Folhas de problemas de Física computacional

Problemas selecionados e traduzidos do livro

Computacional Physics

Mark Newman

Ano lectivo 2018/2019

Docentes:

João Manuel Viana Parente Lopes

José Miguel Nunes da Silva

João Manuel Borregana Lopes dos Santos

Física Computacional

Ano lectivo 2017/2018

Folha 1 - Programação em Python para Físicos

1. Exercício

A órbita espacial de um corpo em torno de outro, como um planeta em torno do Sol, não é necessariamente circular. Em geral assume a forma de uma elipse, com o corpo por vezes mais próximo e por vezes mais afastado. Se na máxima aproximação de um planeta ao Sol, também chamado de *periélio*, a distância é ℓ_1 e sua velocidade linear v_1, então qualquer outra propriedade da órbita pode ser calculado a partir destes dois valores como se segue.

1. A segunda lei de Kepler diz-nos que a distância ℓ_2 e a velocidade v_2 do planeta no ponto de máximo afastamento, o *afélio*, satisfaz $\ell_2 v_2 = \ell_1 v_1$. Também é sabido que a energia mecânica total, cinética mais potencial gravítica, do planeta com velocidade v a uma distância r do Sol é dada por

$$E = \frac{1}{2}mv^2 - G\frac{mM}{r},$$

onde m é a massa do planeta, $M = 1.9891 \times 10^{30}\,\mathrm{kg}$ a massa do Sol, e $G = 6.6738 \times 10^{-11}\,\mathrm{m^3\,kg^{-1}\,s^{-2}}$ a constante de gravitação universal. Dado que a energia mecânica deve ser conservada, mostre que v_2 é a menor das raízes da equação quadrática

$$v_2^2 - \frac{2GM}{v_1\ell_1}v_2 - \left[v_1^2 - \frac{2GM}{\ell_1}\right] = 0.$$

Uma vez obtido o valor de v_2 podemos calcular ℓ_2 usando a relação $\ell_2 = \ell_1 v_1/v_2$.

2. Conhecidos os valores de v_1, ℓ_1, e ℓ_2, outros parâmetros da órbita são dados por expressões simples obtíveis da geometria elíptica da órbita e das leis de Kepler:

$$\begin{aligned}
\text{Semi-eixo maior:} \quad & a = \frac{1}{2}(\ell_1 + \ell_2), \\
\text{Semi-eixo menor:} \quad & b = \sqrt{\ell_1\ell_2}\,, \\
\text{Período orbital:} \quad & T = \frac{2\pi ab}{\ell_1 v_1}\,, \\
\text{Excentricidade orbital:} \quad & e = \frac{\ell_2 - \ell_1}{\ell_2 + \ell_1}.
\end{aligned}$$

Desenvolva um programa que peça ao utilizador a distância do planeta ao Sol e a sua velocidade no periélio e, depois, calcule e mostre os valores das quantidades ℓ_2, v_2, T, e e.

3. Teste o programa fazendo o cálculo das propriedades orbitais da Terra (para a qual $\ell_1 = 1.4710 \times 10^{11}\,\mathrm{m}$ e $v_1 = 3.0287 \times 10^4\,\mathrm{m\,s^{-1}}$) e do cometa Halley ($\ell_1 = 8.7830 \times 10^{10}\,\mathrm{m}$ e $v_1 = 5.4529 \times 10^4\,\mathrm{m\,s^{-1}}$).

Entre outras coisas, deve obter para a Terra um período orbital de um ano, e de 76 anos para o cometa Halley.

2. Exercício

Os números de Catalan C_n são uma sequência de inteiros 1, 1, 2, 5, 14, 42, 132...com um papel importante em Mecânica Quântica e na Teoria dos Sistemas Desordenados. (Estão na base da demonstração por Eugene Wigner da famosa 'Lei do Semicírculo'.) Estes números podem ser definidos por recorrência:

$$C_0 = 1, \qquad C_{n+1} = \frac{4n+2}{n+2} C_n.$$

Desenvolva um programa que mostre, por ordem crescente, todos os números de Catalan até um milhar de milhão.

3. Exercício

Em Física da Matéria Condensada a constante de Madelung representa a energia potencial elétrica total de um ião na rede cristalina. Ela depende das cargas dos outros iões vizinhos e das suas localizações. Considere, por exemplo, o caso do Cloreto de Sódio (sal das cozinhas) onde os iões estão dispostos alternadamente numa rede cúbica simples: o de sódio com uma carga positiva $+e$ e o de cloro com uma carga negativa $-e$, onde é a carga do eletrão. Se etiquetarmos cada nodo da rede por três números inteiros (i, j, k), então os iões sódio ficam nas posições onde $i + j + k$ é par, e os iões cloro nas posições onde $i + j + k$ é ímpar.

Consideremos o ião sódio situado na origem, $i = j = k = 0$, e calculemos a constante de Madelung. Se o espaçamento entre os iões for$a$ (constante da rede), então a distância da origem ao ião na posição (i, j, k) é

$$\sqrt{(ia)^2 + (ja)^2 + (ka)^2} = a\sqrt{i^2 + j^2 + k^2},$$

e o potencial na origem criado por esse ião é

$$V(i, j, k) = \pm \frac{e}{4\pi\epsilon_0 a\sqrt{i^2 + j^2 + k^2}},$$

(ϵ_0 é a permitividade do vazio e o sinal da expressão depende de $i + j + k$ ser par ou ímpar). O potencial total na origem V_{total} é então a soma sobre todas as restantes posições. Supondo um cristal de formato cúbico, e centrado na origem, com L iões em todas as direções, vem

$$V_{\text{total}} = \sum_{\substack{i,j,k=-L \\ \text{not } i=j=k=0}}^{L} V(i, j, k) = \frac{e}{4\pi\epsilon_0 a} M,$$

onde M é (no limite $L \to \infty$) a constante de Madelung.

Desenvolva um programa que calcule a constante de Madelung do Cloreto de Sódio. Use um valor suficientemente grande para L mas que lhe permita correr o programa num tempo não superior a um minuto. Compare com o valor exato para o NaCl: $-1,748$.

4. Exercício

O coeficiente binomial $\binom{n}{k}$ é um número inteiro igual a

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \times (n-1) \times (n-2) \times \ldots \times (n-k+1)}{1 \times 2 \times \ldots \times k}$$

com $k \geq 1$, ou $\binom{n}{0} = 1$ quando $k = 0$.

1. Usando este formulário para o coeficiente binomial, escreva uma função $binomial(n, k)$ que calcula o coeficiente binomial para n e k. Certifique-se de que sua função retorna a resposta na forma de um inteiro (não um float) e forneça o valor correto de 1 para o caso em que $k = 0$.

2. Usando sua função, escreva um programa para imprimir as primeiras 20 linhas do "triângulo de Pascal". A linha n do triângulo de Pascal contém $n + 1$ números, que correspondem aos coeficientes $\binom{n}{0}$, $\binom{n}{1}$, até $\binom{n}{n}$. As primeiras linhas são,

   1 1
   1 2 1
   1 3 3 1
   1 4 6 4 1

3. A probabilidade de uma moeda justa, lançada n vezes, apresenta k resultados de caras é $\binom{n}{k}/2^n$. Escreva um programa que calcula (a) A probabilidade the uma moeda ser lançada 100 vezes e ter como resultado 60 caras, e (b) a probabilidade de ter como resultado pelo menos 60 caras.

5. Exercício

O programa no Exemplo 2.8 do livro adoptado não corresponde ao modo mais eficiente de calcular números primos: verifica cada número para ver se é divisível por qualquer número menor que ele. Podemos desenvolver um programa muito mais rápido para números primos usando as seguintes observações:

1. Um número n é primo se não tiver nenhum fator primo menor que n. Por isso, só precisamos verificar se é divisível por outros primos.

2. Se um número n é não primo, tendo um fator r, então $n = rs$, onde s também é um fator. Se $r \geq \sqrt{n}$ então $n = rs \geq s\sqrt{n}$, o que implica que $s \leq \sqrt{n}$. Em outras palavras, qualquer não-primo deve ter fatores e, portanto, também fatores primos, menores ou iguais a $\sqrt{n}$. Assim, para determinar se um número é primo, temos que verificar seus fatores primos apenas até e incluindo $\sqrt{n}$ - se não houver nenhum, o número é primo.

3. Se encontrarmos um único fator primo menor que $\sqrt{n}$, então sabemos que o número não é primo e, portanto, não há necessidade de verificar mais nada - podemos abandonar esse número e passar para outra coisa..

Escreva um programa Python que encontre todos os primos até dez mil. Crie uma lista para armazenar os primos, contém inicialmente apenas um elemento, o número primo 2. Então, para cada número n de 3 a $10\,000$, verifique se o número é divisível por qualquer um dos primos da lista, até e incluindo $\sqrt{n}$. Assim que encontrar um único fator primo, poderá parar de verificar os restantes - sabe que n não é um primo. Se não

encontrar fatores primos $\sqrt{n}$ ou menos, então n é primo e você deve adicioná-lo à lista. Pode imprimir a lista de uma só vez no final do programa, ou pode imprimir os números individuais à medida que os encontrar.

6. Exercício

Um recurso útil de funções definidas pelo utilizador é a capacidade de uma função se chamar recursivamente. Por exemplo, considere a seguinte definição do fatorial $n!$ de um inteiro positivo n:

$$n! = \begin{cases} 1 & \text{se } n = 1, \\ n \times (n-1)! & \text{se } n > 1. \end{cases}$$

Isto constitui uma definição completa do fatorial que nos permite calcular o valor de $n!$ para qualquer inteiro positivo. Podemos empregar essa definição diretamente para criar uma função Python para fatoriais, como este:

```
def factorial(n):
    if n==1:
        return 1
    else:
        return n*factorial(n-1)
```

Note como, se n não for igual a 1, a função chama a si própria para calcular o fatorial de $n - 1$. A este procedimentos chamamos recursão. Se executarmos factorial(5) o computador irá imprimir corretamente a resposta 120.

1. Encontramos os números de Catalan C_n no Exercício 2. Com apenas um pequeno rearranjo, a definição pode ser reescrita na forma

$$C_n = \begin{cases} 1 & \text{se } n = 0, \\ \dfrac{4n-2}{n+1} C_{n-1} & \text{se} n > 0. \end{cases}$$

   Escreva uma função Python, usando recursão, que calcule C_n. Use essa função para calcular e imprimir C_{100}.

2. Euclides mostrou que o maior divisor comum $g(m, n)$ de dois inteiros não negativos m e n satisfaz

$$g(m,n) = \begin{cases} m & \text{if } n = 0, \\ g(n, m \bmod n) & \text{if } n > 0. \end{cases}$$

   Escreva uma função Python que empregue recursão para calcular o maior divisor comum de m e n usando essa fórmula. Use sua função para calcular e imprimir o maior divisor comum de 108 e 192.

Comparando o cálculo dos números de Catalan na parte (a) acima com o do Exercício 2, vemos que é possível fazer o cálculo de duas maneiras, diretamente ou usando recursão. Na maioria dos casos, se uma quantidade puder ser calculada sem recursão, será mais rápido fazer isso, e normalmente recomendamos que siga essa rota, se possível. Existem alguns cálculos, no entanto, que são essencialmente impossíveis (ou pelo menos muito mais difíceis) sem recursividade. Veremos alguns exemplos mais adiante no livro.

Física Computacional

Ano lectivo 2017/2018

Folha 2 - Representação gráfica de dados experimentais

1. Exercício

Nos recursos on-line do livro adotado existe um ficheiro de nome `sunspots.txt` que contém o número de manchas solares observadas em cada mês desde Janeiro de 1749. Este ficheiro tem duas colunas de números em que a primeira refere o mês e a segunda refere o número de manchas solares.

1. Desenvolva um programa que leia esses dados e trace um gráfico do número de manchas solares em função do tempo.

2. Modifique o programa para que mostre apenas os primeiros 1000 pontos do gráfico.

3. Adapte esse programa de forma a que também calcule e represente num gráfico as médias móveis definidas por

$$Y_k = \frac{1}{2r+1} \sum_{m=-r}^{r} y_{k+m} \,,$$

onde $r = 5$ (e os y_k são os números de manchas solares). Trace no mesmo gráfico os dados originais e as médias móveis, limitando a representação aos primeiros 1000 pontos.

2. Exercício

Embora a função `plot` seja primeiramente entendida para a construção de gráficos y em função de x, também pode ser adaptada a outros tipos de gráficos.

1. Obtenha um gráfico da curva conhecida por *deltoide* e que é definida parametricamente pelas equações

$$x = 2\cos\theta + \cos 2\theta, \qquad y = 2\sin\theta - \sin 2\theta,$$

onde $0 \leq \theta < 2\pi$. Considere um conjunto de valores para θ entre zero e 2π, calcule os correspondentes x e y para cada das equações anteriores e, então, represente y como função de x.

2. Indo mais além, faça uma representação polar $r = f(\theta)$, com uma função f adequada, de forma a obter a conhecida lei de transformação de coordenadas cartesianas para coordenadas polares: $x = r\cos\theta$, $y = r\sin\theta$. Use este método para obter o gráfico da Espiral Galileana $r = \theta^2$ for $0 \leq \theta \leq 10\pi$.

3. Usando esse mesmo método, obtenha o gráfico da 'função de Fey'

$$r = e^{\cos\theta} - 2\cos 4\theta + \text{sen}^5\frac{\theta}{12}$$

no intervalo $0 \leq \theta \leq 24\pi$.

3. Exercício

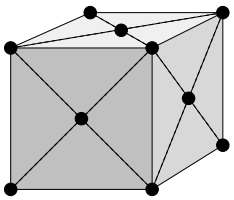
Nos recursos on-line do livro adotado existe um ficheiro de nome `stm.txt`, que contém uma grelha de valores medidos num microscópio de efeito-túnel da superfície (111) do silício. Estes valores representam as alturas na referida superfície. Desenvolva um programa que leia esses dados e depois trace um gráfico de densidades. Estude as melhores opções e variantes de forma a melhor representar a estrutura dessa superfície do silício.

4. Exercício

Usando como ponto de partida o programa do Exemplo 3.2 do livro adoptado, ou se preferir começando do início, faça o seguinte:

1. Um cristal de cloreto de sódio tem átomos de sódio e cloro dispostos em uma rede cúbica, mas os átomos alternam entre o sódio e o cloro, de modo que cada átomo sódio é cercado por seis átomos de cloro e cada átomo cloro é cercado por seis átomos de sódio. Crie uma visualização da estrutura de cloreto de sódio usando duas cores diferentes para representar os dois tipos de átomos.

2. A rede cúbica de face centrada (FCC), que é a rede cristalina mais comum na natureza, consiste de uma estrutura cúbica com átomos posicionados não apenas nos cantos de cada cubo, mas também no centro de cada face:



Faça uma visualização de uma rede do FCC com uma única espécie de átomo (como ocorre no ferro metálico, por exemplo).

5. Exercício

Os seis planetas mais profundos do nosso sistema solar giram em torno do Sol em órbitas aproximadamente circulares, todas localizadas aproximadamente no mesmo plano (elíptico). No seguinte quadro podemos observar os seus parâmetros:

Objecto	Raio (km)	Raio da órbita (milhões de km)	Período da órbita (dias)
Mercúrio	2440	57.9	88.0
Vénus	6052	108.2	224.7
Terra	6371	149.6	365.3
Marte	3386	227.9	687.0
Júpiter	69173	778.5	4331.6
Saturno	57316	1433.4	10759.2
Sol	695500	–	–

Usando as facilidades fornecidas pelo módulo *visual*, crie uma animação do sistema solar que mostre o seguinte:

1. O Sol e planetas como esferas nas suas posições apropriadas e com tamanhos proporcionais aos seus tamanhos reais. Como os raios dos planetas são minúsculos em comparação com as distâncias entre eles, represente os planetas por esferas com raios c_1 vezes maiores que seus valores originais, para que possam ser visualmente distinguida a diferença de raios. Encontre um bom valor para c_1 que torne os planetas visíveis. Também é necessário encontrar um bom raio para o sol. Escolha qualquer valor que forneça uma visualização clara. Não funciona para reescalar o raio do Sol pelo mesmo fator dos planetas porque ficará muito grande. Para um realismo adicional, atribua cores diferentes às esferas. Por exemplo, a Terra pode ser azul e o Sol pode ser amarelo.

2. O movimento dos planetas em torno do Sol (fazendo as esferas dos planetas moverem-se). No interesse de aliviar o tédio, construa seu programa para que o tempo de animação seja um fator de c_2 mais rápido que o tempo real. Encontre um bom valor de c_2 que torne o movimento das órbitas facilmente visível, mas não excessivamente rápido. Faça uso da função *rate* para a animação funcionar sem problemas.

Sugestão: Pode ser útil armazenar as coordenadas das esferas numa matriz do tipo descrito na página 115 do livro adoptado.

6. Exercício

Um dos exemplos mais famosos de caos é o *mapa logístico*, definido pela equação

$$x' = rx(1 - x). \tag{2.1}$$

Para um dado valor da constante r, escolhemos um valor de x - digamos $x = 1/2$ - e avaliámos o lado direito equação para calcular x'. Fazendo $x = x'$ recalculamos o lado direito, e obtemos um novo valor x' e assim por diante. Este é um *mapa iterativo*. Fazendo a mesma operação repetidamente, acontece uma de três coisas a x:

1. O valor de x deixa de variar (ponto fixo). Por exemplo, $x = 0$ é sempre um ponto fixo do mapa logístico (colocando $x = 0$ no lado direito e obtemos $x\prime = 0$ à esquerda).

2. O valor de x não converge para um valor único, mas estabelece um padrão periódico, percorrendo periodicamente um conjunto de valores (ciclo limite).

3. O mapa gera uma sequência aparentemente aleatória de números que parecem não ter nenhum padrão. Este é o caos determinista. *Caos* porque parece *caótico* e *determinista* porque, embora os valores pareçam aleatórios, não o são. São totalmente previsíveis, porque são gerados por uma equação a partir de uma condição inicial. O comportamento é determinado, embora possa não o parecer.

Escreva um programa que calcule e exiba o comportamento do mapa logístico. Para um determinado valor de r, comece com $x = 1/2$ e itere a equação do mapa logístico mil vezes. Esta iteração inicial permite chegar a um ponto fixo ou ciclo limite. De seguida, execute por mais de 1.000 iterações e represente os pontos (r, x) num gráfico com r no eixo horizontal e y no eixo vertical (pode usar a função $plot$ com as opções *"ko"* ou *"k."* para representar pontos ou a função *scatter*). Repita o cálculo para diferentes valores de r de 1 a 4 em passos de $0,01$, representando os pontos para todos os valores de r na mesma figura e, finalmente, usando a função *show* uma vez para exibir a figura completa.

O seu programa deve gerar um gráfico muito típico que se parece com uma árvore inclinada para o lado. Esta famosa figura é chamada de *Feigenbaum plot*, depois da sua descoberta por Mitchell Feigenbaum, ou às vezes por *figtree plot*, uma alusão ao fato às semelhanças com uma árvore e ao facto que Feigenbaum significar *figueira* em alemão.

Responda às seguintes perguntas:

1. Para um dado valor de r, como seria um ponto fixo no gráfico de Feigenbaum? E um ciclo limite? E como seria o caos?

2. Com base no gráfico que obteve, em que valor de *r* o sistema move-se de um comportamento ordenado (pontos fixos ou ciclos de limite) para um comportamento caótico? Este ponto é às vezes chamado de *fronteira do caos*.

O mapa logístico é um sistema matemático muito simples, mas o caos determinístico é observado em muitos sistemas físicos mais complexos, incluindo a dinâmica de fluidos e o clima. Por causa de sua natureza aparentemente aleatória, o comportamento de sistemas caóticos é difícil de prever e é fortemente afetado por pequenas perturbações nas condições externas. Provavelmente já ouviu falar do exemplo clássico do caos na metereologia, o *efeito borboleta*, que foi popularizado pelo físico Edward Lorenz em 1972, numa palestra dada na *Associação Americana para o Avanço da Ciência*, intitulada "*O bater de asas de uma borboleta no Brasil desencadeou um tornado no Texas?*" (embora a primeira pessoa a sugerir o efeito borboleta não fosse um físico, mas o escritor de ficção científica Ray Bradbury no seu famoso conto de 1952, *A Sound of Thunder*, em que a destruição descuidada de uma borboleta pelo viajante do tempo durante uma viagem turística à era jurássica muda o curso da história).

Comentário: Existe outra abordagem para calcular o gráfico Feigenbaum, que é mais puro e rápido, fazendo uso da capacidade do Python de realizar aritmética com arrays. Crie um array r com um elemento contendo cada valor distinto de r que quer investigar: $[1.0, 1.01, 1.02, ...]$. Em seguida, crie outra matriz x do mesmo tamanho para manter os valores correspondentes de x, que devem ser inicialmente definidos como $0, 5$. Numa iteração do mapa logístico podemos executar para todos os valores de r de uma só vez com uma declaração da forma $x = r * x * (1 - x)$. Devido à velocidade com a qual o Python pode executar cálculos em matrizes, esse método deve ser significativamente mais rápido do que o método mais básico acima.

7. Exercício

O conjunto de Mandelbrot, batizado com o nome do matemático francês Benoît Mandelbrot, é um fractal, um objeto matemático infinitamente ramificado que contém estrutura dentro da estrutura e dentro da estrutura, tão fundo quanto observemos. A definição do conjunto Mandelbrot pode ser feita através de números complexos.

Considere a equação

$$z' = z^2 + c,$$

onde z é um número complexo e c é uma constante complexa. Para qualquer valor dado de c, esta equação transforma um número de entrada z num número de saída $z\prime$. A definição do conjunto Mandelbrot envolve a iteração repetida dessa equação: a partir de um valor inicial inicial de z, calculamos a equação para obter um novo valor $z\prime$. Com esse novo valor o fornecemos novamente para obter outro valor, e assim por diante. O conjunto Mandelbrot é o conjunto de pontos no plano complexo que satisfaz a seguinte definição:

> Para um dado valor complexo de c, comece com $z = 0$ e repita o procedimento iterativo. Se a magnitude $|z|$ do valor resultante alguma vez for maior que 2, então o ponto no plano complexo na posição c não está no conjunto Mandelbrot, caso contrário, ele está no conjunto.

Para usar essa definição, seria necessário, em princípio, iterar infinitas vezes para provar que um ponto está no conjunto Mandelbrot, já que um ponto está no conjunto apenas se a iteração *nunca* passar $|z| = 2$. Na

prática, no entanto, apenas executamos um grande número de iterações, digamos 100, e se $|z|$ não excedeu 2, assumimos que não o fará. Escreva um programa para fazer uma imagem do conjunto Mandelbrot executando a iteração para todos os valores de $c = x + iy$ numa grelha $N \times N$ abrangendo a região onde $-2 \leq x \leq 2$ e $-2 \leq y \leq 2$. Faça um gráfico de densidade em que os pontos da grelha dentro do conjunto de Mandelbrot são coloridos em preto e os de fora são coloridos em branco. O conjunto Mandelbrot tem uma forma muito distinta que se parece com um escaravelho com um focinho comprido - saberá quando o vir.

Sugestão: Provavelmente achará útil começar com uma grelha bastante grossa, ou seja, com um pequeno valor de N - talvez $N = 100$ - para que seu programa seja executado rapidamente enquanto testa. Quando tiver certeza de que está funciona corretamente, aumente o valor de N para produzir uma imagem final de alta qualidade da forma do conjunto.

Se estiver entusiasmado, pode fazer outra variante do mesmo exercício que produz imagens incríveis. Em vez de pontos de cor apenas preto ou branco, atribua a cor de acordo com o número de iterações da equação antes de $|z|$ se tornar maior que 2 (ou o número máximo de iterações se $|z|$ nunca se tornar maior que 2). Se usar um dos esquemas de cores que o Python fornece para gráficos de densidade, como os esquemas *"hot"* ou *"jet"*, pode fazer algumas imagens espetaculares. Outra variante interessante é colorir de acordo com o logaritmo do número de iterações, o que ajuda a revelar algumas das estruturas mais finas fora do conjunto.

Física Computacional

Ano lectivo 2017/2018

Folha 3 - Exactidão e velocidade

1. Exercício

Escreva um programa para calcular e imprimir o fatorial de um número digitado pelo utilizador. Se quiser, pode basear seu programa na função para fatorial na Seção 2.6, mas escreva seu programa para que calcule o fatorial usando variáveis inteiras, não variáveis de ponto flutuante. Use seu programa para calcular o fatorial de 200.

Modifique seu programa para usar variáveis de ponto flutuante e, novamente, calcule o fatorial de 200? Discuta o resultado.

2. Exercício

1. Escreva um programa que tome como dados três números, a, b e c, e imprima as duas soluções na equação quadrática $ax^2 + bx + c = 0$ usando a fórmula resolvente,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

   Use seu programa para calcular as soluções de $0.001x^2 + 1000x + 0.001 = 0$.

2. Existe outra maneira de escrever as soluções para uma equação quadrática. Multiplicando a parte superior e inferior da solução acima por $-b \mp \sqrt{b^2 - 4ac}$, mostre que as soluções também podem ser escritas como

$$x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}.$$

   Adicione código ao seu programa para imprimir esses valores além dos anteriores e use novamente o programa para resolver $0.001x^2 + 1000x + 0.001 = 0$. O que vê? Como o explica ?

3. Usando o que aprendeu, escreva um novo programa que calcule ambas as raízes de uma equação quadrática com precisão em todos os casos.

Este é um bom exemplo de como os computadores nem sempre funcionam como se espera. Se aplicar simplesmente a fórmula padrão para a equação quadrática, pode receber uma resposta errada. Na prática, o método que elaborou aqui é a maneira correta de resolver uma equação quadrática num computador, mesmo que seja mais complicado do que a fórmula padrão. Se estivesse em escrever um programa que envolvesse a solução de muitas equações quadráticas, este método poderia ser um bom candidato para definir função: poderia colocar os detalhes do método dentro de uma função para evitar o problema de passar passo a passo todas as vezes que tiver uma nova equação para resolver.

3. Exercício

Suponha que temos função $f(x)$ e queremos calcular sua derivada num ponto x. Podemos fazer isso com lápis e papel, se soubermos a forma matemática da função, ou podemos fazê-lo no computador, fazendo uso da definição da derivada:

$$\frac{df}{dx} = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta}.$$

No computador, não podemos realmente tomar o limite, pois δ vai para zero, mas podemos obter uma aproximação razoável apenas fazendo δ pequeno.

1. Escreva um programa que defina uma função `f(x)` que devolve o valor $x(x-1)$, calcula a derivada da função no ponto $x = 1$ usando a fórmula acima com $\delta = 10^{-2}$. Calcule o valor real da mesma derivada analiticamente e compare com a resposta do seu programa. Os dois não concordarão perfeitamente. Por que não?

2. Repita o cálculo para para $\delta = 10^{-4}$, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12}, e 10^{-14}. Verá que a precisão do cálculo inicialmente melhora à medida que o delta diminui, mas piora novamente. Porquê ?

Vamos examinar derivadas numéricas em mais detalhe na Secção 5.10, onde estudaremos técnicas para lidar com esses problemas e maximizar a precisão de nossos cálculos.

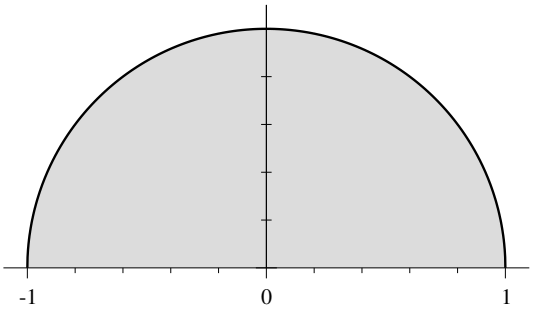
4. Exercício

Cálculo de integrais

Suponha que queremos calcular o valor do integral

$$I = \int_{-1}^{1} \sqrt{1-x^2}dx.$$

o Gráfico da função integranda é um semicírculo de raio 1:



e, portanto, o valor do integral—a área sob a curva—deve ser $\pi/2 = 1.57079632679\ldots$

Alternativamente, podemos avaliar o integral no computador dividindo o domínio de integração em um grande número N de fatias de largura $h = 2/N$ cada, e usando a definição de Riemann da integral:

$$I = \lim_{N \to \infty} \sum_{k=1}^{N} h y_k \,,$$

em que

$$y_k = \sqrt{1-x_k^2} \qquad \text{and} \qquad x_k = -1 + hk.$$

Na prática, não podemos tomar o limite $N \to \infty$, mas podemos fazer uma aproximação razoável tornando N grande.

1. Escreva um programa para avaliar a integral acima com $N = 100$ e compare o resultado com o valor exato. Os dois não concordarão muito bem, porque $N = 100$ não é um número suficientemente grande de fatias.

2. Aumente o valor de N para obter um valor mais preciso para o integral. Se precisar que o programa seja executado em cerca de um segundo ou menos, qual a precisão que consegue?

O cálculo de integrais é uma tarefa comum em física computacional. Estudaremos técnicas para fazer integrais em detalhe no próximo capítulo. Como veremos, existem métodos substancialmente mais rápidos e precisos do que o que usamos aqui.

Física Computacional

Ano lectivo 2017/2018

Folha 4 - Integrais e derivadas

1. Exercício

Nos recursos on-line encontra um ficheiro com o nome velocities.txt que contém duas colunas de números, a primeira representando o tempo t em segundos e a segunda a velocidade x em metros por segundo de uma partícula, medida uma vez por segundo desde o tempo $t = 0$ até $t = 100$. As primeiras linhas têm o aspecto:

```
0 0
1 0.069478
2 0.137694
3 0.204332
4 0.269083
5 0.331656
```

Escreva um programa para fazer o seguinte:

1. Ler os dados e, usando a regra trapezoidal, calcular a distância aproximada percorrida pela partícula na direção x em função do tempo. Veja a Seção 2.4.3 na página 57 se quiser recordar como ler dados de um arquivo.

2. Fazer um gráfico que mostre, na mesmo janela, a curva de velocidade original e a distância percorrida em função do tempo.

2. Exercício

1. Escreva um programa para calcular um valor aproximado para o integral $\int_0^2 (x^4 - 2x + 1)dx$ do Exemplo 5.1, mas usando a regra de Simpson com 10 intervalos, em vez da regra trapezoidal. Pode querer basear seu programa no programa de regras trapezoidais na página 142.

2. Execute o programa e compare seu resultado com o valor correto conhecido de 4.4. Qual é o erro fraccionário no seu cálculo?

3. Modifique o programa para usar uma centena de intervalos, e depois mil. Observe a melhoria no resultado. Como se comparam os resultados aos do Exemplo 5.1 para a regra trapezoidal com o mesmo número de intervalos?

3. Exercício

Considere o integral

$$E(x) = \int_0^x e^{-t^2} dt.$$

1. Escreva um programa para calcular $E(x)$ para valores de x de 0 a 3 em passos de 0.1. Escolha o método que você usará para executar a integral e um número adequado de intervalos.

2. Quando estiver convencido de que seu programa está a funcionar, complete-o para fazer um gráfico de $E(x)$ como uma função de x. Se quiser lembrar-se de como fazer um gráfico, consulte a Seção 3.1, na página 88.

4. Exercício

The diffraction limit of a telescope

Nossa capacidade para resolver detalhes em observações astronómicas é limitada pela difração de luz nos telescópios. A luz das estrelas pode ser tratada efetivamente como proveniente de uma fonte pontual no infinito. Quando essa luz, com comprimento de onda λ, passa pela abertura circular de um telescópio (que assumimos ter raio unitário) e é focada pelo telescópio no plano focal, não produz um único ponto, mas um padrão circular de difração, consistindo num ponto central rodeado por uma série de anéis concêntricos. A intensidade da luz neste padrão de difração é dada por

$$I(r) = \left(\frac{J_1(kr)}{kr}\right)^2,$$

onde r é a distância no plano focal do centro do padrão de difração, $k = 2\pi/\lambda$, e $J_1(x)$ é uma função de Bessel. As funções de Bessel,$J_m(x)$, são dadas por

$$J_m(x) = \frac{1}{\pi}\int_0^\pi \cos(m\theta - x\sin\theta)d\theta,$$

onde m é um inteiro não negativo e $x \geq 0$.

1. Escreva uma função Python `J(m, x)` que calcule o valor de $J_m(x)$ usando a regra de Simpson com $N = 1000$ pontos. Use a sua função num programa para fazer um gráfico, numa única janela , das funções de Bessel $J_0, J_1 e J_2$ como um funções de x, de x = 0 a $x = 20$.

2. Escreva um segundo programa que faça um gráfico de densidade da intensidade do padrão de difração circular de uma fonte de luz pontual com $\lambda = 500$ nm, num quadrado do plano focal, usando a fórmula dada acima. A sua imagem deve cobrir valores de$r$ de zero a cerca de 1 μm.

Sugestão 1: Você pode achar útil saber que$\lim_{x\to 0} J_1(x)/x = 1/2$. Sugestão 2 : O ponto central no padrão de difração é tão brilhante que pode ser difícil ver os anéis ao seu redor no ecran do computador. Se se deparar com este problema, uma maneira simples de lidar com ele é usar um dos outros esquemas de cores para gráficos de densidade descritos na Secção 3.3. O esquema `hot` funciona bem. Para uma solução mais sofisticada do problema, o `imshow` tem um argumento adicional `vmax` que permite definir o valor que corresponde ao ponto mais claro da gráfico. Por exemplo, se disser `imshow (x, vmax = 0.1)`, os elementos com valor 0,1, ou qualquer valor maior, produzirão a cor mais brilhante (mais positiva) na tela. Baixando `vma`, pode reduzir o intervalo total de valores entre o brilho mínimo e máximo e, portanto, aumentar a sensibilidade do gráfico,

tornando detalhes subtis visíveis. (Há também um argumento vmin que pode ser usado para definir o valor que corresponde à cor mais escura (mais negativa).) Para este exercício, um valor $vmax = 0.01$ parece funcionar bem.

5. Exercício

Considere o integral

$$I = \int_0^1 \sin^2 \sqrt{100x}dx$$

1. Escreva um programa que use o método de regra trapezoidal adaptativa da Seção 5.3 e Eq. (5.34) para calcular o valor deste integral com uma precisão aproximada de $\epsilon = 10^{-6}$ (isto é, correcto até seis dígitos após o ponto decimal). Comece com uma único intervalo de integração e aumente a partir daí para dois, quatro, oito e assim por diante. Peça ao seu programa para imprimir o número de intervalos, a sua estimativa do integral e a sua estimativa do erro no integral, para cada valor do número de intervalos$N$, até que a precisão desejada seja atingida. (Dica: deverá encontrar o resultado em torno de I = 0,45)

2. Agora modifique seu programa para avaliar o mesma integral usando a técnica de integração Romberg descrita nesta seção. Peça ao seu programa para imprimir uma tabela triangular de valores, como na página 161, de todas as estimativas de Romberg do integral. Calcule o erro em suas estimativas usando Eq. (5.49) e continue novamente o cálculo até chegar a uma precisão de $\epsilon = 10^{-6}$. Deverá descobrir que o método Romberg atinge a precisão requerida consideravelmente mais rápido que a regra trapezoidal por si só.