Métodos Computacionais em Engenharia (FIS3022)
Problemas 0
DFA–FCUP 2021-22
25 de Fevereiro de 2022

1. Escreva um código para calcular a precisão de trabalho do seu computador.

2. **Numpy arrays**

   O `numpy` permite-nos definir objectos mais parecidos com matrizes (num sentido geral, incluindo escalares e vectores) do que as listas do Python "básico". Podem ser definidos como:

```
import numpy as np
a = np.array([1,2,3,4])
b = np.array([[1,2,3],[4,5,6]]).
```

   A estrutura `array` possui vários métodos (operações na estrutura, em geral para obter informação das suas propriedades). Podemos determinar a dimensão (`a.ndim`), a forma (`a.shape`), o tamanho segundo uma dimensão (`a.len`), ou o tipo de e lementos (`a.dtype`). Tem também alguns métodos para criar matrizes sem ter que inserir os elementos à mão (`arange`, `linspace`, `zeros`, `ones`, `eye`,...), ou alterar a forma de arrays (`a.reshape`). Temos também o conceito de *slicing*(fatiar!), e *logical indexing* que nos permitem escolher apenas alguns elementos, com uma regra regular ou obedecendo a uma condição lógica, respectivamente.

   (a) Produza um array que descreva o vector v=(1,3,4,0,20,23,0,21,1,2,5,3,7,8).

   (b) De um modo elegante (isto quer dizer poucas linhas de código, preferencialmente uma! Mas sem ser críptica!) escreva um código que determina o tipo de variáveis, o tipo dos elementos, a forma do array, o tamanho e número de dimensões;

   (c) mude o 3º elemento para -5;

   (d) mude os últimos 6 elementos para 1;

   (e) determine os índices dos elementos do array que têm o valor 0;

   (f) gere um segundo array que contenha apenas os elementos pares do primeiro;

   (g) gere um terceiro array que tenha elementos alternados do array original.

   (h) o que obtem com a instrução `v[::-1]`? E com `v[...,0]`? E `v[0,...]`?

   (i) o que acontece quando tenta somar dois arrays com tipos diferentes? Por exemplo, tente somar os arrays definidos por `a = np.ones(4,dtype='int64')` e `b = np.arange(4,dtype='complex128')`.

3. **Numpy arrays 2**

   Uma das vantagens dos arrays do numpy versus as listas do Python é a possibilidade de usar os símbolos habituais para operações algébricas (+,-,*,/). Mas o resultado nem sempre é o que poderíamos esperar. Todas estas operações ocorrem elemento a elemento. Se quizer fazer o produto habitual de matrizes tenho que usar uma outra função ou método: `dot`. Defina as matrizes:

```
a = np.arange(4).reshape(2,2)
b = np.array([[1,1],[2,2]])
```

e obtenha os produtos:

(a) `a*b`

(b) `a.dot(b)`

4. **Numpy arrays 3**

   Em Álgebra Linear (AL), se $\mathbf{v}$ é um vector linha, então o produto $\mathbf{v}.\mathbf{v}^T$ resulta num escalar, enquanto que $\mathbf{v}^T.\mathbf{v}$ resulta numa matriz de *rank* 2. Considere o vector real linha (1., 2., ..., 5).

   (a) Obtenha o seu vector transposta. Verifique que obteve o que queria imprimindo no ecrã o resultado. O que lhe dá o método `shape`?

   (b) calcule o quadrado da norma (sem usar o método `v.norm` no `numpy`). Procure info sobre a construção `None` ou `v.newaxis`.

   (c) calcule agora a matriz obtida multiplicando os vectores $\mathbf{v}$ e $\mathbf{v}^T$ pela ordem inversa.

5. **Numpy arrays em 2D**

   Um elemento recorrente nesta UC será uma grelha de pontos (equidistantes) a representar um domínio rectangular. Como veremos, descrever um problema no espaço começa por discretizar o domínio (passar do contínuo real para o discreto numérico), em que cada ponto da grelha terá uma coordenada x e outra y. Mas aqui chocam duas convenções habituais. Em AL o primeiro índice refere-se à linha (que cresce de cima para baixo), enquanto o segundo à coluna (que cresce da esquerda para a direita). Mas em gráficos em geral a direcção x é horizontal (e cresce da esquerda para a direita), e a direcção y é vertical e cresce de baixo para cima! E um ponto no plano é em geral descrito pelo par de coordenadas (x, y), em que o x é a primeira coordenada. Isto pode levar a confusões, já que a primeira coordenada (x) é horizontal, enquanto o primeiro índice da matriz varia verticalmente (e cresce para baixo). Há várias maneiras de abordar a questão (até ignorá-la pode resultar, mas também pode levar a erros!).

   Vamos considerar um domínio espacial centrado na origem, de lado 10 segundo x e 4 segundo y ($x \in [-5, 5]$ e $y \in [-2, 2]$). Vamos tomar 11 pontos na direcção x e 9 na direcção y.

   (a) gere um vector para as coordenadas do eixo dos x, que contenha os pontos extremos do domínio. Gere outro para o eixo dos y.

   (b) gere matrizes rank 2 para as coordenadas x e y dos pontos do domínio. Leia a descrição do método `meshgrid` do `matplotlib`.

   (c) Imprima no ecrã os resultados e compare cada matriz com os vectores que criou para os eixos. Em que direcção têm as matrizes mais elementos? É o que esperava?

   (d) Defina em cada ponto do domínio a função $f(x, y) = \exp(-0.2 * x^2 - 2 * y^2)$.

   (e) use a função imagesc para fazer um plot dessa função no domínio. Rotule os eixos e use `ticklabels`.

   (f) o que pode alterar no procedimento seguido para melhorar a correspondência entre linhas/-colunas das matrizes e o que esperamos para o comportamento dos eixos x e y?

6. Gere dois arrays, sejam x e y, com 200 elementos cada, cujos valores são obtidos aleatoriamente no intervalo [0,15]. Produza um gráfico "scatter"(de pontos) em que as coordenadas de cada ponto correspondem aos elementos dos arrays x e y (procure na web informação sobre geradores de números aleatórios em numpy/scipy, e sobre gráficos "scatter"com matplotlib).

7. Faça a representação gráfica do erro cometido ao calcular a derivada da função $y = \exp(x)$ usando a aproximaçao de primeira ordem com:

   (a) diferenças finitas "forward"(para a frente);

   (b) diferenças finitas "backward"(para trás);

   (c) diferenças finitas simétricas (centrais)

   para diferentes valores do passo h e determine o passo óptimo para cada método. Compare o erro de cada método. Tente fazer a impressão dos resultados na forma de tabelas, com as colunas com iguais dimensões e os valores alinhados.

8. Escreva um código que faça um plot da família de funções

$$f_n(x) = \sin(nx)$$

   na gama

$$x \subset [0, 2\pi]$$

   para

$$n \subset 1, 2, 4, , 16.$$

   Para distinguir as diferentes funções, use diferentes formatos de linha (mude a côr, símbolos e tipo de linha, etc.). Não se esqueça de incluir um título e rótulos nos eixos, tal como nos gráficos que produz para os relatórios da UCs de laboratório.

   Alguns argumentos úteis para "fazer plots":

```
estilos de linhas: - = sólido          marcadores: . = pontos
--= tracejado                       o = círculos
:= pontilhado                      s = quadrados
-.= traço-ponto                     D = losângulos
h = hexágonos
8 = octágonos
^ = triângulos para cima
v = triângulos para baixo
cores:r = vermelho
g = verde
b = azul
k = preto
c = turqueza
m = magenta
y = amarelo
w = branco
```

   Um exemplo:

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 8), dpi=80)

x    = np.array( [ 1, 2, 3,4, 5 ] )
y    = np.array([0.9,4.1,8.7,16.5,24.9])
```

```
xerr = np.array([0.1,0.1,0.1,0.1,0.1])
yerr = np.array([0.6,0.9,0.75,0.9,1.])
plt.errorbar(x,y,yerr,xerr,ls='N ne',marker= 'o',ms = 4 )
plt.plot(t,y)
plt.xlabel('time (s)')
plt.ylabel('voltage(mV)')
plt.title('A Simple Plot')
plt.grid()
plt.xlim(0, 1.5)
plt.ylim(-1.5, 1.5)
plt.legend(loc = 'upper left')
plt.show()
```

9. Escreva um código que faça o plot da função $g(x, y) = \sin(x)\cos(y)$ no domínio $[-\pi, \pi] \times [-\pi, \pi]$, usando a funções `contour` e `contourf`, para 12 níveis.

10. Repita o problema anterior agora usando a função `imshow` e `colorbar`, e `colormap`.