B216C Application de méthodes numériques

Labo 3 : Calcul numérique en Python avec la librairie SciPy

Le but de ce troisième labo est d'utiliser la suite SciPy afin de réaliser des calculs numériques avec le langage Python. Vous allez donc tenter de résoudre plusieurs petits exercices qui vous permettront de prendre en main plusieurs librairies de la suite Scipy, à savoir `numpy`, `scipy` et `matplotlib`.

Dans ce labo, on vous demande également de faire des recherches sur Internet et/ou dans les documentations des librairies afin de trouver les notations et fonctions adéquates à utiliser. Voici par ailleurs quelques liens vers des ressources en ligne pour vous entrainer :

— http://www.labri.fr/perso/nrougier/teaching/numpy.100/
— http://www.scipy-lectures.org/index.html

1 D'Octave à Python

Pour commencer en douceur, on va *« traduire »* une fonction Octave en Python à l'aide de la librairie Scipy. Cette fonction que l'on vous demande de comprendre puis traduire calcule la valeur du courant I_D d'un transistor MOS en fonction des tensions V_{GS} et V_{DS}, étant donné des valeurs pour les paramètres k, V_{T0} et λ. Pour rappel, ce courant dépend du mode de fonctionnement du transistor :

— En mode **bloqué** lorsque $V_{GS} \leq V_{T0}$:

$$I_D = 0$$

— En mode **linéaire** lorsque $V_{GS} > V_{T0}$ et $V_{DS} < V_{Dsat}$:

$$I_D = k \cdot V_{DS} \left(V_{GS} - V_{T0} - \frac{\lambda}{2} V_{DS} \right)$$

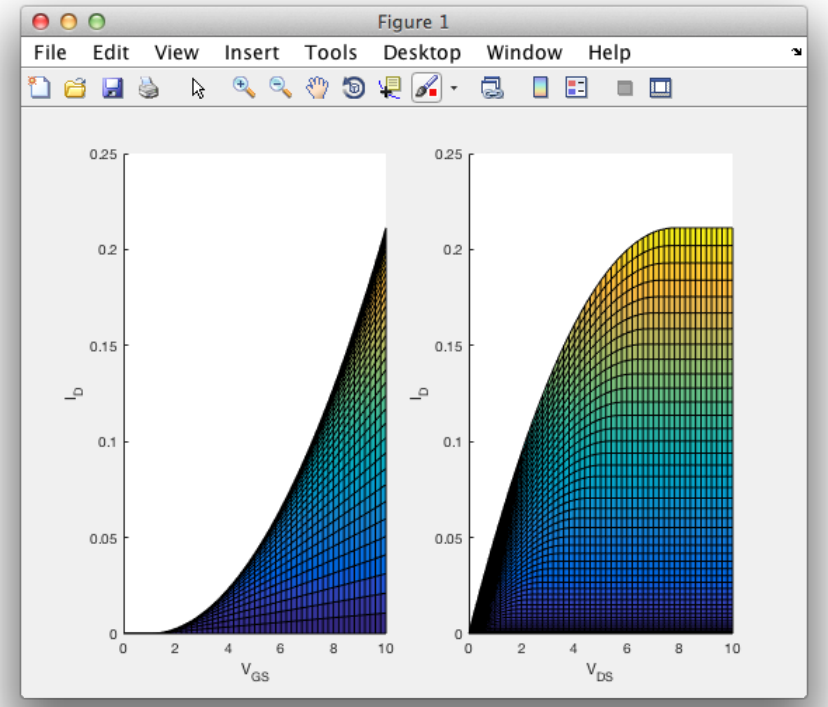
— En mode **saturé** lorsque $V_{GS} > V_{T0}$ et $V_{DS} \geq V_{Dsat}$:

$$I_D = k \cdot \frac{1}{2\lambda} \left(V_{GS} - V_{T0} \right)^2$$

Vous devez écrire une fonction `vgsvds2id` qui reçoit deux structures en paramètre : la première contient les paramètres k, V_{T0} et λ, des scalaires, et la deuxième contient les vecteurs tensions V_{GS} et V_{DS}.

```
function h = vgsvds2id(prm, V)
% Réalisation de graphiques par surface pour ID(VGS, VDS)
% IN:  prm.k    def: 6e-3 mA/V^2
%         .VT0  def: 1 V
%         .lam  def: 1.15
%      V.GS  col vector  def: (0:.1:10)'
%       .DS  row vector  def: 0:.1:10
% OUT: h.a  handle to axes 121, 122
%        s            surfaces
%
%                                          ^
%                                          |    |   /
%                                     VGS|----+---·--
%                                          |####| /·
%                                          |#ID#|/ ·
%                                          |####|  ·
%                                          |###/|  ·
%                                          |##/ |  ·
%                                          |#/  |  ·
%                                          |/   |  ·
%                                     VT0·    |  ·
%                                          |    |  ·   Vcanal
%                                          o----+--+--------->
%                                               VDS VDsat
%
```

Votre fonction doit calculer les valeurs du courant I_D pour toutes les paires possibles de V_{GS} et V_{DS} et dessiner la surface 3D que cela va représenter à l'aide de la fonction `surface`. Réalisez ensuite une figure avec deux sous-figures, chacune d'elle présentant une vue différente de la surface, tel qu'on ait l'évolution de I_D en fonction de V_{GS} à gauche et en fonction de V_{DS} à droite.



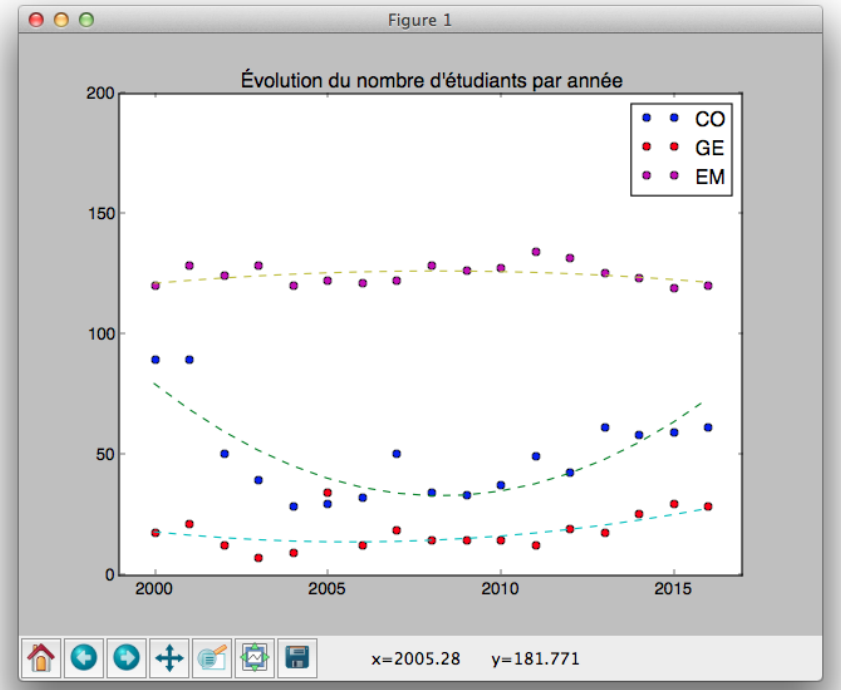
Une fois cette fonction écrite en Octave, on vous demande de la *« traduire »* en Python, à l'aide du document d'aide que vous trouverez sur Claco.

2 Dessin de graphe

Le fichier `data.txt` se trouvant sur Claco contient le nombre d'étudiants de deuxième année inscrits dans les options construction, génie électrique et électromécanique, depuis l'année 2000 à aujourd'hui [1]. Afin d'analyser les tendances, vous devez réaliser plusieurs graphes.

1. On va commencer par présenter les données de manière brute et puis de faire une régression quadratique (ajuster un polynôme du second degré) sur ces données [2]. Écrivez un code qui reproduit le graphe suivant. N'oubliez pas de soigner la présentation du graphe, c'est-à-dire ajouter un titre, des noms aux axes, un titre...

1. L'importation de ces données peut se faire facilement avec la `loadtxt` de `numpy`.
2. L'utilisation des fonctions `polyfit` et `polyval` de `numpy` vous seront sans doute utiles.



2. Calculez le nombre total d'étudiants inscrits par année, et faites la même genre analyse que celle réalisée à la question précédente. Créez donc un nouveau graphe contenant un nuage de points pour le nombre total d'étudiants. Réalisez ensuite une régression linéaire avec `linearfit` et affichez la pente de cette droite sur le graphique.
3. Sauvegardez vos deux graphiques à l'aide de la fonction `savefig` de `matplotlib.pyplot`. Vous pouvez par exemple sauvegarder vos graphiques au format PDF.

3 Manipulation d'images

On va, dans cet exercice, manipuler des images à l'aide de `numpy`. Une image peut être vue comme une matrice de nombres. On peut facilement charger une image prédéfinie à l'aide de `scipy.misc` :

```
1 from scipy import misc
2
3 face = misc.face(gray=True)
4 misc.imsave('face_orig.png', face)
```

L'instruction de ligne 3 permet de charger l'image sous forme d'une matrice, et celle de la ligne 4 permet de sauvegarder la matrice sous forme d'un fichier image. Une fois une image chargée, on peut faire toutes les modifications qu'on veut avant de la sauvegarder. Par exemple, on peut changer tous les pixels dont la valeur est supérieure à 100 en blanc avec le code suivant, dont le résultat est présenté plus bas.

```
1 c = f.copy()
2 c[c>100] = 255
3 misc.imsave('face_trans.png', c)
```

Une opération graphique assez courante que l'on fait sur les images consiste à appliquer un masque sur tous les pixels d'une image. Chaque pixel d'une image est remplacé par une somme pondérée de ses voisins. Par exemple, un masque de taille trois aura la forme suivante :

$$M = \begin{pmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{pmatrix}$$

Si on possède une image comme la suivante :

$$A = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ \vdots & \vdots & \ddots \end{pmatrix},$$

l'application du masque sur l'image produira le résultat suivant :

$$B = \begin{pmatrix} v_5p_1 + v_6p_2 + v_8p_4 + v_9p_5 & v_4p_1 + v_5p_2 + v_6p_3 + v_4p_4 + v_5p_5 + v_6p_6 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix},$$

Définissez une fonction Python `apply` qui prend en paramètre une matrice représentant une image et un matrice représentant le masque. La fonction applique le masque sur l'image et renvoie la matrice de l'image transformée.

4 Calcul symbolique

La librairie `sympy` permet de réaliser du calcul symbolique, c'est-à-dire que les opérations mathématiques qu'elle propose conservent les symboles que vous avez défini préalablement.

Exercice 1

Un magicien demande à une personne de penser à un nombre, puis de lui ajouter 3. Ensuite, il lui demande de multiplier le résultat par 2, puis lui soustraire 4, ensuite lui soustraire deux fois le nombre qu'il a choisi au départ, et d'enfin ajouter 3 au résultat obtenu.

Le magicien devine correctement que le résultat final qu'a obtenu la personne est 5. Écrivez une formule représentant les opérations faites par la personne à l'aide de `sympy` pour vérifier que la réponse sera en fait toujours 5.

Exercice 2

Dans cet exercice[3], on va définir et dériver des fonctions et expressions symboliques.

1. Définissez l'expression suivante à l'aide de `sympy` :

$$y(t, v_0, g) = v_0 t - \frac{1}{2} g t^2$$

2. Dérivez la fonction y par rapport au temps t, une première fois pour obtenir la vitesse $v(t)$ et une deuxième fois pour obtenir l'accélération $a(t)$.
3. Trouvez les racines de y, c'est-à-dire les valeurs de t pour lesquelles $y(t) = 0$. Évaluez la fonction y aux valeurs trouvées pour vérifier qu'il s'agit bien de ses racines.
4. Tracez le graphe de la fonction y depuis la première racines t_0 à la seconde racines t_{end}, avec $v_0 = 5$ et $g = 9.81$ (pour cela, vous devez convertir certains des symboles en valeurs numériques Python.
5. Définissez une fonction Python (ou modifiez celle que vous avez déjà définie) permettant de calculer la fonction f pour ensuite définir une version vectorisée de cette même fonction.

3. Source : http://folk.ntnu.no/fredf/teaching/tkt4140/exercises/pythonExercises/pexercise3/.