



AMILKA DANIELA LOPEZ
AGUILAR

AVANCES

ARCHIVOS

ensembleKF.py - código original que encontramos en el repositorio de Liz del filtro de Kalman y almacenamiento de los csvs y resultados.

givensMethod.py - transformar vectores o matrices en una forma especial con ceros en ciertos coeficientes.

householder.py - código de Jocelyn con el método de Householder. Reduce un sistema de ecuaciones lineales a la forma triangular superior en $n - 1$ pasos

pruebaJocelyn.py - archivo donde copié el código de ensembleKF.py, cambiando la función de potterAlg por carlsonFilter, empleando el código de carlson de Jocelyn. Se cambió también cada instancia de givens por householder.



PRUEBAJOCELYN.PY

```
for m in range(2,21): #SESSIONS TO ANALYZE
    session_start = time.time()
    numb = m
    signalName1 = os.path.join(script_dir, name + "_Pre" + str(m) + ".csv")
    signalName2 = os.path.join(script_dir, name + "_Post" + str(m) + ".csv")

    if not os.path.exists(signalName1) or not os.path.exists(signalName2):
        print(f"Skipping session {m}: One or both files are missing.")
        continue # Skip to the next iteration if files are missing
```

- Reconstrucción de los paths para guardar en la carpeta en la que se encuentre
- Saltarse sesiones que no se encuentren sin limitar el loop

```
# Plot
plt.figure(figsize=(12, 6))
plt.plot(time_axis_pre, amplitudePre_All, label='amplitudePre_All', alpha=0.7)
plt.plot(time_axis_yPre, amplitudeYPre_All, label='amplitudeYPre_All', alpha=0.7)

plt.title("Amplitude vs Time (Pre vs yPre)")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

- Un gráfico de la amplitud contra tiempo en segundos de la señal contemplando todos los sensores, antes y después del filtro de Kalman.
- Se nota que la señal tiene un comportamiento similar pero con un ligero desfase.

```
# Compute Welch PSD
freq_all, psd_all = welch(amplitudeYPre_All, fs=samplingRate, nperseg=512)
freq_wc, psd_wc = welch(amplitudeYPre_WC, fs=samplingRate, nperseg=512)
freq_nwc, psd_nwc = welch(amplitudeYPre_NWC, fs=samplingRate, nperseg=512)
freq_orig, psd_orig = welch(amplitudeOriginal, fs=samplingRate, nperseg=512)

freq_all_post, psd_all_post = welch(amplitudeYPost_All, fs=samplingRate, nperseg=512)
freq_wc_post, psd_wc_post = welch(amplitudeYPost_WC, fs=samplingRate, nperseg=512)
freq_nwc_post, psd_nwc_post = welch(amplitudeYPost_NWC, fs=samplingRate, nperseg=512)
freq_orig_post, psd_orig_post = welch(amplitudePostOriginal, fs=samplingRate, nperseg=512)

# Convert to dB
psd_all_db = 10 * np.log10(psd_all)
psd_wc_db = 10 * np.log10(psd_wc)
psd_nwc_db = 10 * np.log10(psd_nwc)
psd_orig_db = 10 * np.log10(psd_orig)

psd_all_db_post = 10 * np.log10(psd_all_post)
psd_wc_db_post = 10 * np.log10(psd_wc_post)
psd_nwc_db_post = 10 * np.log10(psd_nwc_post)
psd_orig_db_post = 10 * np.log10(psd_orig_post)
```

- SEI el método Welch es una técnica de estimación de densidad espectral que reduce el ruido y mejora la precisión de las estimaciones de densidad espectral de potencia (PSD) al promediar periodogramas calculados a partir de segmentos superpuestos de una serie temporal.
- Se utiliza scipy.signal.welch
- Se hace la conversión a dB/Hz

PRUEBAJOCELYN.PY

```
# Pre plot
plt.subplot(1, 2, 1)
plt.plot(freq_all, psd_all_db, label='yPre_All', linewidth=1.2)
plt.plot(freq_wc, psd_wc_db, label='yPre_WC', linestyle='--')
plt.plot(freq_nwc, psd_nwc_db, label='yPre_NWC', linestyle='-.')
plt.plot(freq_orig, psd_orig_db, label='Original', linestyle=':')
plt.title("Pre Session 2 Carlson x Householder")
plt.xlabel("Frequency (Hz)")
plt.ylabel("PSD (dB/Hz)")
plt.grid(True)
plt.legend()

# Post plot
plt.subplot(1, 2, 2)
plt.plot(freq_all_post, psd_all_db_post, label='yPost_All', linewidth=1.2)
plt.plot(freq_wc_post, psd_wc_db_post, label='yPost_WC', linestyle='--')
plt.plot(freq_nwc_post, psd_nwc_db_post, label='yPost_NWC', linestyle='-.')
plt.plot(freq_orig_post, psd_orig_db_post, label='Original', linestyle=':')
plt.title("Post Session 2 Carlson x Householder")
plt.xlabel("Frequency (Hz)")
plt.grid(True)
plt.legend()

output_filename = (f"CarlsonHouseholder_Session{session_number}_PreAndPost.png")
output_path = os.path.join(script_dir, output_filename)
plt.savefig(output_path)
```

- subplots de pre y post para cada una de las sesiones en una misma figura.
- Graficando después de aplicar el filtro de Kalman para las señales que usan todos los sensores, sensores ganadores y sensores no ganadores. Graficando la señal original.
- Se guarda como un png
- Se guarda el tiempo que tarda en ejecutarse para cada sesión en un txt. Se puede comparar el tiempo entre HouseholderCarlson contra GivensPotter