Cálculo Numérico

Notas de aula - Solução de Sistemas Não-Lineares

Prof. Yuri Dumaresq Sobral

Departamento de Matemática
Universidade de Brasília

2025

- Queremos resolver sistemas de equações não-lineares do tipo

$$\begin{cases} x^3 - 3xy^2 = 1 \\ 3x^2y - y^3 = 0 \end{cases} \qquad \begin{cases} 3x - \cos(yz) = \frac{1}{2} \\ x^2 - 81(y+0.1)^2 + \sin(z) = -1.06 \\ e^{-xy} + 20z = \frac{10\pi - 3}{3} \end{cases}$$

- Estes sistemas podem ser muito complicados de se resolverem analiticamente, pois é possível que não seja possível isolar as variáveis de maneira adequada!
- Além disto, é complicado determinar o que são soluções destes sistemas não-lineares! Estes sistemas podem admitir vários tipos de soluções!
- Antes de prosseguirmos, precisamos redefinir, pontualmente (só neste capítulo), nossa notação:

- Considere um sistema de M equações e M incógnitas dado por

$$\begin{cases} f_1(x_1, x_2, x_3, \ldots, x_M) = 0 \\ f_2(x_1, x_2, x_3, \ldots, x_M) = 0 \\ f_3(x_1, x_2, x_3, \ldots, x_M) = 0 \\ \quad \vdots \\ f_M(x_1, x_2, x_3, \ldots, x_M) = 0 \end{cases} \Leftrightarrow \begin{array}{l} (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})) = \\ \\ = \ \mathbf{f}(\mathbf{x}) = \mathbf{0}. \end{array}$$

- Neste capítulo, x_i denotará a i-ésima componente de um vetor $\mathbf{x}$ qualquer.
- A n-ésima iteração de um processo iterativo baseado na variável escalar x_i será denotada por $x_i^{(n)}$,

$$x_i^{(n+1)} = g(x_i^{(n)}).$$

- Por outro lado, a n-ésima iteração da variável vetorial $\mathbf{x}$ será denotada por $\mathbf{x}_n$,

$$\mathbf{x}_{n+1} = \mathbf{h}(\mathbf{x}_n).$$

- Vamos construir processos iterativos para tentar aproximar a solução de sistemas não-lineares do tipo $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.
- Para isto, vamos querer que as soluções buscadas sejam soluções isoladas do sistema, isto é, se $\mathbf{x}^*$ for solução de $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, então $\exists\ \delta > 0$ tal que se $0 < |\mathbf{x} - \mathbf{x}^*| < \delta$, então $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$. Ou seja, não há nenhuma outra solução nas vizinhanças de $\mathbf{x}^*$.
- Note que, aqui, não exigimos que a solução seja única. Podem existir mais de uma solução, mas todas elas têm que ser isoladas.
- Agora, vamos formular nosso problema: dado um sistema $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, queremos construir um processo iterativo $\mathbf{x}_{n+1} = \mathbf{g}(\mathbf{x}_n)$ tal que seu ponto fixo $\mathbf{x}^*$ seja solução do sistema.

- Já sabemos bastante sobre isto nos casos escalares, em que $g : I\!R \to I\!R$. Agora, precisamos estudar os casos vetoriais em que $\mathbf{g} : I\!R^M \to I\!R^M$. E as coisas se complicam um pouco.
- O primeiro passo que devemos analisar é a estabilidade de $\mathbf{x}^*$: precisamos que ele seja assintoticamente estável.
- Vamos estudar como se comporta o processo iterativo nas vizinhanças de $\mathbf{x}^*$. Para isso, vamos usar uma Série de Taylor:

$$\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{x}^*) + \frac{d\mathbf{g}}{d\mathbf{x}}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + \frac{1}{2!}\frac{d^2\mathbf{g}}{d\mathbf{x}^2}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)^2 + \cdots$$

- Temos um problema! Quem são essas derivadas que envolvem quantidades vetoriais? Faz sentido escrever $(\mathbf{x} - \mathbf{x}^*)^2$?
- Detalhes sobre o cálculo de funções de várias variáveis vão ficar para o curso de Cálculo 3. PAUSA! Vamos mostrar o que vamos precisar aqui!

- A primeira derivada $\frac{d\mathbf{g}}{d\mathbf{x}}$ é, na verdade, o conjunto de todas as possíveis primeiras derivadas organizadas da seguinte maneira:

$$\frac{d\mathbf{g}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_M} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_M}{\partial x_1} & \frac{\partial g_M}{\partial x_2} & \cdots & \frac{\partial g_M}{\partial x_M} \end{pmatrix} = \nabla \mathbf{g}.$$

- A matriz que mostramos acima é chamada de matriz Jacobiana de $\mathbf{g}$. Esta matriz é muito importante em diversas áreas da matemática!
- A segunda derivada $\frac{d^2\mathbf{g}}{d\mathbf{x}^2}$ já é um tensor (matriz 3D!) e contém todas as possíveis segundas derivadas de $\mathbf{g}$. Ele é chamado de tensor Hessiana de $\mathbf{g}$, $H_{\mathbf{g}}$, e o termo associado a ela na série de Taylor é escrito como $\frac{1}{2!}(\mathbf{x}-\mathbf{x}^*)^T \cdot H_{\mathbf{g}} \cdot (\mathbf{x}-\mathbf{x}^*)$.

- As contas se complicam muito nos termos de ordem superior e uma notação tensorial é a mais adequada. (Google!)
- Vamos voltar à série de Taylor:

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}(\mathbf{x}^*) + \nabla\mathbf{g}(\mathbf{x}^*) \cdot (\mathbf{x} - \mathbf{x}^*)$$

e vamos escrever o processo iterativo como:

$$\mathbf{x}_{n+1} = \mathbf{g}(\mathbf{x}_n) = \mathbf{g}(\mathbf{x}^*) + \nabla\mathbf{g}(\mathbf{x}^*) \cdot (\mathbf{x}_n - \mathbf{x}^*) \Leftrightarrow$$

$$\Leftrightarrow \mathbf{x}_{n+1} - \mathbf{g}(\mathbf{x}^*) = \nabla\mathbf{g}(\mathbf{x}^*) \cdot (\mathbf{x}_n - \mathbf{x}^*) \Leftrightarrow \mathbf{x}_{n+1} - \mathbf{x}^* = \nabla\mathbf{g}(\mathbf{x}^*) \cdot (\mathbf{x}_n - \mathbf{x}^*)$$

e, portanto, o erro do processo é dado por:

$$\mathbf{e}_{n+1} = \nabla\mathbf{g}(\mathbf{x}^*) \cdot \mathbf{e}_n.$$

- Desta forma, o ponto fixo $\mathbf{x}^*$ será assintoticamente estável se, e somente se, $\nabla\mathbf{g}(\mathbf{x}^*)$ for uma matriz convergente! Isto é, se seu raio espectral $\rho(\nabla\mathbf{g}(\mathbf{x}^*)) < 1$.

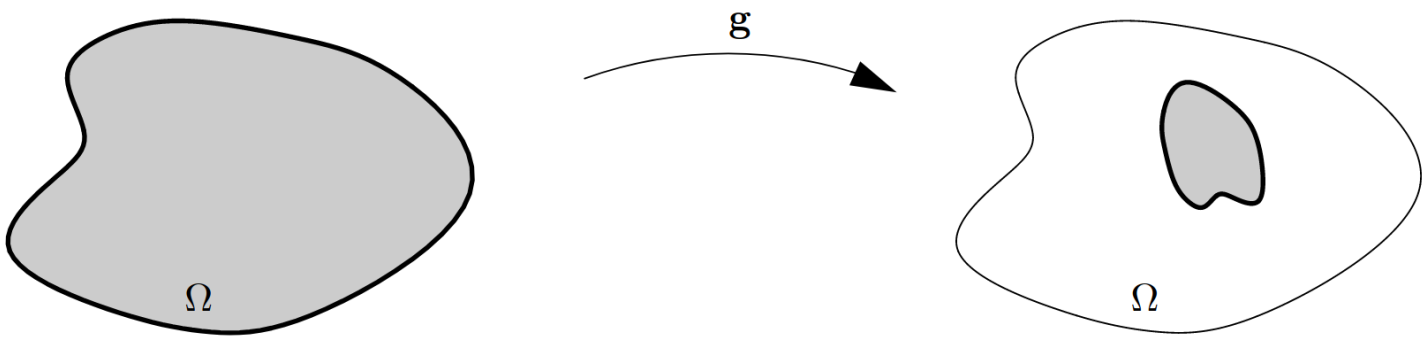
- Já conhecemos as dificuldades que isto traz. Podemos usar o resultado

$$\rho(\nabla \mathbf{g}(\mathbf{x}^*)) \quad \leq \quad \max_{i=1,\ldots,M} \quad \sum_{j=1}^{M} \left| \frac{\partial g_i}{\partial x_j} \Big|_{\mathbf{x}=\mathbf{x}^*} \right|$$

  para estimar o raio espectral da matriz Jacobiana em $\mathbf{x}^*$.

- Existe um resultado que pode ser útil em algumas aplicações: se conseguirmos construir uma $\mathbf{g}(\mathbf{x})$ tal que
  - $\mathbf{g} : \Omega \to \Omega$, com $\Omega \subset I\!R^M$ (isto é, seu conjunto imagem é o mesmo conjunto domínio),
  - exista uma constante $0 < \sigma < 1$ tal que $|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})| \leq \sigma |\mathbf{x} - \mathbf{y}|$,

  então $\mathbf{g}(\mathbf{x})$ adimite um único $\mathbf{x}^* \in \Omega$, e tomando qualquer ponto inicial $\mathbf{x}_0 \in \Omega$, o processo necessariamente converge para $\mathbf{x}^*$, isto é, $\mathbf{x}_n \to \mathbf{x}^*$ com $n \to \infty$.



- Observações:
  - uma função $\mathbf{g} : I\!R^M \to I\!R^M$ com estas propriedades é chamada de uma contração;
  - se $\mathbf{g}$ for uma contração, não apenas o problema da convergência está resolvido, como também do chute inicial! Encontrar chutes iniciais pode ser bastante complicado em altas dimensões;
  - a solução $\mathbf{x}^*$ do sistema não-linear $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ é isolada se e somente se
$$\det\left(\nabla \mathbf{g}(\mathbf{x}^*)\right) \neq 0.$$

- Vamos, agora, pensar em uma maneira de construir um processo iterativo que tenha um convergência mais rápida. Para isto, vamos usar uma metodologia muito similar à que utilizamos no caso dos processos iterativos escalares.
- Vamos querer construir processos que tenham convergência quadrática e, para tal, precisamos construir processos com
$$\nabla \mathbf{g}(\mathbf{x}^*) = \mathbb{O}.$$

- Vamos fazer uma análise totalmente análoga à que fizemos para o caso escalar para resolver uma equação algébrica. Vamos propor

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} + \mathcal{H}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}),$$

em que $\mathcal{H}(\mathbf{x})$ é uma matriz de $M \times M$ em que cada termo é uma função de $\mathbf{x}$.

- Com isto, impondo a condição de que a matriz Jacobiana seja nula no ponto fixo:

$$\nabla \mathbf{g}(\mathbf{x}^*) = \mathbb{O} = I + \nabla \mathcal{H}(\mathbf{x}^*) \cdot \mathbf{f}(\mathbf{x}^*) + \mathcal{H}(\mathbf{x}^*) \cdot \nabla \mathbf{f}(\mathbf{x}^*).$$

- Como $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, a expressão acima se reduz a:

$$\mathcal{H}(\mathbf{x}^*) \cdot \nabla \mathbf{f}(\mathbf{x}^*) = -I \quad \Leftrightarrow \quad \mathcal{H}(\mathbf{x}^*) = -\Big(\nabla \mathbf{f}(\mathbf{x}^*)\Big)^{-1},$$

- Ou seja, a matriz $\mathcal{H}(\mathbf{x})$ é o oposto da inversa da Matriz Jacobiana avaliada no ponto fixo $\mathbf{x}^*$.

- Vamos usar a mesma idéia usada no caso escalar de uma equação algébrica e vamos assumir que a igualdade seja válida para qualquer valor de $\mathbf{x}$, isto é:

$$\mathcal{H}(\mathbf{x}) = -\Big(\nabla \mathbf{f}(\mathbf{x})\Big)^{-1},$$

e, assim, podemos definir

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} - \Big(\nabla \mathbf{f}(\mathbf{x})\Big)^{-1} \cdot \mathbf{f}(\mathbf{x}) \quad \forall \mathbf{x}.$$

- Com isto, podemos construir o seguinte processo iterativo:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \Big(\nabla \mathbf{f}(\mathbf{x}_n)\Big)^{-1} \cdot \mathbf{f}(\mathbf{x}_n).$$

Método de Newton-Raphson para Sistemas

- Note que este método é totalmente análogo ao Método de Newton-Raphson para equações algébricas

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \Big(f'(x_n)\Big)^{-1} f(x_n).$$

- PROBLEMA!!! A cada passo do Método de Newton-Raphson precisaremos avaliar as funções que compõem a Matriz Jacobiana em $\mathbf{x}_n$ ($\mathcal{O}(M^2)$ operações), posteriormente, teremos que inverter esta matriz ($\mathcal{O}(M^3)$ operações), e depois temos que multiplicar este resultado por $\mathbf{f}(\mathbf{x}_n)$ ($\mathcal{O}(M^2)$ operações)!

MUITO CARO!!

- Normalmente, substitui-se a inversão da Matriz Jacobiana por uma solução de sistema linear:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \underbrace{\Big(\nabla \mathbf{f}(\mathbf{x}_n)\Big)^{-1} \cdot \mathbf{f}(\mathbf{x}_n)}_{\mathbf{y}_n},$$

$$\mathbf{y}_n = \Big(\nabla \mathbf{f}(\mathbf{x}_n)\Big)^{-1} \cdot \mathbf{f}(\mathbf{x}_n) \quad \Leftrightarrow \quad \nabla \mathbf{f}(\mathbf{x}_n)\mathbf{y}_n = \mathbf{f}(\mathbf{x}_n).$$

- Então, cada iteração do Método de Newton-Raphson seria dada por:

$$\nabla \mathbf{f}(\mathbf{x}_n)\mathbf{y}_n = \mathbf{f}(\mathbf{x}_n), \quad \mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{y}_n.$$

- O sistema para determinar o vetor $\mathbf{y}_n$ pode ser resolvido por qualquer método (Eliminação Gaussiana, fatoração LU, Gauss-Jacobi, Gauss-Seidel, SOR, etc).
- Algumas simplificações podem ser bem-vindas, mesmo penalizando a ordem quadrática do método.
- Às vezes, calcular exatamente as M^2 derivadas parciais pode não ser prático. Uma possibilidade é aproximar numericamente as derivadas:

$$\frac{\partial f_i}{\partial x_j}(\mathbf{x}_n) = \lim_{h \to 0} \frac{f_i(\mathbf{x}_n + h\mathbf{e}_j) - f_i(\mathbf{x}_n)}{h} \approx \frac{f_i(\mathbf{x}_n + h\mathbf{e}_j) - f_i(\mathbf{x}_n)}{h}$$

  para $|h|$ pequeno.
- Às vezes, a Matriz Jacobiana não muda tanto de uma iteração para outra, pois $\mathbf{x}_n$ e $\mathbf{x}_{n+1}$ estão muito próximos um do outro.

- Então, é possível economizar algumas soluções de sistema implementando o seguinte algoritmo:

1. Defina $\mathbf{x}_0$ e faça $\mathbf{x}_{ref} = \mathbf{x}_0$
2. Calcule $\nabla\mathbf{f}(\mathbf{x}_0)$
3. Resolva $\nabla\mathbf{f}(\mathbf{x}_0)\mathbf{y}_0 = \mathbf{f}(\mathbf{x}_0)$
4. Faça de $n = 0$ até N
   - 4.1 Calcule $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{y}_n$
   - 4.2 Se $|\mathbf{x}_{n+1} - \mathbf{x}_{ref}| < TOL$
   - 4.3 Então resolva $\nabla\mathbf{f}(\mathbf{x}_{ref})\mathbf{y}_{n+1} = \mathbf{f}(\mathbf{x}_{n+1})$
   - 4.4 Senão
     - 4.4.1 Calcule $\nabla\mathbf{f}(\mathbf{x}_{n+1})$
     - 4.4.2 Resolva $\nabla\mathbf{f}(\mathbf{x}_{n+1})\mathbf{y}_{n+1} = \mathbf{f}(\mathbf{x}_{n+1})$
     - 4.4.3 Faça $\mathbf{x}_{ref} = \mathbf{x}_{n+1}$

- As aproximações mencionadas aqui geram métodos que são chamados de Métodos de Quase-Newton.
- Estes métodos normalmente têm convergência superlinear (> 1), e não mais quadrática. Mas exigem consideravelmente menos operações por iteração.

- Apesar da desejada convergência quadrática do método **Newton-Raphson**, encontrar chutes iniciais para iniciar o método de maneira adequada pode ser MUITO complicado, especialmente em altas dimensões...
- Some-se a isto, também, o custo computacional de resolver um sistema de equações lineares a cada passo!
- Portanto, precisamos encontrar uma alternativa robusta... de maneira simiar ao que fizemos com o método da bisseção!
- Não podemos mais usar a ideia de dividir intervalos no contexto de funções de várias variáveis... Porém, há uma alternativa viável!

Vamos voltar à formulação inicial do problema de resolver um sistema não-linear de M equações e M incógnitas:

$$\begin{cases} f_1(x_1, x_2, x_3, \ldots, x_M) = 0 \\ f_2(x_1, x_2, x_3, \ldots, x_M) = 0 \\ f_3(x_1, x_2, x_3, \ldots, x_M) = 0 \\ \quad \vdots \\ f_M(x_1, x_2, x_3, \ldots, x_M) = 0 \end{cases} \quad \Leftrightarrow \quad \mathbf{f}(\mathbf{x}) = \mathbf{0}.$$

Note que a solução $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ deste sistema acontecerá onde $\mathbf{f}(\mathbf{x}) = \mathbf{0} \Leftrightarrow |\mathbf{f}(\mathbf{x})| = 0$, ou seja, podemos pensar em termos da norma da função $\mathbf{f}(\mathbf{x}) = (f_1, f_2, f_3, \ldots, f_M)$:

$$|\mathbf{f}(\mathbf{x})| = \sqrt{f_1^2 + f_2^2 + f_3^2 + \ldots + f_M^2} = 0.$$

Portanto, podemos trocar o problema de encontrar a solução $\mathbf{x}^*$ do sistema não-linear $\mathbf{f}(\mathbf{x}) = \mathbf{0} \ldots$

... pelo problema de encontrar o ponto $\mathbf{x}^{\mathbf{min}}$ que minimiza a função

$$h(\mathbf{x}) = |\mathbf{f}(\mathbf{x})|^2 = f_1^2 + f_2^2 + f_3^2 + \ldots + f_M^2 \quad \geq 0,$$

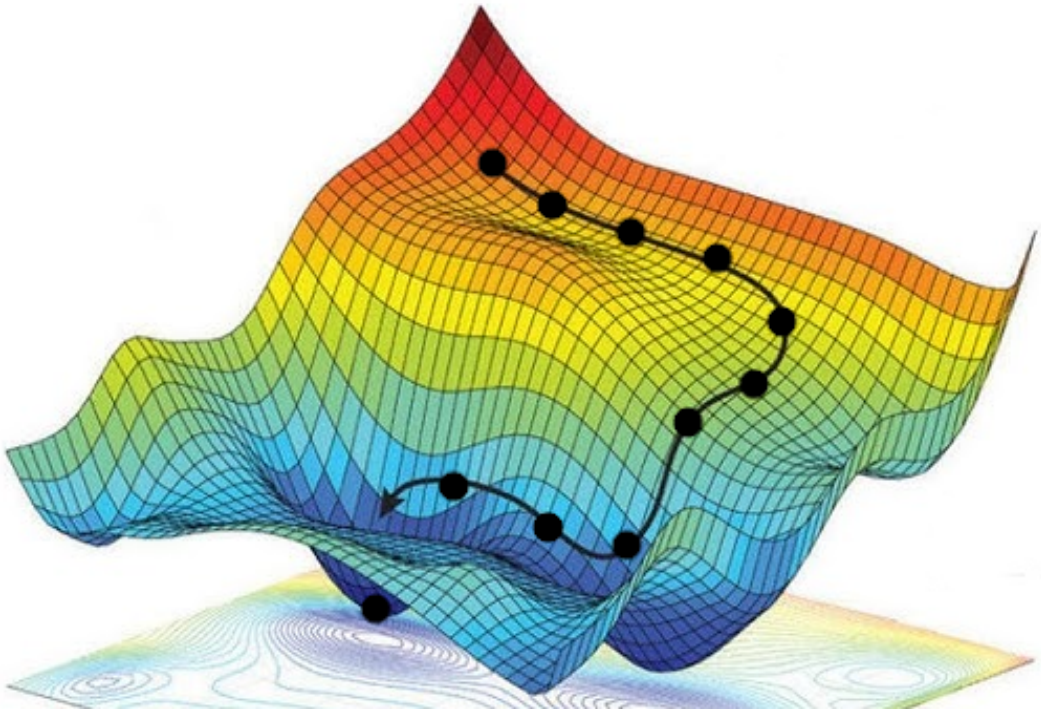
isto é, $h(\mathbf{x}^{\mathbf{min}}) = 0 \leq h(x)\ \forall x$!

Nossa tarefa, portanto, é construir um algoritmo que parta de um valor inicial (chute) $\mathbf{x}_0$ e convirja para o valor $\mathbf{x}^{\mathbf{min}}$.

Pela maneira como construímos a função $h(\mathbf{x})$, para qualquer $\mathbf{x}_0$ tal que $\mathbf{f}(\mathbf{x}_0) \neq \mathbf{0}$, teremos que $h(\mathbf{x}_0) > 0$. Então, precisamos criar um processo iterativo que gere uma sequência de $\mathbf{x}_n$ tal que

$$h(\mathbf{x}_{n+1}) < h(\mathbf{x}_n),$$

isto é, uma sequência tal que o valor de $h(\mathbf{x})$ decaia à medida em que avançamos no processo, até encontrarmos $\mathbf{x}^{\mathbf{min}}$! Graficamente:



Precisamos, assim, encontrar a direção do caminho a seguir! Se conhecêssemos esta direção $\mathbf{d}_n$ em cada passo do método, poderíamos escrever:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{d}_n,$$

em que $\alpha_n \in I\!R_0^+$ é um número que determina o tamanho do passo na direção $\mathbf{d}_n$ (em geral, $|\mathbf{d}_n| = 1$).

- Métodos deste tipo são chamados de Métodos de Descida Gradiente e, para que funcionem, precisamos determinar adequadamente $\mathbf{d}_n$ e α_n.
- Existem várias maneiras de determinar estes parâmetros do método. Vamos estudar uma que se baseia em um conceito do Cálculo 3.
- Seja $\mathbf{v}$ um vetor unitário, isto é, $|\mathbf{v}| = 1$. Definimos a derivada direcional de uma função $h(\mathbf{x})$ na direção de $\mathbf{v}$ como sendo

$$\frac{dh(\mathbf{x})}{d\mathbf{v}} = \lim_{\epsilon \to 0} \frac{h(\mathbf{x} + \varepsilon \mathbf{v}) - h(\mathbf{x})}{\varepsilon} = \nabla h(\mathbf{x}) \cdot \mathbf{v}.$$

- Portanto, a taxa com que uma função $h(\mathbf{x})$ varia ao longo de uma direção arbitrária $\mathbf{v}$ é dada pelo valor da projeção de $\mathbf{v}$ na direção do vetor gradiente $\nabla h(\mathbf{x})$.

- Da Álgebra Linear, sabemos que o produto escalar entre dois vetores pode ser reescrito em termos de seus módulos e do ângulo θ que se forma entre eles, isto é,

$$\frac{dh(\mathbf{x})}{d\mathbf{v}} = \nabla h(\mathbf{x}) \cdot \mathbf{v} = |\nabla h(\mathbf{x})| \, |\mathbf{v}| \cos(\theta) = |\nabla h(\mathbf{x})| \cos(\theta).$$

- Portanto, o valor MÁXIMO da derivada direcional ocorrerá quando $\cos(\theta) = 1$, isto é, quando $\nabla h(\mathbf{x})$ e $\mathbf{v}$ forem paralelos! De fato, a direção do vetor gradiente é a direção de MAIOR CRESCIMENTO DA FUNÇÃO.
- Então, para que o Método de Descida Gradiente funcione, precisamos escolher $\mathbf{d}_n$ tal que

$$\nabla h(\mathbf{x}_n) \cdot \mathbf{d}_n < 0,$$

pois só assim poderemos construir um caminho em que $h(\mathbf{x})$ seja decrescente, isto é, que ela diminua seu valor a cada passo.

De fato, note que se escolhermos, a cada passo,

$$\mathbf{d}_n = -\frac{\nabla h(\mathbf{x}_n)}{|\nabla h(\mathbf{x}_n)|},$$

estaremos tomando a direção oposta à direção do gradiente, isto é, estaremos tomando a direção de

MAIOR DECRESCIMENTO DA FUNÇÃO!

O método que utiliza esta escolha para o vetor $\mathbf{d}_n$ é chamado de Método da Descida Mais Íngrime e é bastante utilizado neste contexto de solução de sistemas não-lineares!

Note que precisamos que $\nabla h(\mathbf{x}_n) \neq \mathbf{0}\ \forall n$ para que o método possa avançar até a aproximação desejada para $\mathbf{x}^{\mathbf{min}}$. PERIGO!

- Uma vez determinada a direção de descida $\mathbf{d}_n$, precisamos determinar a constante α_n que levará o processo iterativo de $\mathbf{x}_n$ para $\mathbf{x}_{n+1}$.

- Em princípio, podemos tomar α_n como uma constante pequena, $\alpha_n = \alpha$. Isto costuma dar certo apenas com funções muito bem comportadas, e é possível que α tenha que ser muito pequeno, deixando o algoritmo muito lento! É possível fazer melhor!

- Vamos definir a função de uma variável

$$p(\alpha) = h(\mathbf{x}_n + \alpha \mathbf{d}_n),$$

isto é, define-se $p(\alpha)$ como a restrição de $h(\mathbf{x})$ à reta $\mathbf{x}_n + \alpha \mathbf{d}_n$. A ideia consiste em determinar $\alpha_n \neq 0$ tal que $p(\alpha_n) < h(\mathbf{x}_n)$.

- Existem vários métodos que fazem diferentes tipos de buscas para α_n ao longo da reta $\mathbf{x}_n + \alpha \mathbf{d}_n$: esta classe de métodos é chamada de algoritmos de busca na linha (line search algorithms).

- Podemos propor encontrar o valor de α_n tal que $p(\alpha_n)$ seja mínima. Para isto, poderíamos calcular (numericamente) a raiz da equação $p'(\alpha_n) = 0$ a cada passo do processo. CARO!

- Uma simplificação comumente feita neste procedimento é aproximar localmente $p(\alpha)$ por uma parábola, isto é,

$$p(\alpha) = a\alpha^2 + b\alpha + c, \quad a, b, c \in I\!R.$$

  e tomar α_n como o valor que minimiza a parábola em um intervalo previamente selecionado $[\alpha_1, \alpha_3]$.

- Tomando três pontos $\alpha_1 < \alpha_2 < \alpha_3$ tais que $h(\mathbf{x}_n) = p(0) \geq p(\alpha_1) > p(\alpha_3)$, encontramos os valores de a, b, c resolvendo o sistema linear:

$$\begin{cases} a\alpha_1{}^2 + b\alpha_1 + c = p(\alpha_1) \\ a\alpha_2{}^2 + b\alpha_2 + c = p(\alpha_2) \\ a\alpha_3{}^2 + b\alpha_3 + c = p(\alpha_3) \end{cases}.$$

  Já sabemos fazer isto no computador! *(Depois vamos ver mais sobre este problema em particular!)*

- Determinada a aproximação parabólica para $p(\alpha)$, escolhemos
  - $\alpha_n = -\dfrac{b}{2a}$, se a parábola tiver um mínimo (ponto crítico) no intervalo $[\alpha_1, \alpha_3]$;
  - $\alpha_n = \alpha_3$, caso contrário.
- Em geral, a escolha $\alpha_1 = 0,\ \alpha_2 = 0.5,\ \alpha_3 = 1$ tende a funcionar. Se não funcionar, tentar $\alpha_3 = 2^{-k}$, $k = 1, 2, \ldots$ até obter $p(\alpha_3) < p(\alpha_1)$.

- O método da descida mais íngrime apresentado aqui costuma convergir, porém de maneira muito lenta. Pode ser usado como um gerador de chute inicial para o método de **Newton-Raphson**!
- É possível que a função $h(\mathbf{x})$ tenha vários mínimos, isto é, é possível que o sistema $\mathbf{f}(\mathbf{x})$ admita várias soluções. Não é possível controlar para qual solução o método irá! Isto dependerá apenas do chute inicial.
- Este método pode ser usado para problemas gerais de otimização, isto é, para encontrar máximos e mínimos de funções de várias variáveis.