Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Kierunek: Informatyczne Systemy Automatyki

Techniki regulacji

Michał Wróblewski
272488

Termin zajęć:
wtorek 15:15 - 17:00

29 kwietnia 2024

Spis treści

Spis rysunków

1 Wprowadzenie

Celem tego sprawozdania jest analiza stabilności systemu dynamicznego oraz badanie wpływu parametru k na jego zachowanie.

1.1 Zalety wykorzystania kryteriów do oceny stabilności systemu

Kiedy przygotowujemy się do analizy stabilności systemu, korzystanie z kryteriów, takich jak kryterium Nyquista czy kryterium Michajłowa, może mieć kilka znaczących zalet nad wyliczaniem biegunów.

Po pierwsze, kryteria te pozwalają na ocenę stabilności systemu w oparciu o jego charakterystykę w dziedzinie częstotliwości. Badając zachowanie systemu dla różnych częstotliwości, możemy lepiej zrozumieć jego reakcję na różne sygnały wejściowe oraz potencjalne problemy stabilności.

Po drugie, korzystanie z kryteriów jest często bardziej efektywne i szybsze niż wyliczanie wszystkich biegunów systemu.

Kryteria te pozwalają nam szybko ocenić stabilność systemu, co jest szczególnie przydatne w przypadku analizy wielu różnych systemów lub w warunkach ograniczonego czasu.

Korzystanie z kryteriów do oceny stabilności systemu może być bardziej wydajne, praktyczne i pomocne w identyfikacji problemów stabilności w porównaniu do wyliczania biegunów jako wstępnego kroku analizy.

2 Analiza stabilności

2.1 Układ otwarty

Rozpoczęto od wyznaczenia analitycznej funkcji $M(j\omega)$ dla układu otwartego. Następnie sporządzono jej wykres jako wykres zmiany argumentu funkcji.

2.1.1 Funkcja $M(j\omega)$

Analitycznie funkcja $M(j\omega)$ została wyznaczona na podstawie układu otwartego. Wyniki zostały przedstawione na wykresie 1.

2.1.2 Kryterium Mikhajłowa

Zgodnie z kryterium Mikhajłowa, oceniono stabilność systemu na podstawie analizy zmian kąta.

Warto zauważyć, że kąt nie zmierza do 2π, co istotnie wpływa na charakterystykę stabilności systemu.

Dodatkowo, obserwacja, że linia wykresu przechodzi przez cztery ćwiartki, dodaje pewności co do stabilności tego systemu.

2.1.3 Kod w python

```
omega = np.linspace(0, 25, 100)
Re = M(omega * 1j).real
Im = M(omega * 1j).imag
angle = np.angle(M(omega * 1j), deg=False)
```

```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

ax1.axhline(linewidth=1, color='black')
ax1.axvline(linewidth=1, color='black')
ax1.plot(Re, Im, label='M(s)')
ax1.set_xlabel('Real Part')
ax1.set_ylabel('Imaginary Part')
ax1.set_title('Real vs Imaginary Parts')
ax1.grid(True)

omega_points = [0, 2*np.sqrt(33)/11, 3.1, 10.28]
for omega_point in omega_points:
    M_point = M(omega_point * 1j)
    ax1.scatter(M_point.real, M_point.imag, color='red', label=f' =
    ↪  {omega_point}', zorder=5)

ax1.legend()
ax1.set_xlim([-500, 500])
ax1.set_ylim([-500, 500])

ax2.axhline(linewidth=1, color='black')
ax2.axvline(linewidth=1, color='black')
ax2.plot(omega, angle, label='Angle of M(s)', color='green')
ax2.axhline(2*np.pi, linestyle='--', color='red', label='$2\pi$')
ax2.set_xlabel('')
ax2.set_ylabel('Angle (radians)')
ax2.set_title('Angle of M( * j)')
ax2.grid(True)

for omega_point in omega_points:
    M_point = M(omega_point * 1j)
    angle_point = np.angle(M_point, deg=False)
    ax2.scatter(omega_point, angle_point, color='red', label=f' =
    ↪  {omega_point}', zorder=5)

ax2.legend()
plt.tight_layout()
plt.show()
```

2.2 Układ zamknięty

Dla układu zamkniętego wyznaczono analitycznie funkcję $K_o(j\omega)$. Następnie sporządzono jej wykres oraz wykres zmiany argumentu funkcji $1 + K_o(j\omega)$.

2.2.1 Funkcja $K_o(j\omega)$

Analitycznie funkcja $K_o(j\omega)$ została wyznaczona na podstawie układu zamkniętego. Wyniki zostały przedstawione na wykresie 3.

2.2.2 Wykres zmiany argumentu funkcji $1 + K_o(j\omega)$

Wykres zmiany argumentu funkcji $1 + K_o(j\omega)$ został przedstawiony na wykresie 4 oraz 5.

2.2.3 Kryterium Nyquista

Na wykresach obserwujemy, że krzywa Nyquista w PART 1 nie przechodzi przez punkt (-1,0), co sugeruje, że system jest stabilny. Z punktu widzenia kryterium Nyquista, brak obejścia tego punktu przez krzywą oznacza, że nie ma pierwiastków transmitancji na prawo od osi rzeczywistej.

Natomiast po dodaniu +1 do funkcji transmitancji i analizie w PART 2, widzimy, że krzywa Nyquista nie przechodzi przez punkt (0,0). Mimo że dodanie tego elementu zmienia kształt charakterystyki, brak obejścia punktu (0,0) potwierdza, że system pozostaje stabilny.

2.2.4 Uwaga

Jeśli jednak krzywa Nyquista obejmuje punkt (-1,0), może to wskazywać na obecność pierwiastków na prawo od osi rzeczywistej, co może prowadzić do niestabilności systemu.

2.2.5 Kod w python - sposob liczenia symulacyjny

```
num_closed = [15, 165, 660, 1140, 720]
den_closed = [1, 22, 209, 1120, 3719, 7909, 10660, 8436, 3204]
s1_closed = signal.TransferFunction(num_closed, den_closed)

# PART 1
omega_closed, H_closed = signal.freqresp(s1_closed)
angle_closed = np.angle(H_closed, deg=False)

# PART 2
H_closed_plus = H_closed + 1
angle_closed_plus_1 = np.angle(H_closed_plus, deg=False)

fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# Charakterystyka Nyquista - PART 1
axs[0, 0].axhline(linewidth=1, color='black')
axs[0, 0].axvline(linewidth=1, color='black')
axs[0, 0].plot(H_closed.real, H_closed.imag, "b")
axs[0, 0].set_title('Charakterystyka Nyquista - PART 1')

# Wykres zmiany argumentu funkcji zamkniętej - PART 1
axs[1, 0].axhline(linewidth=1, color='black')
axs[1, 0].axvline(linewidth=1, color='black')
axs[1, 0].plot(omega_closed, angle_closed, label='Angle of H Closed(s)',
↪  color='green')
axs[1, 0].set_xlabel('')
axs[1, 0].set_ylabel('Angle (radians)')
axs[1, 0].set_title('Angle of K(j) - PART 1')
axs[1, 0].grid(True)
```

```
axs[1, 0].legend()

# Charakterystyka Nyquista - PART 2
axs[0, 1].axhline(linewidth=1, color='black')
axs[0, 1].axvline(linewidth=1, color='black')
axs[0, 1].plot(H_closed_plus.real, H_closed_plus.imag, "b")
axs[0, 1].set_title('Charakterystyka Nyquista - PART 2')

# Wykres zmiany argumentu funkcji zamkniętej - PART 2
axs[1, 1].axhline(linewidth=1, color='black')
axs[1, 1].axvline(linewidth=1, color='black')
axs[1, 1].plot(omega_closed, angle_closed_plus_1, label='Angle of H
↪  Open(s) + 1', color='red')
axs[1, 1].set_xlabel('')
axs[1, 1].set_ylabel('Angle (radians)')
axs[1, 1].set_title('Angle of K(j) + 1 - PART 2')
axs[1, 1].grid(True)
axs[1, 1].legend()

plt.tight_layout()
plt.show()
```

2.2.6 Kod w python - sposob liczenia analityczny

```
k = 15
w = np.linspace(0, 100, 1000)

Re = k*(w**4 - 44*w**2 + 48 + k) / ((w**4 - 44*w**2 + 48 + k)**2 +
↪  (-11*w**3 + 76*w)**2)
Im = -k*(-11*w**3 + 76*w) / ((w**4 - 44*w**2 + 48 + k)**2 + (-11*w**3 +
↪  76*w)**2)

plt.axhline(linewidth=1, color='black')
plt.axvline(linewidth=1, color='black')
plt.plot(Re + 1, Im, "b")
plt.plot(0, 0, 'ro')
plt.xlabel('Im')
plt.ylabel('Re')
plt.title('Charakterystyka Nyquista')
plt.legend()
plt.grid(True)
plt.show()
```

3 Badanie wpływu parametru k

Następnie przeprowadzono symulację wpływu parametru k na odpowiedź skokową. Wyniki zostały przedstawione na wykresie 2.

3.1 Wartości k dla niestabilności

Przeprowadzając analizę, określono wartości parametru k, dla których system przestaje być stabilny.

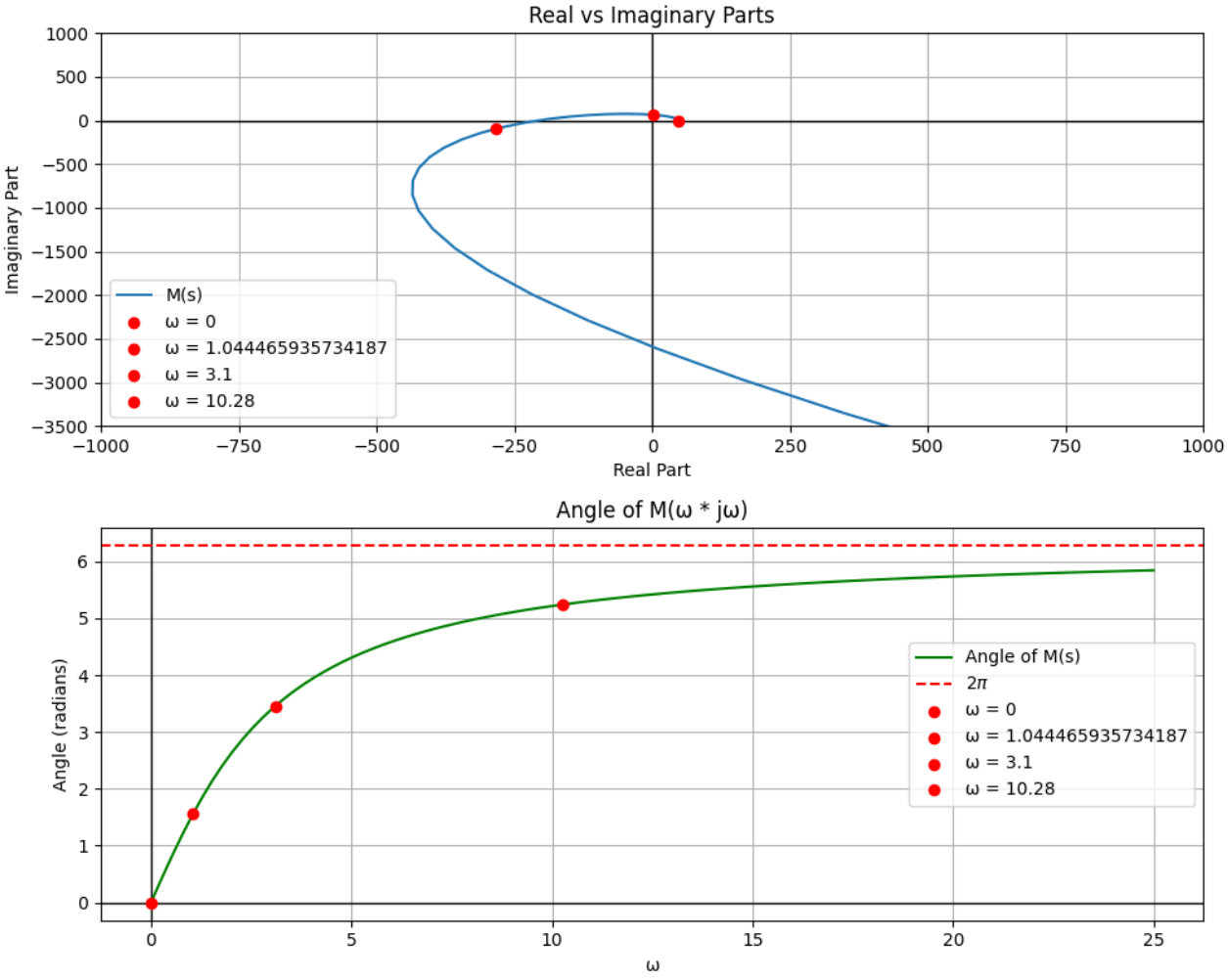
W przypadku układu otwartego zaobserwowano, że system zachowuje stabilność niezależnie od wartości k.

Natomiast dla układu zamkniętego stwierdzono, że traci stabilność, gdy k osiągnie wartość równą 208.37.

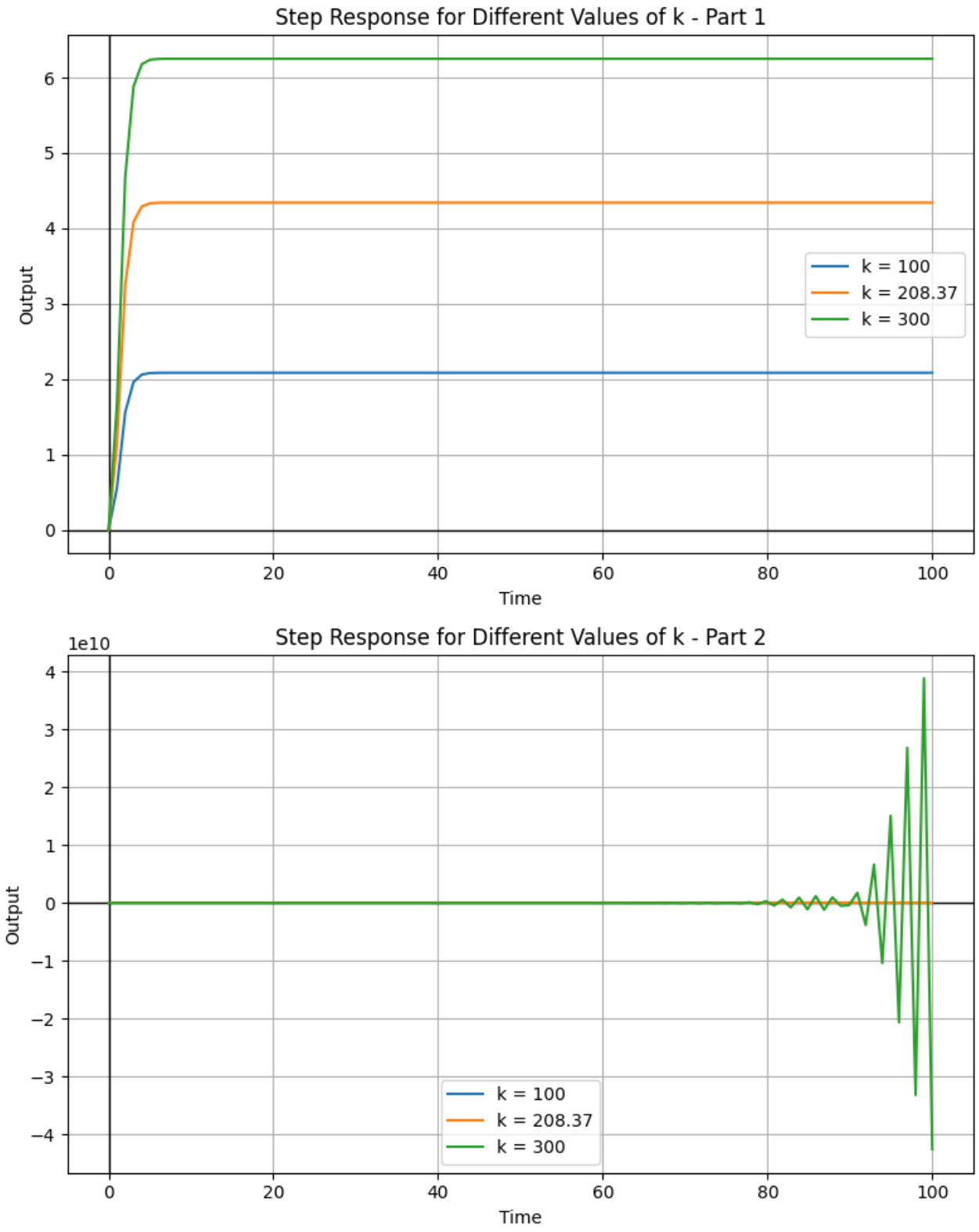
3.2 Kod w python

```
# Step Response - PART 1
plt.figure(figsize=(8, 10))
plt.subplot(2, 1, 1)
plt.axhline(linewidth=1, color='black')
plt.axvline(linewidth=1, color='black')
for k in k_values:
    lti = signal.lti([k], [1, 11, 44, 76, 48])
    t1 = np.linspace(0, 100, 100)
    t, y = signal.step(lti, T=t1)
    plt.plot(t, y, label=f'k = {k}')
    poles = lti.poles
    if np.any(np.real(poles) > 0):
        print(f"For k = {k}, the system is unstable.")
    else:
        print(f"For k = {k}, the system is stable.")

plt.title('Step Response for Different Values of k - Part 1')
plt.xlabel('Time')
plt.ylabel('Output')
plt.grid(True)
plt.legend()

# Step Response - PART 2
plt.subplot(2, 1, 2)
plt.axhline(linewidth=1, color='black')
plt.axvline(linewidth=1, color='black')
for k in k_values:
    lti = signal.lti([1*k, 11*k, 44*k, 76*k, 48*k], [1, 22, 209, 1120, k +
    ↪  3704, 11*k + 7744, 44*k+10000, 76*k+7296, 48*k+2304])
    t1 = np.linspace(0, 100, 100)
    t, y = signal.step(lti, T=t1)

    plt.plot(t, y, label=f'k = {k}')

    poles = lti.poles
    if np.any(np.real(poles) > 0):
        print(f"For k = {k}, the system is unstable.")
    else:
```

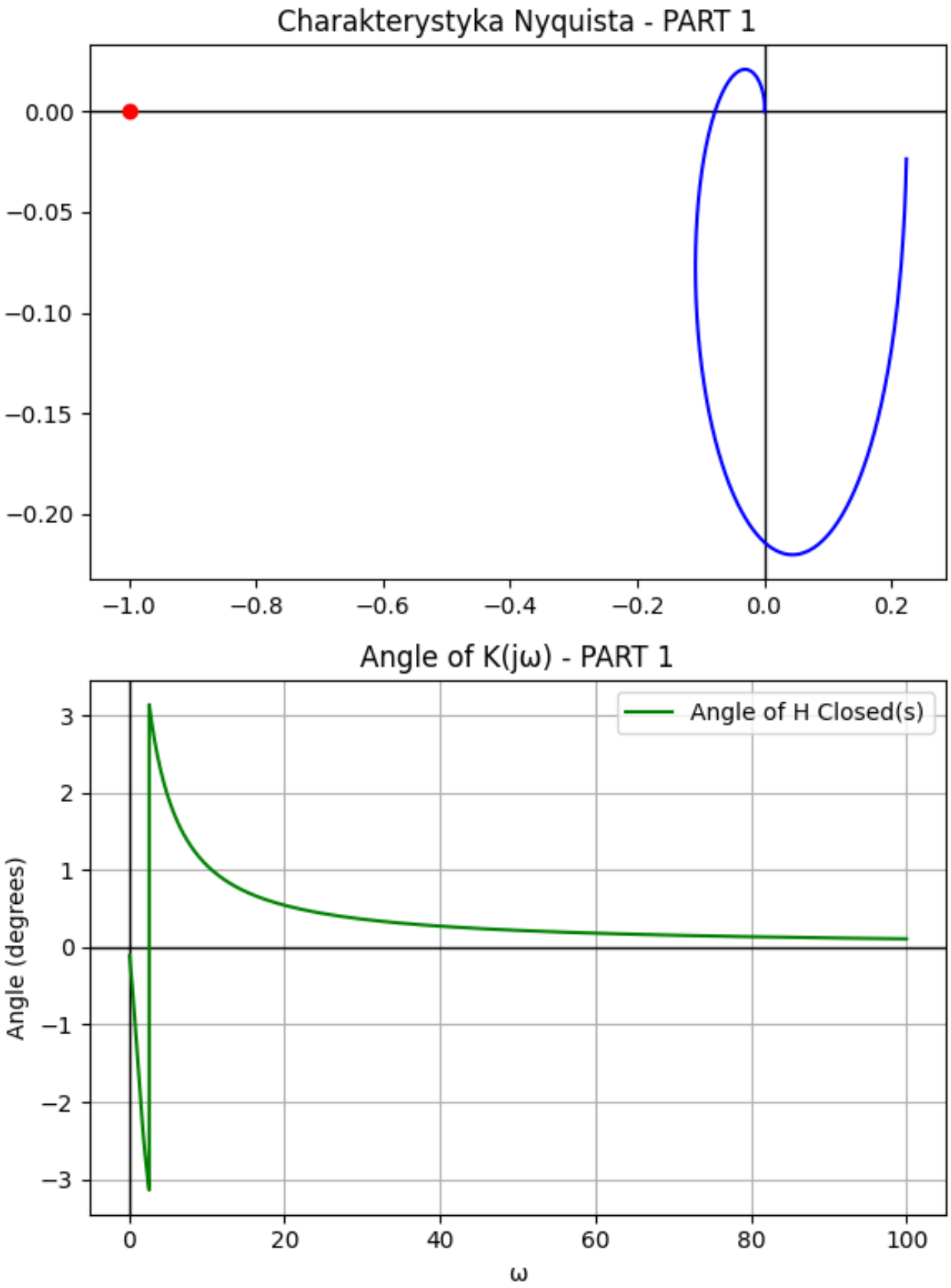
```
        print(f"For k = {k}, the system is stable.")

plt.title('Step Response for Different Values of k - Part 2')
plt.xlabel('Time')
plt.ylabel('Output')
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```



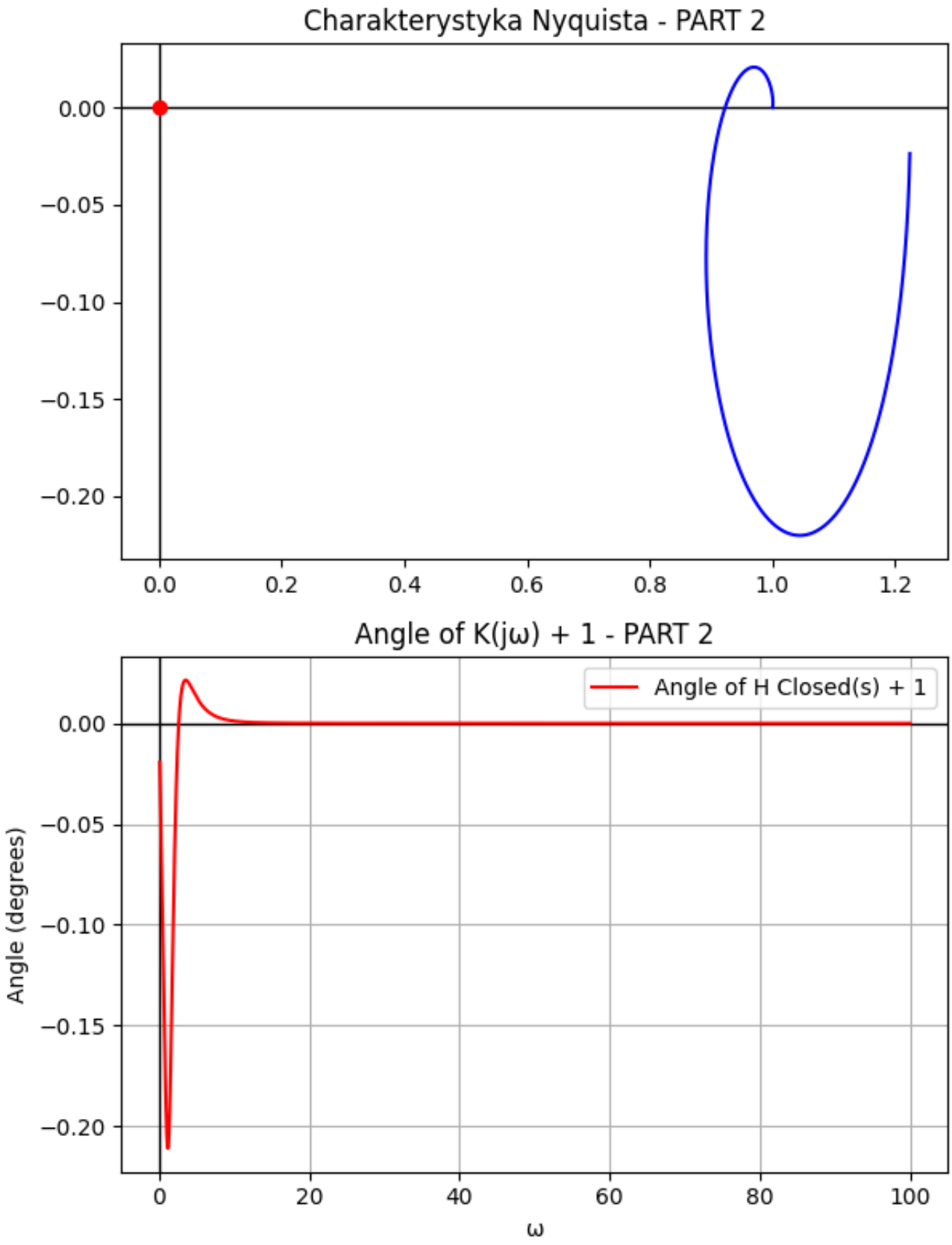
Rysunek 1: Wykres funkcji $M(j\omega)$



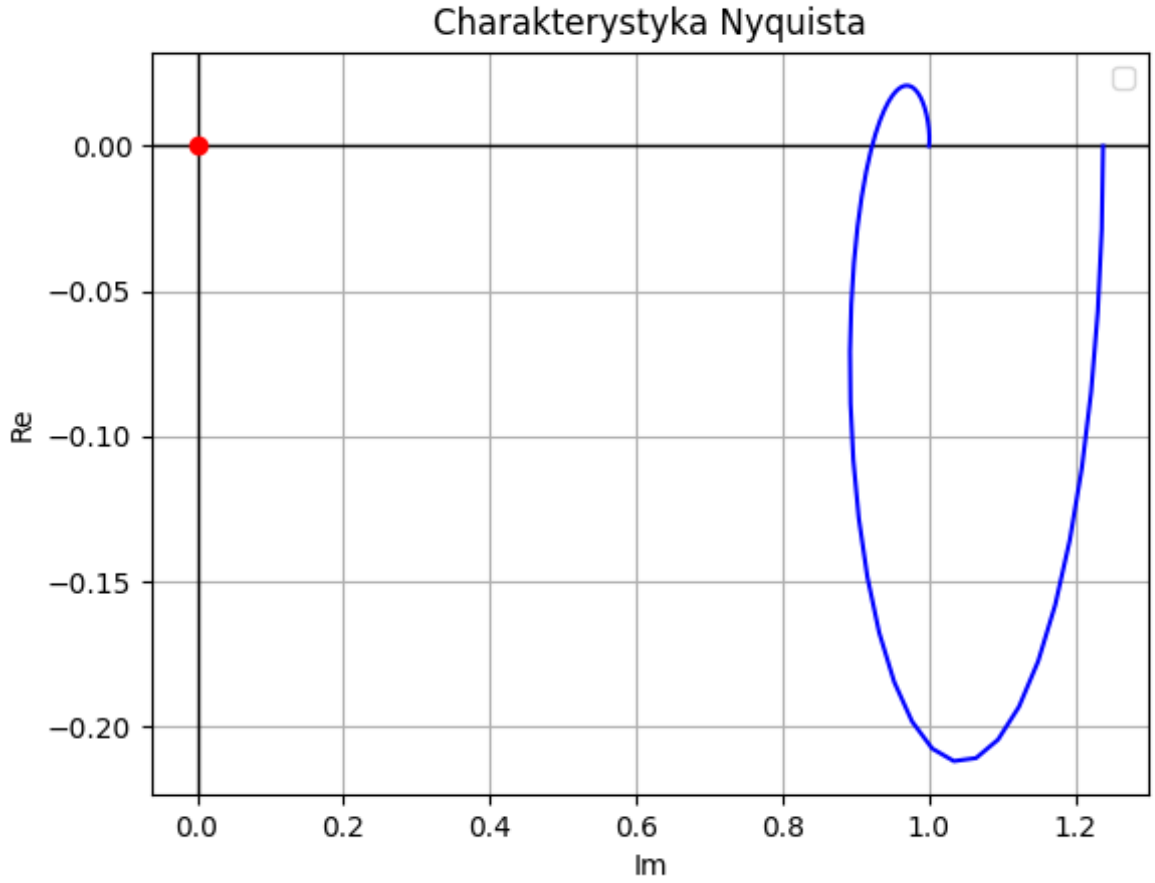
Rysunek 2: Odpowiedź skokowa dla różnych wartości k



Rysunek 3: Wykres funkcji $K_o(j\omega)$



Rysunek 4: Wykres funkcji $1 + K_o(j\omega)$



Rysunek 5: Wykres funkcji $1 + K_o(j\omega)$

4 Podsumowanie

Przeanalizowanie stabilności systemu i ocena wpływu parametru k na jego działanie umożliwiły głębsze poznanie jego właściwości oraz określenie wartości, dla których system może być stabilny.