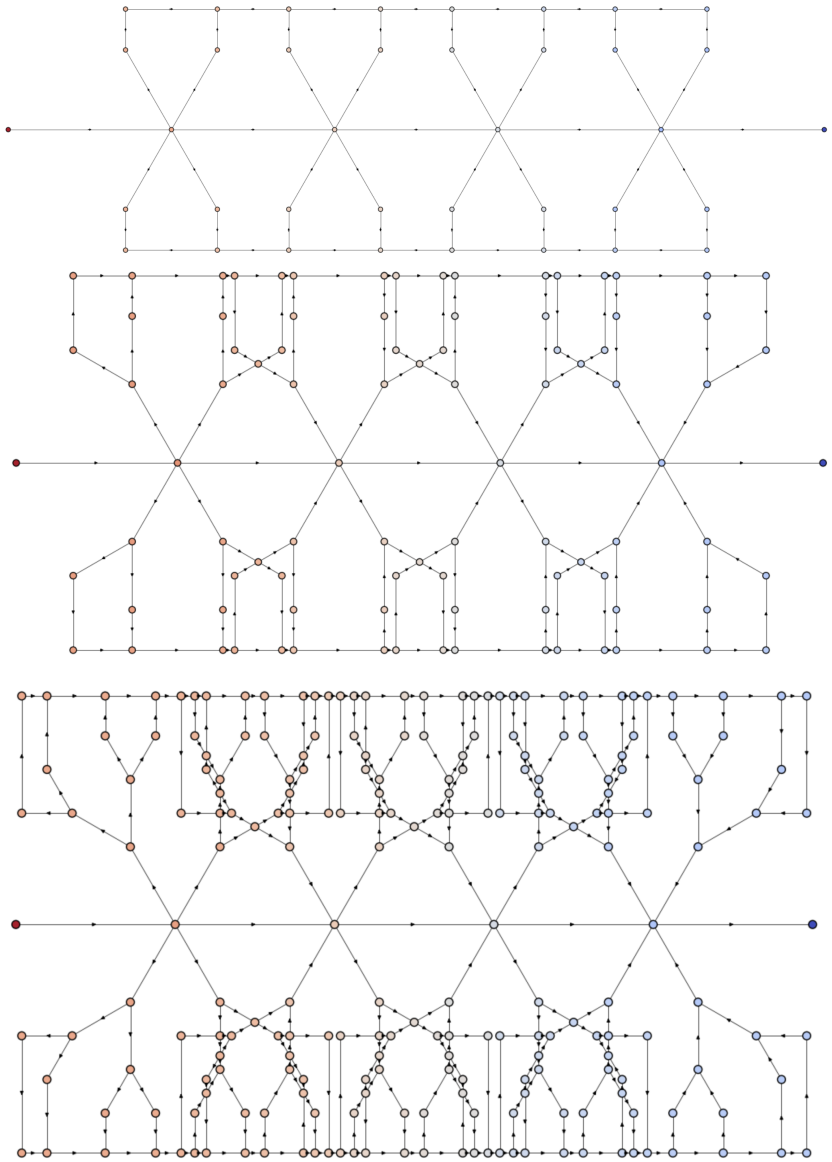


Figure 1.4: Redes hidráulicas com diferentes níveis de complexidade.

1.4.2 Cálculo das condutâncias dos canais

Recordando que, para um canal k, vale a relação

$$Q_k = C_k (p_i^k - p_j^k), \tag{1.1}$$

em que a condutância hidráulica é dada por

$$C_k = \frac{\kappa_k}{L_k}. \tag{1.2}$$

No caso de sistemas microfluídicos contendo fluidos Newtonianos, como a água, podemos assumir que

$$\kappa_k = \frac{\pi D_k^4}{128\,\mu}, \tag{1.3}$$

em que μ é a viscosidade dinâmica do fluido[5] e D_k é o chamado diâmetro hidráulico, definido por

$$D_k = \sqrt{\frac{4\,A_k}{\pi}}, \tag{1.4}$$

sendo A_k a área da seção transversal do canal.

5: Alguns valores típicos de viscosidade:

- Água: $\mu = 0.001$ Pa · s;
- Óleo: $\mu = 0.01$ Pa · s;
- Ar: $\mu = 1.8 \times 10^{-5}$ Pa · s.

Para fixar valores numéricos, consideraremos:

- **Canais:** $A_k = 500\,\mu\text{m} \times 500\,\mu\text{m} = 2.5 \times 10^{-7}\ \text{m}^2$;
- **Fluido:** $\mu = 0.001$ Pa · s (água a 20°C);
- **Inlet:** $Q_{\text{in}} = 0.1$ mL/s $= 10^{-7}\ \text{m}^3/\text{s}$;
- **Outlet:** $p_{\text{atm}} = 0$ [6]

6: Ao adotar a pressão na saída como nula, estamos assumindo que as pressões calculadas representam o excesso de pressão em relação à pressão atmosférica (isto é, pressão manométrica).

(mL denota mililitros). Essas informações devem ser incorporadas na etapa de montagem do sistema linear de equações associado à rede.

1.4.3 Investigando o comportamento do sistema

Nesta seção, investigamos alguns aspectos relevantes do problema. Considere uma rede com nível de complexidade 3, isto é,

```
Xno, conec = GeraGrafo(levels=3)
```

Introduza as modificações necessárias no código e gere os resultados, que deverão ser apresentados ao professor.

1. Suponha que queremos injetar uma vazão em mais de um ponto. Modifique o código atual para permitir a entrada de uma lista de nós e das respectivas vazões injetadas.

   ```
   Qs = {'0' : 1.0e-7, '175' : 1.0e-6, ...}
   ```

2. Suponha que queremos impor o valor da pressão em mais de um ponto. Modifique o código atual para permitir a entrada de uma lista de nós e dos respectivos valores de pressão.

   ```
   ps = {'5' : 0.0, '215' : 0.0, ...}
   ```

3. No problema anterior, considere agora que não temos uma bomba injetando vazão, mas que estamos impondo o valor de pressão 100 no ponto de *Inlet* e o valor 0 no *Outlet*[7].
   Uma vez resolvida a rede, como você pode determinar qual é a vazão que está entrando pelo ponto de *Inlet*[8]?

   Linearidade do problema[9]

4. Suponha que a vazão Q_{in}^0 injetada no nó 0 seja uma função do tempo:

$$Q_{\text{in}}^0(t) = 1 + 0.1\,\sin(\omega t) \quad \text{mL/s}, \qquad t \in [0, 10].$$

   Assuma que estamos fixando a pressão apenas no nó de *Outlet*. Proponha uma estratégia eficiente para calcular a pressão na rede como função do tempo. Plote o comportamento da pressão máxima na rede como função do tempo. Considere, por exemplo, $\omega = 3\,\text{s}^{-1}$ e discretize o intervalo de tempo em $N = 1000$ passos.

5. No exercício anterior, suponha agora que também estamos injetando uma vazão Q_{in}^{175} no nó 175, dada por

$$Q_{\text{in}}^{175}(t) = 0.1 + 0.01\cos(\omega t) \quad \text{mL/s}, \qquad t \geq 0.$$

   Proponha uma estratégia eficiente para calcular a pressão na rede como função do tempo. Plote o comportamento da pressão máxima na rede como função do tempo. Considere, por exemplo, $\omega = 4\,\text{s}^{-1}$ e discretize o intervalo de tempo em $N = 1000$ passos.

6. Suponha agora que a temperatura do sistema evolui segundo a lei

$$T(t) = 20 + 0.9\,t^2, \qquad t \in [0, 10]\ \text{s},$$

   e que a viscosidade do fluido depende da temperatura de acordo com

$$\mu(T) = \frac{0.001791}{1 + 0.03368T + 0.000221T^2},$$

   onde T está em °C e μ em Pa · s.
   Neste caso, suponha que apenas estamos injetando no nó 0 uma vazão $Q_{\text{in}}^0 = 0.1$ mL/s e fixando pressão nula no nó de *Outlet* 5. É possível utilizar as estratégias anteriores para acelerar os cálculos?

7. Considere redes de tamanho variável alterando o nível de complexidade[10]. Construa uma tabela mostrando o tempo computacional necessário para resolver cada rede como função do número de nós. Reporte separadamente os tempos envolvidos na criação das matrizes e o tempo necessário para resolver o sistema linear[11].

7: Apesar de parecer óbvio, note que não é possível impor simultaneamente pressão e vazão em um mesmo nó.

8: Note que esta é a situação dual do caso em que impomos a vazão Q_{in}^0, no qual só conseguimos determinar o valor da pressão no nó 0 após a resolução do problema.

9: Para resolver estes exercícios é fundamental explorar a linearidade do sistema. Isto é, se $\mathbf{p}^{(1)}$ é solução para o lado direito $\mathbf{b}^{(1)}$ e $\mathbf{p}^{(2)}$ é solução para o lado direito $\mathbf{b}^{(2)}$, e α e β são escalares, então $\mathbf{p} = \alpha\,\mathbf{p}^{(1)} + \beta\,\mathbf{p}^{(2)}$ será solução do problema

$$\widetilde{A}\mathbf{p} = \alpha\,\mathbf{b}^{(1)} + \beta\,\mathbf{b}^{(2)}.$$

Explique por quê.

10: Considere `levels=1,2,3` e `4`

11: Os tempos reportados devem ser a média dos valores obtidos em 10 execuções do experimento, a fim de reduzir a influência de fatores externos que possam afetar a medição.

A PLACA TÉRMICA 2

2.1 Preludio

Neste capítulo, estudaremos o problema de transferência de calor por condução em materiais. Para isso, é necessário introduzir alguns conceitos de modelagem que nos permitam formular as equações que governam a distribuição de temperaturas em um material, como ilustrado na figura ao lado. Neste capítulo, abordaremos os seguintes tópicos:

Objetivos

- ↪ Compreender a modelagem do problema de transferência de calor por condução;
- ↪ Compreender o conceito de discretização espacial do domínio;
- ↪ Formular o sistema de equações a ser resolvido;
- ↪ Estudar técnicas computacionais para acelerar a resolução desses sistemas;

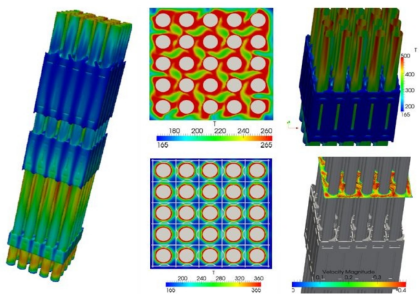


Figure 2.1: Exemplo da distribuição de temperaturas num elemento combustível de uma central nuclear sendo refrigerado por um fluido.

2.2 Transferência de Calor por Condução

Como no problema das redes hidráulicas, precisamos introduzir alguns conceitos para formular o problema:

- **Lei constitutiva:** relaciona o fluxo de energia, na forma de calor, neste caso, devido a uma variação (ou diferença) de temperatura:

$$\mathbf{q}(x, y, t) = -k\,\nabla T(x, y, t)$$

em que (x, y, t) denota o ponto no espaço e o instante de tempo t, k é a condutividade térmica — uma propriedade do material, assumida conhecida —, T é a distribuição de temperaturas, que se deseja determinar, e ∇ denota o operador gradiente, que, no caso bidimensional, é[1]:

$$\nabla T(x, y, t) = \begin{bmatrix} \dfrac{\partial T}{\partial x} \\ \\ \dfrac{\partial T}{\partial y} \end{bmatrix}$$

- **Lei de balanço de energia:**

$$\underbrace{\frac{d}{dt}\iiint_V \rho\, c\, T\, dv}_{\substack{\text{Taxa de variação} \\ \text{da energia interna}}} = \underbrace{\iint_S \mathbf{q}\cdot\mathbf{n}\, ds}_{\substack{\text{Calor que entra} \\ \text{ou sai pela superfície}}} + \underbrace{\iiint_V f\, dv}_{\substack{\text{Calor gerado} \\ \text{no interior do volume}}}$$

em que c é a capacidade calorífica do material, ρ é a densidade, f representa uma fonte de calor (por exemplo, uma reação nuclear, uma reação química ou uma corrente elétrica) e $\mathbf{n}$ é o vetor normal unitário **exterior** à superfície do sólido.

1: Na prática, o gradiente da função é aproximado por uma expressão do tipo:

$$\nabla T(x, y, t) \approx \frac{T_B - T_A}{|d_{AB}|}\frac{\vec{d}_{AB}}{|d_{AB}|},$$

isto é, a partir da diferença de temperatura entre dois pontos próximos A e B, sendo d_{AB} a distância entre eles.

Neste capítulo, assumiremos que o problema é estacionário, isto é, a temperatura não varia no tempo. Dessa forma, podemos negligenciar o termo do lado esquerdo da equação de balanço, obtendo o seguinte problema matemático a ser resolvido para determinar a distribuição de temperaturas $T(x, y)$[2]:

$$\oiint_S k \nabla T \cdot \mathbf{n} \, ds = \oiiint_V f \, dv \qquad (2.1)$$

2: A equação 2.1 deve valer para qualquer volume arbitrário V, limitado por um contorno ou superfície S.

Para se ter uma ideia, o coeficiente de condutividade térmica, denotado por k, é uma propriedade que pode assumir valores bastante distintos dependendo do material:

Material	Condutividade k [$\mathrm{W \cdot m^{-1} \cdot K^{-1}}$]
Aluminio	230
Aço	20
Concreto	1.5
Plástico isolante	0.03
Ar	0.02

O valor de k pode depender da própria temperatura, da posição no espaço ou de outras variáveis. Por ora, assumiremos que k é constante e uniforme, por simplicidade, e veremos como formular e resolver o sistema de equações resultante da aplicação da lei de conservação introduzida anteriormente.

2.3 Discretização do problema

Para resolver o problema, em geral, é necessário realizar o que se conhece como **discretização**. Esse procedimento consiste em dividir o domínio computacional em partes menores, chamadas **células**[3].

3: Assumiremos que todas as células são quadradas, com lado h. Assim, quando $h \to 0$, quanto maior for o número de células, mais acurada será a solução numérica obtida.

Esse processo é ilustrado na figura 2.2, na qual são mostrados um domínio computacional quadrado (à esquerda) e sua discretização em células quadradas (à direita). Na figura, também indicamos as temperaturas impostas nas bordas do domínio, no topo T_T, na base T_B, à esquerda T_L e à direita T_R, as quais devem ser conhecidas para que o problema possa ser efetivamente resolvido.

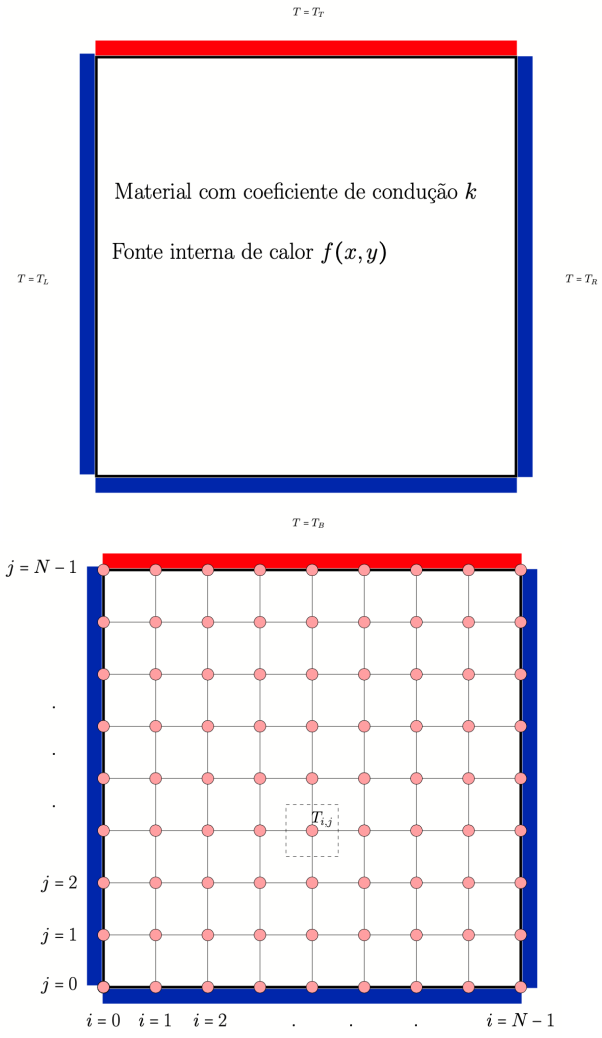


Figure 2.2: Volume e partição em células onde será resolvido o problema de condução de calor.

Observando a figura, vemos uma malha quadrada cujos vértices, indicados pelos pontos rosas, são indexados pelos índices (i, j). Ao redor de cada um desses pontos, representamos, por meio de linhas tracejadas, um volume[4] V arbitrário de forma quadrada. O objetivo é determinar as temperaturas $T_{i,j}$, para todo i, j.

Para formular o problema, devemos escrever a lei de balanço (2.1) para cada volume da discretização. Para isso, consideremos a figura 2.3. Nela, podemos calcular o fluxo de calor que atravessa cada face do contorno do volume quadrado.

Lembrando que h representa a distância entre pontos da malha, obtemos uma boa aproximação para os fluxos dada por:

$$
\begin{aligned}
\mathbf{q}_e &\approx -k\,\frac{T_{i+1,j} - T_{i,j}}{h}\mathbf{e}_x \\
\mathbf{q}_w &\approx -k\,\frac{T_{i,j} - T_{i-1,j}}{h}\mathbf{e}_x \\
\mathbf{q}_n &\approx -k\,\frac{T_{i,j+1} - T_{i,j}}{h}\mathbf{e}_y \\
\mathbf{q}_s &\approx -k\,\frac{T_{i,j} - T_{i,j-1}}{h}\mathbf{e}_y
\end{aligned}
$$

4: Esses volumes são usualmente denominados volumes de controle.

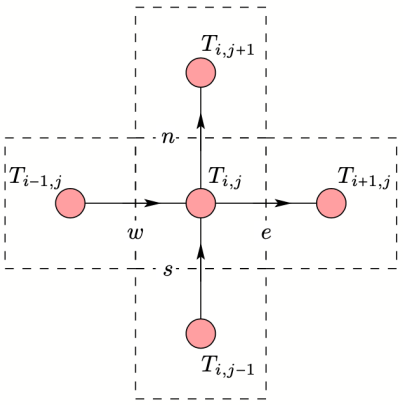


Figure 2.3: Célula (i, j) e seus 4 vizinhos, intercambiando calor a través da sua supeficies, limitada pelos contornos n (*north*), s (*south*), w (*west*) e e (*east*).

Agora, somando todos os fluxos e calculando o termo de geração de calor:

$$\oiint_S k\,\nabla T\cdot\mathbf{n}\,ds \approx L_z \sum_{\substack{\text{Faces do}\\ \text{contorno de } V}} \mathbf{q}_f\cdot\mathbf{n}_f\,s_f \approx$$

$$\approx L_z\,h\left[\frac{k}{h}(T_{i,j}-T_{i+1,j})+\frac{k}{h}(T_{i,j}-T_{i-1,j})+\frac{k}{h}(T_{i,j}-T_{i,j+1})+\frac{k}{h}(T_{i,j}-T_{i,j-1})\right] = \iiint_V f\,dv \approx L_z\,h^2 f_{i,j}$$

em que L_z é a espessura da placa[5]. Note-se que, no lado direito, o valor da integral sobre V dentro da célula (i,j) foi aproximado pelo valor da função $f(\mathbf{x}_{i,j}) = f_{i,j}$, vezes a área da célula, h^2. Se f for a função constante ou linear, a integral é exata.

5: Claramente, o valor de L_z acaba sendo irrelevante no nosso caso pois estamos considerando o problema bidimensional e implicitamente estamos assumindo que não há variação de temperatura ao longo da espessura.

Observação

Para o caso em que $f = 0$ e o k é o mesmo em todos os ponto, a distribuição interna de temperaturas é apenas uma consequência das temperaturas que estivermos impondo nas bordas. Ainda, nesse caso, sendo que lado direito da equação fica igual a zero, teremos que:

$$-T_{i+1,j} - T_{i-1,j} + 4\,T_{i,j} - T_{i,j+1} - T_{i,j-1} = 0$$

O que significa que a temperatura num ponto é a média das temperaturas nos seus vizinhos[6]:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4}$$

Mas, não conhecemos as temperaturas! Isso é justamente o que queremos calcular.

6: Notar que teremos uma destas equações para cada possível valor de i, j, ou seja, o que temos é de fato um sistema de equações, para determinar as temperaturas $T_{i,j},\ i = 1,\ldots,N_x-1, j = 1,\ldots,N_y-1$.

A fórmula anterior motiva alguns métodos iterativos[7] para resolver essas equações, porém estes métodos não são muito eficientes. Por este motivo, continuaremos usando métodos de fatoração baseados no escalonamento da matriz, para o qual precisamos ter a forma explícita da matriz.

7: Os mais conhecidos são os métodos de Jacobi e Gauss-Seidel, que iremos estudar depois em outro contexto.

2.4 Montagem da matriz do sistema

Primeiramente precisamo definir um único índice para cada ponto (ver figura 2.4). No caso da figura estamos considerando $N_x = N_y = N$ por simplicidade.

Na numeração natural, a fórmula para construir o índice global sería: $I = i + (j-1)\,N_x$, porém, como em python a numeração começa em 0 e o índice I chega até $N\,x\,N_y - 1$, usaremos a função de `python`

```
def ij2n (i, j, Nx):
      return i + j*Nx
```

Agora, lembremos a equação para o nó i, j, no caso em que o coeficiente de condutividade k é constante:

$$k\,(-T_{i+1,j}-T_{i-1,j}+4\,T_{i,j}-T_{i,j+1}-T_{i,j-1}) = h^2\,f_{i,j},\ \ i = 1,\ldots,N_x-2,\ j = 1,\ldots N_y-1$$

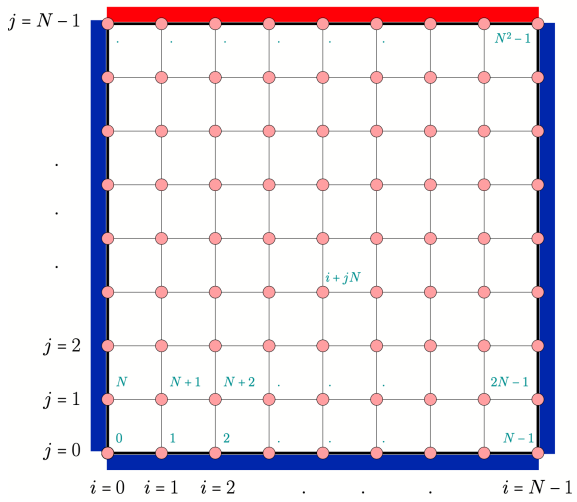


Figure 2.4: Domínio computacional indexando os pontos com um único índice.

O que precisamos saber é a qual índice global de temperaturas corresponde cada um dos termos na equação, i.e.,

```
Ic = ij2n(i,   j,   Nx)
Ie = ij2n(i+1, j,   Nx)
Iw = ij2n(i-1, j,   Nx)
In = ij2n(i,   j+1, Nx)
Is = ij2n(i  , j-1, Nx)
```

Então, a equação resulta:

$$k\,(-T_{I_e} - T_{I_w} + 4\,T_{I_c} - T_{I_n} - T_{I_s}) = h^2\,f_{I_c}, \qquad I_c = 0, \ldots, N_x\,N_y - 1$$

O código para montar a matriz é:

```
def Assembly(Nx, Ny, k):
    nunk = Nx*Ny;
    A = np.zeros(shape=(nunk,nunk))
    for i in range(1,Nx-1):
        for j in range(1,Ny-1):
            Ic = ij2n(i,   j,   Nx)
            .
            .
            .
            A[Ic,[Ic,Ie,Iw,In,Is]] = ...
    return A
```

Por exemplo, para uma grade de 4×4 e $k = 1$, (`A = MatAssTemp(4, 4,1.0)`) resulta:

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.]
```

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]
```

Todas as linhas nulas correspondem aos pontos situados nas bordas, cujas equações veremos como construir na sequência.

2.4.1 Temperatura nas bordas

Falta ainda impor as condições de temperatura nas paredes. Para isso, procederemos de forma análoga ao que foi feito no caso das redes hidráulicas: como conhecemos a temperatura nos nós situados sobre as fronteiras do domínio, podemos escrever, para esses nós, equações triviais. Por exemplo, se o nó (i, j), com índice global I_c, está na parede direita, escrevemos:

$$0\,T_0 \;+\; 0\,T_1 \;+\; \ldots \;+\; 1\,T_{I_c} \;+\; \ldots \;+\; 0\,T_n = T_R. \tag{2.2}$$

em que $n = N_x\,N_y - 1$. Para fixar ideias, suponhamos que a matriz do sistema seja de 5×5 e que desejamos impor a temperatura no nó de índice 2 com valor T_R, isto é:

$$T_2 = T_R$$

O resultado seria:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & 0 & 1 & 0 & 0 \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ T_R \\ b_3 \\ b_4 \end{pmatrix}$$

O código para montar o sistema completo de equações, juntamente com as condições de contorno, deve ter a seguinte estrutura:

```
def SolveSystem(Nx, Ny, h, k, TL, TR, TB, TT, fonte):

    A = Assembly(Nx, Ny, k)
    Atilde = A.copy()

    nunk = Nx*Ny

    # Build right hand side
    b = np.zeros(nunk) # right-hand-side
    .

    iauxh = np.array(range(0,Nx)) # Auxiliary array
    iauxv = np.array(range(0,Ny)) # Auxiliary array
    Iden = np.identity(nunk) # Auxiliary matrix

    Ic = ij2n(0,iauxv,Nx)
    Atilde[Ic,:], b[Ic] = Iden[Ic,:], TL # Tleft

    Ic = ij2n(iauxh,0,Nx)
    Atilde[Ic,:], b[Ic] = Iden[Ic,:], TB # Tbottom
```

```
    .
    .
    .
    .

    Temperature = np.linalg.solve(Atilde, b)

    return Temperature
```

Para implementar o código, será necessário fornecer as funções que calculam as temperaturas nas bordas e o termo de fonte. Por enquanto, podem ser considerados valores constantes para todos (p.e., $T_L = 20°C$, $T_R = 30°C$, $T_B = 40°C$, $T_T = 50°C$ e $f = 500 W \cdot m^{-3}$.

Para o mesmo exemplo de uma malha de 4×4, com $k = 1$, a matriz do sistema resultante é:

```
[[ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  4. -1.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]]
```

Note que, essencialmente, o que estamos fazendo é substituir, nessas linhas, as correspondentes linhas da matriz identidade $\mathbb{I}_{N^2 \times N^2}$. Embora essa forma de impor as condições de contorno seja bastante prática, ela apresenta uma desvantagem: **a matriz do sistema deixa de ser simétrica**.

Existem métodos diretos, como o método de Cholesky[8] e métodos iterativos, como o método dos **gradientes** ou dos gradientes conjugados, que requerem uma matriz simétrica[9].

Nesse contexto, é necessário impor as condições de contorno sem perder a simetria. Do ponto de vista algébrico, isso pode ser feito da seguinte forma (para o exemplo anterior):

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ T_R \\ b_3 \\ b_4 \end{pmatrix}$$

O que podemos fazer para preservar a simetria do sistema, é passar essa incógnita para o lado direito do sistema. Se estamos no ponto 3 por exemplo:

8: O método de Cholesky é um tipo de método de fatoração aplicável apenas a matrizes simétricas e definidas positivas. Nesse método, calcula-se uma matriz triangular superior H tal que

$$A = H^\intercal H.$$

Essa decomposição é conhecida como fatoração de Cholesky (ver Apêndice B).

9: Além disso, em uma matriz simétrica, basta armazenar metade dos seus coeficientes, o que implica economia de memória, aspecto relevante em sistemas de grande porte.

$$\begin{pmatrix} a_{00} & a_{01} & 0 & a_{03} & a_{04} \\ a_{10} & a_{11} & 0 & a_{13} & a_{14} \\ 0 & 0 & 1 & 0 & 0 \\ a_{30} & a_{31} & 0 & a_{33} & a_{34} \\ a_{40} & a_{41} & 0 & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} b_0 - a_{02}\,T_R \\ b_1 - a_{12}\,T_R \\ T_R \\ b_3 - a_{32}\,T_R \\ b_4 - a_{42}\,T_R \end{pmatrix}$$

2.4.2 Esparsidade do sistema

Deve-se notar que esse tipo de discretização leva a matrizes com um grande número de coeficientes nulos (tipicamente, apenas 5 elementos não nulos por linha). Isso motiva o uso de técnicas de armazenamento e algoritmos específicos para matrizes esparsas.

Nesse tipo de estrutura de dados, armazenam-se essencialmente apenas os coeficientes não nulos e suas respectivas posições (índices de linha e coluna) na matriz. Para isso, uma vez montada a matriz, ela pode ser convertida para um formato esparso e, então, o sistema pode ser resolvido:

```
from scipy import sparse
from scipy.sparse.linalg import spsolve
.
.
Atilde = ...
Atildesp = sparse.csr_matrix(Atilde)
temperatura = spsolve(Atildesp, rhs)
```

No caso em que k é uniforme, e observando que a matriz do sistema é essencialmente pentadiagonal, uma forma mais eficiente de construir a matriz A é[10]:

```
print('Warning: This only works for uniform conductivity')
d1 =  np.ones(self.nunk) * 4.0*k
d2 = -np.ones(self.nunk-1) * k
d3 = -np.ones(self.nunk-self.Nx) * k
A = sparse.diags([d3, d2, d1, d2, d3], [-Nx, -1, 0, 1, Nx], format='csr')
```

10: Note que, no caso em que k é uma função da posição (x, y), essa estratégia de montagem da matriz precisaria ser revista, de modo a calcular adequadamente cada elemento das diagonais.

Neste ponto, sugere-se a leitura do Apêndice B, no qual se explicam com mais detalhes os fundamentos por trás da função `linalg.solve`, isto é, dos chamados métodos diretos de resolução de sistemas lineares.

2.5 A placa térmica do GD

Considerando aplicações em microfluídica, assumiremos que o material da placa é o *polidimetilsiloxano puro* (PDMS).

Para fixar valores numéricos, consideraremos:

- **Dimensões:** $L_x \times L_y = 2\text{cm} \times 1\text{cm}$;
- **Condutividade térmica:** $k = 0.2 - 0.3\ \text{W} \cdot \text{K}^{-1} \cdot \text{m}^{-1}$
- **Fonte de calor:** 10^5–$10^6\ \text{W} \cdot \text{m}^{-3}$

- $T_L = 10°\text{C}, T_R = 30°\text{C}, T_B = T_T = 10 + 20\,\dfrac{x}{L_x}$

Note que, para implementar as condições de temperatura nas bordas inferior e superior, que dependem de x, será necessário identificar a coordenada correspondente de cada ponto.

2.5.1 Investigando o comportamento do sistema

1. Monte um caso nominal considerando os valores apresentados acima. Utilize discretizações com diferentes valores de (N_x, N_y), por exemplo,

   $$(21, 11),\ (41, 21),\ (81, 41),\ (161, 81),\ (321, 161),$$

   e estude:

   - Plote as curvas de nível (*contours*) da temperatura para cada caso.
   - Meça o tempo computacional envolvido nas operações (por exemplo, montagem da matriz, resolução do sistema etc.) e compare, em uma tabela, os tempos obtidos para matrizes densas e esparsas.
   - Avalie a temperatura máxima do sistema.
   - Plote a temperatura ao longo do eixo central.

2. Implemente as modificações necessárias para introduzir uma região circular de raio $R = 0{,}2$ cm, centrada em $(0{,}75\,L_x,\ 0{,}5\,L_y)$, na qual a temperatura é fixada em $T_C = 30°\text{C}$. Para isso, considere que, em todos os pontos dessa região, assim como nos pontos sobre a borda, o sistema de equações deve ser modificado para impor essa condição. O resultado esperado é ilustrado na figura 2.5. Neste caso, também utilize diferentes níveis de discretização e reporte a temperatura máxima e o perfil de temperatura ao longo do eixo central.

3. Implemente o caso de condutividade variável considerando a função:

   $$k(x, y) = 0.2 + 0.05\,\sin\left(\frac{3\pi x}{L_x}\right)\sin\left(\frac{3\pi y}{L_y}\right).$$

   Observando a figura 2.3, note que a condutividade aparece na definição dos fluxos. Assim, para cada ponto (i, j) da malha, a função k deve ser avaliada nas posições $w : (x_i - \frac{1}{2}h, y_j)$, $e : (x_i + \frac{1}{2}h, y_j)$, $s : (x_i, y_j - \frac{1}{2}h)$, $n : (x_i, y_j + \frac{1}{2}h)$.

4. Considere novamente o caso de condutividade uniforme k e uma discretização com $(N_x, N_y) = (101, 51)$. Plote a temperatura máxima e a temperatura média do sistema como funções da temperatura na região circular T_C.

5. Pela linearidade do problema, sabe-se que a temperatura em um ponto interno T_k (por exemplo, $k = 233$) depende linearmente de T_R e T_C, isto é,

   $$T_k = a\,T_R + b\,T_C + c.$$

   Determine os coeficientes a, b e c[11].

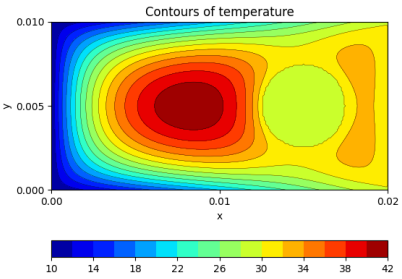


Figure 2.5: Linhas de nível da temperatura esperadas no exercício 2.

11: Atenção: ao determinar esses coeficientes, T_R e T_C não devem possuir uma relação linear entre si.

2.6 Aprendizado por refinamento numérico: Iterações

Consideremos um sistema complexo, como os estudados até agora, descrito por um conjunto de variáveis de estado $\mathbf{x} \in \mathbb{R}^n$, relacionadas por meio de uma função $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$, isto é,

$$\mathbf{F}(\mathbf{x}) = 0.$$

Como exemplos, podemos citar as pressões em uma rede hidráulica ou as temperaturas em uma placa térmica. Em ambos os casos, tratava-se de sistemas lineares de equações, ou seja,

$$\mathbf{F}(\mathbf{x}) = A\mathbf{x} - \mathbf{b},$$

os quais resolvemos por meio de métodos diretos, como o escalonamento (por exemplo, `linalg.solve`), que fornecem a solução do problema em uma única etapa.

No entanto, esses não são os únicos métodos disponíveis para resolver tais problemas. Existem também os chamados métodos iterativos.

O que é um método iterativo?

Partindo de uma condição inicial $\mathbf{x}^{(0)}$, um método iterativo gera uma sequência de aproximações $\{\mathbf{x}^{(k)}\}$, $k = 1, 2, \ldots$, que converge para a solução exata $\mathbf{x}^\star$ do problema $\mathbf{F}(\mathbf{x}) = 0$, isto é,

$$\lim_{k \to \infty} \mathbf{x}^{(k)} = \mathbf{x}^\star. \tag{2.3}$$

A maioria dos métodos iterativos segue o seguinte esquema:

Método Iterativo Geral

Dados $\mathbf{x}^{(0)}$, $\mathbf{F}^{(0)} = \mathbf{F}(\mathbf{x}^{(0)})$, TOL, MAXIT, $k = 0$

Enquanto $\|\mathbf{F}^{(k)}\| >$ TOL e $k <$ MAXIT

1. Resolver $M\,\mathbf{d}^{(k+1)} = -\mathbf{F}^{(k)}$
2. Determinar o escalar β_{k+1}
3. Avançar: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \beta_{k+1}\,\mathbf{d}^{(k+1)}$
4. Calcular residual: $\mathbf{F}^{(k+1)} = \mathbf{F}(\mathbf{x}^{(k+1)})$
5. Incrementar k

Fim

em que a matriz M e o escalar β dependem do problema e do método específico.

2.6.1 Métodos iterativos para problemas lineares

Para problemas lineares da forma

$$\mathbf{F}(\mathbf{x}) = A\mathbf{x} - \mathbf{b},$$

a ideia central é escolher uma matriz M que seja uma "boa aproximação" de A, mas que seja mais simples de inverter[12] do que A.

12: Quando dizemos "mais simples", referimo-nos a menor custo computacional (por exemplo, menor tempo de execução ou menor uso de memória).

Naturalmente, se $M = A$, o método converge em uma única iteração. De fato, denotando por $\mathbf{x}^{(\star)}$ a solução do sistema, temos:

$$d^{(1)} = -A^{-1}\mathbf{r}^{(0)} = A^{-1}\left(\mathbf{b} - A\mathbf{x}^{(0)}\right) = A^{-1}\mathbf{b} - \mathbf{x}^{(0)} = \mathbf{x}^{(\star)} - \mathbf{x}^{(0)},$$

ou seja, a primeira direção já corresponde exatamente ao vetor que leva do chute inicial $\mathbf{x}^{(0)}$ até a solução $\mathbf{x}^{(\star)}$.

Dois métodos clássicos, com forte apelo pedagógico, são:

- **Jacobi**: $M = \text{diag}(A)$;
- **Gauss-Seidel**: $M = \text{tril}(A)$[13].

13: A parte triangular inferior de uma matriz, em Python, pode ser obtida com np.tril(A).

Em ambos os casos, considera-se $\beta_k = 1$ para todo k.

Para ilustrar o método de Jacobi, consideremos o sistema:

$$\begin{aligned} 3x - y &= 2, \\ -2x + 4y &= 1. \end{aligned}$$

O método de Jacobi parte de uma condição inicial $[x_0, y_0]$ e atualiza as incógnitas a cada iteração segundo as regras:

$$\begin{aligned} x_{k+1} &= \frac{1}{3}(2 + y_k), \\ y_{k+1} &= \frac{1}{4}(1 + 2x_k). \end{aligned}$$

A ideia é que, em cada equação, atualiza-se uma incógnita utilizando os valores das demais na iteração anterior. O processo iterativo é repetido até que x_k e y_k estejam suficientemente próximos da solução do sistema.

Como seria o procedimento no caso do método de Gauss-Seidel?

A seguir, propõem-se alguns exercícios adicionais para estudar o comportamento da placa térmica:

1. Implemente os métodos de Jacobi e Gauss-Seidel para resolver o problema da placa térmica. Procure desenvolver uma implementação o mais eficiente possível e compare o tempo de cálculo em função do tamanho da malha e da tolerância escolhida (TOL) para atingir a convergência.
2. Elabore uma animação que mostre a evolução do campo de temperaturas ao longo das iterações[14].
3. Os métodos iterativos não se limitam à resolução de sistemas lineares. Um método iterativo simples pode ser formulado para resolver o seguinte problema não linear:

   Qual é a temperatura T_C que deve ser aplicada na região circular de modo que:

   $$T_{\max} = \max_{i,j} T_{i,j} = T^{\star},$$

   sendo $T^{\star}$ o valor desejado para a temperatura máxima no sistema.

14: Para evitar um custo computacional elevado, inclua um *frame* a cada 10 ou 20 iterações. Além disso, não inclua a etapa de geração da animação nas medições de tempo, pois o salvamento de gráficos pode ser significativamente mais lento.

Nesse caso, define-se a função $F(T_C) = T_{\max} - T^\star$. Pode-se adotar a matriz identidade ($M = \mathbb{I}$) e escolher o coeficiente β como um valor próximo de 1. Implemente, então, o método iterativo geral para determinar o valor de T_C que permita atingir a temperatura máxima desejada $T^\star = 39.5°\text{C}$. Estude como a solução depende da malha utilizada. Para os demais parâmetros e condições, considere os valores nominais.

A MEMBRANA ELÁSTICA 3

3.1 Prelúdio

Existem diversos problemas em engenharia e matemática aplicada nos quais é necessário calcular os autovalores e autovetores de uma matriz. Alguns exemplos são apresentados nas figuras abaixo. Para compreender adequadamente esses problemas, é importante estudar os fundamentos dos métodos numéricos utilizados no cálculo de autovalores e autovetores.

Objetivos

- ↪ Revisar algumas noções de álgebra linear relacionadas ao problema de autovalores;
- ↪ Introduzir o **Método de Francis** para o cálculo do espectro completo de autovalores, baseado na fatoração QR;
- ↪ Implementar os algoritmos em Python e utilizar bibliotecas de métodos iterativos disponíveis no `scipy`;
- ↪ Realizar a modelagem computacional do problema de vibração de membranas.

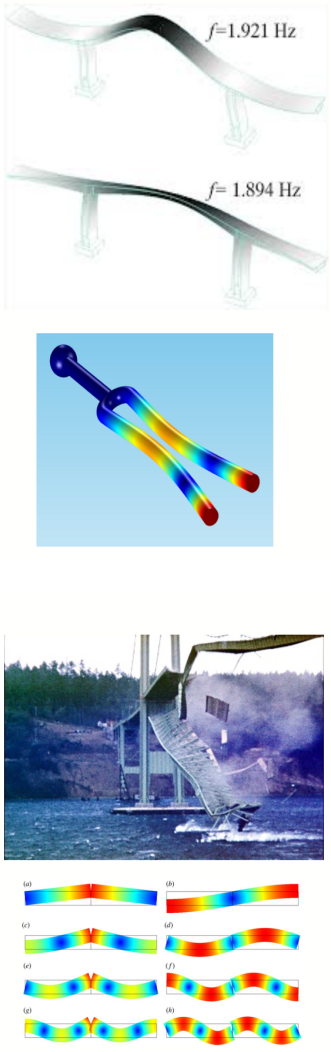


Figure 3.1: Exemplos de problemas que envolvem o cálculo de autovalores.

3.2 O problema de autovalores e autovetores

Seja uma matriz $A \in \mathbb{R}^{n\times n}$. Dizemos que um escalar $\lambda \in \mathbb{R}$ é um autovalor de A se existir um vetor $\mathbf{v} \in \mathbb{R}^n$ não nulo* tal que

$$A\,\mathbf{v} = \lambda\,\mathbf{v}.$$

Em outras palavras, buscamos vetores $\mathbf{v} \neq 0$ tais que a ação da matriz A sobre $\mathbf{v}$ resulte em um múltiplo escalar do próprio vetor $\mathbf{v}$.

Assim, se λ é um autovalor de A, então existe $\mathbf{v} \in \mathbb{R}^n$, com $\mathbf{v} \neq 0$, satisfazendo

$$(A - \lambda\,\mathbb{I})\,\mathbf{v} = 0, \tag{3.1}$$

em que $\mathbb{I}$ denota a matriz identidade de ordem n.

Pelos resultados da Álgebra Linear, sabemos que isso só é possível se a matriz $(A - \lambda\,\mathbb{I})$ for singular, isto é, não invertível. Consequentemente, seu determinante deve ser nulo:

$$P(\lambda) = \det(A - \lambda\,\mathbb{I}) = 0.$$

A expressão acima define um polinômio na variável λ, conhecido como *polinômio característico* da matriz A.

* Note que o caso $\mathbf{v} = 0$ não é de interesse, pois a equação é satisfeita trivialmente.

Exemplo 1

Considerar o seguinte exemplo:

$$A = \begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}$$

Então, calculando o polinômio característico

$$P(\lambda) = \det\left(\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = \det\left(\begin{bmatrix} 2-\lambda & 2 \\ 5 & -1-\lambda \end{bmatrix}\right)$$

Calculando o determinante, chegamos a:

$$P(\lambda) = \lambda^2 - \lambda - 12$$

cujas raizes (i.e., os λ tais que $P(\lambda) = 0$, são $\lambda_1 = -3$ e $\lambda_2 = 4$, que são os autovalores de A.

Em particular, se a matriz A é simétrica, isto é, $A^\intercal = A$, então podemos garantir que ela possui exatamente n autovalores[1].

1: O caso das matrizes simétricas é muito importante, pois aparece frequentemente na engenharia.

De fato, se $\mathbf{v}$ e $\mathbf{w}$ são autovetores associados a autovalores distintos λ e μ de uma matriz simétrica, isto é,

$$A, \mathbf{v} = \lambda, \mathbf{v}\, A, \mathbf{w} = \mu, \mathbf{w}$$

então $\mathbf{v}$ e $\mathbf{w}$ são ortogonais, ou seja,

$$\mathbf{v} \cdot \mathbf{w} = \mathbf{v}^T, \mathbf{w} = 0$$

Exemplo 2

Considerar o seguinte exemplo:

$$A = \begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

Então, calculando o polinômio característico

$$P(\lambda) = \det\left(\begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

Chegamos em que:

$$P(\lambda) = (\lambda + 1)^2 (\lambda - 8)$$

cujas raizes são $\lambda_1 = -1$, $\lambda_2 = -1$ e $\lambda_3 = 8$. Já, os autovetores correspondentes a eles são:

$$\mathbf{v}_{(1)} = [-0.4941, -0.47202, 0.73011]^\intercal$$

$$\mathbf{v}_{(2)} = [-0.55805, 0.81614, 0.14998]^\intercal,$$

$$\mathbf{v}_{(3)} = [0.6667, 0.3334, 0.6667]^\intercal,$$

onde é facil notar que os produtos escalares $\mathbf{v}_{(1)} \cdot \mathbf{v}_{(2)} = 0$ e $\mathbf{v}_{(1)} \cdot \mathbf{v}_{(3)} = 0$ e $\mathbf{v}_{(2)} \cdot \mathbf{v}_{(3)} = 0$, ou seja, são ortogonais.

Na sequência introduzimos um conceito muito importante:

Matrizes semelhantes

Duas matrizes A e B dizem-se semelhantes, se existe uma matrix Q não singular tal que:

$$A = Q\,B\,Q^{-1}$$

ou, de forma equivalente

$$B = Q^{-1}\,A\,Q$$

Então, se $(\lambda, \mathbf{v})$ é par Autovalor-Autovetor de A, $(\lambda, Q^{-1}\mathbf{v})$ será par Autovalor-Autovetor de B.

Outro conceito importante é o de matriz diagonalizável:

Matriz diagonalizável

Uma matriz diz-se diagonalizável se ela for semelhante a uma matriz diagonal, ou seja, existe Q tal que

$$A = Q\,D\,Q^{-1}$$

em que D é uma matriz diagonal, i.e.,

$$D = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

e pode-se demonstrar que se $A \in \mathbb{R}^{n\times n}$ for **simétrica**, então, existirá uma matriz Q ortogonal (i.e., $Q^{-1} = Q^T$), tal que

$$Q^T\,A\,Q = D = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

em que os $\lambda_i,\ i = 1, \dots, n$ são os autovalores de A.

Notemos que as colunas de Q são de fato os autovetores de A: Para ver isto, tomamos um vetor da base canónica, i.e., o vetor $\mathbf{e}_{(i)}$ está feito de zeros exceto na sua compenente i que vale 1:

$$\mathbf{e}_{(i)} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix}$$

Agora fazemos $(Q^T\, A\, Q)\, \mathbf{e}_{(i)} = D\, \mathbf{e}_{(i)} = \lambda_i\, \mathbf{e}_{(i)}$

Então:

$$(A\, Q\,)\, \mathbf{e}_{(i)} = Q\, (\lambda_i\, \mathbf{e}_{(i)}) = \lambda_i\, (Q\, \mathbf{e}_{(i)})$$

Agora chamemos $\mathbf{v}_{(i)} = Q\, \mathbf{e}_{(i)}$

$$A\, \mathbf{v}_{(i)} = \lambda_i\, \mathbf{v}_{(i)}$$

Isto significa que $\mathbf{v}_{(i)} = Q\, \mathbf{e}_{(i)}$ é o autovetor associado ao autovalor λ_i da matriz A. Note que $Q\, \mathbf{e}_{(i)}$ corresponde exatamente à i-ésima coluna da matriz Q, ou seja:

$$Q = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \mathbf{v}_{(1)} & \mathbf{v}_{(2)} & \dots & \mathbf{v}_{(n)} \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}$$

Em outras palavras, a matriz Q é formada ao "pendurar" os autovetores da matriz A.

Observação importante

O que vimos anteriormente nos mostra que, se dispusermos de um método para diagonalizar uma matriz, teremos então um caminho para calcular **todos** os seus autovalores e autovetores.

3.3 Método de Francis

Na sequência apresentamos um método iterativo que permite diagonalizar uma matriz. O método é baseado na decomposição ou fatoração QR da matriz, para o qual temos:

Teorema da fatoração QR

Toda matriz $A \in \mathbb{R}^{m \times n}$ $(m \geq n)$ possui uma fatoração

$$A = Q\, R$$

onde $Q \in \mathbb{R}^{m\times m}$ é ortogonal e $R \in \mathbb{R}^{m\times n}$ é triangular (trapezoidal) superior, com $r_{ii} \geq 0\ \forall i$.

$$A = QR = \begin{pmatrix} q_{11} & q_{12} & \dots & \dots & q_{1m} \\ q_{21} & q_{22} & \dots & \dots & q_{2m} \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ q_{m1} & q_{m2} & \dots & \dots & q_{mm} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & 0 & \dots & r_{nn} \\ - & - & - & - & - \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

Se em particular a matriz for quadrada ($m = n$), então fica:

$$A = QR = \begin{pmatrix} q_{11} & q_{12} & \dots & \dots & q_{1n} \\ q_{21} & q_{22} & \dots & \dots & q_{2n} \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ q_{n1} & q_{n2} & \dots & \dots & q_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ 0 & 0 & 0 & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & 0 & \dots & r_{nn} \end{pmatrix}$$

Não demonstraremos o teorema anterior, cuja prova pode ser realizada por meio de um processo conhecido como ortogonalização de vetores. Para os nossos propósitos, basta saber que o teorema é válido e, portanto, que qualquer matriz pode ser escrita como o produto de uma matriz Q ortogonal por uma matriz R triangular superior. Isso é suficiente para compreender o funcionamento do seguinte método iterativo:

3.3.1 Algoritmo iterativo para cálculo de autovalores

O processo iterativo funciona da seguinte forma: a ideia é gerar uma sequência de matrizes $\{A_1, A_2, \dots, A_k, \dots\}$ tal que:

- Definir a primeira matriz da sequência $A_1 = A$
- Calcular a fatoração QR de $A_1 \Rightarrow A_1 = Q_1 R_1$
- Calcular $A_2 = R_1 Q_1$
- Calcular a fatoração QR de $A_2 \Rightarrow A_2 = Q_2 R_2$
- Calcular $A_3 = R_2 Q_2$
- .
- .
- .
- Calcular $A_{k-1} = R_{k-2} Q_{k-2}$
- Calcular a fatoração QR de $A_{k-1} \Rightarrow A_{k-1} = Q_{k-1} R_{k-1}$
- Calcular $A_k = R_{k-1} Q_{k-1}$
- .
- .

...e assim sucessivamente, até que um determinado critério de parada seja atingido.

É possível provar duas propriedades fundamentais:

- as matrizes A e A_k possuem os mesmos autovalores;
- a matriz A_k converge para uma matriz diagonal, na qual os autovalores ficam explícitos.

Para verificar a primeira propriedade, consideremos que, em uma determinada iteração, temos $A_{k-1} = Q_{k-1} R_{k-1} \Rightarrow R_{k-1} = Q_{k-1}^\intercal A_{k-1}$. Como definimos $A_k = R_{k-1} Q_{k-1}$, segue que:

$$A_k = Q_{k-1}^\intercal A_{k-1} Q_{k-1}$$

Agora, procedemos de forma recursiva:

$$\begin{aligned}
A_k &= Q_{k-1}^\intercal A_{k-1} Q_{k-1} \\
&= Q_{k-1}^\intercal (Q_{k-2}^\intercal A_{k-2} Q_{k-2}) Q_{k-1} \\
&= Q_{k-1}^\intercal (Q_{k-2}^\intercal (Q_{k-3}^\intercal A_{k-3} Q_{k-3}) Q_{k-2}) Q_{k-1} \\
&\vdots \\
&= Q_{k-1}^\intercal Q_{k-2}^\intercal Q_{k-3}^\intercal \dots Q_1^\intercal A_1 Q_1 \dots Q_{k-3} Q_{k-2} Q_{k-1} \\
&= (Q_1 \dots Q_{k-3} Q_{k-2} Q_{k-1})^\intercal A (Q_1 \dots Q_{k-3} Q_{k-2} Q_{k-1}) \\
&= V^\intercal A V
\end{aligned}$$

em que $V = Q_1 \dots Q_{k-3} Q_{k-2} Q_{k-1}$, ou seja, as matrizes A_k e A são semelhantes à través da matriz V e por tanto possuem os mesmos autovalores, e como visto anteriormente, os autovetores serão as colunas da matriz V. Isto se traduz no seguinte algoritmo de diagonalização[2]:

Algoritmo de Francis

Dada A, MAX_ITER, TOL
Inicializar $k = 1$ e $A_1 = A, V_1 = \mathbb{I}$
Enquanto $\varepsilon > TOL$ e $k < MAX\_IT$

1. Calcular a fatoração $A_k = Q_k R_k$
2. Definir $A_{k+1} = R_k Q_k$
3. $V_{k+1} = V_k Q_k$
4. Calcular o erro: $\varepsilon = \max \|A_k^{ij}\|, i, j = 1, \dots, n,\ i \neq j$
5. Incrementar k

2: Este método também se conhece como algoritmo de Francis, ou simplesmente, algoritmo baseado na fatoração QR.

3.3.2 Problema de autovalores generalizado

Uma variante do clássico problema de autovalores é o chamado problema generalizado, que consiste em encontrar os vetores não nulos **v** e escalares

λ tais que:

$$A\,\mathbf{v} = \lambda\, M\,\mathbf{v} \tag{3.2}$$

em que M é uma matriz simétrica definida positiva[3] Em muitos casos, M será uma matriz diagonal com coeficientes positivos. Nessas condições, verifica-se a ortogonalidade dos autovetores de A em relação ao produto escalar **generalizado**, ou seja:

3: Uma matriz $M \in \mathbb{R}^{n\times n}$ é dita definida positiva se $\mathbf{x}^\intercal M\mathbf{x} > 0 \;\; \forall \mathbf{x} \neq 0$.

$$\mathbf{v}_{(i)}^\intercal\, M\, \mathbf{v}_{(j)} = \delta_{ij} \tag{3.3}$$

em que $\delta_{ij} = 0$ se $i \neq j$ e $\delta_{ij} = 1$ se $i = j$. Este problema tem interesse na resolução de problemas envolvendo oscilações de sistemas que possuim inercia, por isto a matriz M chame-se matriz de massas.

3.3.3 Cálculo de autovalores em `python`

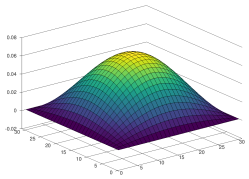
Em `python`, temos disponíveis dois métodos principais para resolver problemas de autovalores:

- **Matrizes densas**: O método baseado na fatoração QR que acabamos de apresentar, o qual fornece todos os autovalores e autovetores da matriz:

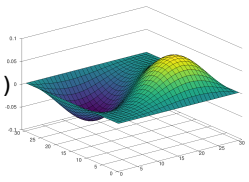
  ```
  Lam, Q = scipy.linalg.eigh(A, M)
  ```

  A versão para matrizes não simétricas é `scipy.linalg.eig`.

- **Matrizes esparsas**: Um método que ainda não discutimos, baseado no chamado método das potências. Este método é projetado para calcular apenas um número restrito de autovalores (por exemplo, os n maiores ou os n menores), o que geralmente é suficiente. Na análise de frequências de oscilação de estruturas ou de circuitos, por exemplo, em geral interessam apenas os chamados modos fundamentais.

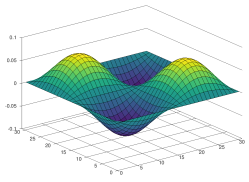
  

  ```
  Lam, Q = scipy.sparse.linalg.eigsh(K, k=20, M=M, which='SM')
  ```

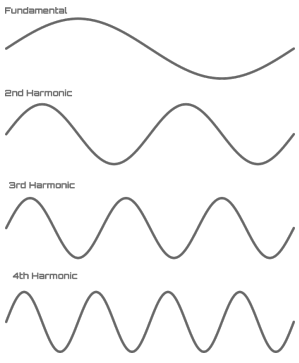
  em que `k` indica o número de autovalores desejado e `'SM'` denota *Smallest in magnitude* (menor magnitude), sendo a outra opção comum `'LM'` (*Largest in magnitude*).

  A versão para matrizes não simétricas é `scipy.sparse.linalg.eigs`.

Em ambos os casos, se a matriz `M` não for fornecida, assume-se o problema padrão de autovalores (ou seja, $M = \mathbb{I}$).

3.4 Vibração de membranas em `python`



Nesta parte, estudaremos uma aplicação do cálculo de autovalores ao problema da oscilação de membranas tensionadas. Trata-se de um corpo aproximadamente bidimensional que ocupa uma região do plano $x - y$ e está submetido a uma tensão. Nas figuras ao lado se mostram alguns modos fundamentais de oscilação de uma membrana de forma quadrada.

Outro exemplo é uma "membrana unidimensional" (uma corda tensionada), como mostrado também na figura ao lado.

A dinâmica da membrana decorre do princípio de conservação do momento linear, que é, essencialmente, a equação de Newton aplicada a uma membrana representada por um domínio Ω do plano. Assim, o problema de equilíbrio dinâmico pode ser formulado como uma equação de derivadas parciais, em que a incógnita é o deslocamento vertical $w(x, y, t)$ no ponto $(x, y) \in \Omega$ no instante t.

$$\begin{cases} \rho\, e\, \dfrac{\partial^2 w}{\partial t^2} - \sigma\, \nabla^2 w = f(x, y, t)\,, & (x, y) \in \Omega \subset \mathbb{R}^2 \\ w(x, y, t) = 0, & (x, y) \in \partial\Omega \\ w(x, y, 0) = u_0(x, y) & (x, y) \in \Omega \\ \dfrac{\partial w}{\partial t}(x, y, 0) = v_0(x, y) & (x, y) \in \Omega \end{cases}$$

em que conhecemos:

- $\partial\Omega$ denota a borda do dominio Ω;
- $\rho(x, y)$ é a densidade do material da membrana;
- $e(x, y)$ é a espessura da membrana;
- σ é a tensão membranal (em N/m);
- $f(x, y, t)$ é a força vertical aplicada (em $\mathrm{N/m^2}$);
- ∇^2 é o operador Laplaciano (para modelizar as forças internas), o qual em duas dimensões é dado por:

$$\nabla^2 w = \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}$$

As duas últimas equações representam as condições iniciais, para a posição e a velocidade da membrana.

Para resolver este problema, em geral, precisamos introduzir uma discretização do domínio Ω, assim como foi feito no problema de transferência de calor. Vamos considerar um domínio retangular e uma grade regular tal como mostrado na figura:

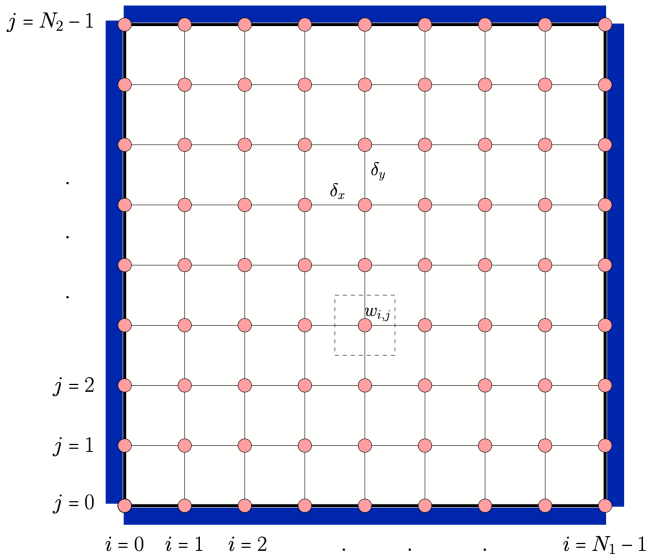


Figure 3.2: Domínio discretizado para o problema de oscilação de membranas.

A ideia é escrever a equação para cada parcela de membrana identificada pelos indices i, j

$$\rho_{ij}\, e_{ij}\, \frac{d^2 w_{ij}}{dt^2} - \sigma\, \nabla^2_{ij} w = f_{ij}(t)$$

O ponto agora é como expressar o Laplaciano no caso discreto. Uma possibilidade é dada pelo operador de diferenças de segundo grau

$$\nabla^2_{ij} w = \frac{w_{i+1,j} + w_{i-1,j} + w_{i,j+1} + w_{i,j-1} - 4w_{ij}}{h^2}$$

resultando na equação para o ponto i, j

$$\rho_{ij}\, e_{ij}\, \frac{d^2 w_{ij}}{dt^2} - \sigma\, \frac{w_{i+1,j} + w_{i-1,j} + w_{i,j+1} + w_{i,j-1} - 4w_{ij}}{h^2} = f_{ij}(t)$$

Agora transformamos os "arrays" em vetores, por exemplo:

$$W_k = w_{ij} \qquad \Leftrightarrow \qquad \texttt{k = i + j*N1}$$

onde N_1 é o número de nós em x, dando

$$\rho_k\, e_k\, \frac{d^2 w_k}{dt^2} - \sigma\, \frac{w_{k_e} + w_{k_w} + w_{k_n} + w_{k_s} - 4w_k}{h^2} = f_k(t)$$

em que k_e (east), k_w (west), k_n (north), k_s (south) são os indices globais dos pontos vizinhos do ponto k. Notar que os w_k ainda são funções do tempo. Em notação vetorial podemos escrever o seguinte sistema de ODEs acoplado[4]:

$$M\, \frac{d^2 \mathbf{w}(t)}{dt^2} + K\mathbf{w}(t) = F(t)$$

em que M é chamada de matriz de massas e K de matriz de rigidez da membrana e vem dadas por:

$$M = \begin{bmatrix} \rho_1 e_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \rho_2 e_2 & \dots & 0 & \dots & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & \dots & \rho_k e_k & \dots & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & \dots & 0 & \dots & \rho_n e_n \end{bmatrix}$$

$$K = \frac{\sigma}{h^2} \begin{bmatrix} . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & -1 & \dots & -1 & 4 & -1 & \dots & -1 & . & . \\ . & . & -1 & \dots & -1 & 4 & -1 & \dots & -1 & . \\ . & . & . & -1 & \dots & -1 & 4 & -1 & \dots & -1 & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \end{bmatrix}$$

que é essencialmente a mesma matriz pentadiagonal que temos visto em outras ocasiões. A implementação em python pode ser feita como mostrado no código da sequência[5].

4: Notar que este é um sistema de equações ordinárias de segunda ordem de grande porte.

5: Notar que na implementação da matriz K estamos usando o comando `scipy.sparse.diags`, que já cria uma matriz pentadiagonal em formato esparso.

```
def BuildMatrizesEigen(N1, N2, sigma, rho, e, h):
    nunk = N1*N2

    # Stiffness matrix K: Build it as a sparse matrix
    d1 = 4.0*np.ones(nunk)
    d2 = -np.ones(nunk-1)
```

```
    d3 = -np.ones(nunk-N1)
    K = (sigma/h**2)*scipy.sparse.diags([d3, d2, d1, d2, d3],
              [-N1, -1, 0, 1, N1], format='csr')

    # Force the eigenvalues associated to restricted points
    # to be a big number as compared to fundamental modes
    big_number = 10000
    Iden = big_number*scipy.sparse.identity(nunk, format='csr')

    # Lados verticais
    for k in range(0,N2):
        Ic = ij2n(0,k,N1) # Left
        K[Ic,:], K[:,Ic] = Iden[Ic,:], Iden[:,Ic]

        Ic = ij2n(N1-1,k,N1) # Right
        K[Ic,:], K[:,Ic] = Iden[Ic,:], Iden[:,Ic]

    # Lados horizontais
    for k in range(0,N1):
        Ic = ij2n(k,0,N1) # Bottom
        K[Ic,:], K[:,Ic] = Iden[Ic,:], Iden[:,Ic]

        Ic = ij2n(k,N2-1,N1) # Top
        K[Ic,:], K[:,Ic] = Iden[Ic,:], Iden[:,Ic]

    # Processar os pontos da mascara para uma membrana circular
    ...
    ...

    # Mass matrix: Simple case, multiple of identity
    M = rho*e*scipy.sparse.identity(nunk, format='csr')

   return K, M
```

Observação importante

No código apresentado, observa-se uma particularidade na estrutura da matriz K. Seguindo a abordagem adotada anteriormente, incorporamos as restrições de movimento diretamente na construção da matriz. Para os pontos da grade com movimento restrito, substituímos a linha e a coluna correspondentes por linhas e colunas da matriz identidade, multiplicadas por um fator denominado `big_number`. Essa estratégia numérica garante que cada ponto restrito gere um autovalor igual ao coeficiente posicionado na diagonal. Como se trata de um artifício para impor a restrição física da membrana, esses autovalores não possuem significado físico. Portanto, é conveniente que seus valores sejam numericamente distantes das frequências de oscilação reais. Para assegurar esse isolamento, define-se o `big_number` com um valor elevado, como 10^4.

Uma vez que temos as matrizes K e M podemos proceder de duas formas:

- Podemos resolver o sistema de ODEs usando algum método numérico, i.e., discretizando também a variável temporal[6].
- ou, podemos calcular os autovalores e os autovetores do problema generalizado e escrever a solução de forma explícita, como explicamos na sequência

6: Isto será estudado posteriormente

Evolução do sistema (oscilações livres)

Considerando o caso com $f = 0$ (sem forças aplicadas). A evolução do sistema

$$M\,\mathbf{w}'' + K\,\mathbf{w} = 0 \tag{3.4}$$

a partir de uma condição inicial arbitrária para o deslocamente $\mathbf{w}(0) = U$ e para a velocidade $\mathbf{w}'(0) = V$ é:

$$\mathbf{w}(t) = \sum_{k=1}^{n} \Phi^{(k)}\, c_k\, \sin(\omega_k t + \phi_k) \tag{3.5}$$

em que os vetores $\Phi^{(k)}$ e as frequências ω_k são solução do problema de autovalores

$$K\,\Phi^{(k)} = \omega_k^2\, M\,\Phi^{(k)}$$

- Notar que a relação entre os autovalores e as frequências é: $\omega_k = \sqrt{\lambda_k}$.
- Os vetores $\Phi^{(k)}$ são os **modos naturais** da estrutura, e os ω_k são as **frequências naturais** dela.
- Os coeficientes c_k e ϕ_k, de amplitude e fase, são definidos pela condição inicial. Notar em particular que, se a condição inicial é proporcional a um dos vetores da base (ou seja, a excitação inicial tem uma forma proporcional ao modo natural de oscilação j), então a solução do sistema será:

  $$\mathbf{w}(t) = \Phi^{(j)}\, \sin(\omega_j t + \phi_j)$$

  então, a membrana ficará oscilando com essa forma na frequência ω_j, indefinidamente, já que, pela equação (3.3), $\Phi^{(j)\mathsf{T}}\, M\,\Phi^{(k)} = 0$ se $k \neq j$ e só é distinto de zero se $k = j$.

Evolução do sistema (oscilações forçadas amortecidas)

Primeiro, vamos colocar um amortecimento na equação diferencial proporcional à matriz de massa[7]:

$$M\,\frac{d^2W}{dt^2} + \beta\, M\,\frac{dW}{dt} + K\,W \;=\; F(t) \tag{3.6}$$

Agora, assumamos agora que $F(t)$ é cosenoidal,

$$F(t) = Z\,\cos(\omega_* t)\,, \tag{3.7}$$

e analizemos a resposta da membrana postulando uma solução da forma

$$W(t) = \sum_i c_i\,\cos(\omega_* t + \phi_i)\,\Phi^{(i)}\,. \tag{3.8}$$

A solução do problema homogêneo (com lado direito nulo) poderia ser adicionada, mas, esta tende para zero quando $t \to \infty$ e por tanto prevalece a solução particular que acabamos de formular.

7: Isto torna o problema mais realista, pois na prática sempre haverá forçãs de atrito.

Para obter os coeficientes, primeiro calculamos as derivadas:

$$W'(t) = -\sum_i c_i\, \omega_*\, \sin(\omega_* t + \phi_i)\, \Phi^{(i)}$$

$$W''(t) = -\sum_i c_i \omega_*^2\, \cos(\omega_* t + \phi_i)\, \Phi^{(i)}$$

Assim resulta

$$K\,W = \sum_i c_i\, \cos(\omega_* t + \phi_i)\, K\,\Phi^{(i)} = \sum_i c_i\, \omega_i^2\, \cos(\omega_* t + \phi_i)\, M\,\Phi^{(i)}$$

e substituindo,

$$\sum_i c_i \left[(-\omega_*^2 + \omega_i^2)\cos(\omega_* t + \phi_i) - \beta\, \omega_* \sin(\omega_* t + \phi_i)\right]\, M\,\Phi^{(i)} = \\ = Z\, \cos(\omega_* t)$$

No lado direito, decompomos Z na base[8] $M\,\Phi^{(i)}$,

$$Z = \sum_i \alpha_i\, M\,\Phi^{(i)}$$

e utilizamos

$$\cos(\omega_* t) = \cos\phi_i\, \cos(\omega_* t + \phi_i) + \sin\phi_i\, \sin(\omega_* t + \phi_i)$$

para obter, igualando coeficientes,

$$c_i = \frac{\alpha_i\, \cos\phi_i}{-\omega_*^2 + \omega_i^2} = \frac{-\alpha_i\, \sin\phi_i}{\beta\, \omega_*}\ .$$

Eliminando ϕ_i, resulta

$$c_i = \frac{\alpha_i}{\sqrt{(-\omega_*^2 + \omega_i^2)^2 + \beta^2 \omega_*^2}} \tag{3.9}$$

8: Isto é claramente possível, pois como sabemos os autovetores formam uma base de $\mathbb{R}^n$

A energia elástica da membrana é

$$E_e(t) = \frac{1}{2} W(t)^T\, K\, W(t) = \frac{1}{2}\, \sum_i c_i^2\, \omega_i^2\, \cos^2(\omega_* t + \phi_i)$$

o que permite calcular a energia elástica média do modo sobre um periodo como

$$A_e = \frac{1}{4}\, \sum_i c_i^2\, \omega_i^2\ . \tag{3.10}$$

3.5 A membrana elástica do GD

Considerando aplicações em microfluídica, assumiremos que o material da membrama é um elastômero.

Para fixar valores numéricos, consideraremos:

- **Dimensões:** Membrana circular de raio 0.4 cm;

- **Espessura:** 0.1 mm
- **Tensão:** $\sigma = 200\ \mathrm{N}\cdot\mathrm{m}^{-1}$
- **Densidade:** $900\ \mathrm{Kg}\cdot\mathrm{m}^{-3}$

Para evitar instabilidades numéricas decorrentes das escalas e ordens de grandeza dos parâmetros físicos envolvidos, recomenda-se a adimensionalização do sistema. Esse procedimento permite que os cálculos sejam realizados utilizando as matrizes K e M em uma forma adimensional. Para tanto, definem-se as seguintes variáveis:

$$\hat{x} = \frac{x}{R}, \quad \hat{w} = \frac{w}{w_0}, \quad \hat{t} = \frac{t}{R}\sqrt{\frac{\sigma}{\rho\, e}}$$

A partir dessas definições, obtém-se a equação de onda adimensional:

$$\frac{\partial^2 \hat{w}}{\partial \hat{t}^2} - \hat{\nabla}^2 \hat{w} = \hat{f}$$

na qual a força normalizada é expressa por $\hat{f} = \dfrac{f \cdot R^2}{\sigma \cdot w_0}$. É importante observar que, na implementação numérica, o domínio adimensional terá dimensões $\hat{L}_x \times \hat{L}_y = \frac{L_x}{R} \times \frac{L_y}{R}$, o que resulta em um raio unitário para a membrana ($\hat{R} = 1$).

Após a obtenção dos autovalores adimensionais $\hat{\omega}_k$, as frequências físicas em Hertz (Hz) podem ser recuperadas pela relação:

$$f_k = \frac{\hat{\omega}_k}{2\pi} \cdot \frac{1}{R}\sqrt{\frac{\sigma}{\rho\, e}}$$

3.5.1 Investigando o comportamento do sistema

1. Implementar o caso de uma membrana de forma circular, empregando os parâmetros indicados acima. Como sugerido no código acima, e feito em casos anteriores, para ristringir os pontos fora do circulo, será possível definir uma mascara.
2. Calcular as frequências de oscilação e plotar os primeiros 10 modos fundamentais da membrana. Considere diferentes discretizações (N_x, N_y), por exemplo,

   $$(21, 21),\ (41, 41),\ (61, 61),\ (81, 81),\ (101, 101)$$

   Mostrar numa tabela os valores das frequências de oscilação como função da discretização.
3. (Exo. teórico) Fazer os cálculos necessários para obter as formulas analíticas que permitam calcular os coeficientes c_i e ϕ_i na equação (3.5).
4. Considerar o problema de oscilações forçadas com o termo forçante (adimensional) dado por:

   $$\hat{f}(\hat{x}, \hat{y}, \hat{t}) = [(\hat{x} - 0.5)^2 + (\hat{y} - 0.5)^2] \cos(\hat{\omega}_* \hat{t})$$

   Obter a representação desse termo forçante na base de autovetores generalizados $M\,\Phi^{(i)}$. Para isto será necessário encontrar

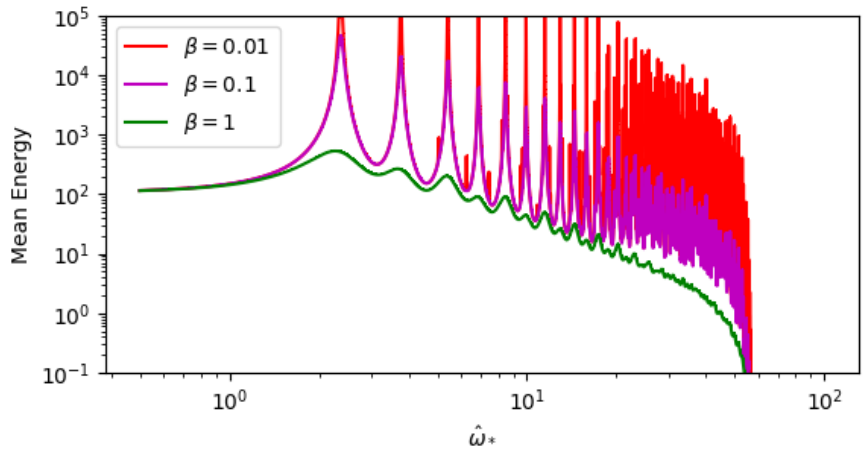


Figure 3.3: Energia média da membrana como função da frequência do termo forçante ω_*.

o vetor de $\mathbb{R}^n$ cujas componentes são $(\hat{x}_{i,j} - 0.5)^2 + (\hat{y}_{i,j} - 0.5)^2$.

5. Para o caso do ponto anterior, plotar a função que dá a energia elástica média para diferentes valores de $\hat{\omega}_*$ (pode trabalhar usando as varáveis adimensionais por simplicidade). Fazer gráficos considerando $\beta = 0.01,\ 0.1,\ 1$ e tirar conclusões. Considerar $\hat{\omega}_* \in [0.5, 100]$. Usar escala `loglog` para realizar os plots. O resultado deverá ser parecido com o mostrado na figura.

ACOPLAMENTO HIDRÁULICO-TÉRMICO

4

4.1 Preludio

Neste capítulo, são estudados os acoplamentos entre a rede hidráulica e a placa térmica. Primeiramente, considera-se a influência da temperatura na rede, oportunidade em que são introduzidos conceitos fundamentais de integração numérica. Em seguida, analisa-se o impacto da rede na distribuição de temperaturas da placa. Em ambos os casos, assume-se um acoplamento fraco entre os sistemas (*one-way coupling*), investigando-se o efeito de uma física mantendo a outra fixa. Por fim, se introduzirmos uma física mais complexa na rede de microcanais, poderá ser necessário resolver o acoplamento forte (*two-way coupling*), avaliando a influência mútua e simultânea entre ambos os sistemas.

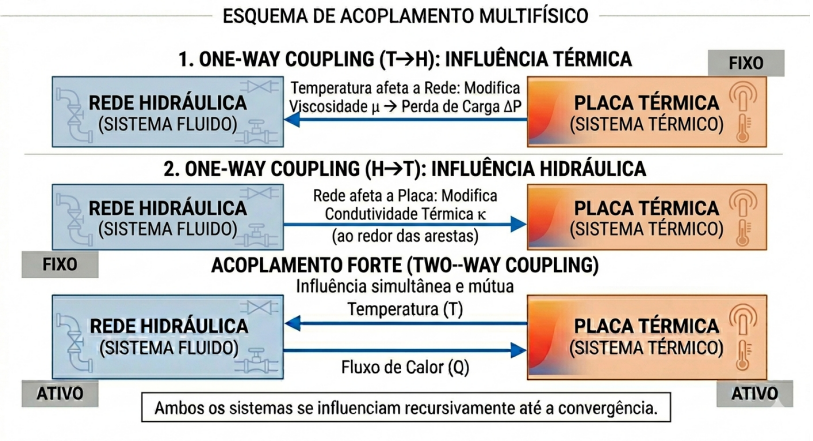


Figure 4.1: *one-way coupling* and *two-way coupling*.

Consideramos inicialmente uma rede com as seguintes características:

Rede hidráulica

- **Canais:** $A_k = 500\,\mu\text{m} \times 500\,\mu\text{m} = 2.5 \times 10^{-7}\,\text{m}^2$;
- **Inlet 0:** $Q_{\text{in}}^0 = 0.1\,\text{mL/s} = 10^{-7}\,\text{m}^3/\text{s}$;
- **Inlet 175:** $Q_{\text{in}}^{175} = 1\,\text{mL/s} = 10^{-6}\,\text{m}^3/\text{s}$;
- **Outlet:** $p_{\text{atm}} = 0$
- **Viscosidade do fluido:**

$$\mu(T) = \frac{0.001791}{1 + 0.03368T + 0.000221T^2},$$

  onde T está em °C e μ em Pa · s.
- **Nivel de complexidade:** 3

e uma placa com as seguintes características:

Placa térmica

- **Dimensões:** $L_x \times L_y = 3\text{cm} \times 1.5\text{cm}$;
- **Condutividade térmica da matriz:** $k = 0.25\ \text{W} \cdot \text{K}^{-1} \cdot \text{m}^{-1}$
- **Fonte de calor:** $5 \times 10^5\ \text{W} \cdot \text{m}^{-3}$
- $T_L = 10°\text{C}$, $T_R = 30°\text{C}$, $T_B = T_T = 10 + 20\,\dfrac{x}{L_x}$
- Região circular de raio $R = 0.25$ cm, centrada em $(2 + R,\ 0.75)$ cm, na qual a temperatura é fixada em $T_C = 35°\text{C}$.

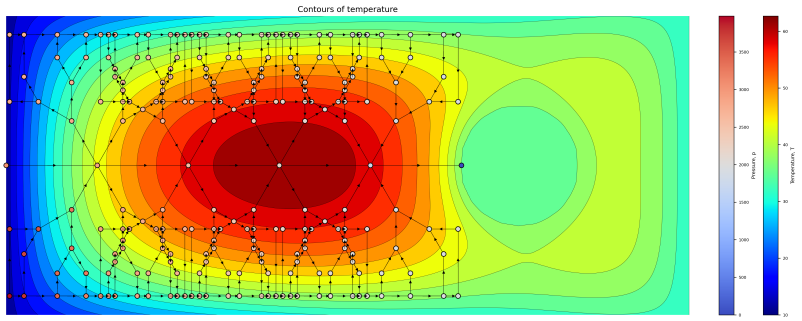


Figure 4.2: Rede hidráulica embutida na placa térmica.

4.2 Influência da temperatura na rede hidráulica

Nesta primeira etapa, analisa-se o efeito da temperatura na viscosidade do fluido e, consequentemente, no fator de perda de carga ao longo dos microcanais. Por simplicidade, adota-se uma viscosidade fixa por cada aresta, baseada na temperatura média da aresta. Dessa forma, para a k-ésima aresta, determina-se a seguinte grandeza:

$$\langle T_k \rangle = \frac{1}{\ell_k} \int_0^{\ell_k} T_k\left(\mathbf{p}(s)\right)\, ds \tag{4.1}$$

em que ℓ_k representa o comprimento da aresta e s é a variável de parametrização do trajeto do microcanal. Explicitamente, para um canal k que conecta o ponto $\mathbf{p}_i$ ao ponto $\mathbf{p}_j$, com índices de conectividade dados por $i = \texttt{conec}(k, 0)$ e $j = \texttt{conec}(k, 1)$, um ponto arbitrário $\mathbf{p}(s)$ ao longo da aresta é definido por:

$$\mathbf{p}(s) = \mathbf{p}_i + \frac{s}{\ell_k}\left(\mathbf{p}_j - \mathbf{p}_i\right)$$

sendo $\ell_k = \|\mathbf{p}_j - \mathbf{p}_i\|_2$. Rigorosamente, a função composta deveria ser denotada por $T\left(\mathbf{p}(s)\right) = \widetilde{T}(s)$; no entanto, por conveniência e simplicidade de notação, adota-se apenas $T(s)$. Como a função de temperatura não possui forma analítica conhecida ao longo do domínio, o cálculo da integral requer o emprego de métodos de aproximação numérica. Entre as alternativas disponíveis, destacam-se:

- **Regra do ponto médio:** Consiste em aproximar a função $T_k(s)$ por uma constante ao longo da aresta[1], isto é:

$$T_k(s) \approx T_k\left(s = \frac{\ell_k}{2}\right) \quad \Rightarrow \quad \int_0^{\ell_k} T_k\left(\mathbf{p}(s)\right)\, ds \approx T_k\left(s = \frac{\ell_k}{2}\right) \ell_k$$

1: Denominada interpolada de ordem 0, trata-se de uma função constante que coincide com a função original no ponto médio do intervalo.

A regra do ponto médio composta baseia-se na subdivisão do segmento de integração em subintervalos, aplicando-se a regra unidimensional em cada um deles. Ao dividir o intervalo $[0, \ell_k]$ em N subintervalos de comprimento $\Delta s = \ell_k/N$, definem-se os pontos de quadratura como $s_n = \left(n - \frac{1}{2}\right)\Delta s$ para $n = 1, \ldots, N$, de modo que:

$$\int_0^{\ell_k} T_k\left(\mathbf{p}(s)\right)\, ds \approx \sum_{n=1}^{N} T_k\left(s_n\right)\, \Delta s \tag{4.2}$$

- **Regra do trapézio:** Consiste em aproximar a função $T_k(s)$ por uma função linear ao longo da aresta[2], isto é:

$$T_k(s) \approx T_k(0) + s\,\frac{T_k(\ell_k) - T_k(0)}{\ell_k} \;\Rightarrow\; \int_0^{\ell_k} T_k\left(\mathbf{p}(s)\right)\, ds \approx \ell_k\,\frac{T_k(0) + T_k(\ell_k)}{2}$$

Geometricamente, a integral é aproximada pela área de um trapézio, o que justifica a nomenclatura do método. De maneira análoga, define-se a regra do trapézio composta (ilustrada na porção inferior da Figura 4.3), cujo desenvolvimento formal segue de forma direta a partir do mapeamento dos subintervalos.

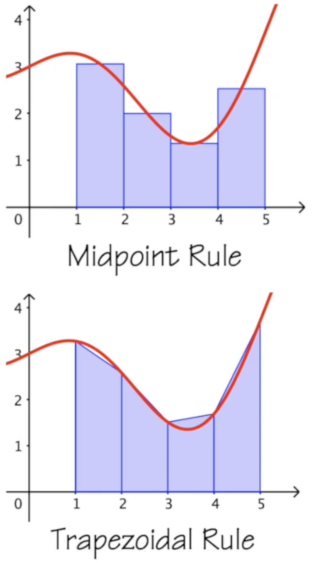


Figure 4.3: Ilustração das regras do ponto médio e do trapézio na forma composta.

Cálculo das temperaturas

Para o cálculo das integrais mencionadas, é necessário avaliar a temperatura na localização exata dos pontos $\mathbf{p}(s)$ ao longo das arestas da rede de microcanais. No entanto, devido à discretização numérica do problema de condução de calor, os valores do campo de temperaturas são conhecidos apenas nos nós da grade computacional, os quais, em geral, não coincidem com as posições $\mathbf{p}(s)$.

Dessa forma, faz-se uso de uma função de interpolação bidimensional para estimar a temperatura em coordenadas arbitrárias do domínio $[0, L_x] \times [0, L_y]$. Para essa finalidade, pode-se empregar o módulo `scipy.interpolate` do Python, conforme ilustrado no código abaixo:

```
from scipy.interpolate import RegularGridInterpolator

data = T.reshape(myPlaca.Ny, myPlaca.Nx).T
interp = RegularGridInterpolator((x, y), data, method='linear')
# Avaliar num determinado ponto
pts = np.array([[0.015, 0.005]])
print(interp(pts))
```

Esse procedimento gera um interpolador local com base nos valores de temperatura disponíveis na vizinhança do ponto de interesse. Dependendo do método selecionado, a aproximação pode ser do tipo `linear`, `cubic`, `nearest`, entre outras[3]. Propriamente, se o interpolador for avaliado em uma coordenada que coincide exatamente com um nó da grade original, o valor retornado será o da temperatura nodal correspondente; contudo, em pontos arbitrários intermediários, haverá um erro associado ao processo de interpolação.

4.2.1 Investigando o comportamento do sistema

1. Deduzir e escrever a expressão análoga a (4.2) para o caso da

2: Denominada interpolada de ordem 1, define-se como uma reta que coincide com a função original nos pontos extremos do intervalo.

3: No caso de métodos polinomiais (`linear` e `cubic`), o interpolador constrói um polinômio bidimensional $p(x, y)$ a ser avaliado no ponto desejado. Já a abordagem `nearest` (vizinho mais próximo) simplesmente atribui ao ponto (x, y) o valor de temperatura do nó da grade que lhe for geometricamente mais próximo.

regra do trapézio composta.

2. Implementar o interpolador bidimensional para estimar o campo de temperaturas em coordenadas arbitrárias do domínio. Para tanto, adote como referência a solução do problema térmico obtida em uma malha de 241×121 pontos. A partir dessa solução, defina uma grade secundária mais grosseira e avalie as temperaturas em suas posições nodais. Apresente os mapas de contorno (curvas de nível) nessa nova grade utilizando, comparativamente, as opções de interpolação `linear`, `cubic` e `nearest`. Em seguida, repita o procedimento tomando como base a solução do problema térmico resolvida em uma malha de 61×31 pontos e discuta as diferenças observadas.
   Utilizando a mesma metodologia, calcule a temperatura nas posições correspondentes aos nós da rede hidráulica e gere uma visualização do grafo da rede, mapeando a temperatura calculada como uma escala de cores nos nós.
3. Calcular a temperatura média em cada aresta da rede empregando as regras de quadratura apresentadas anteriormente. Avalie inicialmente as versões simples (com apenas um intervalo de integração) e, na sequência, as versões compostas considerando subdivisões de $10, 100$ e 1000 subintervalos por aresta. Compare os resultados numéricos e quantifique o tempo de computação associado a cada configuração. Adicionalmente, proponha uma estratégia matemática que permita estimar o erro de integração cometido em cada cenário.
   Por fim, plote o grafo da rede hidráulica, mapeando os valores obtidos de temperatura média em uma escala cromática aplicada diretamente sobre as arestas.
4. Resolver o circuito hidráulico da rede, atualizando a condutância de cada aresta em função das respectivas temperaturas médias calculadas. Avalie quantitativamente de que maneira a escolha da malha térmica (refinada vs. grosseira) e a regra de quadratura adotada influenciam as variáveis globais do sistema, especificamente a pressão máxima da rede e a potência total consumida.
5. Em vez de calcular a temperatura média das arestas, que outra coisa poderia ser feito para calcular uma viscosidade média das arestas?

4.3 Influência dos microcanais na placa térmica

A rede de microcanais impacta o comportamento térmico da placa por meio de dois mecanismos principais:

- Modificação da condutividade térmica na vizinhança dos canais;
- Introdução de um termo fonte (geração) ou sumidouro (extração) de calor na placa, dinâmica que pode ser decorrente, por exemplo, de reações químicas no fluido que escoa pelos microcanais.

4.3.1 Influência na condutividade

Inicialmente, aborda-se a alteração local da condutividade térmica. Para tanto, é necessário identificar quais canais da rede hidráulica encontram-

se geometricamente próximos ao ponto onde a condutividade da placa será avaliada durante a montagem da matriz do sistema linear.

Conforme ilustrado na Figura 2.3, a condutividade térmica é um parâmetro central na definição dos fluxos numéricos nas interfaces. Assim, para cada nó (i, j) da malha, o coeficiente k deve ser estimado nas posições de interface: $w : (x_i - \frac{1}{2}h, y_j)$, $e : (x_i + \frac{1}{2}h, y_j)$, $s : (x_i, y_j - \frac{1}{2}h)$ e $n : (x_i, y_j + \frac{1}{2}h)$.

Adota-se a premissa física de que apenas os microcanais posicionados dentro de um raio de distância limite $d_{\max}$ em relação ao ponto de interesse exercem influência sobre ele. Dado que tanto a grade de discretização da placa quanto a rede hidráulica podem conter milhares de nós e elementos, essa busca espacial deve ser implementada de forma computacionalmente eficiente. Para essa finalidade, utiliza-se uma rotina baseada em estruturas de dados espaciais, a qual mapeia e armazena os canais situados abaixo da distância de corte $d_{\max}$ para cada coordenada da grade. O procedimento para a chamada e execução dessa função é apresentado a seguir:

```
# Geometria
Lx, Ly = 0.03, 0.015
Nx, Ny = 241, 121
RR = 0.0025 # raio da inclusao
spine_length = 6
factor_units = 0.001
complex_level = 3
Xno, conec = generate_graph_arrays(complex_level, spine_length)
Xno = Xno*factor_units
# Deslocar em y para centralizar a rede
Xno[:,1] += 0.5*Ly
xnout = factor_units*(spine_length-1)*4
# Centro do circulo tal que o no da saida espete justo a inclusao
xincl, yincl = xnout + RR, 0.5*Ly
d_max = ...
mapa_proxim = CreateMapDistance(Lx, Ly, Nx, Ny, Xno, conec, d_max)
```

A estrutura `mapa_proxim` armazena, para cada nó da grade computacional, uma lista contendo os identificadores das arestas da rede hidráulica vizinhas e suas respectivas distâncias euclidianas. Estruturalmente, o retorno configura-se como uma lista de listas (ou dicionário indexado pelos nós), mapeada da seguinte forma:

```
...
10: [(205, np.float64(0.0009369099909052838)), (206, np.float64(0.0005386004333789896)), ...]
...
30: [(43, np.float64(0.0007929360811527958)), (44, np.float64(0.0009690546379224702)), ...]
...
50
[]
....
70
[]
...
100
[(189, np.float64(0.0001479571311137604)), (190, np.float64(0.0007807696311137608)),  ...]
...
```

O código anterior indica por exemplo, que o ponto de índice global `10` na grade, está próximo das arestas `205, 206`, etc, o ponto `50` não tem arestas próximas o suficiente, e assim por diante.

Uma vez definida essa estrutura, torna-se possível calcular a condutividade térmica modificada em cada interface por meio da seguinte expressão:

$$k_f = k_0 \left(1 + \sum_{j \in \mathcal{V}_f} \frac{1}{1 + d_j}\right)$$

em que $k_0 = 0.25$ representa a condutividade nominal do material original da placa. Desse modo, a condutividade térmica na face f associada ao volume de controle (i, j) assumirá um valor perturbado em função da proximidade com o circuito hidráulico.

Na formulação acima, $\mathcal{V}_f$ denota o conjunto de todas as arestas da rede que se encontram na vizinhança do ponto médio da face f, e d_j representa a menor distância euclidiana entre esse ponto médio e a correspondente aresta j da rede.

Por exemplo, para o cálculo da condutividade na face leste do elemento (i, j), localizada em $e : (x_i + \frac{1}{2}h, y_j)$, deve-se considerar a contribuição de todas as arestas da rede hidráulica que estejam próximas tanto do nó (i, j) quanto do nó vizinho $(i + 1, j)$, procedendo-se de maneira análoga para as demais direções cardeais.

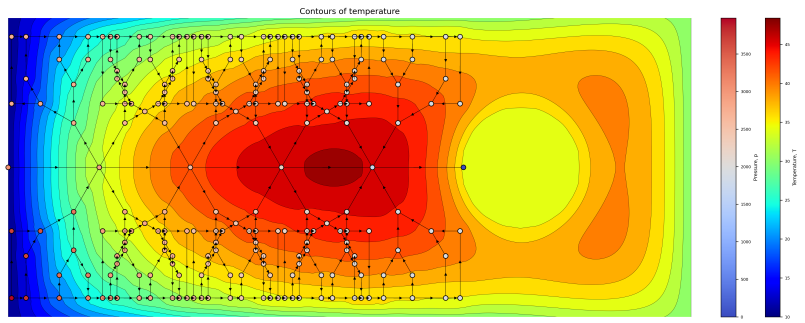


Figure 4.4: Rede hidráulica embutida na placa térmica considerando a influência da rede de microcanais na condutividade térmica do material da placa.

4.3.2 Influência no termo fonte e sumidouro

Considera-se, agora, o cenário em que o escoamento no interior dos microcanais é acompanhado por reações químicas que liberam ou absorvem energia térmica. Fisicamente, esse fenômeno requer a inclusão de uma parcela adicional ao termo fonte original do problema de condução na placa, cuja intensidade também é modelada em função da distância geométrica entre os nós da grade e os microcanais. Para essa finalidade, adota-se a seguinte formulação baseada em um perfil gaussiano:

$$S_p = S_0 \sum_{j \in \mathcal{V}_p} I_j \cdot \exp\left(-\frac{d_j^2}{2\sigma^2}\right)$$

em que $\mathcal{V}_p$ representa o conjunto de arestas da rede hidráulica situadas dentro do raio de corte $d_{\max}$ em relação ao ponto p da grade, e S_0 é a intensidade base da fonte.

O parâmetro de espalhamento é definido como $\sigma = \frac{d_{\max}}{2}$, o que implica que a uma distância igual a $d_{\max}$ a contribuição da aresta decai para $\exp(-2) \approx 0.135$ (cerca de 13.5% do seu valor de pico), tornando-se desprezível à medida que se aproxima do limite externo de influência.

O coeficiente I_j quantifica a intensidade específica da fonte na aresta j. No cenário ideal simplificado, assume-se uma distribuição homogênea com

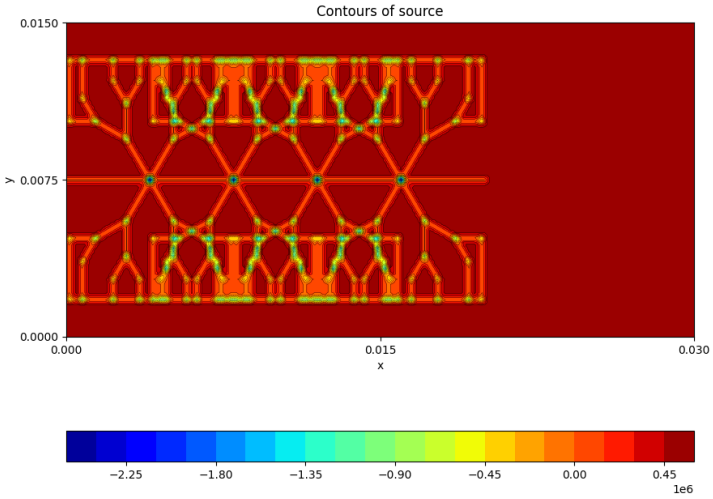


Figure 4.5: Termo de fonte modificado pela presença da rede.

$I_j = 1,\ \forall j$. Em formulações mais complexas, nas quais o termo I_j dependa dinamicamente de variáveis locais do escoamento, como a velocidade do fluido ou a concentração de reagentes dentro do canal, o sistema passa a exibir um comportamento de acoplamento forte, exigindo estratégias de resolução iterativa para a convergência mútua das físicas.

4.3.3 Investigando o comportamento do sistema

1. Implementar a rotina para o cálculo das condutividades térmicas modificadas nas interfaces, incorporando o efeito de proximidade da rede hidráulica. Para a análise paramétrica, considere os valores de raio de corte $d_{\max}$ iguais a 0.00025, 0.0005 e 0.001.
   Para cada cenário, realize as seguintes tarefas:
   - Gerar os mapas de contorno bi-dimensionais para o campo de temperaturas;
   - Comparar os valores máximos de temperatura obtidos na placa;
   - Extrair e plotar perfis unidimensionais de temperatura ao longo de seções retas horizontais e verticais estratégicas do domínio;
   - Mensurar e reportar o tempo de processamento computacional demandado em cada etapa do cálculo.

   Avalie o impacto do refinamento espacial no problema térmico adotando as seguintes malhas computacionais: $(61, 31)$, $(121, 61)$ e $(241, 121)$.
2. Incorporar ao modelo térmico o cálculo do termo fonte e sum-

idouro originado pela rede de microcanais. Para tanto, avalie o comportamento do sistema sob as seguintes intensidades de base S_0:

$$\pm 10^5, \quad \pm 5 \times 10^5, \quad \pm 10^6$$

Para cada valor de S_0, apresente os respectivos mapas de temperatura, estabeleça uma comparação entre os valores máximos e analise os perfis térmicos verticais e horizontais. A resolução deve contemplar duas distribuições distintas para o coeficiente de intensidade local I_j:

- $I_j = 1, \quad \forall j$ (distribuição homogênea);
- $I_j = 100$ para as arestas que integram o alinhamento central (espinha principal) da rede hidráulica, e $I_j = 0.1$ para as arestas remanescentes.

ACOPLAMENTO HIDRÁULICO-MECÂNICO 5

5.1 Preludio

O GÊMEO DIGITAL 6

6.1 Preludio

PROGRAMAÇÃO EM `python` A

A.1 Preludio

Neste apêndice introduzimos algumas noções de programação que podem ser uteis para começar a trabalhar. Lembremos que a ideia do curso é resolver problemas complexos de Engenharia, que não poderiam ser resolvidos manualmente, sem o auxilio de um computador.

De maneira muito geral, as linguagens de programação, podem ser divididas em duas categorias:

- **Linguagens compiladas:** Dentre as primeiras temos linguagens tais como `C`, `C++` e `Fortran`. A escrita de código neste tipo de linguagens de programação requer um domínio maior da sintaxe e das funcionalidades da linguagen. Como contrapartida, este tipo de linguagens produzem códigos que são muito rápidos pois tem sido optimizadas ao longo dos anos, e por tanto são usadas amplamente na Engenharia. De fato, a maioria dos códigos de cálculo usados na indústria (*Open Source* ou comerciais) estão feitos com elas. Ao **compilar** o código e gerar um arquivo binário optimizado, é possivel tirar o maior proveito do poder de processamento de um computador.

- **Linguagens interpretadas:** Por outra parte, as linguagens interpretadas, como `python` e `Matlab`, são relativamente simples e intuitivas, pela sua flexibilidade na sintaxe, tornando o processo de desenvolvimento mais rápido e ágil. De maneira grosseira, o código vai sendo executado a medida que se **interpreta**. Porém, se não são tomados alguns recaudos no desenho do código, a performance computacional delas pode estar bastante aquém do necessário para resolver problemas de grande porte. Um exemplo prototípico disto, é quando utilizamos uma estrutura de repetição (como um `for`) no qual realizamos um grande número de operações em cada passo. Ao longo de curso iremos chamando a atenção sobre isto, tentando introduzir boas práticas de programação.

Como comparação, vejamos o exemplo de um código simples para criar um array com números de ponto flutuante de dupla precisão, popular ele com os números $0, \ldots, N$ e imprimir o resultado na tela, usando a linguagem compilada `C` (acima) e a linguagem interpretada `python` (embaixo).

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int i, N = 10;
  double *array;
  array=(double *) malloc(N*sizeof(double));
  for(i=0; i < N; i++) {
       array[i] = (double) i;
       printf("%lf\n", array[i]);
     }
  free(array);
  return 0;
}
```

```python
import numpy as np
N = 10
array = np.arange(N)
print(array)
```

Olhando para o exemplo, já vemos que uma linguagem interpretada como `python` se torna mais prática para um curso de cálculo numérico, pois o objetivo é entender os métodos numéricos, os algoritmos e como resolver problemas práticos no computador, sem gastar muito tempo no desenvolvimento de código.

A.2 Noções iniciais de python

Ao longo do curso iremos incorporando diversas ferramentas e funções disponíveis em várias bibliotecas, mas antes vamos a introduzir alguns conceitos essenciais de programação específicos de `python`. Se sugere ir digitando em algum editor de código ou algum ambiente de programação.

A.2.1 Tipos de variáveis

Alguns dos tipos de variáveis mais usados são apresentados na sequência:

Números

São os objetos mais simples. Podem ser inteiros, de ponto flutuante, complexos e booleanos, por exemplo:

```
1, -2, 3.1415, 6.02e23, 1 + 1j, True, False
```

Strings

São basicamente, listas de caracteres e se escrevem entre aspas simples ou duplas:

```
"a", "USP", "21", "AbCdE", "---"
```

Listas

Servem para agrupar vários objetos.

```
mylist = [1, 3.14, "USP", True, -10000, 1+1j, myfunc]
```

A lista é indexada simplesmente pela posição do objeto, começando desde 0, p.e.,

`mylist[1]` é `3.14`
`mylist[6]` é myfunc

Diccionarios

É uma forma mais prática de definir listas com identificadores:

```
mydic = {"valor": 3.14, "minhauni": "USP", "lista": ["a", 2,
1+1j]}
```

Então,

`mydic{"minhauni"}` é `"USP"`
`mydic{"lista"}[2]` é `1 + 1j`

A.2.2 Estruturas condicionais

Uma das estruturas de programação mais usadas é a estrutura condicional. A sintaxe é simples:

```
if (Expressão lógica):
    .
    .
else:
    .
    .
```

ou em situações com mais de duas opções para decidir:

```
if (Expressão lógica 1):
    .
    .
elif (Expressão lógica 2):
    .
    .
else:
    .
    .
```

Notar os `:` no final das sentencias e a indentação dentro de cada bloco.

Advertência

A indentação é fundamental em `python`. Se esta não for respeitada o

interpretador não saberá quais instruções ficam dentro da estrutura e por tanto o programa poderá ter comportamentos não esperados ou em alguns casos parar a execução .

A.2.3 Estruturas de repetição

Existem duas estruturas de repetição que são muito usadas. A primeira estrutura é o famoso `for` (o "para" em portugués):

```
for i in range(N):
    .
    .
```

A segunda estrutura de repetição que iremos usar as vezes é o `while` (o "enquanto" em portugués):

```
while (Expressão lógica):
    .
    .
```

Com elas podemos fazer qualquer tipo de cálculo em que precisamos iterar sobre os elementos de algum objeto ou repetir uma operação várias vezes.

A.2.4 Funções

As declaração de funções é fundamental para poder organizar um código, encapsulando uma serie de operações as quais pode ser necessário realizar muitas vezes. A sintaxe para declarar uma função é:

```
def minhafunc(arg1, arg2, ...):
    .
    .
    return var1, var2, ...
```

Novamente, notar os `:` no final da definição e a indentação dentro do bloco da função.

A.2.5 A biblioteca `numpy`

Neste curso iremos adotar a linguagem `python`, a qual tem-se tornado bastante popular nos últimos anos. Para facilitar a implementação dos diferentes métodos numéricos, contamos com algumas bibliotecas específicas que facilitarão o trabalho:

- `numpy`: Para manipulação eficiente de matrizes e vetores, operações de algebra linear computacional e vários métodos numéricos.
- `scipy`: Para métodos numéricos mais avançados ou específicos, não cobertos pela anterior.
- `matplotlib`: Para plotagens e geração de gráficos em 2D e 3D.

Em particular, numpy é uma das bibliotecas que mais serão usadas ao longo do curso. Basicamente permite definir vetores, matrizes e tensores em geral, populados com números ou dados de um tipo homegêneo (i.e., todos números inteiros, ou todos números de ponto flutuante, etc.). Esta biblioteca é fundamental para poder realizar cálculos científicos com uma eficiencia razoável, possivelmente similar à da uma linguagem compilada. Alguns exemplos de uso na sequência:

```
import numpy as np


vec_ints = np.array([1,2,3,4], dtype=np.int32)
vec_doubles = np.array([1,2,3,4], dtype=np.float64)
x = np.linspace(start, end)
y = np.sin(x)
vetor_nulo = np.zeros(10, dtype=np.int32)
matriz_nula = np.zeros(shape=(5,3), dtype=np.float64)
```

A ideia era mostrar alguns exemplos simples para o aluno se familiarizar um pouco com a sintaxe. Na sequência podem colocar os conceitos em prática desenvolvindo códigos para resolver alguns problemas. A lista na sequência não precisa ser entregue.

A.3 Exercícios para praticar

1. Fazer uma função que calcula o produto escalar de dois vetores randômicos $\mathbf{a}$ e $\mathbf{b}$ de $\mathbb{R}^3$ e determina o ângulo θ que eles formam, usando a fórmula:
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \, \cos\theta$$
2. Fazer uma função que calcula o produto vetorial de dois vetores randômicos $\mathbf{a}$ e $\mathbf{b}$ de $\mathbb{R}^3$
$$\mathbf{a} \times \mathbf{b}$$
3. Fazer um código que define dois vetores `vec1` e `vec2` de tamanho 5 com números de ponto flutuante inventados. Depois cria um outro vetor de números inteiros do mesmo tamanho, tal que na i-éssima posição ele toma o valor 1 se a componente correspondente de `vec1` é maior que a do `vec2` e 0 em caso contrário.
4. Fazer um código que define um ponto de $\mathbf{R}^2$ e imprime `True` ou `False` dependendo se o ponto está emcima ou embaixo da reta $y = x$.
5. Calcular o número π usando a serie:
$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$$
Comparar o resultado dependendo do número de termos usados na série com o verdadeiro valor de π que pode ser obtido usando `np.pi`.
6. Declarar uma lista que possui nomes de pessoas inventados, outra lista que possui os seus sobrenomes e outra lista que possui a suas idades. Depois gerar listas individuais associadas a cada pessoa, na qual se encapsulem todos os dados dessa pessoa. Para isto, pode ser util usar uma propriedade das listas que é a possibilidade de acrescentar elementos usando a função

append, por exemplo:

```
minha_lista = []          # Lista vazia
minha_lista.append('kkk') # Acrescento o string kkk
minha_lista.append('jjj') # Acrescento o string jjj
```

que cria a lista `['kkk', 'jjj']`.

7. Fazer um programa que cria 10 pontos inventados em $\mathbf{R}^2$ e imprime quantos desses pontos estão fora do círculo de raio 1 centrado na origem e quantos dentro.
8. Repetir o exercício anterior para o caso de pontos em $\mathbf{R}^3$ e uma esfera.
9. Fazer um script que define um vetor randômico de dimensão N e calcula a média dos valores das suas componentes, i.e.,

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Considerar $N = 10, 100, 1000, 10000, 100000, 1000000$. Pode usar a função de `numpy` `np.random.rand(N)` que irá criar um `array` unidimensional (i.e., um vetor), com N números randômicos com valores entre 0 e 1. Estes número possuim o que se chama uma distribuição uniforme, pois qualquer um tem a mesma chance de sair.

Gráficos simples

10. Fazer um gráfico da função $f(x) = x^m$ para diferentes valores de m considerando o intervalo $x \in [1, 4]$. Usar escala linear e escala loglog.
11. Fazer um gráfico da função $f(x) = \sin(m\,x)$ para diferentes valores de m considerando o intervalo $x \in [0, 2\,\pi]$.
12. Fazer um gráfico que mostra pontos distribuídos randomicamente na região do plano $[0, 2] \times [-2, 1]$.
13. Considerar 10 dados que são jogados, podendo sair números do 1 até o 6. A soma dos valores será

$$S = \sum_{i=1}^{10} d_i$$

em que d_i é o que saiu em cada um dos dados. Jogar os 10 dados 100000 vezes e construir um histograma que mostre o comportamento de S. Um histograma seria um gráfico de frequencia de um certo evento, ou seja, quantas vezes a soma deu 10, quantas vezes a soma deu 11, quantas vezes a soma deu 12, ..., quantas vezes a soma deu 60. Para isto precissará usar a função

```
counts, bins = np.histogram(s)
plt.stairs(counts, bins)
```

em que `s` será um vetor que guardou o resultados das 100000 realizações. Pode usar a função `np.random.randint(1,7,10)` que irá gerar 10 número inteiros entre 1 e 6, com distribuição uniforme (ou seja, os dados não estão cargados)

MÉTODOS DIRETOS PARA SISTEMAS LINEARES

B

B.1 Revisão de Algebra linear

Antes de começar, lembremos o seguinte teorema importante:

Theorem B.1.1 ***Teorema 1:*** *Para um sistema de equações cuja matriz $A \in \mathbb{R}^{n\times n}$, a solução é única se uma das seguintes proposições equivalentes se cumpre:*

- *A é não singular;*
- *$det(A) \neq 0$;*
- *As linhas de A são linearmente independentes;*
- *Existe a matriz inversa A^{-1};*
- *$imagem(A) = \mathbb{R}^n$ (São todos os vetores que podem-se escrever como combinação linear das colunas de A);*
- *$nucleo(A) = \{0\}$ (São todos os vetores z para os que $A\,z = 0$);*

Métodos de resolução

Há dois tipos de métodos para resolver sistemas de equações:

1. **Métodos diretos**: Dão a solução exata a menos dos erros de arredondamento.
    - Eliminação de Gauss - Escalonamento
    - Métodos baseados em decomposições: LU, Cholesky, QR
2. **Métodos iterativos**: Iteramos até atingir uma solução aproximada.
    - Jacobi e Gauss-Seidel
    - Métodos dos Gradientes, Gradientes conjugados e outros mais sofisticados.

Nesta parte damos alguns detalhes de como funcionam os métodos **Diretos**.

Que sabemos resolver?

As únicas matrizes que são fácil de resolver são:

- **Matriz triangular superior**: $a_{ij} = 0$ sempre que $i > j$.
- **Matriz triangular inferior**: $a_{ij} = 0$ sempre que $i < j$.

que são matrizes da forma:

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}, \quad L = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & 0 \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix}$$

Para um sistema triangular superior: $U\,x = y$, usamos a substituição regressiva (*Backward substitution*)

$$\begin{array}{ccccccccc} u_{11}x_1 & + & u_{12}x_2 & + & \ldots & + & u_{1n}x_n & = & y_1 \\ & & u_{22}x_2 & + & \ldots & + & u_{2n}x_n & = & y_2 \\ & & & & & & \vdots & & \vdots \\ & & & & & & u_{nn}x_n & = & y_n \end{array}$$

$$\begin{aligned} x_n &= \frac{1}{u_{nn}}\, y_n \\ x_i &= \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right), \quad i = n-1, \ldots, 1 \end{aligned}$$

e para um sistema triangular inferior: $L\,y = b$, usamos a substituição progressiva (*Forward substitution*)

$$\begin{array}{ccccccccc} l_{11}y_1 & & & & & & & = & b_1 \\ l_{21}y_1 & + & l_{22}y_2 & & & & & = & b_2 \\ \vdots & & & & & & & & \vdots \\ l_{n1}y_1 & + & l_{n2}y_2 & + & \ldots & + & l_{nn}y_n & = & b_n \end{array}$$

$$\begin{aligned} y_1 &= \frac{1}{l_{11}}\, b_1 \\ y_i &= \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}y_j\right), \quad i = 2, \ldots, n \end{aligned}$$

Então, aproveitando isso, vamos supor que conseguimos escrever:

$$A = L\,U$$

$\Rightarrow$ Os passos para resolver $Ax = b$ são:

- Achar os fatores L e U

$$L\,U = A$$

- Notando que $Ax = (LU)x = L(Ux) = b$, resolver com forward substitution:

$$L\,y = b$$

- Resolver com backward substitution:

$$U\,x = y$$

Outro caso que sabemos resolver é o das matrizes ortogonais. Se Q é ortogonal, então:

$$Q^{-1} = Q^T$$

Vamos supor que por acaso conseguimos fazer a seguinte fatoração:

$$A = QR = \begin{pmatrix} q_{11} & q_{12} & \dots & \dots & q_{1n} \\ q_{21} & q_{22} & \dots & \dots & q_{2n} \\ q_{31} & q_{32} & \dots & \dots & q_{3n} \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ . & . & \dots & \dots & . \\ q_{n1} & q_{n2} & \dots & \dots & q_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & . & \dots & . \\ 0 & 0 & 0 & \dots & r_{nn} \end{pmatrix}$$

$\Rightarrow$ Os pasos para resolver $Ax = b$ são:

- Achar os fatores Q e R

$$Q\,R = A$$

- Notando que $(QR)x = Ax = b$, fazer

$$Q^T\,Q\,Rx = Q^T\,b = c$$

  Mas, sendo Q uma matriz ortogonal $Q^T\,Q = \mathbb{I}$
- Resolver com backward substitution:

$$R\,x = c$$

B.2 Escalonamento - Decomposição LU

Primeiro precisamos lembrar as operações elementares:

As seguintes operações aplicadas a um sistema linear geram um sistema linear que é equivalente (i.e., os sistemas têm a mesma solução):

(i) Multiplicar uma equação por um escalar;
(ii) Mudar uma equação pela soma dela mesma e de um multiplo não zero de qualquer outra equação;
(iii) Trocar a ordem de duas equações;

Cada uma dessas operações pode ser representada por uma matriz que, multiplicada a esquerda da matriz A do sistema, produz a operação desejada. Vejamos por exemplo (ii):

Utilizando matrizes triangulares inferiores da forma (para $n = 4$ por exemplo):

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & 0 & 1 & 0 \\ l_{41} & 0 & 0 & 1 \end{pmatrix}$$

cuja inversa é:

$$L^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -l_{21} & 1 & 0 & 0 \\ -l_{31} & 0 & 1 & 0 \\ -l_{41} & 0 & 0 & 1 \end{pmatrix}$$

que também é triangular inferior, temos

$$L^{-1} \cdot A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -l_{21} & 1 & 0 & 0 \\ -l_{31} & 0 & 1 & 0 \\ -l_{41} & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \dots$$

$$\dots = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} - l_{21}\, a_{11} & a_{22} - l_{21}\, a_{12} & a_{23} - l_{21}\, a_{13} & a_{24} - l_{21}\, a_{14} \\ a_{31} - l_{31}\, a_{11} & a_{32} - l_{31}\, a_{12} & a_{33} - l_{31}\, a_{13} & a_{34} - l_{31}\, a_{14} \\ a_{41} - l_{41}\, a_{11} & a_{42} - l_{41}\, a_{12} & a_{43} - l_{41}\, a_{13} & a_{44} - l_{41}\, a_{14} \end{pmatrix}$$

i.e., pegamos a primeira linha, a multiplicamos pelo fator l_{i1} e a substraimos da linha i para $i = 2, 3, 4$.

Similarmente, se tivermos os fatores l's em otra coluna, p.e.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -l_{32} & 1 & 0 \\ 0 & -l_{42} & 0 & 1 \end{pmatrix}.$$

Escalonamento - Decomposição LU

Vamos resolver o sistema $Ax = b$ (Ex. para $n = 4$)

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
a_{21}	a_{22}	a_{23}	a_{24}	b_2
a_{31}	a_{32}	a_{33}	a_{34}	b_3
a_{41}	a_{42}	a_{43}	a_{44}	b_4

1. Substrair a linha 1 multiplicada por $l_{i1} = a_{i1}/a_{11}$ da linha i, $i = 2, \dots, 4$
2. Definir $a'_{ik} = a_{ik} - l_{i1}a_{1k}$, $i, k = 2, \dots, 4$
3. Definir $b'_i = b_i - l_{i1}b_1$, $i = 2, \dots, 4$

1. Substrair a linha 2 multiplicada por $l'_{i2} = a'_{i2}/a'_{22}$ da linha i, $i = 3, \dots, 4$
2. Definir $a''_{ik} = a'_{ik} - l'_{i2}a'_{2k}$, $i, k = 3, \dots, 4$
3. Definir $b''_i = b'_i - l'_{i2}b'_2$, $i = 3, \dots, 4$

1. Substrair a linha 3 multiplicada por $l''_{43} = a''_{43}/a''_{33}$ da linha i, $i = 3, \dots, 4$

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
0	a'_{22}	a'_{23}	a'_{24}	b'_2
0	a'_{32}	a'_{33}	a'_{34}	b'_3
0	a'_{42}	a'_{43}	a'_{44}	b'_4

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
0	a'_{22}	a'_{23}	a'_{24}	b'_2
0	0	a''_{33}	a''_{34}	b''_3
0	0	a''_{43}	a''_{44}	b''_4

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
0	a'_{22}	a'_{23}	a'_{24}	b'_2
0	0	a''_{33}	a''_{34}	b''_3
0	0	0	a'''_{44}	b'''_4

2. Definir $a'''_{44} = a''_{44} - l''_{43}a''_{34}$
3. Definir $b'''_4 = b''_4 - l''_{43}b''_3$.

É uma matriz triangular superior!

Agora, vamos supor que guardamos os fatores $l's$ e vamos armar umas matrizes:

$$L_1 = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & & 1 & \\ l_{41} & & & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & l'_{32} & 1 & \\ & l'_{42} & & 1 \end{pmatrix}, \quad L_3 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & l''_{43} & 1 \end{pmatrix}$$

cujas inversas são

$$L_1^{-1} = \begin{pmatrix} 1 & & & \\ -l_{21} & 1 & & \\ -l_{31} & & 1 & \\ -l_{41} & & & 1 \end{pmatrix}, \quad L_2^{-1} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & -l'_{32} & 1 & \\ & -l'_{42} & & 1 \end{pmatrix}, \quad L_3^{-1} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -l''_{43} & 1 \end{pmatrix}$$

Então, tudo esse processo que temos feito para achar uma matriz triangular superior, pode-se ver que de fato foi:

$$U = L_3^{-1} L_2^{-1} L_1^{-1} A$$

Já que o produto de matrizes triangulares inferiores também é triangular inferior, e a inversa também é, resulta:

$$U = L^{-1} A \Rightarrow A = LU$$

O que acabamos de fazer se chama fatoração LU, em que as matrizes L e U são:

$$U = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a'''_{44} \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l'_{32} & 1 & 0 \\ l_{41} & l'_{42} & l''_{43} & 1 \end{pmatrix}$$

A matriz L só tem 1's na diagonal.

Decomposição LU:

Teorema 2: Seja $n \geq 2$ e $A \in \mathbb{R}^{n\times n}$ tal que todas as submatrices principais $A_k = A(1:k, 1:k),\ k = 1, \ldots, n-1$ são não singulares. Então A pode ser fatorada na forma $A = LU$ em que $L \in \mathbb{R}^{n\times n}$ é triangular inferior com 1's na diagonal e $U \in \mathbb{R}^{n\times n}$ é triangular superior.

Matrizes comuns na matemática aplicada

Em algumas matrizes pode ser provado que as condições do teorema se verificam:

- Matrizes definidas positivas:
  Uma matriz $A \in \mathbb{R}^{n\times n}$ diz-se definida positiva, se $\forall\, x \in \mathbb{R}^n,\ x \neq 0,\ x^T Ax > 0$.
- Matrizes com diagonal estritamente dominante:
  Uma matriz é de diagonal estritamente dominante por linha se:

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, \ldots, n$$

  e por columas se

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ji}|, \quad i = 1, \ldots, n$$

B.3 Eliminação de Gauss com Pivoting

Para outras matrizes em geral, o processo de eliminação de Gauss não pode ser completado sem recorrer a troca de linhas da matriz. Por exemplo:

$$\begin{pmatrix} 9 & 3 & 2 & 1 \\ 0 & 3 & 1 & 2 \\ 0 & 6 & 2 & 1 \\ 0 & 3 & 5 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 9 & 3 & 2 & 1 \\ 0 & 3 & 1 & 2 \\ 0 & 0 & 0 & -3 \\ 0 & 0 & 4 & 1 \end{pmatrix} \xrightarrow[\text{troca } \ell_3 \text{ e } \ell_4]{} \begin{pmatrix} 9 & 3 & 2 & 1 \\ 0 & 3 & 1 & 2 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & -3 \end{pmatrix}$$

Na prática, é pouco provável que apareça um coeficiente exatamente igual a zero, mas, podem aparecer fatores a_{kk} muito pequenos $\Rightarrow$ para minimizar erros de arredondamento e evitar divisões por números pequenos, se realiza troca de linhas, procurando que a_{kk} seja o maior possível dentre todos os coeficientes da coluna (desde $k+1$ em diante). Aqui que aparecem as matrizes de permutação (Lembrar das operações elementares) $\rightarrow$ Isto se chama pivoting!

B.3.1 Matrizes de permutação

Uma matriz de permutação $P \in \mathbb{R}^{n\times n}$ é uma matriz em que cada linha e coluna tem apenas uma entrada igual a 1 e o resto das entradas são 0. As linhas de P são permutações de linhas da matriz identidade.

- A inversa de uma matriz de permutação é a sua trasposta, i.e., $P^{-1} = P^T$.
- O produto de uma matriz de permutação à esquerda/direita por uma matriz A, da como resultado a matriz A com suas linhas/colunas permutadas do mesmo jeito.

Ex. $n = 4$

$$PA = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} a_{21} & a_{22} & a_{23} & a_{24} \\ a_{11} & a_{12} & a_{13} & a_{14} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Então, vamos aplicar as permutações ao longo do processo:

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
a_{21}	a_{22}	a_{23}	a_{24}	b_2
a_{31}	a_{32}	a_{33}	a_{34}	b_3
a_{41}	a_{42}	a_{43}	a_{44}	b_4

1. Permutar linhas $i = 1, \ldots, 4$ (se necessário) para que $a_{11} \neq 0$ e o maior possivel. Esse elemento será o Pivot (guardar P_1)
2. Substrair a linha 1 multiplicada por $l_{i1} = a_{i1}/a_{11}$ da linha i, $i = 2, \ldots, 4$
3. Definir $a'_{ik} = a_{ik} - l_{i1}a_{1k}, i, k = 2, \ldots, 4$
4. Definir $b'_i = b_i - l_{i1}b_1, i = 2, \ldots, 4$

1. Permutar linhas $i = 2, \ldots, 4$ (se necessário) para que $a'_{22} \neq 0$ e o maior possivel. Esse elemento será o próximo Pivot (guardar P_2)
2. Substrair a linha 2 multiplicada por $l'_{i2} = a'_{i2}/a'_{22}$ da linha i, $i = 3, \ldots, 4$
3. Definir $a''_{ik} = a'_{ik} - l'_{i2}a'_{2k}, i, k = 3, \ldots, 4$
4. Definir $b''_i = b'_i - l'_{i2}b'_2, i = 3, \ldots, 4$

1. Permutar linhas $i = 3, \ldots, 4$ (se necessário) para que $a''_{33} \neq 0$ e o maior possivel. Esse elemento será o próximo Pivot (guardar P_3)
2. Substrair a linha 3 multiplicada por $l''_{43} = a''_{43}/a''_{33}$ da linha i, $i = 3, \ldots, 4$
3. Definir $a'''_{44} = a''_{44} - l''_{43}a''_{34}$
4. Definir $b'''_4 = b''_4 - l''_{43}b''_3$.

É uma matriz triangular superior!

Agora, vamos supor que guardamos duas coisas

- Os fatores $l's$
- E as permutações, que poderiam se pensar como matrizes de permutação P_1, P_2, P_3

Como antes, vamos pegar os fatores $l's$ e vamos armar umas matrizes:

$$L_1 = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & & 1 & \\ l_{41} & & & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & l'_{32} & 1 & \\ & l'_{42} & & 1 \end{pmatrix}, \quad L_3 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & l''_{43} & 1 \end{pmatrix}$$

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
0	a'_{22}	a'_{23}	a'_{24}	b'_2
0	a'_{32}	a'_{33}	a'_{34}	b'_3
0	a'_{42}	a'_{43}	a'_{44}	b'_4

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
0	a'_{22}	a'_{23}	a'_{24}	b'_2
0	0	a''_{33}	a''_{34}	b''_3
0	0	a''_{43}	a''_{44}	b''_4

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	b_1
0	a'_{22}	a'_{23}	a'_{24}	b'_2
0	0	a''_{33}	a''_{34}	b''_3
0	0	0	a'''_{44}	b'''_4

Cujas inversas já sabiamos calcular.

Então, tudo esse processo que temos feito para achar uma matriz triangular superior, pode-se ver que de fato foi:

$$U = L_3^{-1} P_3 L_2^{-1} P_2 L_1^{-1} P_1 A$$

ou, fazendo um reacomodamento:

$$U = \underbrace{L_3^{-1}(P_3 L_2^{-1} P_3^{-1})\,(P_3 P_2 L_1^{-1} P_2^{-1} P_3^{-1})}_{L^{-1}}\ \underbrace{(P_3 P_2 P_1)}_{P} A$$

Finalmente

$$U = L^{-1} P A \Rightarrow P A = L U$$

O que acabamos de fazer se chama fatoração LU com pivoting parcial, em que as matrizes L e U são:

$$U = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a'''_{44} \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l'_{32} & 1 & 0 \\ l_{41} & l'_{42} & l''_{43} & 1 \end{pmatrix}$$

Igual que antes, veja que na matriz L só tem 1's na diagonal.

$\Rightarrow$ Os pasos para resolver $Ax = b$ são:

- Achar os fatores L e U e guardar as permutações P tal que

$$L U = P A$$

- Notando que $(LU)x = (PA)x = P(Ax) = Pb$, resolver com forward substitution:

$$L y = P b$$

- Resolver com backward substitution:

$$U x = y$$

Este processo está garantido para matrizes não singulares pelo teorema:

Teorema 3: Se $A \in \mathbb{R}^{n\times n}$ é não singular, então podemos achar uma matriz de permutação P tal que PA satisfaz as condições do Teorema 2, i.e., $PA = LU$ existe e é unica.

B.4 Complexidade computacional

Nos algoritmos que utilizamos resulta importante contar o número de operações de ponto flutuante que são realizadas (i.e., somas, multiplicações, etc.). Isto se denota por (#flops) (number of floating point operations)

Vejamos alguns exemplos simples:

- **Produto escalar de vetores:** Sejam $x, y \in \mathbb{R}^n$

  $$\Rightarrow \langle x, y \rangle = x^\intercal y = \sum_{i=1}^{n} x_i\, y_i = x_1\, y_1 + x_2\, y_2 + \cdots + x_n\, y_n$$

  o qual envolve n produtos e $(n-1)$ somas, portanto:

  $$\#\text{flops}_{pe} = n + (n-1) = 2\,n - 1 \quad \sim \mathcal{O}(n)$$

  em que $\mathcal{O}(n)$ denota "ordem n", que seria o comportamento dominante para n grande.
- **Produto matriz-vetor:** Sejam $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$ e $A\,x = y \in \mathbb{R}^n$

  $$\Rightarrow y_i = \sum_{i=1}^{n} A_{ik}\, x_k, \;\; i = 1, \ldots, n$$

  que são n produtos escalares de vetores de $\mathbb{R}^n$, i.e., a componente i de y é o produto escalar da linha i de A com o vetor coluna x, portanto:

  $$\#\text{flops}_{pmv} = n\,(2\,n - 1) = 2\,n^2 - n \quad \sim \mathcal{O}(n^2)$$

  em que $\mathcal{O}(n^2)$ denota "ordem n^2", que seria o comportamento dominante para n grande.

Que acontece no caso de resolver um sistema com fatoração LU

- Decomposição LU:

  $$\#\text{flops}_{LU} = \frac{2}{3}n^3 + \mathcal{O}(n^2)$$
- Forward substitution:

  $$\#\text{flops}_{F} = n^2 + \mathcal{O}(n)$$
- Backward substitution:

  $$\#\text{flops}_{B} = n^2 + \mathcal{O}(n)$$
- Em total:

  $$\#\text{flops} = \frac{2}{3}n^3 + \mathcal{O}(n^2) + 2n^2 + \mathcal{O}(n)$$

  portanto, o comportamento dominante para n grande é $\mathcal{O}(n^3)$.

i.e., se para resolver um sistema de 100 equações precisamos 1 milisegundo → para resolver um sistema com 1000, precisaremos 1 segundo!

B.5 Decomposição de Cholesky

Se A é simétrica e definida positiva, podemos construir uma fatoração melhor:

$$A = H\,H^T$$

Em que H é uma matriz triangular inferior que tem elementos positivos na diagonal. Por exemplo, para $n = 4$ teriamos

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} h_{11} & 0 & 0 & 0 \\ h_{21} & h_{22} & 0 & 0 \\ h_{31} & h_{32} & h_{33} & 0 \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix} \begin{pmatrix} h_{11} & h_{21} & h_{31} & h_{41} \\ 0 & h_{22} & h_{32} & h_{42} \\ 0 & 0 & h_{33} & h_{43} \\ 0 & 0 & 0 & h_{44} \end{pmatrix}$$

Decomposição de Cholesky: Algoritmo

$$h_{11} = \sqrt{a_{11}}$$

para $i = 2, \ldots, n$

$$h_{ij} = \frac{1}{h_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} h_{ik}\, h_{jk} \right), \quad j = 1, \ldots, i-1$$

$$h_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} h_{ik}^2}$$

Resolução via Cholesky: Complexidade computacional

- Em total:

$$\#\text{flops} = \frac{1}{3} n^3 + \mathcal{O}(n^2)$$

i.e., que demora a metade do tempo que a decomposição LU!

B.6 Funções de `scipy`

Temos várias funções para resolver sistemas por método diretos: Em `scipy`

- `scipy.linalg.lu(A)`
- `scipy.linalgy.cholesky(A)`
- `scipy.linalg.inv(A)` → Tentar não utilizar
- `scipy.linalg.solve(A,b)`
- `...`

Estas a sua vez estão baseadas em outras bibliotecas, tais como a LAPACK.

B.7 Matrizes Esparsas

De maneira informal, uma matriz esparsa é uma matriz com uma quantidade suficiente de zeros, para valer a pena tomar vantagem disso.

A ideia básica, é so guardar os elementos diferentes de zero da matriz, e as posições em que eles se encontram. Desta forma vamos economizar:

- Memoria;
- Número de operações;
- Tempo de cálculo;

Temos varias funções para trabalhar com matrizes esparsas:

- `Asp = sparse.csr_matrix(A)` → Converter de formato denso a formato esparso CSR (compress sparse row)
- `Asp = sparse.csc_matrix(A)` → Converter de formato denso a formato esparso CSC (compress sparse column)
- `Aden = Asp.to_dense()` → Converter de formato esparso a formato denso
- `Asp.nnz` → Contabilizar o número de não zeros da matriz
- `iden = sparse.speye` → Matriz identidade em formato esparso
- `sparse.coo_matrix` → Criar uma matriz esparsa a partir de um triplete
- `plt.spy(Asp)` → Visualizar a estrutura da matriz
- `scipy.sparse.linalg.spsolve(A)` → Resolver sistemas esparsos
- `scipy.sparse.linalg.splu(A)` → Calcular a fatoração *LU*
- ...

Um exemplo: Consideremos a matriz de `9 x 9` da sequência:

```
A =
3  -1   0  -2   0   0   0   0   0
-1   4  -1   0  -2   0   0   0   0
0  -1   3   0   0  -2   0   0   0
-2   0   0   5  -1   0  -2   0   0
0  -2   0  -1   6  -1   0  -2   0
0   0  -2   0  -1   5   0   0  -2
0   0   0  -2   0   0   3  -1   0
0   0   0   0  -2   0  -1   4  -1
0   0   0   0   0  -2   0  -1   3
```

Como podemos armar a estrutura simbólica desta matriz de uma vez, i.e., sem precisar primeiro armar a matriz densa e depois convertir a formato esparso?

Uma possibilidade é definir um triplete, formado por os índices de linha, coluna e os coeficientes:

```
il   = np.array([1,  1,  1,    2,  2,  2,  2,    3,  3,  3,    4,  4,  4,  4, ..])
jc   = np.array([1,  2,  4,    1,  2,  3,  5,    2,  3,  6,    1,  4,  5,  7, ..])
il = il - 1
jc = jc - 1
```

```
coef = np.array([3, -1, -2,   -1,  4, -1, -2,   -1,  3, -2,   -2,  5, -1, -2, ..])
Aesp = sparse.coo_matrix((coef, (il, jc)), shape=(9,9))

  Aesp =
  (0, 0) 3
  (0, 1) -1
  (0, 3) -2
  (1, 0) -1
  (1, 1) 4
  (1, 2) -1
  (1, 4) -2
  .
  .
  .
```

Alphabetical Index