Some idea on P1

Jie Wang

> *A quick overview and further co-work*

Instructions: Please turn in your project typed. Handwritten reports for projects are not allowed. Each report should contain a separate cover page containing title, course info, your personal info. In the text, write in clear idiomatic English. Each report must be organized as follows: 1. Introduction explaining the problem statement and objectives. 2. Methodology and main results. 3. Discussion of the results. 4. Conclusion. 5. Appendix containing your main code and any other supporting materials. Please do not include the code in the main text of the report. You can use any computer language (Python, MATLAB, etc). You are also allowed to use existing functions and packages in these programming language as long as you provide a documentation about the functions you use (put in Appendix if used). Project Statement: Non-periodic Signals: Let xptq be a triangular pulse with duration T0. 1. Generate the pulse and plot it for variable T0 (choose any amplitude). 2. Compute the Fourier transform Xpfq and study the influence of T0 on the shape of the spectrum.

1. Report foundation

Since `Please turn in your project typed. Handwritten reports for projects are not allowed.`, I suggest that we use Overleaf for cooperation.

https://www.overleaf.com/project/6525814070294613cb62192d

It took 9 dollars per month, but I do believe it worth it, as it greatly enhance the speed of cooperation task online.

If we choose Overleaf, we can:

- Practice our latex skill
- Get elegant output easily
- Co-work seemlessly

Then we need to separate 9 / 5 = 1.8 usd / month

Co-working with Other Students Efficiently

1. Define Clear Roles and Responsibilities

When working on a group project, it's essential to assign specific roles to each member. This avoids confusion, prevents overlap, and ensures that all parts of the project are covered. For instance:

- **Research and Data Collection:** Assign someone the task of gathering relevant data or researching necessary background information.
- **Coding and Implementation:** If the project requires programming or any hands-on work, designate a member or two for this.
- **Analysis and Discussion:** Someone should be in charge of interpreting the results and leading the discussion section.
- **Proofreading and Finalization:** Ensure that at least one person (preferably more) is responsible for proofreading and finalizing the document for submission.

2. Regular Communication

- **Schedule Regular Meetings:** Whether in-person or online, regular group meetings are essential to discuss progress, address concerns, and ensure everyone is on the same page.
- **Use Collaboration Tools:** Platforms like Slack, Microsoft Teams, or even WhatsApp groups can facilitate quick communication.
- **Shared Document Platforms:** Tools like Google Docs or Overleaf (as you mentioned) are excellent for real-time collaboration.

3. Set Deadlines

- **Internal Deadlines:** Before the official deadline, set mini-deadlines for different parts of the project. This ensures that work is spaced out and not left to the last minute.
- **Review Points:** Set specific dates where the group will review the current state of the project. This allows time for revisions and refinements.

4. **Constructive Feedback**

- Encourage every member to provide and accept feedback positively. It’s crucial for refining your work and ensuring the best possible outcome.

5. **Use Version Control (if coding is involved)**

- Platforms like GitHub or Bitbucket can track changes, help merge different members' codes, and prevent any accidental overwrites.

6. **Document Everything**

- Whether it's research, meeting minutes, or code, document everything. This not only helps in referencing but also ensures transparency among group members.

7. **Respect and Team Spirit**

- Respect every member's input and appreciate their efforts. A positive environment is key to successful collaboration.

8. **Distribute Work Equally**

- Ensure that every member has an equal share of the workload. This not only ensures fairness but also promotes responsibility and accountability.

9. **Backup Regularly**

- Always keep backups of your work, especially when using shared platforms. Unexpected changes or deletions can occur, and having a backup can be a lifesaver.

10. **Plan for Contingencies**

- What if a team member falls ill or is unable to complete their part? Always have a contingency plan in place.

Incorporate these practices, and you'll find that group projects can be smooth, efficient, and rewarding experiences. Collaboration is a skill in itself, and mastering it will be beneficial in both academic and professional settings.

2. Advice on teamwork

1. **Define Clear Roles and Responsibilities**

When working on a group project, it's essential to assign specific roles to each member. This avoids confusion, prevents overlap, and ensures that all parts of the project are covered. For instance:

- **Research and Data Collection:** Assign someone the task of gathering relevant data or researching necessary background information.
- **Coding and Implementation:** If the project requires programming or any hands-on work, designate a member or two for this.
- **Analysis and Discussion:** Someone should be in charge of interpreting the results and leading the discussion section.
- **Proofreading and Finalization:** Ensure that at least one person (preferably more) is responsible for proofreading and finalizing the document for submission.

2. **Regular Communication**

- **Schedule Regular Meetings:** Whether in-person or online, regular group meetings are essential to discuss progress, address concerns, and ensure everyone is on the same page.
- **Use Collaboration Tools:** Platforms like Slack, Microsoft Teams, or even WhatsApp groups can facilitate quick communication.
- **Shared Document Platforms:** Tools like Google Docs or Overleaf (as you mentioned) are excellent for real-time collaboration.

3. **Set Deadlines**

- **Internal Deadlines:** Before the official deadline, set mini-deadlines for different parts of the project. This ensures that work is spaced out and not left to the last minute.
- **Review Points:** Set specific dates where the group will review the current state of the project. This allows time for revisions and refinements.

4. **Constructive Feedback**

- Encourage every member to provide and accept feedback positively. It’s crucial for refining your work and ensuring the best possible outcome.

5. **Use Version Control (if coding is involved)**

- Platforms like GitHub or Bitbucket can track changes, help merge different members' codes, and prevent any accidental overwrites.

6. **Document Everything**

- Whether it's research, meeting minutes, or code, document everything. This not only helps in referencing but also ensures transparency among group members.

7. **Respect and Team Spirit**

- Respect every member's input and appreciate their efforts. A positive environment is key to successful collaboration.

8. **Distribute Work Equally**

- Ensure that every member has an equal share of the workload. This not only ensures fairness but also promotes responsibility and accountability.

9. **Backup Regularly**

- Always keep backups of your work, especially when using shared platforms. Unexpected changes or deletions can occur, and having a backup can be a lifesaver.

10. **Plan for Contingencies**

- What if a team member falls ill or is unable to complete their part? Always have a contingency plan in place.

Incorporate these practices, and you'll find that group projects can be smooth, efficient, and rewarding experiences. Collaboration is a skill in itself, and mastering it will be beneficial in both academic and professional settings.

GPT report template

1. Introduction

Problem Statement and Objectives:

This project focuses on the analysis of non-periodic signals, specifically a triangular pulse (x(t)) with variable duration (T_0). The primary objectives of the project are:

1. Generation and visualization of the triangular pulse for different values of (T_0).
2. Computation and analysis of the Fourier Transform (X(f)) of the pulse.
3. Studying the impact of varying (T_0) on the spectral shape of (X(f)).

2. Methodology and Main Results

Triangular Pulse Generation:

A triangular pulse of duration (T_0) can be mathematically defined as a piecewise function. Using Python, we can employ the `numpy` library to generate and plot the pulse for a given amplitude and (T_0).

Fourier Transform Calculation:

Using the `scipy` library in Python, the Fourier Transform (X(f)) of the generated triangular pulse is computed. By varying (T_0), the shape of the spectrum is observed.

3. Discussion of the Results

Upon computation, it's anticipated that the triangular pulse will display a sinc-squared shape in its Fourier spectrum. Additionally, as (T_0) increases, the width of the main lobe in the frequency domain will decrease and vice versa. The width of the main lobe is inversely proportional to (T_0). Thus, as we vary (T_0), the frequency components spread changes, which could be pivotal in signal processing and communication applications.

4. Conclusion

This project successfully demonstrates the relationship between a time-domain triangular pulse and its frequency-domain representation. The impact of the pulse duration (T_0) on the Fourier spectrum showcases the fundamental principles of time-frequency duality.

5. Appendix

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft

def triangular_pulse(t, T0, amplitude=1.0):
    return np.where((t >= -T0/2) & (t <= T0/2), amplitude * (1 -
(2/T0)*np.abs(t)), 0)

def plot_pulse_and_spectrum(T0, amplitude=1.0):
    t = np.linspace(-2*T0, 2*T0, 4000)
    f = np.fft.fftfreq(t.shape[-1])

    x_t = triangular_pulse(t, T0, amplitude)
    X_f = fft(x_t)

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.plot(t, x_t)
    plt.title('Triangular Pulse')
    plt.xlabel('Time')
    plt.ylabel('Amplitude')

    plt.subplot(1, 2, 2)
    plt.plot(f, np.abs(X_f))
    plt.title('Fourier Transform')
    plt.xlabel('Frequency')
    plt.ylabel('Magnitude')

    plt.tight_layout()
```

```
    plt.show()

# Sample Execution
plot_pulse_and_spectrum(2)
plot_pulse_and_spectrum(4)
```

Used Functions Documentation:

- `numpy.linspace()`: Generates evenly spaced numbers over a specified interval.
- `numpy.where()`: Returns elements based on conditional expression.
- `scipy.fft.fft()`: Computes the one-dimensional n-point discrete Fourier Transform.
- `numpy.fft.fftfreq()`: Returns the discrete Fourier Transform sample frequencies.

Note: This is a generalized report. For more detailed results and discussions, actual code execution and observations should be documented.