

Lab 2 : Série de Fourier et Réseau

CEG 3585 [A] – Introduction à la communication de données et au réseautage

Hiver 2023
École de science informatique et de génie électrique
Université d'Ottawa

Professeur : Mohamed Ali Ibrahim, ING., PhD.

Vendredi, 17 février, 2023
CEG 3585 [A] Lundi Groupe 7:

Patrick Loranger (300112374)
Pierre Akladios (300114467)

But et théorie du problème :

Ce laboratoire présente une relève de conception et d'implémentation de la série de Fourier. Le premier but est de trouver les valeurs de a_n et de b_n nécessaire pour bien afficher la courbe correspondante à l'une des six différentes fonctions communes dans la série de Fourier. Dans ce laboratoire, on implémente la fonction square wave, rectangular pulse train, sawtooth wave, triangular wave, half-wave rectified sine et full-wave rectified sine. Le deuxième but important pour ce laboratoire est de créer un système client / serveur qui prend l'input du client pour savoir quel graphe afficher. Le serveur de suite affiche le bon graphe dans une machine distante.

Explication de l'algorithme de notre solution :

Nous avons utilisé Python 3 pour notre solution de l'application de fonction de la série de Fourier.

Comment utiliser le code :

1. Assurer d'avoir télécharger tous les fichiers py, et les mettre dans le même répertoire.
2. Créer une instance de serveur dans le terminal de choix avec la commande suivante :
`python3 server.py`
3. Créer une instance de client dans le terminal de choix avec la commande suivante :
`python3 client.py`
4. Lire la liste de commande dans le terminal client et voir les graphiques produit par le serveur.
5. Pour voir un nouveau graphique, fermer la fenêtre avec le graphique et entrer une nouvelle commande avec le graphique désirer.

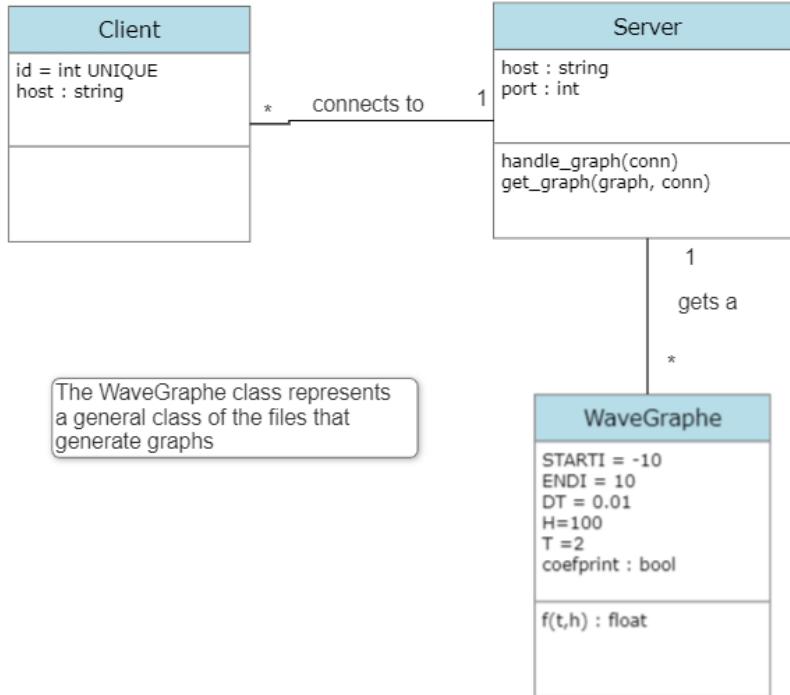
Les parties les plus importantes de cette application sont les fichiers des graphiques / fonctions communes de la série de Fourier. Nous avons 6 fichiers py différents, un pour chacune des fonctions communes. Les algorithmes varient de fichiers à fichiers pour avoir un affichage de graphique différent. Généralement, les fichiers ont la même structure. On commence avec les imports important comme matplotlib.pyplot, math et numpy. Après, il y a les déclarations de variables, STARTI, ENDI, DT, H, T et A. Nous continuons avec la fonction f qui produit le graphe et finalement une série de commande pour ouvrir un fichier avec le graphique de la fonction de la série de Fourier.

De suite, il est important de comprendre le fonctionnement du serveur. Nous utilisons des sockets pour bien suivre le protocole TCP IP. Premièrement, on fait la connexion au host 127.0.0.1 et au port 4444. On fait les étapes préliminaires pour bien établir le socket dans le serveur.

Dans le serveur, la fonction handle_graph décode le texte entré par le client et fait un appel à la fonction get_graph. Ceci fait un appel à un des 6 fichiers py pour afficher la bonne fonction.

Finalement, nous avons le code pour le fichier client. On initialise les host, port, socket de la même manière que dans le code du serveur. En se connectant étant un client, une liste de commande ce fait imprimer sur l'écran, « get » avec les types de graphes. Il y a seulement une fonction pour envoyer des messages, car le client dans cet application ne reçoit rien.

Document de conception et diagrammes UML :



Captures d'écrans de la démonstration de l'application :

```

py - Python server.py - 140x25
Last login: Sat Feb 11 21:17:58 on ttys001
patrickloranger@patrick-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LAB2/py
patrickloranger@patrick-MacBook-Pro py % python3 server.py
Server is running on host: 127.0.0.1, on port: 4444
[]

py - Python client.py - 140x25
Last login: Sat Feb 11 21:17:35 on ttys000
patrickloranger@patrick-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LAB2/py
patrickloranger@patrick-MacBook-Pro py % python3 client.py
Connected to Server running on host: 127.0.0.1, on port: 4444
#####
# Commands: <get> <graph>
# Example: get square_wave
# Graphs Types:
#   full_wave_rectified_sine
#   half_wave_rectified_sine
#   rectangular_pulse_train
#   sawtooth_wave
#   square_wave
#   triangular_wave
#####
[1]

```

Le client et le serveur sont mis en marche.

```

py -- Python server.py — 140x25
> a0: 0 -- a79 : 0 -- b79 : 0.161
> a0: 0 -- a80 : 0 -- b80 : 0.0
> a0: 0 -- a81 : 0 -- b81 : 0.157
> a0: 0 -- a82 : 0 -- b82 : 0.0
> a0: 0 -- a83 : 0 -- b83 : 0.153
> a0: 0 -- a84 : 0 -- b84 : 0.0
> a0: 0 -- a85 : 0 -- b85 : 0.15
> a0: 0 -- a86 : 0 -- b86 : 0.0
> a0: 0 -- a87 : 0 -- b87 : 0.146
> a0: 0 -- a88 : 0 -- b88 : 0.0
> a0: 0 -- a89 : 0 -- b89 : 0.143
> a0: 0 -- a90 : 0 -- b90 : 0.0
> a0: 0 -- a91 : 0 -- b91 : 0.14
> a0: 0 -- a92 : 0 -- b92 : 0.0
> a0: 0 -- a93 : 0 -- b93 : 0.137
> a0: 0 -- a94 : 0 -- b94 : 0.0
> a0: 0 -- a95 : 0 -- b95 : 0.134
> a0: 0 -- a96 : 0 -- b96 : 0.0
> a0: 0 -- a97 : 0 -- b97 : 0.131
> a0: 0 -- a98 : 0 -- b98 : 0.0
> a0: 0 -- a99 : 0 -- b99 : 0.129
> a0: 0 -- a100 : 0 -- b100 : 0.0
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.

```



```

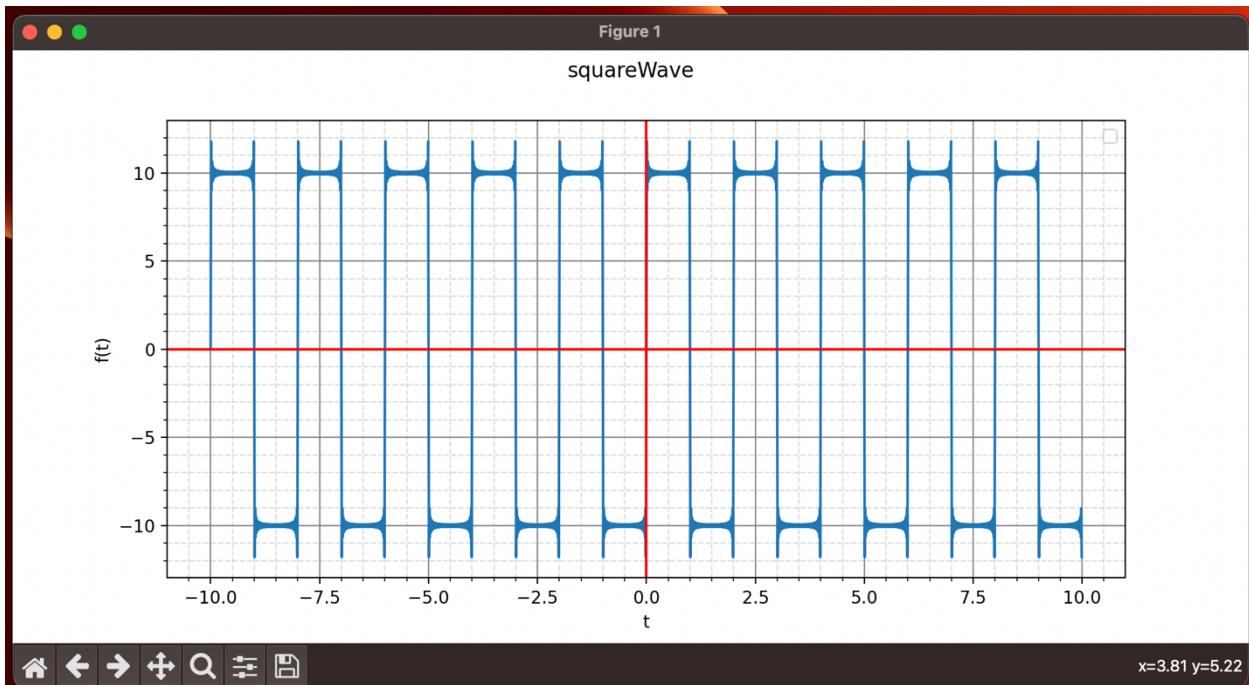
py -- Python client.py — 140x25
Last login: Sat Feb 11 21:17:35 on ttys000
patrickloranger@Patricks-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585.LABS/LAB2/py
patrickloranger@Patricks-MacBook-Pro py % python3 client.py

Connected to Server running on host: 127.0.0.1, on port: 4444
#####
# Command: <get> <graph>
# Example: get square_wave
# Graphs Types:
#   full_wave_rectified_sine
#   half_wave_rectified_sine
#   rectangular_pulse_train
#   sawtooth_wave
#   square_wave
#   triangular_wave
#####

get square_wave

```

Le client fait une demande pour voir le graphique square_wave. Le serveur affiche les 100 premières harmoniques.



Le graphique square_wave est ouvert dans une nouvelle fenêtre.

```

py -- Python server.py - 140x25
> a0: 4.712 -- a79 : 0 -- b79 : -0.04
> a0: 4.712 -- a80 : 0 -- b80 : -0.04
> a0: 4.712 -- a81 : 0 -- b81 : -0.039
> a0: 4.712 -- a82 : 0 -- b82 : -0.039
> a0: 4.712 -- a83 : 0 -- b83 : -0.038
> a0: 4.712 -- a84 : 0 -- b84 : -0.038
> a0: 4.712 -- a85 : 0 -- b85 : -0.037
> a0: 4.712 -- a86 : 0 -- b86 : -0.037
> a0: 4.712 -- a87 : 0 -- b87 : -0.037
> a0: 4.712 -- a88 : 0 -- b88 : -0.036
> a0: 4.712 -- a89 : 0 -- b89 : -0.036
> a0: 4.712 -- a90 : 0 -- b90 : -0.035
> a0: 4.712 -- a91 : 0 -- b91 : -0.035
> a0: 4.712 -- a92 : 0 -- b92 : -0.035
> a0: 4.712 -- a93 : 0 -- b93 : -0.034
> a0: 4.712 -- a94 : 0 -- b94 : -0.034
> a0: 4.712 -- a95 : 0 -- b95 : -0.034
> a0: 4.712 -- a96 : 0 -- b96 : -0.033
> a0: 4.712 -- a97 : 0 -- b97 : -0.033
> a0: 4.712 -- a98 : 0 -- b98 : -0.032
> a0: 4.712 -- a99 : 0 -- b99 : -0.032
> a0: 4.712 -- a100 : 0 -- b100 : -0.032
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.

```



```

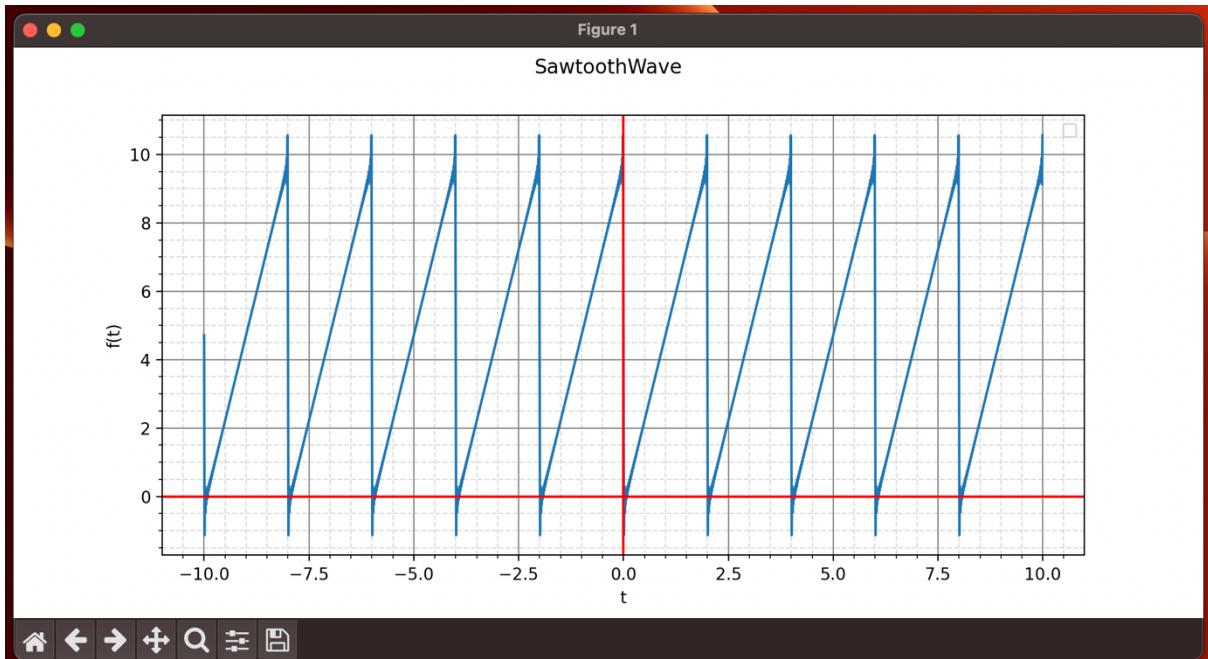
py -- Python client.py - 140x25
Last login: Sat Feb 11 21:17:35 on ttys000
patrickloranger@Patricks-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LAB2/py
patrickloranger@Patricks-MacBook-Pro py % python3 client.py

Connected to Server running on host: 127.0.0.1, on port: 4444
#####
# Command: <get> <graph> #
# Example: get square_wave #
# Graphs Types: #
#   full_wave_rectified_sine #
#   half_wave_rectified_sine #
#   rectangular_pulse_train #
#   sawtooth_wave #
#   square_wave #
#   triangular_wave #
#####

get square_wave
get sawtooth_wave

```

Le client fait une demande pour voir le graphique `sawtooth_wave`. Le serveur affiche les 100 premières harmoniques.



Le graphique `sawtooth_wave` est ouvert dans une nouvelle fenêtre.

```

py -- Python server.py — 140x25
> a0: 3.142 -- a79 : 0 -- b79 : 0
> a0: 3.142 -- a80 : -0.001 -- b80 : 0
> a0: 3.142 -- a81 : 0 -- b81 : 0
> a0: 3.142 -- a82 : -0.001 -- b82 : 0
> a0: 3.142 -- a83 : 0 -- b83 : 0
> a0: 3.142 -- a84 : -0.001 -- b84 : 0
> a0: 3.142 -- a85 : 0 -- b85 : 0
> a0: 3.142 -- a86 : -0.001 -- b86 : 0
> a0: 3.142 -- a87 : 0 -- b87 : 0
> a0: 3.142 -- a88 : -0.001 -- b88 : 0
> a0: 3.142 -- a89 : 0 -- b89 : 0
> a0: 3.142 -- a90 : -0.001 -- b90 : 0
> a0: 3.142 -- a91 : 0 -- b91 : 0
> a0: 3.142 -- a92 : -0.001 -- b92 : 0
> a0: 3.142 -- a93 : 0 -- b93 : 0
> a0: 3.142 -- a94 : -0.001 -- b94 : 0
> a0: 3.142 -- a95 : 0 -- b95 : 0
> a0: 3.142 -- a96 : -0.001 -- b96 : 0
> a0: 3.142 -- a97 : 0 -- b97 : 0
> a0: 3.142 -- a98 : -0.001 -- b98 : 0
> a0: 3.142 -- a99 : 0 -- b99 : 0
> a0: 3.142 -- a100 : -0.001 -- b100 : 0
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
]

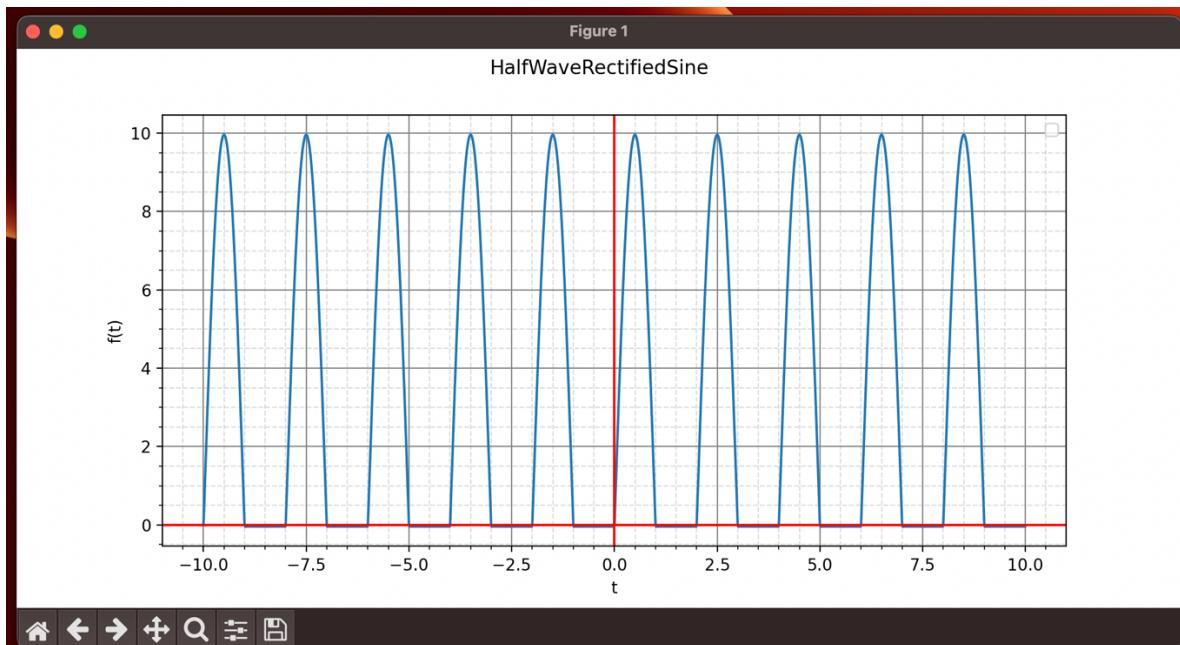
py -- Python client.py — 140x25
Last login: Sat Feb 11 21:17:35 on ttys000
patrickloranger@Patricks-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LAB2/py
patrickloranger@Patricks-MacBook-Pro py % python3 client.py

Connected to Server running on host: 127.0.0.1, on port: 4444

#####
# Command: <get> <graph>
# Example: get square_wave
# Graphs Types:
#   full_wave_rectified_sine
#   half_wave_rectified_sine
#   rectangular_pulse_train
#   sawtooth_wave
#   square_wave
#   triangular_wave
#####
get square_wave
get sawtooth_wave
get half_wave_rectified_sine

```

Le client fait une demande pour voir le graphique `half_wave_rectified_sine`. Le serveur affiche les 100 premières harmoniques.



Le graphique `half_wave_rectified_sine` est ouvert dans une nouvelle fenêtre.

```

py - Python server.py — 140x25
> a0: 6.283 -- a79 : 0 -- b79 : 0
> a0: 6.283 -- a80 : -0.002 -- b80 : 0
> a0: 6.283 -- a81 : 0 -- b81 : 0
> a0: 6.283 -- a82 : -0.002 -- b82 : 0
> a0: 6.283 -- a83 : 0 -- b83 : 0
> a0: 6.283 -- a84 : -0.002 -- b84 : 0
> a0: 6.283 -- a85 : 0 -- b85 : 0
> a0: 6.283 -- a86 : -0.002 -- b86 : 0
> a0: 6.283 -- a87 : 0 -- b87 : 0
> a0: 6.283 -- a88 : -0.002 -- b88 : 0
> a0: 6.283 -- a89 : 0 -- b89 : 0
> a0: 6.283 -- a90 : -0.002 -- b90 : 0
> a0: 6.283 -- a91 : 0 -- b91 : 0
> a0: 6.283 -- a92 : -0.002 -- b92 : 0
> a0: 6.283 -- a93 : 0 -- b93 : 0
> a0: 6.283 -- a94 : -0.001 -- b94 : 0
> a0: 6.283 -- a95 : 0 -- b95 : 0
> a0: 6.283 -- a96 : -0.001 -- b96 : 0
> a0: 6.283 -- a97 : 0 -- b97 : 0
> a0: 6.283 -- a98 : -0.001 -- b98 : 0
> a0: 6.283 -- a99 : 0 -- b99 : 0
> a0: 6.283 -- a100 : -0.001 -- b100 : 0
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.

```



```

py - Python client.py — 140x25
Last login: Sat Feb 11 21:17:35 on ttys000
patrickloranger@Patricks-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LAB2/py
patrickloranger@Patricks-MacBook-Pro ~ % python3 client.py

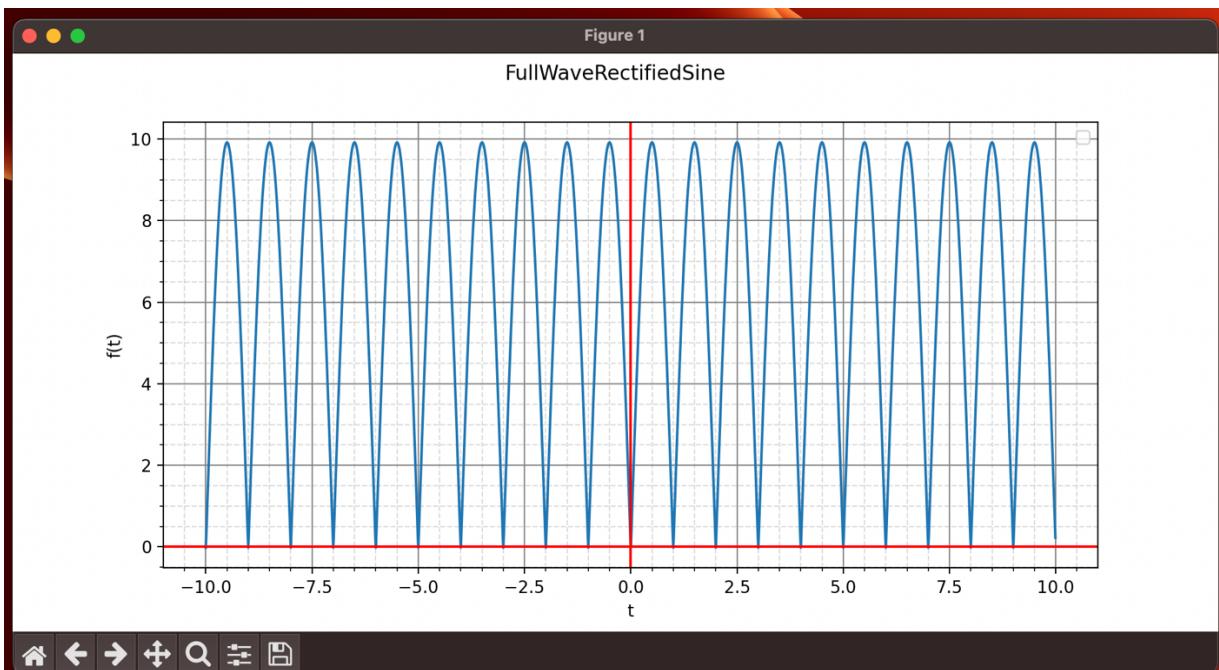
Connected to Server running on host: 127.0.0.1, on port: 4444

#####
# Command: <get> <graph>
# Example: get square_wave
# Graphs Types:
#   full_wave_rectified_sine
#   half_wave_rectified_sine
#   rectangular_pulse_train
#   sawtooth_wave
#   square_wave
#   triangular_wave
#####

get square_wave
get sawtooth_wave
get half_wave_rectified_sine
get full_wave_rectified_sine

```

Le client fait une demande pour voir le graphique `full_wave_rectified_sine`. Le serveur affiche les 100 premières harmoniques.



Le graphique `full_wave_rectified_sine` est ouvert dans une nouvelle fenêtre.

```

py -- Python server.py -- 140x25
> ab: 1.1894305308612978 -- a79 : 0.0 -- b79 : 0
> ab: 1.1894305308612978 -- a80 : 0.0 -- b80 : 0
> ab: 1.1894305308612978 -- a81 : 0.0 -- b81 : 0
> ab: 1.1894305308612978 -- a82 : 0.0 -- b82 : 0
> ab: 1.1894305308612978 -- a83 : 0.0 -- b83 : 0
> ab: 1.1894305308612978 -- a84 : 0.0 -- b84 : 0
> ab: 1.1894305308612978 -- a85 : 0.0 -- b85 : 0
> ab: 1.1894305308612978 -- a86 : 0.0 -- b86 : 0
> ab: 1.1894305308612978 -- a87 : 0.0 -- b87 : 0
> ab: 1.1894305308612978 -- a88 : 0.0 -- b88 : 0
> ab: 1.1894305308612978 -- a89 : 0.0 -- b89 : 0
> ab: 1.1894305308612978 -- a90 : 0.0 -- b90 : 0
> ab: 1.1894305308612978 -- a91 : 0.0 -- b91 : 0
> ab: 1.1894305308612978 -- a92 : 0.0 -- b92 : 0
> ab: 1.1894305308612978 -- a93 : 0.0 -- b93 : 0
> ab: 1.1894305308612978 -- a94 : 0.0 -- b94 : 0
> ab: 1.1894305308612978 -- a95 : 0.0 -- b95 : 0
> ab: 1.1894305308612978 -- a96 : 0.0 -- b96 : 0
> ab: 1.1894305308612978 -- a97 : 0.0 -- b97 : 0
> ab: 1.1894305308612978 -- a98 : 0.0 -- b98 : 0
> ab: 1.1894305308612978 -- a99 : 0.0 -- b99 : 0
> ab: 1.1894305308612978 -- a100 : 0.0 -- b100 : 0
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.

```



```

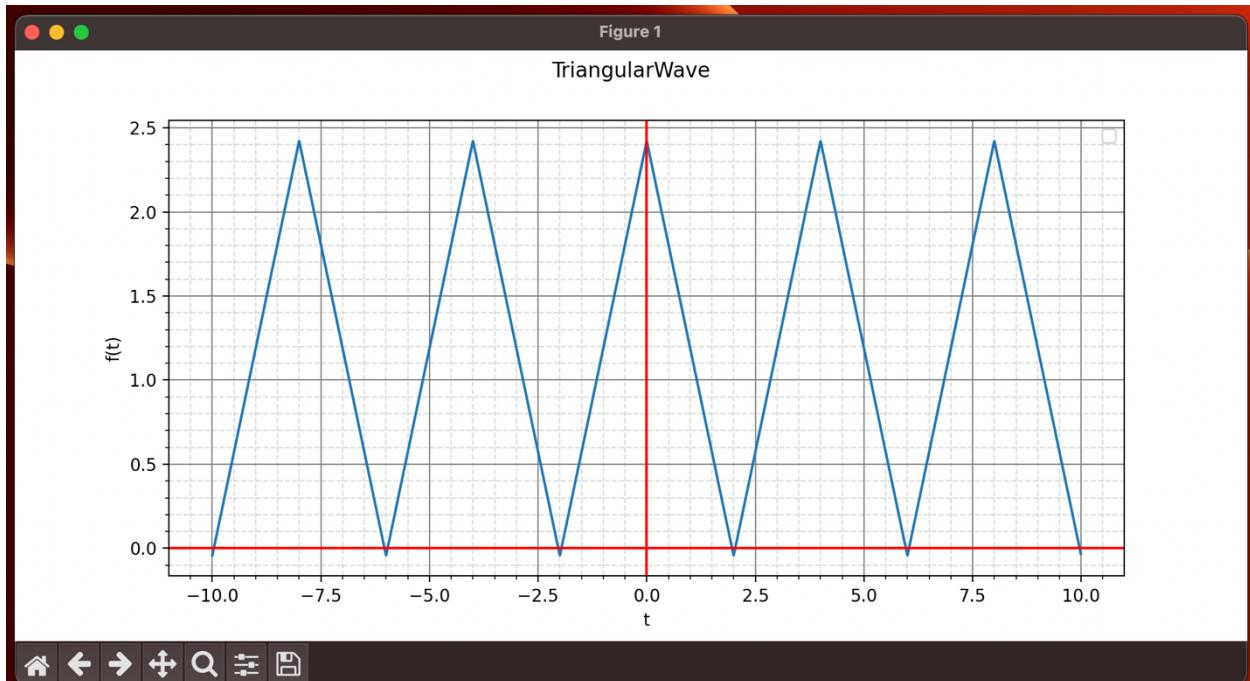
py -- Python client.py -- 140x25
Last login: Sun Feb 12 16:42:58 on ttys001
patrickloranger@patrick-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LAB2/
patrickloranger@patrick-MacBook-Pro ~ % python3 client.py

Connected to Server running on host: 127.0.0.1, on port: 4444
#####
# Command: <get> <graph>
# Example: get square_wave
# Graphs Types:
#   full_wave_rectified_sine
#   half_wave_rectified_sine
#   rectangular_pulse_train
#   sawtooth_wave
#   square_wave
#   triangular_wave
#####

get square_wave
get sawtooth_wave
get half_wave_rectified_sine
get full_wave_rectified_sine
get triangular_wave

```

Le client fait une demande pour voir le graphique triangular_wave. Le serveur affiche les 100 premières harmoniques.



Le graphique triangular_wave est ouvert dans une nouvelle fenêtre.

```

py - Python server.py - 140x25
> a0: 5 -- a79 : 0 -- b79 : 0.081
> a0: 5 -- a80 : 0 -- b80 : 0.0
> a0: 5 -- a81 : 0 -- b81 : 0.079
> a0: 5 -- a82 : 0 -- b82 : 0.0
> a0: 5 -- a83 : 0 -- b83 : 0.077
> a0: 5 -- a84 : 0 -- b84 : 0.0
> a0: 5 -- a85 : 0 -- b85 : 0.075
> a0: 5 -- a86 : 0 -- b86 : 0.0
> a0: 5 -- a87 : 0 -- b87 : 0.073
> a0: 5 -- a88 : 0 -- b88 : 0.0
> a0: 5 -- a89 : 0 -- b89 : 0.072
> a0: 5 -- a90 : 0 -- b90 : 0.0
> a0: 5 -- a91 : 0 -- b91 : 0.07
> a0: 5 -- a92 : 0 -- b92 : 0.0
> a0: 5 -- a93 : 0 -- b93 : 0.068
> a0: 5 -- a94 : 0 -- b94 : 0.0
> a0: 5 -- a95 : 0 -- b95 : 0.067
> a0: 5 -- a96 : 0 -- b96 : 0.0
> a0: 5 -- a97 : 0 -- b97 : 0.066
> a0: 5 -- a98 : 0 -- b98 : 0.0
> a0: 5 -- a99 : 0 -- b99 : 0.064
> a0: 5 -- a100 : 0 -- b100 : 0.0
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.

```



```

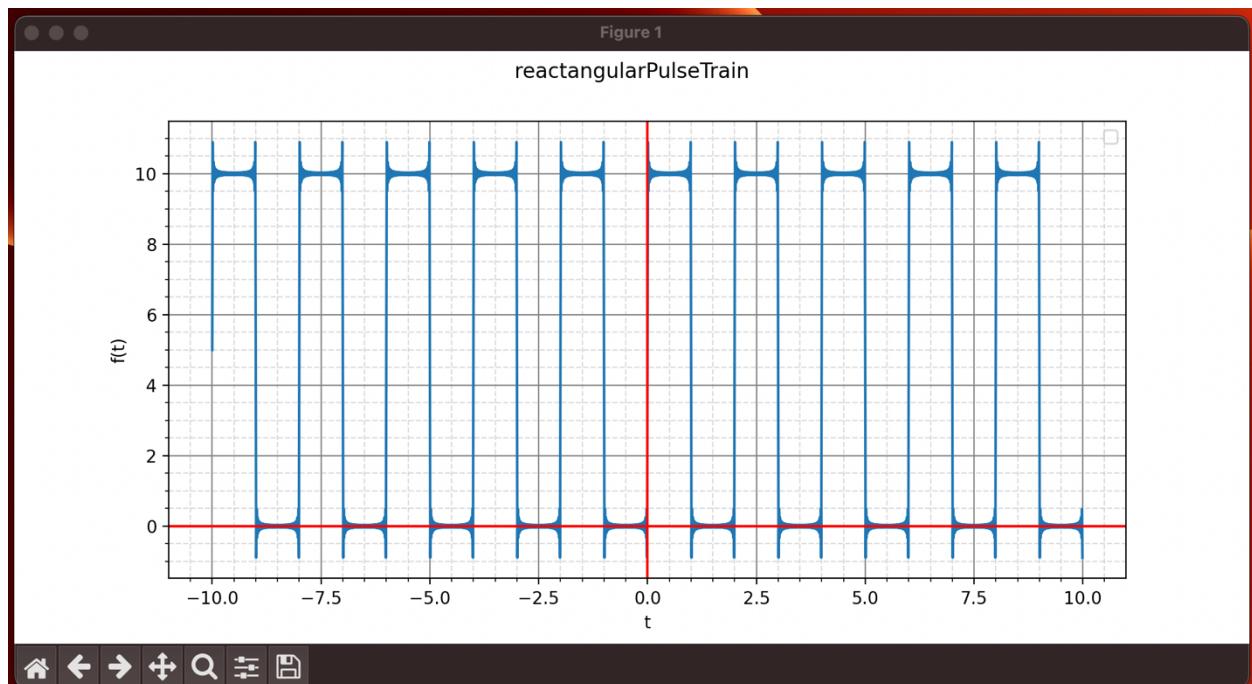
py - Python client.py - 140x25
Last login: Sun Feb 12 16:42:58 on ttys001
[patrickloranger@Patricks-MacBook-Pro ~ % cd /Users/patrickloranger/Documents/uOttawa\ SEG/7.\ Winter\ 2023/CEG\ 3585/CEG3585_LABS/LA82/py
[patrickloranger@Patricks-MacBook-Pro ~ % python3 client.py

Connected to Server running on host: 127.0.0.1, on port: 4444
#####
# Command: <get> <graph> #
# Example: get square_wave #
# Graphs Types:
#   full_wave_rectified_sine #
#   half_wave_rectified_sine #
#   rectangular_pulse_train #
#   sawtooth_wave #
#   square_wave #
#   triangular_wave #
#####

get square_wave
get sawtooth_wave
get half_wave_rectified_sine
get full_wave_rectified_sine
get triangular_wave
get rectangular_pulse_train

```

Le client fait une demande pour voir le graphique rectangular_pulse_train. Le serveur affiche les 100 premières harmoniques.



Le graphique rectangular_pulse_train est ouvert dans une nouvelle fenêtre.

Discussion :

En participant à ce laboratoire on a eu l'opportunité d'apprendre à propos des Séries de Fourrier surtout à propos des 6 fonctions communes qu'on a implémentées. Les Séries de Fourrier sont très utiliser dans le domaine des réseaux car elles peuvent être utiliser pour modéliser le mouvement des signaux. On devait trouver une manière de calculer les valeurs des équations de ces fonctions pour pouvoir générer leurs graphiques. Il fallait faire des ajustements dans la manière dans l'équation était représenter dans le code. Par exemple, l'intégrale de la fonction était représentée par une boucle qui calculait la sommation des valeurs de a_n et b_n . Ces valeurs sont les coefficients pour les fonctions cos et sin de la fonction représentant la série de Fourrier. Dans ce laboratoire, il fallait aussi, implémenter un système client-serveur qui permettrait de transmettre de l'information à propos des séries de Fourrier. Les détaille du fonctionnement de ce système sont discuter dans une section précédente du laboratoire. Ce système était inspiré par le code implémenter dans le cadre du laboratoire 1. L'implémentation du code a été faite en python à cause que ce langage offre plusieurs librairies qui sont utiles à la résolution de ce problème. Par exemple la librairie matplotlib.pyplot qui permet de générer une représentation visuel des graphes des séries qu'on a implémentés. Aussi, la librairie math qui a permis de faire plusieurs calculs par rapport aux valeurs des coefficient dans la fonction.

Conclusion :

En conclusion on a réussi à accomplir les tâches nécessaires pour implémenter le système client-serveur et de générer un graphique de 6 types de fonction communes parmi les fonctions de Fourrier. Le langage de programmation choisi était python à cause de la disponibilité des librairies et pour pouvoir réutiliser du code du laboratoire 1. Les but du laboratoire qui était d'apprendre à propos des séries de Fourrier en développent un système client-serveur et une implémentation utilisant les équations de fonctions communes ont été accomplies.