Kontroler PID w inżynierii lotniczej

Symulacja utrzymywania wysokości przez rakietę przy pomocy autopilota opartego o PID

Cel sprawozdania

Celem sprawozdania jest zbadanie użyteczności kontrolerów PID i ról ich poszczególnych członów w kontekście sterowania rakietą o jednym wektorze ciągu na przykładzie zadania utrzymywania stałej wysokości, bądź autopilota osiągającego daną wysokość i utrzymującego ją. Badając różne zależności parametrów kontrolera PID zostaną wyznaczone role i odpowiedzialności poszczególnych członów kontrolera. Nieformalne kryteria oceny wynikają ze specyfiki problemu. Osiągnięcie wysokości 0m nad terenem w połączeniu z pionową prędkością przekraczającą ~5m/s jest rezultatem nieporządanym (tzw. eksplozja). Również z racji sposobu działania silników rakietowych na paliwo ciekłe z utleniaczem, zbyt częste przełączanie i niepotrzebne oscylacje przepustnicy również będą wykluczane. Oczywiście podstawowym zadaniem kontrolera jest szybkie i stabilne osiągnięcie zadanej wysokości.

Opis symulatora

Symulacja przeprowadzona będzie w symulacyjnej grze komputerowej Kerbal Space Program stworzonej przez firmę Squad. Aby umożliwić programowanie autopilota, dodatkowo w grze dołączony został dodatek Kerbal Operating System wraz z kodem wykorzystującym wbudowany w KOS kontroler PID napisanym na potrzeby tej symulacji.

We wszystkich symulacjach startuje taki sam model rakiety, stworzony na potrzeby tej symulacji ze specjalnie wysokim stosunkiem maksymalnego ciągu do masy w celu większej dynamiki systemu. Symulacja przebiega według następującego schematu:

- Rakieta startuje stojąc na podłożu (środek ciężkości ok. 1.5m nad ziemią)
- Przez pierwsze 20 sekund lotu (T+0 do T+20) wysokość zadana to 50m.
- W czasie (T+20 - T+40) wysokość zadana to 100m.
- Dla (T+40 - T+50) wysokość zadana to 110m, dla (T+50 - T+90) wysokość zadana to 50m.
- W momencie T+90 następuje zakończenie testu i zapisanie danych do formatu .json.

Przez cały czas trwania symulacji nie następuje żadne sterowanie od strony pilota, systemy stabilizacji RCS + SAS są włączone.

Po zakończeniu wszystkich symulacji, dane zostały odczytane przez skrypt w języku python3, który parsuje pliki .json, tworzy odpowiednie listy danych i generuje (w matplotlib) wykresy wysokości nad terenem, łącznego ciągu silników i prędkości pionowej w zależności od czasu.

Symulacja fizyki w grze Kerbal Space program uwzględnia następujące siły działające na rakietę:

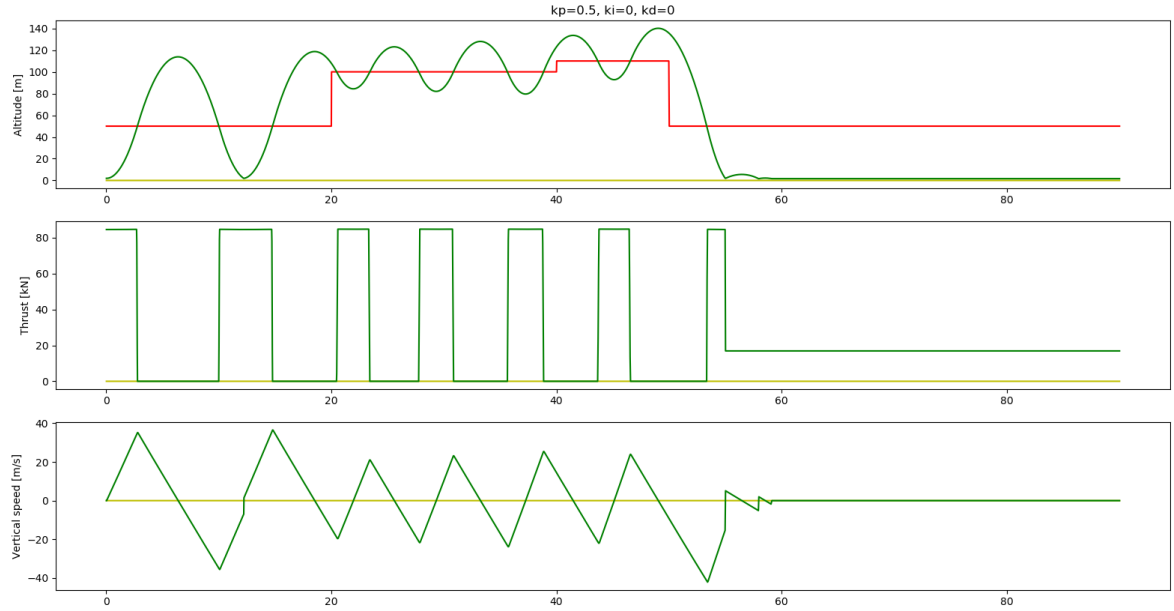
- Siła ciążenia (przyśpieszenie grawitacyjne ~1g na powierzchni i masa rakiety ~3775kg)
- Siła ciągu silników (maksymalnie ~84kN na poziomie morza)
- Opory powietrza (zależne od wysokości i kształtu rakiety [różny spadając i wznosząc się])

Plan symulacji

Wyżej opisana symulacja została przeprowadzona wielokrotnie dla następujących wartości parametrów Kp, Ki, Kd. Pogrubione zostały pomiary warte uwagi, których wykresy umieszczono i skomentowano poniżej. Resztę wykresów i danych można obejrzeć na repozytorium git projektu.

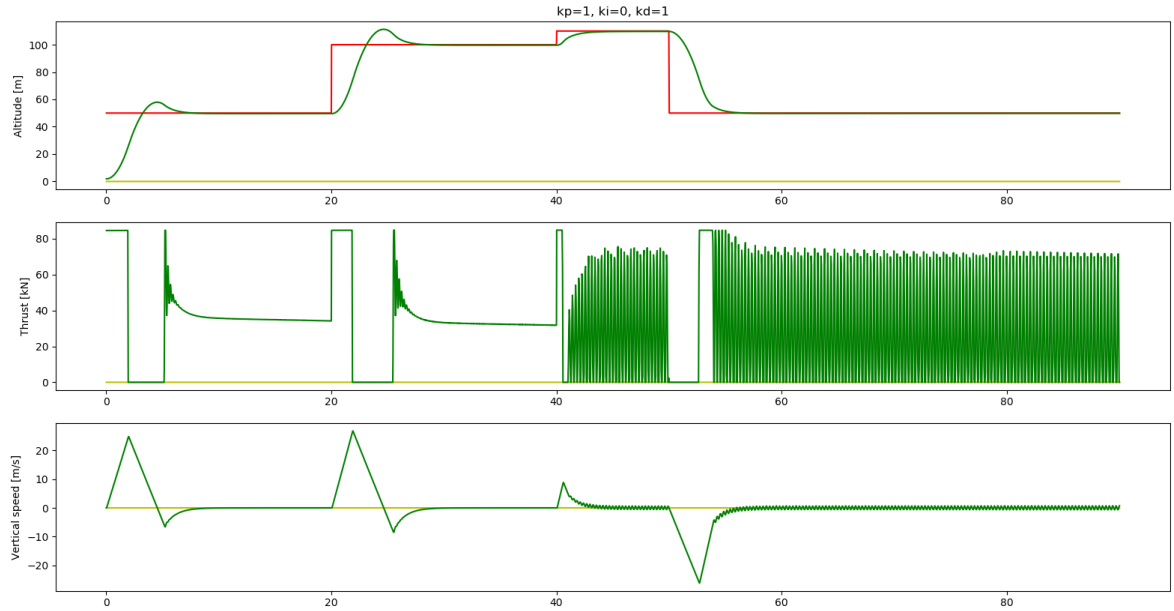
Kp	Ki	Kd
0.5	**0**	**0**
1	**0**	**1**
1	0	0.5
1	1	0.5
1	0.5	1
0.5	0.5	1
0.5	**0.5**	**0.5**
1	1	1
0.5	0.5	0.25
0.5	0.5	0.75
0.25	0.5	0.25
0.5	0.25	0.25
0.5	**0.25**	**0.5**
0.4	**0.35**	**0.5**
0.6	**0.15**	**0.5**
0.25	0.25	0.5
0.12	0.12	0.25

1. Kp=0.5, Ki=0, Kd=0



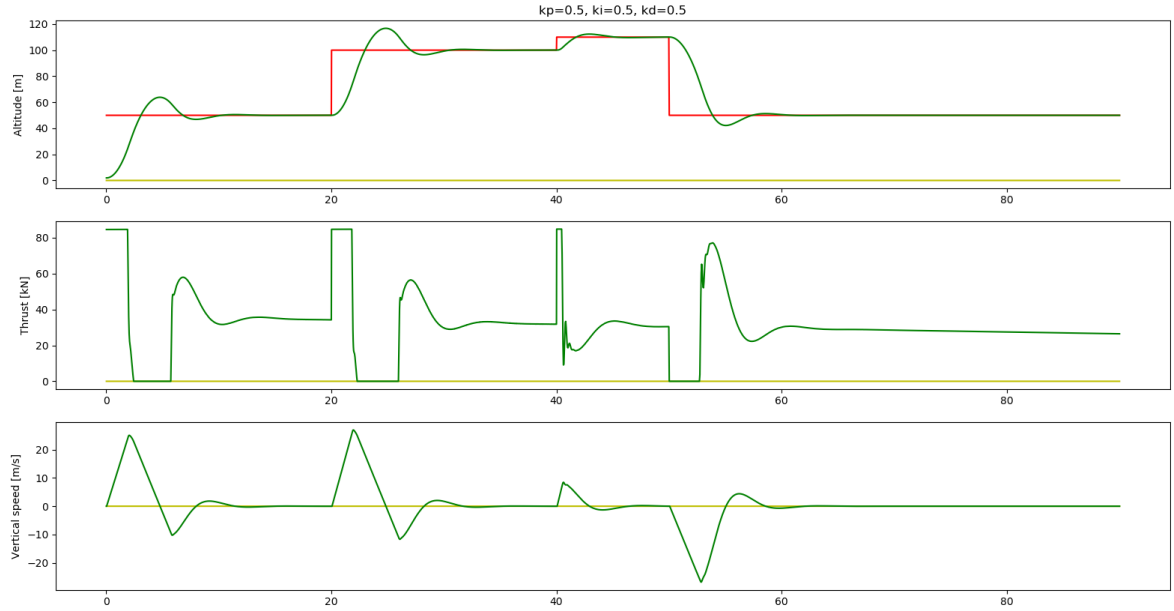
Naiwne ustawienie członu proporcjonalnego jako jedynego systemu sterującego spowodowało odbiciem się od ziemi około 11 sekundy, brakiem stabilności i kolizją niszczącą 4 z 5 silników rakiety około 55 sekundy lotu. Człon proporcjonalny w samotności powoduje stałe oscylacje i nie radzi sobie ze zmianami wartości zadanej.

2. Kp=1, Ki=0, Kd=1



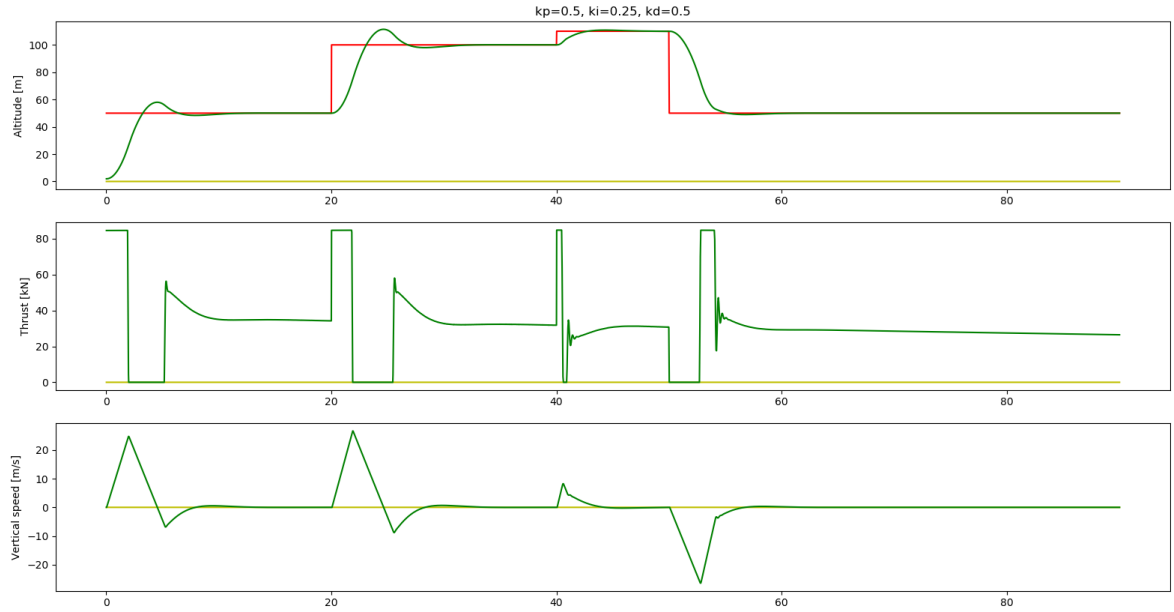
Kontroler PD dobrze trzyma się wartości zadanej, ale bardzo chaotycznie zmienia ustawienie przepustnicy co w prawdziwej rakiecie na paliwo ciekłe z utleniaczem nie jest możliwe.

3. Kp=0.5, Ki=0.5, Kd=0.5



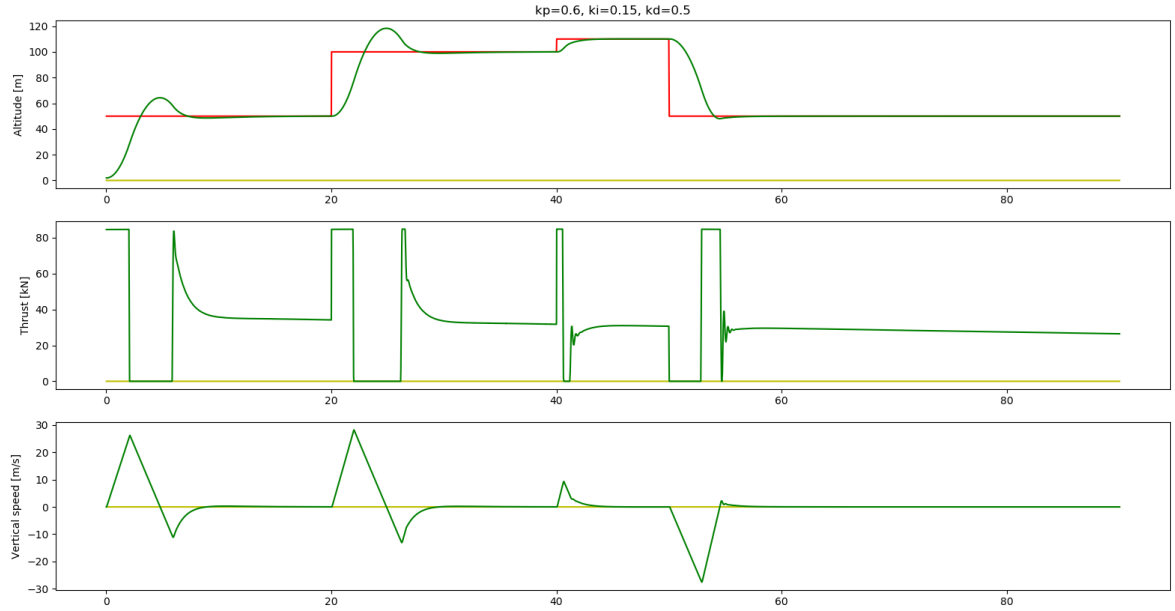
Równa waga wszystkich członów PID jest względnie dobrym rozwiązaniem. Oscylację szybko łagodzą się po niewielkim „przestrzeleniu”. Obsługa przepustnicy jest w miarę płynna

4. Kp=0.5, Ki=0.25 Kd=0.5



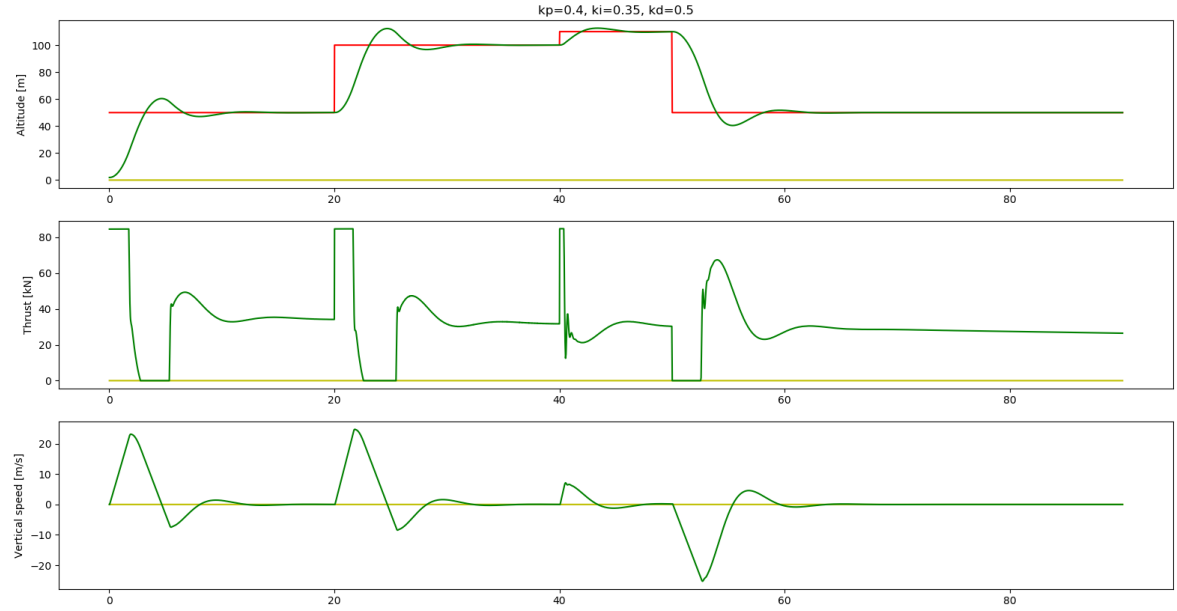
Zmniejszenie wagi członu całkującego skutkuje zmniejszeniem oscylacji, ale przynosi także więcej gwałtownych zmian ciągu, co uniemożliwia wykorzystanie takiej konfiguracji w rzeczywistości.

5. Kp=0.6, Ki=0.15, Kd=0.5



Zwiększenie udziału członu proporcjonalnego i zmniejszenie - całkującego skutkuje brakiem oscylacji po początkowym „przestrzeleniu”, ale wprowadza niebezpieczne dla silnika gwałtowne zmiany w otwarciu przepustnicy.

6. Kp=0.4, Ki=0.35, Kd=0.5



Powyższe ustawienie wag członów kontrolera PID skutkuje najbardziej dopuszczalnym kompromisem precyzji i stabilności sterowania w stosunku do ostrożnego i nieczęstego regulowania wartości otwarcia przepustnicy i w efekcie ciągu.

Wnioski

W tak skomplikowanym systemie zawsze występować będą jakieś kompromisy w dobieraniu parametrów kontrolera PID. W zależności od naszych kryteriów oceny i priorytetów, różne konfiguracje mogą okazać się najlepszymi. Mimo to, całkiem szybko udało się znaleźć zadowalające parametry, które zadziwiająco dobrze sprawują się utrzymując rakietę bezpiecznie i szybko na zadanej wysokości.

Równie zadziwiający okazał się fakt jak dobrym silnikiem do symulacji jest gra Kerbal Space Program. Język KerboScript używany w dodatku KOS jest względnie wygodny w użytkowaniu i precyzja działania silnika gry Kerbal Space Program pozwala na deterministyczne testy i symulacje.

Najważniejszy wniosek jaki można wyciągnąć z tych badań to fakt, że czasami zbyt wysoki udział danego członu w kontrolerze PID może mieć katastroficzne skutki w porównaniu do zupełnej nieobecności tego członu.

Podsumowanie

Cały kod wykorzystany w powyższych symulacjach, jak i logi przeprowadzonych symulacji znaleźć można na repozytorium projektu na platformie Github. Również znajdują się tam wszystkie wygenerowane wykresy. Nienajlepszej jakości filmik prezentujący jeden z testów (nieuwzględniony w sprawozdaniu ze względu na zakłocenia wynikające z nagrywania) można obejrzeć pod tym linkiem