Bibliothèque de calcul numérique: Eigen. Langage Python.

Ce TP propose différents exercices permettant de s'initier dans un premier temps à l'utilisation de la bibliothèque de calcul matricielle Eigen en C++. Puis dans un second temps, au langage de programmation Python.

Contents

1- Eigen

Eigen est une bibliothèque de calcul OpenSource optimisée pour réaliser des opérations matricielles. Elle pourra être utilisée pour le calcul d'inversion de système linéaire, les multiplications de matrices, les inversions de matrices, les factorisations de matrices. La bibliothèque gère le cas des matrices pleines et creuses (Sparse).

1.1 Compilation d'Eigen

La démarche suivante est à suivre si vous ne disposez pas d'une installation par défaut, ou si vous souhaitez vous habituer à la compilation de programmes à partir de leurs sources.

- ▷ Téléchargez l'archive correspondant à ce TP et décompressez celle-ci. Notez qu'il existe un répertoire `eigen/` et `eigen_src/` vide. Nous allons placer les sources d'Eigen dans `eigen_src/` et l'installer dans le répertoire `eigen/`.
- ▷ Récupérez sur le site la dernière version d'Eigen.
- ▷ Placez l'archive dans le répertoire `eigen_src/` et décompressez celle-ci.
- ▷ Ouvrez une ligne de commande, et placez celle-ci dans le répertoire[1] `eigen_src/nomRepertoire/`.
- ▷ Créez un nouveau répertoire `build/` et placez la ligne de commande dans ce répertoire. L'archive d'Eigen devrait avoir la hiérarchie montrée en fig. 1, notez le répertoire `build/` vide dans lequel vous devez avoir placé votre ligne de commande.

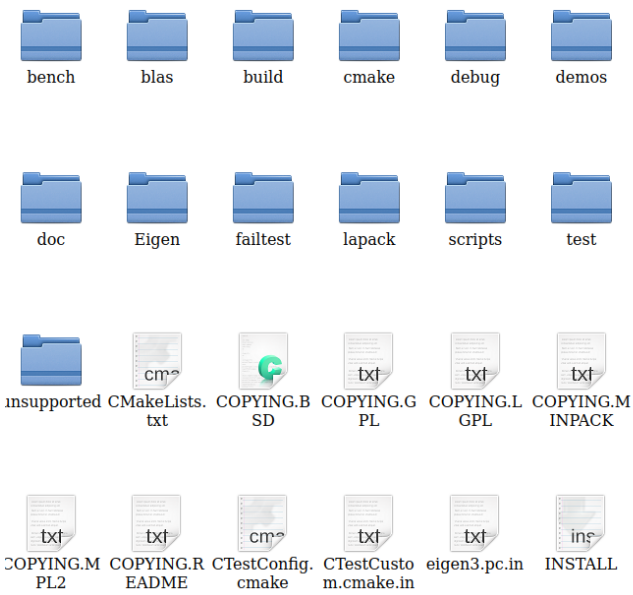

Figure 1: Repertoire des sources d'Eigen avant compilation

[1] où vous aurez remplacé *nomRepertoire* par le nom du répertoire des sources d'Eigen une fois décompressé

Soit `DIR` le chemin complet depuis la racine `/` vers le répertoire `eigen/`. Copiez ce chemin (par exemple en utilisant le navigateur de fichier et en tapant `CTRL+L` pour avoir accès aux chemin des répertoires)

Tapez ensuite dans la ligne de commande (dans le répertoire `build/`)

```
cmake -DCMAKE_INSTALL_PREFIX=DIR ..
```

Où vous aurez remplacé `DIR` par votre propre chemin (utilisez le copié collé pour y coller le nom du chemin complet). Notez également la présence des deux points à la fin de cette ligne. Cette ligne indique de lancer l'outil `CMake` que vous avez déjà utilisé sur le fichier `CMakeLists.txt` présent dans le répertoire parent (d'où la présence des `..`). Il indique également de considérer que le répertoire d'installation est `DIR` et non `usr/` par défaut sur lequel vous n'avez pas les droits d'écritures.

Après configuration, un `Makefile` doit être créé dans le répertoire courant. Lancez désormais la compilation à l'aide de la commande

```
make -j2
```

Ici le `-j2` permet de lancez la compilation sur deux processus en parallèles. Notez que dans le cas général, lorsque vous possédez un ordinateur ayant `N` coeurs, vous pouvez lancer l'option `-jN` afin d'accélérer la compilation.

Une fois la compilation terminée, lancez l'installation d'Eigen avec la commande

```
make install
```

Cette étape viens copier les fichiers d'en-tête et potentiellement de bibliothèque statique ou dynamique dans les répertoires indiqués lors de la configuration du `Makefile` par `CMake`. Si vous n'avez pas indiqué correctement le répertoire d'installation (première étape), cette étape d'installation échouera en tentant de copier des fichiers dans le répertoire `usr/` sur lequel vous n'avez pas les droits d'écriture.

Vérifiez que votre répertoire `eigen/` contient désormais un répertoire `include/eigen3/` contenant lui même des sous-répertoires avec les fichiers d'en-tête de la bibliothèques.

Notez qu'en cas de difficulté d'installation, il est préférable de supprimer tous vos répertoires temporaires (notamment le répertoire des sources d'Eigen: supprimez le puis décompressez l'archive à nouveau) et de recommencer à partir de 0.

Notez également qu'optionnellement, il est possible de recourir à l'interface graphique de paramétrage de `CMake` à l'aide de la commande

```
cmake-gui ..
```

depuis le répertoire `build/`.

1.2 Utilisation d'Eigen

▷ Compilez et exécutez le programme contenu dans le répertoire `01_test_eigen` à partir du `CMakeLists.txt` ou du `Makefile` fourni.

▷ Observez les différents appels dans le code du fichier `main.cpp`. Assurez vous que vous compreniez chaque ligne.

1.3 Approximation d'un ensemble de points par un polynome

Soit un ensemble de N points du plan (x_i, y_i) pour $i \in [0, N-1]$. Nous souhaitons approximer ces points par un polynome de la forme

$$P(x) = \sum_{k=0}^{k=d} \alpha_k \, x^k \, ,$$

où d est le degré du polynome. Un exemple de tel approximation est présenté en figure 2.

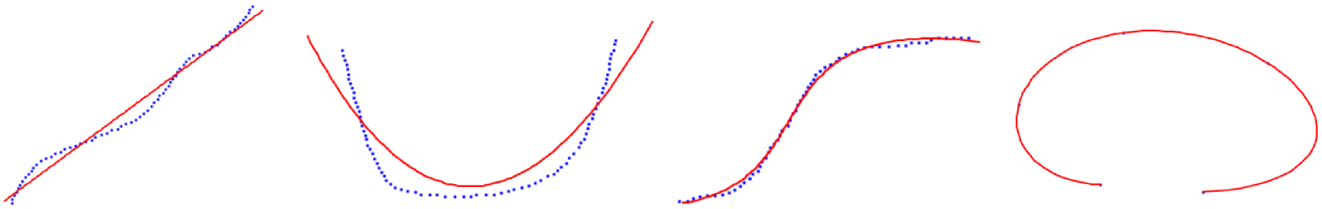

Figure 2: Exemple d'approximation (fitting) de différents ensemble de points par des polynome d'ordre 1 à 4.

Nous cherchons à approximer cet ensemble de point au sens des moindres carrés, c'est à dire que nous cherchons les coefficients α_k du polynome P qui minimise l'erreur quadratique E suivante

$$E = \sum_{i=0}^{i=N-1} \left(P(x_i) - y_i\right)^2 \, .$$

Notez qu'ici les inconnus sont les coefficients $(\alpha_k)_{k\in[0,N-1]}$ et non les (x_i, y_i) qui sont les données d'entrées.

1.3.1 Théorie

On rappel que pour minimiser une énergie E dépendant des variables $(\alpha_k)_{k\in[0,N-1]}$, on cherche les conditions sur les variables α_k permettant d'annuler l'ensemble des dérivées

$$\forall k \in [0, N-1] \, , \quad \frac{\partial E}{\partial \alpha_k} = 0 \, . \tag{1}$$

Si la fonction E est quadratique par rapport aux variables α_k, alors les conditions sur l'annulation des dérivées sont linéaires. On peut exprimer celles-ci sous la forme d'un système linéaire qu'il est possible de résoudre par des approches numériques standards.

Nous allons définir ce système linéaire.

▷ Donnez l'expression de la dérivée de $P(x_i)$ par rapport à α_k, $k \in [0, N-1]$.

▷ Montrez que la relation (1) peut se réécrire dans le cas présent sous la forme suivante

$$\forall k \in [0, N-1] \ , \ \sum_i P(x_i) x_i^k = \sum_i y_i x_i^k$$

▷ Concluez sur le fait que les $(\alpha_k)_{k\in[0,N-1]}$ optimaux vérifient la relation matricielle suivante

$$\mathrm{M}\,\boldsymbol{\alpha} = \mathbf{b} \ , \tag{2}$$

- avec $\boldsymbol{\alpha}$ le vecteur des α_k concaténés $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{N-1})$.
- $\mathbf{b}$ le vecteur de dimension N tel que

$$\forall k \in [0, N-1] \ , \ \mathbf{b}_k = \sum_{i=0}^{N-1} x_i^k \ y_i \ .$$

- M la matrice de dimension $N \times N$ telle que

$$\forall (j,k) \in [0, N-1]^2 \ , \ \mathrm{M}_{jk} = \sum_{i=0}^{N-1} x_i^{j+k} \ .$$

1.3.2 Algorithme

L'algorithme de création du polynôme approximant l'ensemble de points est donc le suivant:

```
def fitting( p =(xi,yi) ):
  Construit matrice M
  Construit vecteur b

  Resout le systeme lineaire A alpha=b
  (méthode de résolution de système linéaire de Eigen)

  return polynome(alpha)
```

1.4 Implémentation

Le répertoire `02_curve_fitting/` propose différentes classes permettant de stocker un ensemble de points (`point_set`), de gérer un polynome (`polynomial`), et fait appel à une fonction `fitting` que vous devez compléter afin de résoudre le système permettant d'approximer l'ensemble de points par le polynôme.

▷ Observez la fonction `main()`. Quelle erreur devrait-elle afficher dans le cas décrit si le fitting fonctionne correctement ?

▷ Complétez la fonction de fitting dans la fonction `fitting` en suivant les notes laissés dans les `to do`.
Aide: La résolution d'équation linéaire par Eigen est détaillé ici.
Note: Ne modifiez pas l'interface (le fichier `fitting.hpp` car votre fichier sera utilisé dans d'autres programmes.

▷ Quel est l'avantage d'utiliser une méthode de résolution de système linéaire avec le pré-calcul d'un solveur (QR, LU, etc) plutôt que de réaliser explicitement le calcul de $\mathrm{M}^{-1}\mathbf{b}$.

▷ Une fois complété, vérifiez que votre fonction gère correctement le cas décrit, et testez également le cas sur-déterminés nécessitant spécifiquement l'utilisation d'un moindre carrés.

1.5 Utilisation interactive avec Qt

Le programme du répertoire `03_curve_fitting_qt_interval` propose une interface où l'on peut directement dessiner un ensemble de points sur un `Widget` Qt.

▷ Compilez et testez ce programme.

▷ Observez la gestion de la souris et de l'affichage dans la classe `render_area`. Notez que vous devriez être en mesure de comprendre l'intégralité du code de ce programme.

▷ Copiez votre fichier `fitting.cpp` que vous avez réalisé au programme 2 précédant pour remplacer l'actuel du programme 3. Votre programme devrait compiler et afficher l'approximation de votre ensemble de points par une courbe.

▷ Modifiez le degré d'interpolation du polynôme. Observez les instabilités liés à l'utilisation de polynôme de degré élevé.

▷ La courbe est modélisée ici comme étant une fonction du type $y = f(x)$, et non une courbe paramétrique du type $(y(t), x(t))$. Quels limitations sont liées à ce choix ?

1.6 Courbe paramétrique

Ne réalisez cette partie pendant le TP que si il vous reste au minimum 2h pour la partie Python.

Le programme 4 du répertoire `04_curve_fitting_qt_parametric` propose le fitting d'une courbe paramétrique $(x(t), y(t))$. Le polynome P est alors défini par

$$P(t) = \sum_{k=0}^{k=d} \alpha_k \, t^k \ ,$$

où $\alpha_k = (\alpha_{x\,k}, \alpha_{y\,k})$ est désormais un vecteur à deux dimensions. Le polynome est donc également vectoriel est peut être décrit comme étant $P(t) = (P_x(t), P_y(t))$.

L'ensemble de points (`point_set`) contient désormais l'ensemble des positions mais également la valeur du paramètre t correspondant. Chaque point $\mathbf{p}_i(t_i) = (x_i(t_i), y_i(t_i))$ est donc stocké par $((x_i, y_i), t_i)$ ce qui correspond à la classe `vec2_parameterized`. Le paramètre t est automatiquement

calculé en fonction de la forme de la courbe en considérant une paramétrisation curviligne. C'est à dire que pour un ensemble discret de points, on considérera

$$\forall i \in [1, N-1] \ , \quad t_i = \|\mathbf{p}_i - \mathbf{p}_{i-1}\| \quad ; \quad t_0 = 0 \ .$$

- ▷ Compilez et executez le programme actuel.
- ▷ Observez la construction des coordonnées curvilignes dans la classe `point_set`. Observez également que les points qui seraient trop proches ne sont pas nécessairement ajoutés (ce qui évite certaines instabilités lors du fitting).

Le polynome P doit désormais minimiser la fonction suivante

$$\sum_{i=0}^{i=N-1} \|P(t_i) - \mathbf{p}_i(t_i)\|^2 \ .$$

Les coordonnées x et y étant indépendantes, il est possible de trouver indépendamment $P_x(t)$ et $P_y(t)$.

- ▷ Quelles fonctions doivent être minimisées par $P_x(t)$ et $P_y(t)$?
- ▷ Ecrivez le système matricielle à résoudre, similairement au cas précédent.
- ▷ Implémentez votre solution dans le fichier `fitting.cpp`.
- ▷ Observez qu'il est désormais possible de tracer des courbes paramétriques lisses dans le plan suivant la trajectoire que vous donnez à la main.

2- Python

2.1 Premier programme

▷ Créez un fichier nommé `[NOM].py`, où `[NOM]` est le nom de votre choix.

▷ Copiez le code suivant dans ce fichier à l'aide d'un éditeur de texte de votre choix (Kate, Emacs, SublimeText, etc.)

```
T=[1,4,7,6,1]
print(T)
T.append(-4)

for valeur in T:
  print(valeur)
```

▷ Lancez l'interpréteur Python sur ce fichier depuis la ligne de commande:

```
python [NOM].py
```

2.2 Affichage de courbes

Python permet l'affichage de courbes. Pour cela, il est possible d'utiliser la bibliothèque `matplotlib` ainsi que la librairie de calcul numérique `numpy`.

▷ Recopiez le code suivant et observez le résultat à l'écran (code disponible dans le répertoire `02_affichage_courbe/`).

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

N=100
t=np.linspace(0,2*np.pi,N)

plt.plot(np.cos(t),np.sin(t))
plt.plot(np.cos(t),np.sin(t),'r.')
plt.axis('equal')
plt.show()
```

Remarques

- `import numpy` permet d'importer un package d'une bibliothèque. Les fonctions du package de numpy sont disponibles à partir de l'appel `numpy.[nom_fonction]`.

- `import numpy as np` permet de donner un alias plus court pour faire appel aux fonction de numpy sous la forme `np.[nom_fonction]`.
- `plt.plot` fait appel à la fonction `plot` de la bibliothèque matplotlib. La syntaxe est proche de celle proposée par `Matlab`.
- `np.cos(t)` est un *cos* définit dans la bibliothèque matplotlib car il s'applique de manière vectorielle, terme à terme sur le vecteur t.

2.3 Image

Il est possible de lire et écrire dans les formats d'images standards en passant par la bibliothèque PIL. Différents exemples sont fournis dans le répertoire `03_images/`.

▷ Utilisez le code suivant pour lire et afficher une image `jpeg`.

```
from PIL import Image

im = Image.open("mon_image.jpg")
im.show()
```

Une fois une image chargée, il est possible de modifier d'accéder aux tableaux de couleurs de manière vectorielle en passant par numpy afin d'y appliquer des modifications quelconques.

```
from PIL import Image
import numpy as np

#Chargement de l'image
im = Image.open("mon_image.jpg")

#Convertion des donnees vers le format numpy.array
couleurs = np.array(im,dtype='float')

#Valeurs rouge,vert,bleu
r = couleurs[:,:,0]
g = couleurs[:,:,1]
b = couleurs[:,:,2]

#Taille de l'image
Nx,Ny = r.shape
print("Image de taille",Nx,"x",Ny)

#Modification des couleurs
g *= 0.9
b *= 0.6
```

```
#Parcours de l'ensemble des pixels de l'image
for kx in range(Nx):
    for ky in range(Ny):
        if pow(kx-100,2) + pow(ky-470,2) < pow(80,2):
            g[kx,ky] *= 0.6


#Construction d'un array [r,g,b]
couleurs_modifiees = np.dstack((r,g,b))

#Reconstruction du format d'image
# Necessite la conversion du type float vers uint8
# (entier non signe stocke sur 1 octet)
im_final = Image.fromarray(np.array(couleurs_modifiees,dtype='uint8'))
im_final.show()
```

▷ Assurez vous de comprendre les différentes lignes du code précédent.

OpenCV est une bibliothèque de *vision par ordinateur*. Elle permet de charger des images et vidéo et possède de nombreuses fonctions permettant d'analyser celle-ci. Notez qu'OpenCV est une bibliothèque C et C++ dont nous utilisons ici les interfaces en Python.

▷ Observez le code et lancez les différents programmes du répertoire utilisant OpenCV qui font respectivement: l'affichage d'une image, l'affichage du flux vidéo d'une webcam, la reconnaissance temps-réelle de visage sur le flux de la webcam. Notez que pour les deux derniers programme, vous avez besoin de brancher une webcam sur les PC.

2.4 Qt

Il est possible de programmer des interfaces graphiques utilisant Qt avec Python. Pour cela, il est possible d'utiliser la bibliothèque PyQt tel que dans les exemples du répertoire `04_python_qt/`.

▷ Observez les codes et lancez les programmes de ce répertoires. Assurez vous de comprendre ce qu'ils réalisent.

2.5 Lecture de fichier

Le répertoire `05_lecture_fichier/` propose un exemple de code permettant de lire un fichier de données csv (données ASCII exportable depuis un tableur) et affiche celles-ci sous forme de courbe.

▷ Observez le code et lancez le programme.

▷ Observez le fichier de données csv à l'aide d'un éditeur de texte Notez qu'il s'agit bien d'un fichier contenant des données structurées ASCII.

▷ À l'aide d'un tableur (par exemple Calc de Libre Office, ou Excel), ajoutez une seconde colonne de valeurs aux données (Notez que le délimiteur entre données est un espace).

▷ Sauvegardez vos nouvelles données dans un fichier au format csv.

▷ Modifiez le code Python pour afficher une seconde courbe correspondant aux données que vous avez ajoutés.

Remarques:

- La lecture de fichier se fait lignes par lignes par défaut sous Python.
- La méthode `split` sur une string (`str`) scinde la chaine de charactère en sous chaine. Le séparateur peut être spécifié en argument, le cas par défaut considère un séparateur sous forme d'espace.

2.6 Parcours de répertoires

Python permet de s'interfacer aisément avec le système d'exploitation et d'accéder aux commandes du bash.

▷ Testez cette fois le code suivant (disponible dans le répertoire `06_os/`)

```
import os

#read environment variable
home_directory=os.environ['HOME']
print("You home directory is",home_directory)

#list files (equivalent of os.system("ls "+home_directory))
directory_in_home=os.listdir(home_directory)
print(len(directory_in_home),"files or dir in your home path")

#Use the bash operation
os.system("mkdir -p new_directory")
print("new_directory/ is created")

#read the output of the OS call
process_txt=os.popen("ps").read()
process=process_txt.split("\n") #split every line
nbr_process=len(process)-1 #first line do not count
print("There is currently",nbr_process,"in this window")
print("The first process is \"",process[1],"\"")
```

2.7 Polynome

Soit un polynôme quelconque dont les coefficients sont quelconques (potentiellement aléatoires). On pourra considérer le polynome de degré d sous la forme

$$P(x) = \sum_{k=0}^{k=d} \alpha_k \, x^k \, ,$$

avec $\alpha_d = 1$.

On rappelle que l'on appelle matrice compagnon du polynome P la matrice C telle que

$$\mathrm{C} = \begin{pmatrix} 0 & 0 & \dots & 0 & -\alpha_0 \\ 1 & 0 & \dots & 0 & -\alpha_1 \\ 0 & 1 & \dots & 0 & -\alpha_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_{d-1} \end{pmatrix} .$$

Les valeurs propres de la matrice C sont les racines du polynôme P.

- ▷ Construisez une fonction Python qui, étant donné un polynôme donné par ses coefficients sous forme de liste de nombre renvoie sa matrice compagnon (matrice numpy).
- ▷ Affichez la position des racines du polynome dans le plan complexe. Quelles propriétées doivent vérifier les racines complexes si les coefficients du polynome sont réels?

Aide

- Un nombre complex appartient à la classe `complex`.
- Le nombre complex unitaire s'écrit `1j`.
- Étant donné un nombre complex z, il est possible d'accéder à sa partie réelle et imaginaire par l'appel à `z.real` et `z.imag`.
- Les valeurs/vecteurs propres d'une matrices peuvent être calculés par la fonction `eigenval,eigenvec=np.linalg.eig([MATRICE])`.
- Les axes x et y peuvent avoir les même échelles de longueurs suite à l'appel à `plt.axis("equal")`.
- Il est possible de limiter la taille des axes de l'affichage par l'appel à `plt.axis([x_min,x_max,y_min,y_max])`.

2.8 Analyse de système

▷ Afficher le graphique de l'histrogramme du nombre de lignes de l'ensemble de vos fichiers sources et affichez tous les fichiers qui sont trop longs.

Pour cela, vous parcourerez l'ensemble de vos fichiers de manière récursive à partir d'un répertoire de départ. Si un fichier termine par .c ou .cpp, il est considéré comme un fichier source. Comptabilisez alors son nombre de ligne et stockez ce nombre dans un tableau. Si le nombre de ligne est supérieur à 500, affichez également ce fichier avec un warning. Enfin, affichez le graphique de l'histogramme du nombre de fichier ayant un certain nombre de lignes.

Aide

- Il est possible de parcourir récursivement vos répertoires à partir d'un répetoire source à l'aide du code suivant (`[ROOT]` désigne le répertoire de départ).

```
for root,dirs,files in os.walk([ROOT]):
  for name in files: #name designe le fichier courant
    path=os.path.join(root,name) #chemin complet du fichier
```

- La commande bash `wc -l [FICHIER]` permet de connaitre le nombre de lignes d'un fichier.
- Il est possible de vérifier si un fichier débute ou termine par une chaine de caractère particulière à l'aide des méthodes `beginswidth(chaine)` et `endswidth(chaine)`.
- Il est possible d'afficher l'histogramme d'un ensemble de valeur à l'aide du code suivant

```
hist,bins=np.histogram(values)
width=0.7*(bins[1]-bins[0])
center=(bins[:-1]+bins[1:])/2
plt.bar(center,hist,align='center',width=width)
plt.show()
```