Programació de Simulacions i d'Instruments de Mesura. Exercicis de programació en Python. Grau de Física. Curs 2019-20.

Artur Carnicer, artur.carnicer@ub.edu

18 d'octubre de 2019

Índex

Exercici 1

Operacions bàsiques

En aquesta pràctica calcularem el valor de π de diverses maneres. A partir dels exercicis introduirem diverses instruccions d'ús freqüent en Python.

1. Calculeu π mitjançant la fórmula Bailey-Borwein-Plouffe, `https://en.wikipedia.org/wiki/Bailey-Borwein-Plouffe_formula`. Itereu els termes de la successió mitjançant un bucle `while`. Indiqueu l'error en cada iteració (atureu el càlcul quan l'error sigui prou petit).

   - Feu una gràfica de l'error en funció del número d'iteracions. Feu servir `matplolib.pyplot.figure`, `matplolib.pyplot.plot`, etcètera. Assegureu-cos que la finestra es veu en una finestra independent (seguiu les indicacions al campus virtual). Feu servir escala logarítmica per a l'eix y. Decoreu la gràfica adequadament: graella, llegenda, tituleu els eixos, etcètera. Noteu que es poden posar caràcter grecs, i símbols matemàtics fent servir codis LaTeX.
   - Guardeu els valors de la gràfica en un arxiu de text pla amb extensió `.csv`. Aquests arxius poden ser oberts en un full de càlcul. Considereu les ordres `numpy.loadtxt` i `numpy.savetxt`. Finalment, guardeu la imatge com un fitxer `.png` amb `matplolib.pyplot.savefig`.

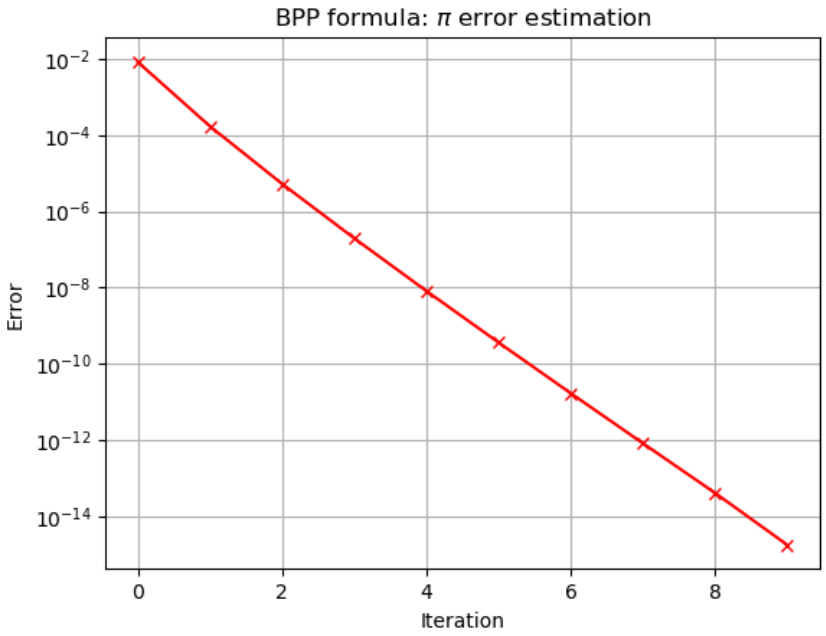


Figura 1.1: Evolució de l'error

2. Calculeu π a partir de les integrals següents:

$$\int_{-\infty}^{\infty} \exp(-x^2)\, dx = \sqrt{\pi} \tag{1.1}$$

$$\int_{-\infty}^{\infty} \frac{1}{1+x^2}\, dx = \pi \tag{1.2}$$

   Estudieu la informació que trobareu a `https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html`. Feu servir `scipy.integrate.quad` i/o `scipy.integrate.simpson`. Quan cal fer servir un o altre? Recordeu que per definir els valor de les abscisses on calcular la integral podeu fer servir `numpy.linspace` o `numpy.arange`. Noteu que Python disposa de valors predefinits (veieu `https://www.numpy.org/devdocs/reference/constants.html`). En aquest exercici us pot ser d'utilitat `np.Inf`.

3. Considereu un quadrat de costat L i el cercle inscrit en aquest quadrat. Determineu π a partir del quocient entre el número de vegades que els punts cauen al cercle i el nombre total de punts calculats: prenent números a l'atzar, genereu les coordenades de punts a l'interior del quadrat. Feu servir `numpy.random.random`. Determineu si aquests punts es troben també a l'interior del cercle. Mostreu i guardeu la gràfica de l'error en funció del nombre d'iteracions.

4. Curiositat 1. Probeu de calcular `np.Inf - np.Inf`, `0 * np.Inf` o `np.Inf / np.Inf`. Què vol dir `np.NaN`? Poseu a la línea de comandes `np.NaN == np.NaN` o `np.NaN != np.NaN`. Té sentit?

5. Curiositat 2. Cerqueu informació sobre sencers de 8, 16, 32 i 64 bits. Estudieu el contingut d'aquest article publicat a 'Wired': `https://www.wired.com/2014/12/gangnam-style-youtube-math/`.

.

Exercici 2

Numpy: Representació dels conjunts de Mandelbrot i Julia

Els objectius d'aquesta primera pràctica són múltiples:

- Familiaritzar-se amb l'ús d'estructures bidimensionals: `numpy.meshgrid`.
- Utilitzar operación lògiques amb estructures indexades sense fer servir `if`.
- Visualitzar i guardar matrius en fals color amb funcions del mòdul `matplotlib.pyplot`.
- Generar arxius dinàmics (`.gif`) amb `imageio.mimsave`.

El Conjunt de Mandelbrot (CM) es defineix com el conjunt de punts $c \in \mathbb{C}$, pels quals la successió

$$z_{n+1} = z_n^2 + c \quad \text{amb} \quad z_0 = 0 \tag{2.1}$$

convergeix. Es pot demostrar que aquesta sèrie convergeix si $\|z_n\| < 2$. El CM es visualitza representant els valors de c en el pla complex. Caldrà calcular diverses iteracions de la sèrie z_n per tots el valors c fins arribar a un comportament estable de la representació del CM. La figura mostra els CM calculats per n =2, 4, 10 i 100. Els píxels blancs de la imatge indiquen el punts que pertanyen al CM.
L'objectiu d'aquesta pràctica és desenvolupar el codi per generar la representació gràfica del CM.

2.1 Indicacions per desenvolupar el codi

1. Considereu inicialment que els valors que pot prendre c es troben a l'interval `-2 - 1.5j` i `1 + 1.5j`. Més tard podreu canviar l'escala i verificar el caràcter fractal del CM en determinades zones d'interès. Genereu la matriu complexa de dimensió MxN que contindrà els valors de c. Els valors de la dimensió de la matriu són variables de disseny i els heu d'escollir vosaltres.
2. Utilitzeu la funció `numpy.meshgrid`. Estudieu el que diu la documentació d'aquesta funció per tal d'entendre el seu funcionament. Construïu la matriu de valors de c fent `c = X + 1j * Y`, on `X` i `Y` surten de fer `X, Y = np.meshgrid(... , ...)`.
3. Genereu la matriu CM que contindrà la representació del conjunt.
4. Feu un bucle que vagi d'1 fins el número d'iteracions que vulgueu. Calculeu el valor de z_{n+1} per cada iteració n fent `z = z**2 + c` i apliqueu el criteri de convergència. Si aquest es verifica en el punt considerat, CM val 1, altrament aquest serà 0 (feu, per exemple, `CM = np.abs(z) < 2`).
5. Representeu la matriu CM obtinguda amb `plt.imshow(CM, cmap='gray')`.
6. Pels valors c fora del CM, compteu el número d'iteracions necessàries per verificar que no es compleix la condició de convergència en el punt c. Genereu una nova matriu M amb aquests valors. Representeu-la amb `plt.imshow()`. Fent servir `subplot()` podreu representar diverses imatges en una sola finestra.

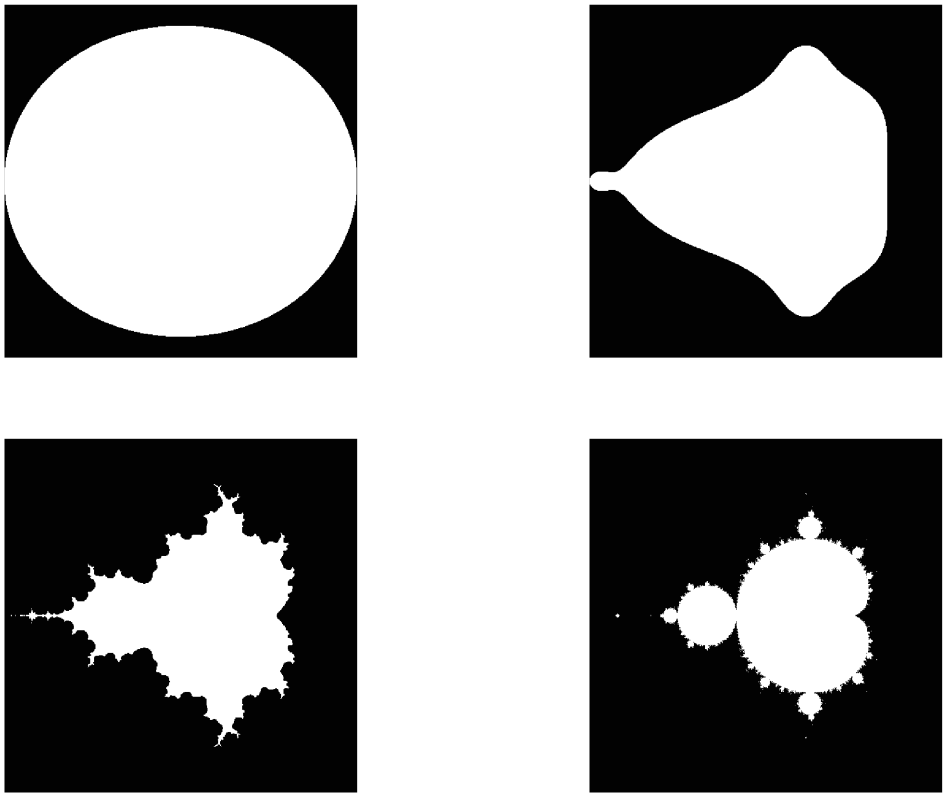


Figura 2.1: CM determinats segons el número d'iteracions

7. Podeu consultar els mapes de color disponibles a `http://matplotlib.org/examples/color/colormaps_reference.html`.

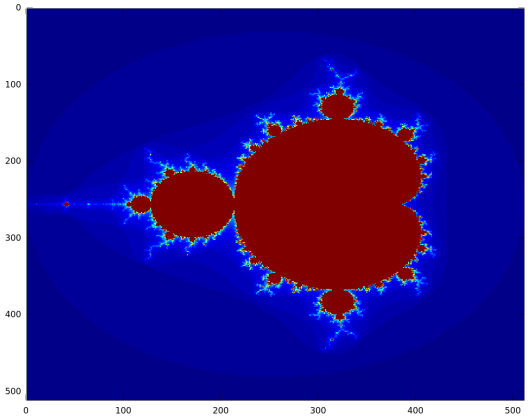


Figura 2.2: CM acolorit representat la velocitat de convergència

8. Si guardeu en fitxer les imatges generades en cada iteració podreu fusionar-les en un fitxer `.gif` i visualitzar la seqüència. Consulteu
   `https://stackoverflow.com/questions/753190/programmatically-generate-video-or-animated-gif-in-python/10376713`

per veure un exemple d'utilització de `imageio.mimsave`.

9. Implementeu fractals del conjunt de Julia. Per exemple $z_{n+1} = z_n^2 - 0.7269 + 0.1889 * 1j$, amb $z_0 = 0.7269 + 0.1889 * 1j$ (o el valor que volgueu). A `https://en.wikipedia.org/wiki/Julia_set` trobareu molts exemples interessants.

2.2 Presentació de resultats

El codi que heu de presentar ha de generar quatre figures: la primera ha de mostrar el CM per cada iteració i l'altra ha de mostrar la velocitat de convergència. Les altres dues han de presentar la mateixa informació però per un fractal del conjunt de Julia. Incloeu comentaris que expliquin el que esteu fent. Assegureu-vos que el codi s'executa correctament. Important: aquest script NO ha de guardar imatges ni generar l'arxiu .gif. Anomeneu el vostre script `nom_cognom1_cognom2.py`; assegureu-vos de no fer servir espais ni accents. Pugeu-lo al campus virtual. La data límit de presentació s'indicarà oportunament.

Exercici 3

Equacions diferencials ordinàries en problemes de Física

En aquest projecte mostrarem les tècniques de resolució d'equacions diferencial ordinàries que aplicarem a problemes de mecànica: pèndols, oscil·ladors acoblats i el moviment en un cos sotmès a una força gravitatòria.

3.1 El pèndol esmorteït i forçat

La segona llei de Newton del moviment és

$$\ddot{\theta} = -\frac{g}{L}\sin\theta + \frac{-b\dot{\theta} + A\cos\Omega t}{mL^2} \tag{3.1}$$

on θ és l'angle del pèndul, g l'acceleració de la gravetat, L i m són la longitud i la massa del pèndul, b el coeficient de fricció, A i Ω l'amplitud i la freqüència de la força externa i t és el temps. La velocitat angular ω és, òbviament, $\dot{\theta} = \omega$.
L'ordre Python que permet resoldre equacions diferencials és:

`scipy.integrate.odeint(equdif, ci, t)`.

La primera de les variables `"equdif"` [1] és una referència a la funció que conté la descripció del sistema d'equacions; `ci`, és un vector amb les condicions inicials i `t`, és un vector amb els punts on s'avalua la solució (el temps en el nostre cas). La funció `odeint` retorna la matriu `x` que conté el resultat.
`odeint` s'utilitza per calcular sistemes d'equacions diferencials de primer ordre. No obstant, en el cas que ens ocupa el que tenim és una única equació diferencial de segon ordre. Per treballar amb aquesta funció cal convertir una equació de segon ordre en un sistema d'equacions diferencials de primer ordre.
Considerem inicialment el cas del pèndol simple. Aquest segueix l'equació diferencial $\dot{\omega} = -\frac{g}{L}\theta$, que escrit com un sistema d'equacions esdevé:

$$\begin{aligned} \dot{\omega} &= -\frac{g}{L}\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{3.2}$$

La part més delicada és com s'escriuen les equacions anteriors en la funció `equdif()` que serà cridada per `odeint`. Per exemple, aquesta funció s'ha de declarar així:

```
 def equdif(y,t):
"""
```

[1] Els noms de les funcions i de les variables utilitzades en el guió poden ser unes altres. S'han utilitzat aquestes només per il·lustrar l'exemple.

```
y[0]=velocitat
y[1]=angle
"""
    return -g * y[1] / L, y[0]
```

Obtingueu els diagrames $\omega(t)$ i $\theta(t)$. Utilitzeu l'ordre `subplot` per organitzar els resultats en una única figura. Podeu trobar més informació sobre `odeint` a la pàgina corresponent del manual [1]. Reproduïu l'exemple per familiaritzar-vos amb el format. Generalitzeu el problema a un oscil·lador esmorteït i forçat descrit per l'equació diferencial de l'equació 3.1.
Presentació de resultats: mostreu els diagrames $\omega(t)$ i $\theta(t)$ d'un pèndol sotmès a fregament i una força externa:

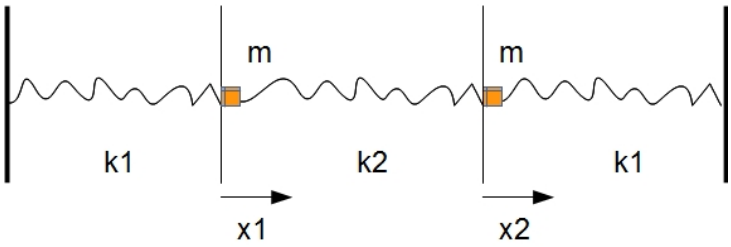
- Considereu primer el cas senzill $b = 0$ i $A = 0$. Doneu valors raonables per a m i L i doneu les condicions inicials.
- Repetiu el punt anterior considerant $b \neq 0$.
- Considereu ara un cas hipotètic en que $A = 1.35$, $m = 1$, $g = 1$, $L = 1$, $b = 0.5$ i $\Omega = 0.666$.

Mostreu tres finestres, una per cada cas considerat: pèndol simple, esmorteït i forçat.

3.2 Oscil·ladors acoblats

Considereu l'equació diferencial d'un sistema acoblat com l'indicat a la figura:

$$m\ddot{x_1} + (k_1 + k_2)x_1 - k_2x_2 = 0 \tag{3.3}$$
$$m\ddot{x_2} + (k_1 + k_2)x_2 - k_2x_1 = 0 \tag{3.4}$$



Resoleu l'equació diferencial per $m = 1$, $k_1 = 10$ i $k_2 = 0.5$ (unitats SI), amb les condicions inicials $x_1(0) = 1$, $x_2(0) = 0$, $v_1(0) = 0$, $v_2(0) = 0$. Feu variar el temps entre 0 i 40 s.

1. Genereu una gràfica que mostri l'evolució de $x_1(t)$ i $x_2(t)$. Poseu una etiqueta a l'eix 'x' que digui 'Temps (s)' i una a l'eix 'y' que digui 'Posició partícula'. Mostreu una graella superposada al gràfic.
2. Genereu una gràfica que mostri l'evolució de $v_1(t)$ i $v_2(t)$. Poseu una etiqueta a l'eix 'x' que digui 'Temps (s)' i una a l'eix 'y' que digui 'Velocitat partícula' . Mostreu una graella superposada al gràfic.
3. Mostreu les dues gràfiques en una finestra.

3.3 Moviment d'un cos en un camp gravitatori

En aquest cas es tracta de resoldre l'equació

$$\vec{F} = -G\frac{Mm}{r^2}\hat{r} \tag{3.5}$$

que en coordenades cartesianes s'escriu

$$\ddot{x} = -\frac{GMx}{(x^2+y^2)^{3/2}} \tag{3.6}$$

$$\ddot{y} = -\frac{GMy}{(x^2+y^2)^{3/2}} \tag{3.7}$$

Les variables en aquest problema seran x, y, v_x i v_y. Transcriviu el sistema d'equacions diferencials anteriors. Feu els càlculs per la Terra i el Cometa Halley al voltant del Sol. Mostreu $x(t)$, $y(t)$, $|v|(t)$ i $y(x)$. Mostreu les quatre gràfiques agrupades adequadament en una figura. Feu servir una finestra per la Terra i una pel Halley. Per aquest darrer cas, feu servir les següents dades[2]:

```
G = 6.67e-11 (SI)
M⊙ = 1.9891e30 Kg
1 au = 1.49598e11 m
aphelion = 35.082*au
vel_ap = 0.869e3 m/s
T = 3.15576e7 * 75.32 s
```

.

3.4 Condicions inicials definides en punts arbitraris.

Les funcions d'Hermite

$$\Psi_n(x) = \left(2^n n!\sqrt{\pi}\right)^{-1/2} \exp(-x^2/2) H_n(x) \qquad \text{amb } n = 0, 1, 2, \ldots \tag{3.8}$$

són solucions analítiques de l'equació d'Schrödinger per a l'oscil·lador harmònic

$$-\frac{\hbar}{2m}\Psi"(x) + \frac{1}{2}m\omega^2 x^2 \Psi(x) = E\Psi(x) \tag{3.9}$$

on $H_n(x)$ són els polinomis d'Hermite amb la normalització utilitzada a Física, $\omega^2 = k/m$ sent m la massa de la partícula i k la constant elàstica i $E = (n + \frac{1}{2})\hbar\omega$ els nivells d'energia ($n = 0, 1, 2, \ldots$). Agrupant les constants, l'equació anterior es pot escriure com

$$\Psi"(x) + \left((2k+1) - Bx^2\right)\Psi(x) = 0. \tag{3.10}$$

Resoleu l'equació diferencial per $x \in [-5, 5]$, pels casos $B = 1$, $k = 0, 2, 4$. Feu servir les condicions inicials següents:

- $\Psi_0(0) = 0.751126, \Psi_0'(0) = 0$
- $\Psi_2(0) = -0.531125, \Psi_2'(0) = 0$
- $\Psi_4(0) = 0.459969, \Psi_4'(0) = 0$

Obriu una finestra amb dos subplots. En el primer, mostreu les funcions d'Hermite per $n = 0, 2, 4$ [Eq. (3.8)]. En el segon, mostreu $\Psi(x)$ calculant l'equació diferencial [Eq. (3.10]. Escaleu l'eix x entre -5 i 5 i l'eix y entre -0.6 i 0.8. Nota: la funció `scipy.special.eval_hermite` us pot resultar d'utilitat.

3.5 Presentació de resultats

El codi que heu de presentar ha de generar totes les figures que s'indiquen. Incloeu comentaris que expliquin el que esteu fent. Assegureu-vos que el codi s'executa correctament. Anomeneu el vostre script `nom_cognom1_cognom2.py`; assegureu-vos de no fer servir espais ni accents. Pugeu-lo al campus virtual. La data límit de presentació s'indicarà oportunament.

Exercici 4

Càlculs que involucren matrius: sistemes multicapes

Normalment, els sistemes òptics estan recoberts per una pel·lícula dielèctrica per evitar reflexos. Treballar amb elements recoberts és especialment important en sistemes formats per múltiples components. En cada superfície que travessa el feix es produeix una reflexió que fa que la llum total a la sortida de l'instrument sigui una fracció de la llum incident. Els recobriments consisteixen en una pila de materials dielèctric de gruix molt prim que es dipositen sobre la superfície del component òptic. Habitualment s'alternen capes d'índex de refracció alt i petit que tenen un gruix de $\lambda/4$, tal i com indica la figura. Un exemple senzill de recobriment és el format per una única capa de gruix $d = \dfrac{\lambda_0}{4n}$ (λ_0 és longitud d'ona en el buit) feta amb un material d'índex n que verifica $n = \sqrt{n_s}$ sent n_s l'index del substrat.

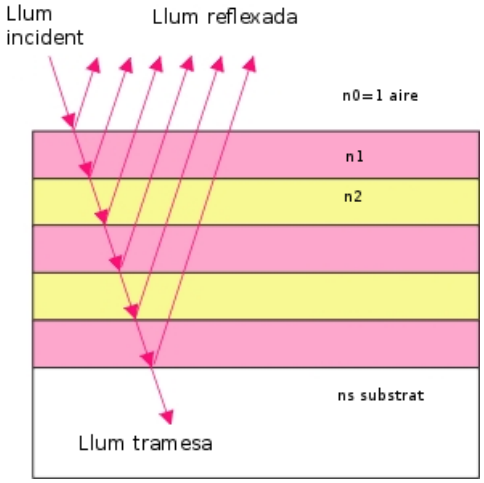


Figura 4.1: Sistema multicapes

En aquest projecte es proposa calcular la reflectància $R(\lambda)$ i la transmitància $T(\lambda)$ d'un sistema de capes primes format per capes que alternen materials d'índex de refracció alt i baix. Treballarem amb dos materials habituals en tecnologia de capa prima: el diòxid de titani (TiO_2) amb índex $n_d = 2.48$ i el fluorur de magnesi MgF_2 amb índex $n_d = 1.37$ [1]. El substrat serà el vidre BK7, amb $n_d = 1.52$.

[1] n_d=n(587 nm)

4.1 Fonamentació Física

Assumint incidència normal, es pot demostrar (veieu per exemple [3, 4]) que cada capa es pot caracteritzar per la següent matriu

$$\mathrm{M} = \begin{pmatrix} \cos\left(\frac{2\pi}{\lambda} n(\lambda) d\right) & \frac{i}{n(\lambda)} \sin\left(\frac{2\pi}{\lambda} n(\lambda) d\right) \\ i n(\lambda) \sin\left(\frac{2\pi}{\lambda} n(\lambda) d\right) & \cos\left(\frac{2\pi}{\lambda} n(\lambda) d\right) \end{pmatrix}$$

on d i $n(\lambda)$ són el gruix de la làmina i l'índex de refracció respectivament, i $i = \sqrt{-1}$. El gruix de les capes es determina a partir de la longitud d'ona λ_r per la qual es dissenya la capa (la heu d'escollir vosaltres. El sistemes multicapes es dissenyen apilant parells de capes que alternen un index de refracció alt i un altre de baix. D'aquesta manera $d_1 = \dfrac{\lambda_r}{4n_1(\lambda_r)}$ i $d_2 = \dfrac{\lambda_r}{4n_2(\lambda_r)}$. Per un sistema d'N capes, la matriu de transferència total es pot calcular com

$$\mathrm{M_T} = \mathrm{M_1 M_2 \ldots M_{N-1} M_N}\,. \tag{4.1}$$

Aleshores introduïm B i C

$$\begin{pmatrix} B \\ C \end{pmatrix} = \mathrm{M_T} \begin{pmatrix} n_0 \\ n_s \end{pmatrix} \tag{4.2}$$

on $n_0 = 1$ és l'index de l'aire. Finalment, R i T es calculen a partir de

$$R(\lambda) = \left| \frac{n_0 B - C}{n_0 B + C} \right|^2 \tag{4.3}$$

$$T(\lambda) = 1 - R(\lambda) \tag{4.4}$$

4.2 Indicacions per desenvolupar el codi

1. Implementeu una funció que retorni l'índex de refracció per cada longitud d'ona pels materials considerats (TiO_2, MgF_2 i BK7). Podeu trobar les relacions de dispersió $n(\lambda)$ per a diferents materials a la pàgina `refractiveindex.info`. La opció més simple és accedir a la fórmula de l'índex fent clic on diu 'Expressions per n'.

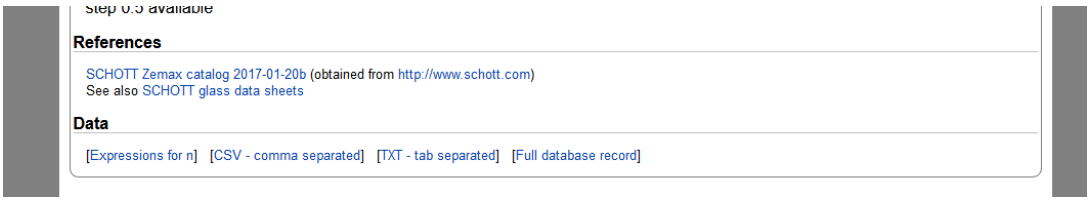


Figura 4.2: Diferents formes d'accedir a la informació de l'índex de refracció.

2. Alternativament (no es obligatori) podeu descarregar-vos els valors en format `.csv` (separats per comes) o `.txt` (separats per tabuladors). En aquest dos casos podeu accedir a les dades mitjançant l'ordre `numpy.loadtxt`. A partir d'aquestes podeu generar un polinomi de grau suficient que les interpoli amb precisió. L'ordre `numpy.polyfit` us retornarà els coeficients del polinomi que millor s'ajusta. Fent servir aquests coeficients en combinació amb `numpy.polyval` generareu la funció que descriu $n(\lambda)$ en cada cas.

3. Representeu gràficament $n(\lambda)$ per als tres materials considerats. El rang de longitud d'ona ha de cobrir el visible i l'infraroig proper (e.g. 450-1200 nm). Noteu que en aquesta pàgina les longituds d'ona es donen en μm.

4. Implementeu una funció que calculi la matriu M per a cada capa i longitud d'ona. Comproveu que funciona bé per un sistema format per dos capes (TiO_2 i MgF_2, gruix $\lambda/4$ cada una) sobre vidre BK7

5. Incrementeu el nombre de blocs de dues capes alternes de TiO_2 i MgF_2 (2, 4, 6, etc). Estudieu l'efecte sobre $R(\lambda)$ i $T(\lambda)$. Representeu-les gràficament.

6. Determineu $R(\lambda)$ i $T(\lambda)$ quan s'utilitzen MgF_2 i SiO_2. Genereu la funció $n(\lambda)$ per al diòxid de silici, repetiu els passos anteriors i mostreu $R(\lambda)$ i $T(\lambda)$. Què ha passat?

Notes d'implementació

1. Tingueu present que els operadors $*$ i $**$ no són adequats per fer operacions amb matrius. Les ordres `numpy.dot` i `numpy.linalg.matrix_power` us poden ser d'utilitat.

2. Per organitzar les dades, eviteu fer servir combinacions d'*arrays* i llistes, i l'`.append()`. Si ho feu, us pot resultar difícil d'accedir a les dades. Una forma transparent i que no genera problemes és utilitzar d'*arrays* multidimensionals. Com que coneixem a priori la quantitat de memòria que necessitarem, podem fer servir les ordres `numpy.empty` o `numpy.zeros`; per exemple, `numpy.empty([2, 2, 100], dtype = complex)` reservarà espai per 100 matrius complexes de 2x2 components.

4.3 Presentació de resultats

El codi que heu de presentar ha de generar dues figures: la primera ha de contenir $n(\lambda)$ pels quatre materials considerats i la segona ha de mostrar $R(\lambda)$ i $T(\lambda)$ pel sistemes multicapes TiO_2-MgF_2 i SiO_2-MgF_2. Incloeu comentaris al codi. Assegureu-vos que el codi s'executa correctament. Anomeneu el vostre script `nom_cognom1_cognom2.py`; assegureu-vos de no fer servir espais ni accents. Pugeu-lo al campus virtual. La data límit de presentació s'indicarà oportunament.

Exercici 5

Transformades de Fourier: Processament d'un senyal d'àudio

En aquest projecte es farà un filtrat de freqüències d'un arxiu d'àudio en format `.wav`. Aquest arxius contenen la informació de l'ona sonora en funció del temps. Com que es tracta d'un format digital, la informació no és continua sino que ha estat mostrejada. En el nostre cas particular, l'ona ha estat discretitzada a un ritme de 44100 mostres per segon (*samples*). D'altra banda, els valors de l'ona (ample de banda, *bandwidth*) tenen una profunditat de 16 bits. Aquesta és la combinació de valors del que s'anomena qualitat CD.
Python disposa del parell d'odres `scipy.io.wavfile.read` i `scipy.io.wavfile.write` per carregar i guardar aquests arxius. En particular, la primera obre l'arxiu i carrega la seva informació en un array. En cada una de les seves components s'emmagatzema una mostra de l'ona. La longitud d'aquest serà el nombre de mostres per segon per la durada. Equivalentment, la longitud del senyal en segons (L) serà el nombre de mostres de l'arxiu (N) per la durada de cada una d'elles (T), $L = NT$.
L'equalització consisteix en manipular el contingut freqüencial, amplificant més o menys una determinada zona de l'espectre audible. L'eina matemàtica que permet descomposar el contingut freqüencial d'un senyal és la Transformada de Fourier (FT).

$$G(f) = \mathrm{FT}[g(t)] = \int_{-\infty}^{\infty} g(t) \exp(-i2\pi f t) dt \tag{5.1}$$

Python disposa d'una implementació d'aquest operador mitjançant les ordres `numpy.fft.fft` i `numpy.fft.ifft`. Aquests funcions tenen com argument el senyal d'àudio $g(t)$ i retornen un vector amb el valor de la transformada. Les freqüències corresponents f s'han de calcular de forma independent i verifiquen

$$f \in \left[-\frac{1}{2T}, \frac{1}{2T}\right] = \left[-\frac{N}{2L}, \frac{N}{2L}\right] \tag{5.2}$$

on $f_c = 1/2T$ és la freqüència de tall o de Nyquist. Aquest resultat es coneix com el Teorema del Mostreig, enunciat per Shannon al 1949. Si volem tenir informació de tot l'espectre audible (de 20 a 20000 Hz), actuarem de la següent manera: prenem $L = 1$s i com que la freqüència de mostreig és de 44100 Hz, aleshores $f_c = 22050$ Hz. Per mes informació podeu consultar [5]. Noteu que la funció `numpy.fft.fft` retorna les freqüències ordenades de 0 a f_c i de $-f_c$ a 0. Per visualitzar la transformada en la ordre habitual de $-f_c$ a f_c, necessitareu fer servir la funció `numpy.fft.fftshift`.
En aquest projecte desenvoluparem diverses tasques. D'un banda analitzarem conceptes relatius a l'anàlisi en freqüència que es poden estudiar amb la transformada de Fourier. Per l'altra, aplicarem aquests conceptes a l'estudi d'uns senyal d'àudio i implementarem un filtratge passa baixes.

5.1 Transformada de Fourier d'un senyal periòdic

Genereu un senyal periòdic de freqüència 5 Hz amb l'ajut de les funcions `scipy.signal.square` i `scipy.signal.sawtooth`. Utilitzeu el teorema de Shannon per determinar l'interval de digitalització

adequat si la longitud del senyal és 1 s. Analitzeu els resultats. Noteu que $G(f)$ és una magnitud complexa. Mostreu el mòdul de la transformada (feu servir `numpy.abs`). Aquesta gràfica us dona informació del pes dels diferents harmònics en els que es descompon l'ona. Elimineu les contribucions d'alta freqüència i convertiu-les en un senyal sinusoïdal.

5.2 Creació d'un senyal d'àudio

Genereu una ona sinusoïdal de freqüència 440 Hz, digitalitzada a 44100 Hz i amb una durada de 5 s. Guardeu aquesta ona fent servir l'ordre `scipy.io.wavfile.write`. Escolteu-la. Heu generat un diapasó.

5.3 Analitzador gràfic

1. Disposeu del fitxer `cello.wav` al Campus Virtual. Preneu les dades corresponents al primer segon del fitxer on els valors siguin diferent de zero. Feu la transformada de Fourier d'aquest senyal. Fixeu-vos que la quantitat de mostres amb que treballeu es molt superior a la resolució de la pantalla. Quin problema representa això?

2. Observeu com canvia el mòdul de la transformada de Fourier amb L=1 s quan la primera mostra comença a t=0 s, 0.1 s, 0.2 s, ... Opcional: feu una animació.

5.4 Filtratge de baixes freqüències

1. Preneu el vector de dades corresponent al primer segon del fitxer que heu fet servir abans. Calculeu la seva transformada.

2. Elimineu la informació dels harmònics que superin una determinada freqüència.

3. Calculeu l'antitransformada i guardeu el resultat en format `.wav`. Mostreu en un gràfic l'ona original, la filtrada i els espectres de la transformada de Fourier abans i després d'aplicar el filtre.

4. Considereu diversos filtres de freqüència. Escolteu el fitxer original i els filtrats. Quines diferències apreciu?

Presentació de resultats

Desenvolupeu un únic script que faci el següent:

1. En una única finestra mostreu (i) el senyal original, (ii) el mòdul de la TF, (iii) el mòdul de la TF filtrada i (iv) el senyal sinusoïdal recuperat. Feu això per l'ona quadrada (Figura 1) i el dent de serra (Figura 2).

2. Figura 3, mostreu en una única finestra: (i) el senyal original en un interval molt curt per assegurar que es mostra correctament (e.g. 0.01 s), (ii) el mòdul de la TF amb L=1 s, (iii) el mòdul de la TF filtrada i (iv) el senyal filtrat.

No cal pujar l'arxiu d'àudio. Assegureu-vos però, que l'script i l'arxiu estan al mateix directori. Anomeneu el vostre script `nom_cognom1_cognom2.py`; assegureu-vos de no fer servir espais ni accents. Pugeu-lo al campus virtual. La data límit de presentació s'indicarà oportunament.

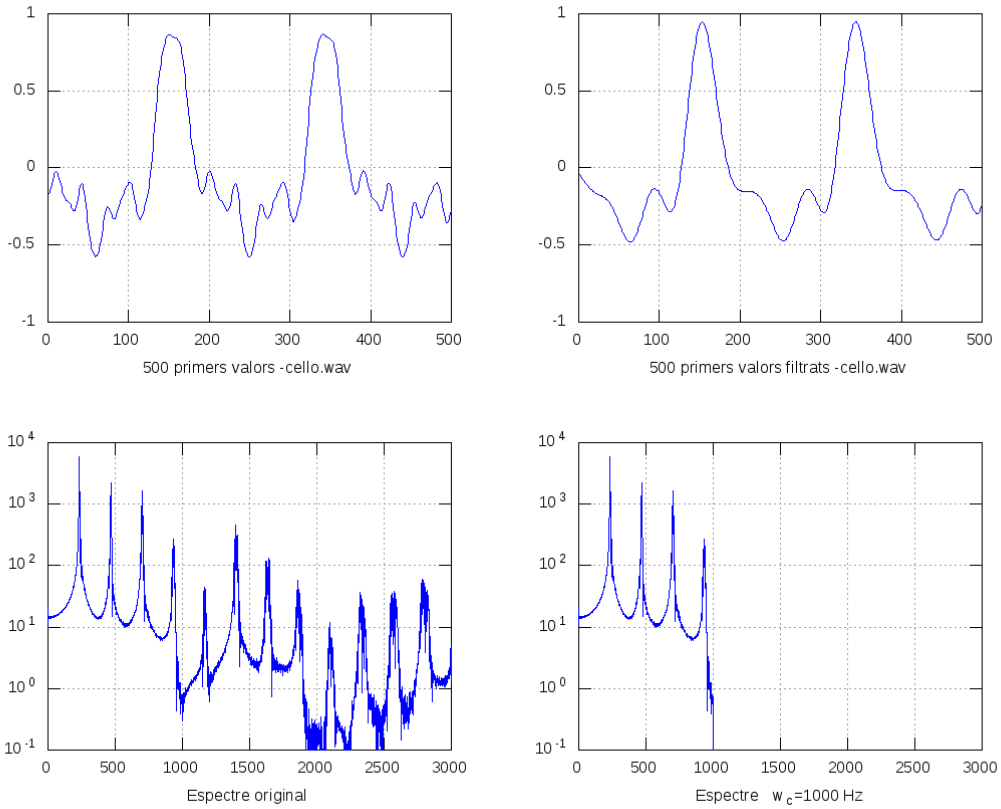


Figura 5.1: Efecte del filtratge de baixes freqüències sobre la forma de l'ona ($f_c = 1000$ Hz). Observeu la pèrdua d'harmònics

Exercici 6

Càlcul de les coordenades de color RGB a partir de l'espectre d'emissió d'un gas.

Escriviu un programa que calculi les coordenades RGB a partir de l'espectre d'emissió $E(\lambda)$. El mètode per passar de la mesura física a les coordenades RGB consisteix dels següents passos:

1. Calcular les següents integrals

$$x = \int E(\lambda)x(\lambda)\,d\lambda \tag{6.1}$$

$$y = \int E(\lambda)y(\lambda)\,d\lambda \tag{6.2}$$

$$z = \int E(\lambda)z(\lambda)\,d\lambda \tag{6.3}$$

$$\tag{6.4}$$

   on $x(\lambda)$, $y(\lambda)$ i $z(\lambda)$ són unes distribucions estàndard denominades valors triestímul (*color matching functions* en anglès) que trobareu a l'arxiu adjunt.

2. Normalitzeu x, y, i z fent

$$X = \frac{x}{x+y+z} \tag{6.5}$$

$$Y = \frac{y}{x+y+z} \tag{6.6}$$

$$Z = \frac{z}{x+y+z} \tag{6.7}$$

$$\tag{6.8}$$

   on X, Y i Z són les coordenades de color en el sistema CIE 1931 (Commission internationale de l'éclairage).

3. Passeu del sistema CIE 1931 al RGB aplicant fent el següent canvi lineal (feu servir `numpy.dot`):

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = 255. \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{6.9}$$

   Finalment els valors R G B s'arrodoneixen al sencer sense signe de 8 bits més proper.

4. Per visualitzar el color a la pantalla de l'ordinador genereu una estructura tridimensional `cl` de NxNx3 dimensions (preneu N=250) amb `cl = np.ones([250,250,3], dtype=...)` i assigneu `cl[:,:,0] = R`, `cl[:,:,1] = G` i `cl[:,:,2] = B`. Fent `plt.imshow(cl)` visualitzareu el color obtingut.

5. Es disposa d'un arxiu problema al campus virtual. Es un arxiu de text separat per espais ' ' i que consta de cinc columnes. La primera és la longitud d'ona (en nm amb valors equiespaiats cada 5 nm), les tres següents els valors de $x(\lambda)$, $y(\lambda)$ i $z(\lambda)$ i la darrera, l'espectre $E(\lambda)$. L'ordre `numpy.loadtxt` us pot resultar útil.

   - Carregueu l'arxiu en memòria. Assegureu-vos que l'script i l'arxiu estan al mateix directori. Mostreu dos (sub)gràfics en una única finestra. El primer ha de mostrar les distribucions triestímul $x(\lambda)$, $y(\lambda)$ i $z(\lambda)$ i el segon, l'espectre $E(\lambda)$. Els eixos han d'estar escalats de la següent manera: abscisses entre 380 i 650 nm; ordenades entre 0 i 2 (u.a.). Afegiu una graella i etiquetes explicatives als eixos per millorar la visualització.
   - Calculeu els valors (x, y, z), (X, Y, Z) i (R, G, B). Mostreu els resultats numèrics per la consola.
   - Visualitzeu el color en una segona finestra.
   - Considereu que $E(\lambda) = 1$. Calculeu de nou les coordenades (x, y, z); (X, Y, Z) i (R, G, B). Mostreu els resultats numèrics per la consola i el color resultant en una tercera finestra.

.

Presentació de resultats

L'script ha de generar les tres figures indicades a l'apartat anterior. Mostreu per consola els valors de (x, y, z), (X, Y, Z) i (R, G, B) per $E(\lambda)$ (obtinguda a l'arxiu) i $E(\lambda) = 1$. Anomeneu el vostre script `nom_cognom1_cognom2.py`; assegureu-vos de no fer servir espais ni accents. Pugeu-lo al campus virtual. La data límit de presentació s'indicarà oportunament.

Bibliografia

[1] Tutorial scipy.integrate `http://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html`. [Online; accedit el 30 de setembre de 2016].

[2] J. Rahe. Komet Halley `https://ui.adsabs.harvard.edu/abs/1982S%26W....21...21R/abstract`, 1982. [Online; accedit 17 de setembre de 2018].

[3] H.A. Macleod. *Thin-film optical filters*. Taylor & Francis, 2001.

[4] James C. Wyant. Multilayer fims `http://wyant.optics.arizona.edu/ThinFilms/multilayerfilms.pdf`. [Online; accedit el 22 d'octubre de 2012].

[5] Wikipedia. Sampling (signal processing) — wikipedia, the free encyclopedia, 2017. [Online; accedit 13 de setembre de 2017].