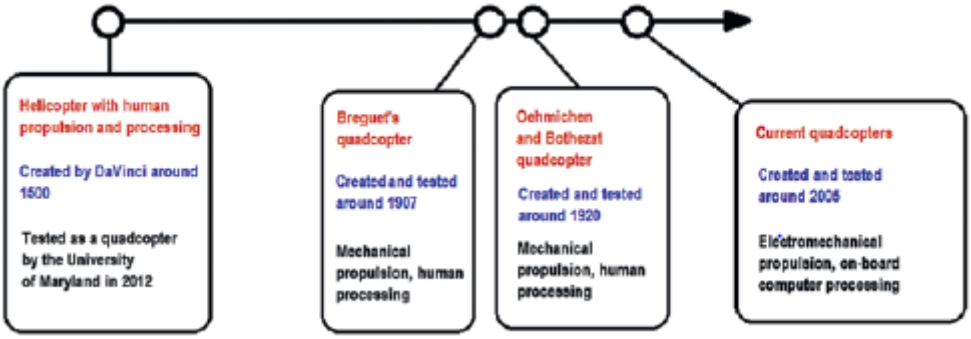
Koncepcje projektowania drona

W tym rozdziale znajdziesz najistotniejsze informacje o tym, czym jest dron, w tym kontekst historyczny i społeczny. W tym rozdziale zostaną omówione międzynarodowe standardy, niektóre etymologie i pewne względy bezpieczeństwa, o których należy pamiętać podczas pracy z tymi statkami powietrznymi. Przejdziemy przez opis ogólnych komponentów oraz ich dobór i połączenie. Dzięki temu poznasz szczegóły techniczne, międzynarodowe standardy oraz niezbędne wymagania dotyczące projektowania dronów i innych pojazdów. Pamiętaj, że ta książka jest szybkim kursem, więc ten rozdział zapewnia tylko szybki sposób przyswojenia treści. Jeśli chcesz poszerzyć tę wiedzę, przejdź do dodatku z przewodnikiem.

Kontekst historyczny

Jeśli spojrzysz na rysunek,



zauważysz, że drony, a zwłaszcza multikoptery, mają wielowiekową historię. Zaczęli jako szkice zaprojektowane przez Da Vinci (przynajmniej tak opisuje to historia Zachodu). Ostatnio te szkice zostały wykonane na gigantycznym quadkopterze w rodzaju retrotechnologii opracowanej przez University of Maryland, gdzie procesorem i silnikiem podstawowym jest ludzkie ciało (odpowiednio mózg i kończyny). Multikoptery powróciły w 1907 roku wraz z braćmi Breguet, a później w latach dwudziestych wraz z Oehmichen i Bothezatem. W tych dronach procesor i silnik podstawowy były odpowiednio pilotem i komputerem mechanicznym (mechanizmem) odpowiedzialnym za koordynację ruchów każdego śmigła i silników spalinowych. Zapomniano o nich aż do 2010 roku, kiedy nastąpił boom handlowy i badawczy w tych pojazdach/robotach. Obecnie podstawą procesora i silnika są odpowiednio autopilot i silniki elektryczne. Teraz, gdy znasz już trochę historii, przejdźmy do kilku przydatnych nomenklatur.

Etymologie i nazwy w użyciu

Choć termin „dron” jest poprawnie używany w odniesieniu do każdego pojazdu bezzałogowego, popularyzacja komercyjna kojarzy go z statkami powietrznymi, a dokładniej z multicopterami. Termin ten pochodzi od owadów, a konkretnie od tych, których funkcją w ulu lub gnieździe jest po prostu utrwalanie gatunku.

Bardziej profesjonalne słowa odnoszące się do drona latającego to

• UAV lub bezzałogowy statek powietrzny

• UAS lub bezzałogowy system powietrzny

• RPAS lub zdalnie sterowany system statku powietrznego

Powinieneś również zapoznać się z terminami „wielowirnik” i „wielokopter”, które z grubsza oznaczają wiele silników obrotowych. Termin „wiele helikopterów” został zaadaptowany z francuskiego „hélicoptère”, słowa wymyślonego przez Gustave'a d'Amécourta w 1861 roku i oznaczającego skrzydło spiralne. Słowo to jest oparte na greckim „helix”, co oznacza helisę lub spiralę i „pteron”, co oznacza skrzydło. Dlaczego warto o tym wiedzieć? Być może wydaje się to trywialne, ale dla tych, którzy chcą zbadać klasyczne koncepcje dronów, ważne jest poznanie archaicznej nomenklatury. Oczywiście, jeśli potrzebujesz tylko aktualnych badań i nowych trendów, wystarczy znać nowoczesne terminy. W przypadku pojazdów przedstawionych w tej książce ewolucja od klasycznej do obecnej terminologii jest następująca:

• Multicopter (22 400 wyników w Google Scholar 5 maja 2020 r.) -> multirotor (10 600 wyników w Google Scholar 5 maja 2020 r.)

• Quadcopter (37 400 wyników w Google Scholar 5 maja 2020 r.) -> Quadrotor (35 500 wyników w Google Scholar 5 maja 2020 r.)

• I w nietypowy sposób słowo quadricopter (674 wyniki w Google Scholar 5 maja 2020 r.)

Mniej więcej na początku 2011 roku słowa „kwadrotor” i „kwadrokopter” były prawie w równym stopniu zatrudnione. W rzeczywistości bardzo często czytano „kwadrotor” w dziedzinie naukowej, ale użycie „kwadrokoptera” rosło wraz z upływem czasu (w tym w badaniach naukowych i technologicznych). Obecnie oba terminy są mniej więcej równe w użyciu ze względu na ograniczenie ich użycia oraz dominację i popularyzację terminu „dron” (rysunek 1-3). Co ciekawe, terminy „kwadrokopter” i „kwadrotor” zostały zepchnięte na pole badawcze, gdzie obecnie rosnącym terminem jest „UAV”. Podsumowując, wartością tej sekcji jest wskazanie trendu niektórych terminów w celu ułatwienia wszelkich niezbędnych badań, a także sposobu zapewnienia tła, w jaki sposób każdy termin powinien być używany w kontekście. Teraz, gdy wiesz już o etymologiach jako narzędziu wyszukiwania, zastanówmy się, jakiego rodzaju drona potrzebujesz.

Jakiego rodzaju drona potrzebujesz?

Istnieją cztery rodzaje użytkowników dronów: specjalista naukowiec, badacz (który używa drona jako narzędzia), projektant i pilot.

Ogólne kwestie bezpieczeństwa i normy międzynarodowe

Ogólnie rzecz biorąc, nikt nie będzie Cię prześladować za przeprowadzanie testów prototypów w Twoim domu, ogrodzie lub laboratorium przy odpowiednim sprzęcie i środkach bezpieczeństwa, o ile ludzie, zwierzęta, budynki lub mienie inne niż Twoje nie zostaną zranione. Jednak, aby zmniejszyć ryzyko i wypadki, dołączamy mały rozdział z ogólnymi zaleceniami. Należy pamiętać, że wpływ tych zaleceń na każdą osobę może się różnić w zależności od lokalnych przepisów.

Komunikacja

Dron może wpływać na lokalną telekomunikację i naruszać ją na dwa sposoby i w obu przypadkach możesz zostać ukarany.

1. System telemetrii i system zdalnego sterowania: Nadajniki telemetrii i zdalnego sterowania muszą być kompatybilne z zakresem częstotliwości transmisji dozwolonym w Twojej miejscowości. Typowym przykładem jest to, że radiotelefony telemetryczne zwykle występują w dwóch prezentacjach: 433 MHz, który jest używany w Europie i krajach o tym samym standardzie oraz 915 MHz, który jest używany w USA i krajach zgodnych z tym samym standardem.

2. Zużycie energii elektrycznej: Osoby, które projektują i używają quadkopterów o wysokim zużyciu energii, zwłaszcza w zastosowaniach wysokoprądowych, wiedzą, że ma to wpływ na działanie czujników pojazdu i konieczne jest ekranowanie kabli, skręcanie ich, zakrywanie, a nawet oddzielić połączenia. Może to mieć wpływ na zachowanie anteny publicznej lub cywilnego sprzętu komunikacyjnego. Tak więc, jako projektant lub użytkownik, powinieneś mieć świadomość, że możesz zostać ukarany.

Regulacje i Normalizacje

Kiedy dojdziesz do Części 2, zauważysz niearbitralny wybór współrzędnych i kątów obrotu, jak pokazano na rysunku

Wybór ten jest zgodny z normą ISO 1151-2: 1985 i ułatwia powiązanie opisanej teorii z wykorzystaniem już wyprodukowanych czujników i komponentów. Jednak, aby uprościć równania, użyjemy skierowanej w górę osi Z. Zauważ, że w przypadku drona, jak zobaczysz później, chociaż kąty pochylenia (teta) i przechyłu (phi) są mierzone odpowiednio w osiach Y i X, ruch w osi X zależy od kontrolowanych obrotów przechyłu w teta, a ruch w osi Y zależy od kontrolowanych obrotów przechyłu w phi, jak pokazano na rysunku na rysunku.

Zalecenia

Oto kilka standardowych zaleceń dotyczących pracy z dronami:

• Zapoznaj się z lokalnymi przepisami dotyczącymi wymienionych tutaj tematów, a nawet dodatkowych rozważań.

• Sprawdź, czy Twój sprzęt jest zgodny z wymaganiami wspomnianych przepisów.

• Ty i Twój zespół musicie używać standardowego sprzętu ochronnego, takiego jak gogle lub rękawiczki.

• Gdy podczas testu obecne są inne osoby, należy pamiętać, że prototyp może spowodować obrażenia i zaleca się, abyś Ty lub osoba kierująca laboratorium posiadali licencję na obsługę bezzałogowych statków powietrznych.

• Unikaj miejsc publicznych lub prywatnych, zarówno wewnątrz, jak i na zewnątrz, które nie są przeznaczone do testów.

• Pamiętaj, że jesteś projektantem, a niekoniecznie pilotem.

• Jeśli posiadasz niezbędne uprawnienia, musisz przygotować miejsce ze ścianami, siatkami ochronnymi lub plastikowymi powierzchniami na wypadek pęknięcia lub oderwania się jakichkolwiek elementów od drona. Powinieneś również umieścić ostrzeżenia/znaki ostrzegawcze, aby ograniczyć dostęp nieupoważnionego personelu.

• Przeprowadź wyczerpujące testy bez śmigieł lub z zakotwiczonym dronem przed testami lotów swobodnych. Aby uniknąć uszkodzenia silników, testując je bez obciążenia, możesz wymienić śmigła na kawałki papieru.

• Gdy masz dostępny prototyp, sprawdź, czy wszystkie jego elementy są legalne w Twojej lokalizacji i określ, czy istnieją funkcjonalne zamienniki tych elementów, które nie są dozwolone. Teraz nadszedł czas na określenie rodzaju dronów, które później przeanalizujemy.

Rodzaje dronów

Według ich przemieszczenia drony można sklasyfikować w ten sposób:

• Skrzydła obrotowe: do poruszania się wymagają jedynie śmigieł. W tym przypadku mają pionowy start i lądowanie i są przeznaczone do transportu ładunków oraz zastosowań na krótkim i średnim zasięgu.

• Skrzydła stałe: Do poruszania się wymagają długich powierzchni, jak większość samolotów. W tym przypadku mają zastosowanie do poziomego startu i lądowania i są przeznaczone do wyższych prędkości oraz dłuższego zasięgu lotu niż samoloty z obrotowymi skrzydłami.

• Trzepoczące skrzydła: do latania potrzebują oscylujących powierzchni, jak skrzydła ptaka lub owada. Mogą być postrzegane jako połączenie skrzydeł nieruchomych i obrotowych i podobnie mają mieszane zastosowania.

• Transformowalne: mogą przechodzić z jednej konfiguracji do drugiej, jak w przypadku ogonowców. Pojazdy te przechodzą z pionowego startu i lądowania za pomocą śmigieł do trybu lotu ze stałym skrzydłem do operacji dalekiego zasięgu.

Nasz tekst obejmuje tematy dotyczące bezzałogowych statków powietrznych typu multicopter, które są oparte na napędzie wiropłatów. Jednak większość koncepcji można rozszerzyć na inne typy statków powietrznych pod dodatkowymi ograniczeniami lub względami (w tym tematy związane z modelowaniem, sterowaniem, symulacją i programowaniem). Przejdźmy do komponentów drona.

Składniki

Opiszemy najistotniejsze elementy drona na podstawie ich zastosowania: elementy działania, elementy konstrukcyjne, elementy pomiarowe, elementy dowodzenia, elementy mocy.

Komponenty akcji

Istnieją dwa rodzaje elementów działania: działanie elektromechaniczne i elektryczne. Typy elektromechaniczne to silniki lub serwa, które po prostu odbierają sygnał elektryczny, a następnie przekształcają ten sygnał w efekt mechaniczny. W przypadku działania elektrycznego są to

komponenty zapewniające stopień wzmocnienia lub adaptację elektryczną. Elektromechaniczne elementy wykonawcze mogą mieć bezpośredni wpływ na prędkość (silniki bezszczotkowe i szczotkowe) lub położenie (silniki serwo).

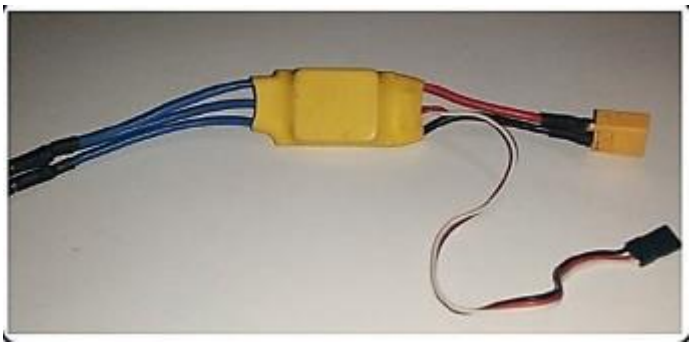
Silniki szczotkowane i bezszczotkowe

Obecnie, ze względu na czas reakcji, najczęstsze silniki stosowane w multikopterach (dla producentów i naukowców) są oparte na silnikach elektrycznych prądu stałego. Tego rodzaju silniki mogą być szczotkowane lub bezszczotkowe. W obu przypadkach wymagana jest modyfikacja prędkości silnika poprzez zwiększanie lub zmniejszanie jej poprzez regulację wartości takich jak napięcie RMS. W przypadku silników szczotkowych są to najczęściej spotykane silniki elektryczne. Stosowane są w dronach o bardzo małych rozmiarach. Szczotki skracają czas reakcji ze względu na tarcie kontaktowe. Ponadto nie są zalecane w środowiskach zagrożonych wybuchem ze względu na wytwarzane przez nie iskry i nie są zalecane w środowiskach wodnych, ponieważ mogą same się zwierać. Ponieważ jednak działają one z sygnałami cyklu pracy PWM (pamiętaj, że modulacja szerokości impulsu to różne techniki regulacji sygnału poprzez zmianę procentu jego działania, w tym przypadku poprzez regulację procentu włączenia/wyłączenia sygnału mocy), ten rodzaj PWM jest zwykle stosowany na przykład w płytce Arduino. Silniki bezszczotkowe są zwykle spotykane w multikopterach ze względu na ich krótszy czas reakcji oraz brak styków elektrycznych, co oznacza zmniejszenie tarcia. Są stosowane w środowiskach zagrożonych wybuchem ze względu na brak iskier i mogą być stosowane w środowiskach wodnych, ponieważ nie mają elementów podatnych na zwarcia (o ile ich przewody są pokryte powłoką). Z drugiej strony są one droższe w porównaniu do silników szczotkowych i działają ze specjalnym rodzajem PWM zwanym RC, gdzie procent włączeń/wyłączeń jest regulowany inaczej niż silniki szczotkowe (tu procent jest funkcją czas), wymagające dodatkowych komponentów. Do prawidłowego użytkowania obu typów silników wymagane są stopnie mocy, jak zobaczysz w następnej sekcji. Należy zauważyć, że PWM typu cyklu pracy i PWM typu vRC nie są bezpośrednio kompatybilne. Główną różnicą jest to, że silniki szczotkowe są zwykle jednofazowe, podczas gdy bezszczotkowe są silnikami trójfazowymi lub wielofazowymi (zwykle trójfazowymi w zastosowaniach lotniczych). Rysunek przedstawia zbyt duży silnik szczotkowy w porównaniu z silnikiem bezszczotkowym, ale może się również zdarzyć odwrotnie; zależy to po prostu od wymagań użytkownika lub projektanta.

Bardziej niezwykłą różnicą, którą powinieneś zauważyć, jest liczba kabli (faz elektrycznych) oraz fakt, że silniki szczotkowane są zazwyczaj typu inrunner (obracający się wałek wewnętrzny), podczas gdy bezszczotkowe są często outrunner (obracająca się obudowa zewnętrzna).

ESC i stopnie mocy

ESC, zwane również wariatorami prędkości , to stopnie mocy silników bezszczotkowych i niektórych silników szczotkowych (w przypadku tych ostatnich są to tylko elektroniczny sygnał RC do adapterów cyklu pracy).



Zasadniczo są one wstrzykiwane sygnałem, który reguluje ich stany włączenia/wyłączenia w formacie PWM RC. W ten sposób sygnał mocy jednofazowej i prądu stałego jest przekształcany na rodzaj sygnału trójfazowego sterującego prędkością obrotową silnika. W przypadku szczotkowanych silników prądu stałego, jak już wyjaśniono, można zastosować adapter ESC. Pobiera to sygnał PWM RC, który jest konwertowany na sygnał cyklu pracy PWM. Inną opcją, która jest częściej stosowana w urządzeniach, jest sterownik elektroniczny, który pobiera PWM cyklu pracy jako wejście, a następnie wzmacnia sygnał w celu uzyskania prądu lub napięcia niezbędnego do regulacji prędkości silnika. Zarówno ESC, jak i sterowniki elektroniczne są nazywane stopniami mocy, ponieważ są to obwody elektroniczne, które wzmacniają sygnały elektryczne wysyłane przez autopiloty lub jednostki przetwarzające (które nie są w stanie dostarczyć wystarczającej ilości energii elektrycznej) do napędzania silników. Zauważ, że jeśli spróbujesz przenieść silnik bezpośrednio z procesora bez użycia odpowiedniego stopnia mocy, możesz spalić lub złamać ten procesor.

Serwa

Serwosilniki (powszechnie znane jako „serwo”) to silniki pozycjonujące, które otrzymują sygnał typu PWM RC, który jest przekształcany na pozycję osi sprzężonej z silnikiem. Zobacz rysunek.

Działają podobnie do silnika bezszczotkowego, z tym wyjątkiem, że BLDC (bezszczotkowy silnik prądu stałego) zmienia swoją prędkość poprzez zmiany sygnału PWM RC, podczas gdy serwo zmienia swoją pozycję. Wymagają również odpowiedniego stopnia mocy. Dodatkowo nie zmieniają pozycji tak szybko, jak silnik bezszczotkowy. W związku z tym istnieje ryzyko pęknięcia lub spalenia siłownika przy nagłej zmianie sygnału PWM. Sygnał PWM używany do serw musi być wolniejszy niż sygnał PWM używany do BLDC.

Śmigła

Silnikowi trudno jest oddziaływać bezpośrednio na środowisko. Taka interakcja wymaga obiektów, takich jak koła lub śmigła, w zależności od tego, czy jest to jednostka pływająca, samolot czy pojazd naziemny.

Ze względów bezpieczeństwa pojazd powinien być testowany bez nich, dopóki nie uzyskasz pożądanego zachowania. Silnik ma dwa rodzaje osiągów. Pierwsza to praca przy pełnym obciążeniu, która ma miejsce, gdy śmigła są umieszczone, a druga to praca swobodne, co oznacza, że silnik nie ma śmigła ani innego przedmiotu przymocowanego do jego konstrukcji. Oba przypadki reprezentują skrajne sytuacje. W przypadku osiągów swobodnych, które są wspólne dla prób wstępnych, zaleca się stosowanie kawałków papieru zamiast śmigieł jako obciążenia bezpieczeństwa (ciężaru). W trybie pełnego obciążenia zaleca się nie sięgać maksymalną wartość obciążenia (masa) zalecana przez producenta.

Elementy struktury

W poniższych sekcjach omówiono komponenty, które nadają kształt i wsparcie pojazdowi.

Rama

Rama to po prostu podwozie lub karoseria pojazdu , na którym zamontowane są inne elementy.

Mocowania tłumiące drgania : Mocowania tłumiące drgania to konstrukcje zaprojektowane do pochłaniania lub redukcji hałasu, który wywołuje wibracje pojazdu na czujnikach lub procesorze, a tym samym pozwala uniknąć nieprawidłowych pomiarów lub błędów obliczeniowych.

Łącznik mechaniczny : Oczywiste jest, że elementy muszą być zamocowane na stałe lub długotrwale (śruby) lub w sposób wymienny lub krótkotrwały (rzepy, paski).

Komponenty pomiarowe : W poniższych sekcjach omówiono komponenty, które pozwalają pojazdowi rozpoznać, gdzie jest umieszczony.

Różne czujniki: multikopter wymaga czujników ruchu i prędkości na płaszczyźnie XY i wysokości, a także trójwymiarowych czujników prędkości obrotowej i kątowych. Mogą to być czujniki analogowe, cyfrowe lub szeregowe, a nawet czujniki pokładowe i zewnętrzne. Najczęściej spotykane są barometry, IMU, żyroskopy, akcelerometry, LIDARy, kamery i systemy śledzenia ruchu. (Zobacz Załącznik, aby dowiedzieć się więcej na ich temat).

GPS: Chociaż jest to trójwymiarowy system czujników ruchu i prędkości (translacyjny), GPS ma swoją własną sekcję, ponieważ jest szeroko stosowany przez twórców i naukowców, zwłaszcza w zastosowaniach na otwartym terenie. Jest to w zasadzie satelitarny system triangulacji, na który wpływa środowisko; dlatego jest zalecany tylko do operacji na otwartej przestrzeni.

Komponenty poleceń

W poniższych sekcjach omówiono komponenty, które umożliwiają działanie pojazdu i sterowanie nim.

Autopilot

Autopilot to wyspecjalizowana jednostka przetwarzająca dla pojazdów bezzałogowych .

Istnieją również mikrokontrolery lub płytki rozwojowe przystosowane do działania jako autopiloty. Mają architekturę zamkniętą i otwartą.

Moduły telemetryczne

Moduły telemetryczne to jednostki do bezprzewodowej i zdalnej transmisji informacji .

Mają ograniczony zasięg działania. Moduły telemetryczne mogą być wykorzystywane do komunikacji między dwoma lub więcej pojazdami lub do stałej bazy z jednym lub kilkoma dronami. Mogą być używane do przesyłania do pojazdu pozycji zajętych za pomocą systemu przechwytywania ruchu. Są to na ogół moduły dwukierunkowe.

Moduły RC

W przeciwieństwie do modułów telemetrycznych, moduły RC pozwalają na ręczne sterowanie dronem za pomocą drążków i przycisków. Mają szeroki zasięg działania, a niektóre z nich umożliwiają transmisję wideo. Zazwyczaj są one jednokierunkowe, co oznacza, że posiadają moduł nadawczy oraz urządzenie odbiorcze.

Komputer towarzyszący

Komputer towarzyszący to pokładowa jednostka przetwarzająca, która służy do wykonywania wymagających zadań, które zwykle wykraczają poza zakres autopilota (takich jak sztuczne widzenie lub SLAM). Przykładami są komputer jednopłytkowy Raspberry Pi, mikrokontroler, mikroprocesor lub FPGA. Zasadniczo autopilot jest bezpośrednio używany do sterowania lotem, a komputer towarzyszący jest używany do zadań pośrednich podczas tego lotu.

Komponenty zasilania

W poniższych sekcjach omówiono komponenty, które zasilają pojazd

Akumulatory/materiały na uwięzi

Użycie baterii lub zasilaczy na uwięzi do zasilania systemu jest oczywiste. Chociaż większość dronów wykorzystuje obecnie akumulatory typu LIPO do długotrwałych zastosowań, dostępne są konwertery mocy lub zasilacze, co oznacza pojazd podłączony do przedłużacza elektrycznego. Oczywiście takie rozszerzenie wiąże się ze specjalnymi warunkami użytkowania i ograniczeniami mobilności. Idealne warunki dla tych zasilaczy są takie, że mają one zredukowane tętnienia napięcia i prądu lub skoki na poziomie, aby dostarczane wartości były tak stałe, jak to możliwe. Ponieważ akumulatory mają obszerną dokumentację, porozmawiamy nieco więcej o dostawach przewodowych lub na uwięzi. Istnieją cztery rodzaje zapasów na uwięzi:

• Przez zasilacze: w tym przypadku największym utrudnieniem jest koszt, ponieważ wielośmigłowy dron oparty na bezszczotkowych silnikach zużywa znacznie więcej prądu niż napięcia (mały pojazd z łatwością zużywa około 12V/60A). W ten sposób, mając źródło zasilania, które zapewnia wspomniane napięcie i prąd, należy wziąć pod uwagę, że kable i przewody muszą być w stanie poradzić sobie z tymi parametrami, w tym fakt, że kabel pobierający tak dużo prądu musi być fizycznie ciężki i nieporęczne, przez co nie nadają się do użytku na dużych wysokościach. Dlatego ta opcja jest zalecana, jeśli Twoja aplikacja jest na poziomie laboratorium, gdzie dron ledwo przekracza półtora metra wysokości.

• Z akumulatorów samochodowych: To samo dzieje się w poprzednim przypadku, ale koszt jest nieco obniżony, biorąc pod uwagę, że akumulatory samochodowe są jednostkami bardzo komercyjnymi. Jednak akumulator samochodowy jest zwykle ciężki i wymaga szczególnej troski podczas transportu i ładowania elektrycznego.

• Przez konwertery mocy: W tym przypadku wymagane jest dostarczenie średniego napięcia i niskiego prądu, na przykład 400V/3A, aby zasilić konwerter mocy zdolny do przekształcenia tego wejścia na 12V/100A. Te konwertery mocy są dostępne u niektórych producentów (na przykład VICOR). Zwykle ważą 200 g wraz z radiatorami i są realną opcją dla tych aplikacji na uwięzi.

• Ogniwa słoneczne: Ograniczenia są podobne do poprzednich przypadków, z implikacjami, jakie mają takie ogniwa (takie jak rozmiar i gęstość mocy).

Można ewentualnie argumentować, że istnieją opcje oparte na paliwie, ponieważ mają one dziesiątki lat użytkowania w standardowych helikopterach; jednak w przypadku multikopterów elektrycznych nadal mają wyzwania badawcze (z wyjątkiem generatorów paliwowych, w których silnik paliwowy jest używany jako źródło zasilania elektrowni).

Wskaźnik baterii

Jeśli używana jest bateria, ważne jest, aby mieć urządzenie, które ostrzega nas, gdy bateria się wyczerpie, aby samolot nie zapadł się i nie uległ uszkodzeniu. Zasadniczo są to małe woltomierze ze wskaźnikami świetlnymi lub dźwiękowymi.

Dystrybutor mocy

Rozdzielacz mocy to urządzenie zaprojektowane do korzystania z wielu silników z jednym zasilaczem. Są to w zasadzie dzielniki prądu w prezentacji uprzęży (zużywają więcej miejsca, ale łatwiej nimi manipulować) lub jako płytki z obwodami elektronicznymi (mają mniej miejsca, ale trudno nimi manipulować). Zobacz Rysunek pośrodku.

Moduł zasilania

Moduł zasilania to urządzenie umożliwiające zasilanie autopilota z baterii głównej. Zobacz Rysunek po lewej.

Złącze elektryczne

Złącza elektryczne to sposób na połączenie różnych urządzeń elektrycznych lub elektronicznych. Istnieją wersje przeznaczone do zasilania lub danych. Złącza zasilania mogą obsługiwać duże ilości napięcia i/lub prądu. Złącza danych służą wyłącznie do wysyłania i odbierania informacji, które zazwyczaj zużywają bardzo niski prąd lub napięcie. Tabele poniższe zawierają więcej informacji na

temat poprzednich komponentów i są przedstawione w trzech kategoriach: komponenty mocy, komponenty mechaniczne i komponenty sterujące.

Element mocy : Główne cechy : Częste procedury : Typy

Bateria : Liczba ogniw, rozmiar, waga, amperogodziny, napięcia robocze, C lub szybkość rozładowania, złącza, akcesoria wymagane do ich użycia (ładowarka) : Ładowanie, rozładowywanie, przechowywanie, transport, lutowanie złączy, zrównoleglenie lub serializacja : Istnieją różne, ale baterie LIPO są najbardziej obfite.

Zasilanie : Moc, prąd, napięcie, tętnienie, częstotliwość, napięcie wejściowe (110/220), liczba faz : Filtrowanie w razie potrzeby, łączenie faz w razie potrzeby, wzmocnienie lub redukcja, regulacja, ochrona przed rozładowaniem, transformacja (AC-DC) : Przełączane i liniowe

Wskaźnik baterii : Dozwolona liczba ogniw, wskaźniki świetlne lub dźwiękowe : Weryfikacja pinów : Na pokładzie lub na zewnątrz

Moduł mocy : Maksymalny obsługiwany prąd, maksymalne obsługiwane napięcie, ekran elektromagnetyczny, maksymalny zmierzony prąd, typ złączy : Ekranowanie elektromagnetyczne w razie potrzeby, dodanie czujników prądu lub napięcia w razie potrzeby, zrównoleglenie w razie potrzeby : Dla niskiego i wysokiego prądu

Rozdzielacz zasilania : Maksymalny obsługiwany prąd, maksymalne obsługiwane napięcie, liczba silników do zasilania, BEC, rozmiar, waga, typ Ekran elektromagnetyczny : Lutowanie zacisków, serializacja lub zrównoleglenie, ekranowanie elektromagnetyczne : Hub (wiązka) lub płyta (obwód)

Złącza zasilania : Maksymalny obsługiwany prąd, maksymalne obsługiwane napięcie, kompatybilność z komponentami, rezystancja termiczna, elastyczność, możliwości podłączania/odłączania : Lutowanie zacisków, wymiana : Każda aplikacja ma swój własny zestaw złączy

Stopnie mocy i esc : maksymalny rozmiar, waga, prąd i napięcie, odwracalność, BEC lub optozłącze : konfiguracja (zwykle za pomocą interfejsu oprogramowania) : odwracalne lub nieodwracalne, cykl pracy PWM-RC lub PWM, standardowe lub sprzężone opto szczotkowane silniki

Element kontrolny : Główne cechy : Częste procedury : Typy

Autopilot : Rozmiar, waga, liczba silników, złącza, dostępne porty, architektura, język programowania, procesor, wymagana moc : Programowanie : Architektury otwarte i zamknięte

Czujniki: Rozdzielczość, rozmiar, waga, liniowość typu, wymagana moc, rodzaj złącza, zakres i warunki pracy: Lutowanie, sprzężenie sygnału, filtrowanie: Analogowe, cyfrowe i szeregowe

Moduły telemetryczne :: Rozmiar, napięcia i prądy pracy, wymagane akcesoria, kompatybilność złącz, maksymalna odległość, równość częstotliwości pracy : Wiązanie i etykietowanie, prędkość danych konfiguracji: standardy amerykańskie i europejskie

Pilot zdalnego sterowania : Maksymalna odległość, obsługa baterii LIPO, legalność częstotliwości pracy, liczba kanałów, tryb wiązania, rodzaj kanałów pomocniczych, czułość, tryby transmisji, ekran lub nie, obecność lub brak kanału PPM w odbiorniku : Łączenie odbiornik z konfiguracją nadajnika, drążków i przycisków: Według liczby kanałów i dodatkowych funkcji jako ekran

Karta pamięci : Rozmiar pamięci, typ, adaptery, dostępność do noszenia, liczba cykli pracy : Zapis, kasowanie, użycie zewnętrzne lub na płycie, konwersja formatu : Zwykle według rozmiaru pamięci i typu komercyjnego

Komputer towarzyszący: ten sam, co autopilot plus dodatkowe funkcje przetwarzania i kompatybilność ze sprzętem i oprogramowaniem: Programowanie: według poziomu abstrakcji kodu (język naturalny a język maszynowy), również według wydajności sprzętowej

Przycisk stop : Dostępność dla użytkownika, rozmiar, kolory, tłumienie mechaniczne : Poziom dostępności, programowanie : Przyciski, drążki, na podstawie czasu lub zdarzeń

Złącza danych : Standardowe w użyciu i kompatybilności, narzędzia do ich użycia, prędkość, ekrany elektromagnetyczne, dostępność na rynku, liczba pinów, uszczelnione lub nie, wersje standardowe lub specjalne, maksymalne napięcie i prąd, łatwe lub sztywne podłączanie : Lutowanie, wymiana, programowanie : To zależy od każdej aplikacji

Komponent mechaniczny : Główne cechy : Częste procedury : Typy

Rama : Materiał, twardość, lekkość, otwory, poziomy, akcesoria, rozmiar i waga, podwozie, systemy przeciwwstrząsowe, składane lub nie, ekranowanie elektromagnetyczne : Połączenie komponentów, wyważanie, przywracanie : Powszechne klasyfikacje dotyczą rozmiaru, materiałów i zdolności składania

Silniki : kv, waga i wymiary, maksymalny prąd i napięcie, maksymalny ciąg i moment : Uzyskanie wartości początkowej i proporcjonalności przyrostów momentu i ciągu w odniesieniu do ich prędkości i śmigła, wiązanie PWM (za pomocą pilota) : Szczotkowane i bezszczotkowe , prędkość lub pozycja, rodzaj sygnału sterującego (DC, BLDC, serwo, stepper, AC, PWM RC lub PWM duty cycle, itp.)

Śmigła: Podziałka, średnica, krawędź, liczba ostrzy, elastyczność, twardość, kierunek obrotu: Wyważanie, cięcie krawędzi: Zwykłe klasyfikacje opierają się na materiale, elastyczności i liczbie ostrzy.

Łączniki : Materiał, twardość, lekkość, akcesoria, rozmiar i waga, szybka lub wolna zmiana : Każde połączenie mechaniczne jest przypadkiem analizy i zależy od zastosowania. : Szybka zmiana lub stałe połączenie

Mocowania antywibracyjne : Rozmiar, kompatybilność z ramą, mobilność, rodzaj tłumienia : Wyważanie : Do silników, do autopilotów, do czujników

Teraz, gdy znasz już wstępne koncepcje dotyczące komponentów dronów, zobaczmy, jak je wybrać.

Wybór komponentów

Proces ten jest podzielony, zgodnie z zalecanym projektem, na trzy podprocesy (analiza jest przeprowadzana z uwzględnieniem silników bezszczotkowych, ponieważ są one najczęściej używane w quadkopterach).

Wybór pojazdu

Wybór pojazdu to proces iteracyjny, na który składają się:

1. Wybór silników. Silniki dobierane są w zależności od wagi i prędkości, z jaką chcesz prowadzić pojazd. Dlatego należy zapoznać się z dwoma ważnymi parametrami w arkuszach danych: maksymalnym

ciężarem, jaki może podnieść każdy silnik i jego maksymalną wartością obrotów (w silnikach bezszczotkowych nazywa się to kv i nie należy go mylić ze stałymi elektromechanicznymi silnika).

2. Zgodnie z powyższym (nośność i prędkość pojazdu) dobierane są śmigła.

3. Po wybraniu silników i śmigieł wybierane są ESC. Ponownie w arkuszu danych silników bezszczotkowych można znaleźć prąd i napięcie wymagane w tych warunkach pracy. Pozwala to na wybór ESC. Zaleca się, aby ESC zużywały o 30% więcej prądu w porównaniu do prądu zużywanego przez ich odpowiednie silniki, aby przezwyciężyć straty mocy.

4. Przy poprzednim punkcie i włączeniu innych elementów, takich jak autopilot i silniki pomocnicze, dobierany jest odpowiedni akumulator lub źródło zasilania.

5. Biorąc pod uwagę gabaryty wszystkich elementów oraz masę końcową (którą można obliczyć korzystając z indywidualnych kart katalogowych dla każdego elementu pojazdu) dobierana jest rama.

6. Biorąc pod uwagę ostateczną wagę, musisz wrócić do kroku 1 i sprawdzić, czy silniki nadal spełniają Twoje wymagania. Jeśli nie, musisz powtórzyć proces projektowania.

Wybór pilota zdalnego sterowania

Zdalne sterowanie jest bezpośrednim połączeniem z pilotem, zakładając, że potrzebujesz ręcznej interwencji lub przynajmniej kontrolowanego przez człowieka zatrzymania awaryjnego. Proces selekcji przebiega następująco:

1. Minimalna liczba kanałów do lotu quadkopterem to cztery (X, Y, Z i kąt odchylenia). W innych pojazdach, na przykład w samochodzie, są tylko dwa (jazda do przodu i skręcanie). Musisz jednak wziąć pod uwagę, czy potrzebujesz również zatrzymania awaryjnego, przycisku, który może być używany do otwierania i zamykania zacisku. Każde z tych zadań wymaga przynajmniej własnego kanału pomocniczego. W związku z tym stosowane są sześcio-, ośmio- lub dziesięciokanałowe sterowanie radiowe. Te dodatkowe kanały mogą być drążkami, przyciskami lub pokrętłami i mogą być używane przez kombinatoryczną logikę i stany aktywacji (aby dowiedzieć się więcej o maszynach stanów, zobacz moją książkę o Ardupilot i inne odniesienia zawarte w Dodatku).

2. Gdy już wiesz, ile kanałów potrzebujesz, następną rzeczą jest określenie maksymalnej odległości, jaką pilot pozwoli od operatora do pojazdu. To określi obszar działania, w którym pojazd będzie oddziaływał z człowiekiem, a to wpłynie na to, czy dron będzie możliwy do odzyskania w sytuacji awaryjnej.

3. Poniższe są subiektywnymi szczegółami obsługi i polegają na zadaniu sobie pytania, czy potrzebny jest monitor kamery na pilocie, czy chcesz obsługiwać więcej niż jeden pojazd tym samym RC, i tak dalej.

4. Następnym krokiem jest ustalenie, jak długo potrzebujesz korzystać z pojazdu, ponieważ wpłynie to na żywotność baterii w pilocie. Dzięki temu piloty ze złączami baterii LIPO polecane są na długie okresy pracy.

5. Na koniec musisz sprawdzić, czy autopilot i odbiornik zdalnego sterowania mają porty PPM. W przeciwnym razie musisz kupić odpowiednie adaptery.

Wybór autopilota

Wybór autopilota będzie zależał od Twoich umiejętności programowania, a także od stopnia zanurzenia w pojeździe.

1. Jeśli nie musisz programować specjalistycznych zadań, wystarczy kupić drona generycznego.

2. Jeśli wymagane są bardziej specjalistyczne zadania, ale nie wiesz, jak programować, zaleca się zakup autopilota, który umożliwia korzystanie z GUI, takiego jak Mission Planner. Zalecanym autopilotem jest tutaj Pixhawk.

3. Jeśli umiesz programować i chcesz projektować własne zadania bez koncentrowania się na projektowaniu pojazdu, zaleca się skorzystanie z uproszczonego SDK, takiego jak Dronekit. Ten SDK wymaga znajomości programowania w języku Python, ale pozwala na użycie pojazdu jako cząstki poruszającej się w osiach X, Y, Z i obracającej się wokół własnej osi. Polecany jest również Pixhawk.

4. Jeśli (oprócz poprzedniego punktu) chcesz po prostu zaprojektować lub przetestować prawa kontrolne w wstępnie zaprojektowanych pojazdach, konieczne jest użycie rozszerzonego SDK. Ten rodzaj SDK pozwala na prowadzenie pojazdu w trybie cząstek, ale poprzez oddziaływanie siły i momentu obrotowego, a nie tylko definiowanie trajektorii. Przykładem jest Bebop Autonomy SDK lub biblioteki Simulink dla autopilota Pixhawk.

5. Jeśli oprócz punktu 4 projektujesz pojazd eksperymentalny lub pojazd, który nie mieści się w standardowych modelach, które są kompatybilne z rozszerzonymi SDK, potrzebujesz autopilota kompatybilnego z pełnym SDK. Przykładami tego typu SDK są biblioteki Ardupilot i biblioteki PX4. Ponownie można tutaj użyć autopilota Pixhawk.

6. Jeżeli oprócz tego, co opisano w punkcie 5, musisz wykonywać inne specjalistyczne czynności, takie jak używanie sztucznego widzenia, zaleca się korzystanie z komputera towarzyszącego wraz z autopilotem. Na przykład możesz użyć Raspberry

Pi, Odroid lub laptop z ROS do wykonywania algorytmu sztucznego widzenia w połączeniu z Pixhawkiem realizującym algorytm lotu. Lub, jeszcze lepiej, możesz użyć wbudowanej jednostki, która łączy autopilota i komputer towarzyszący, taki jak karty NavIO lub ErleBrain.

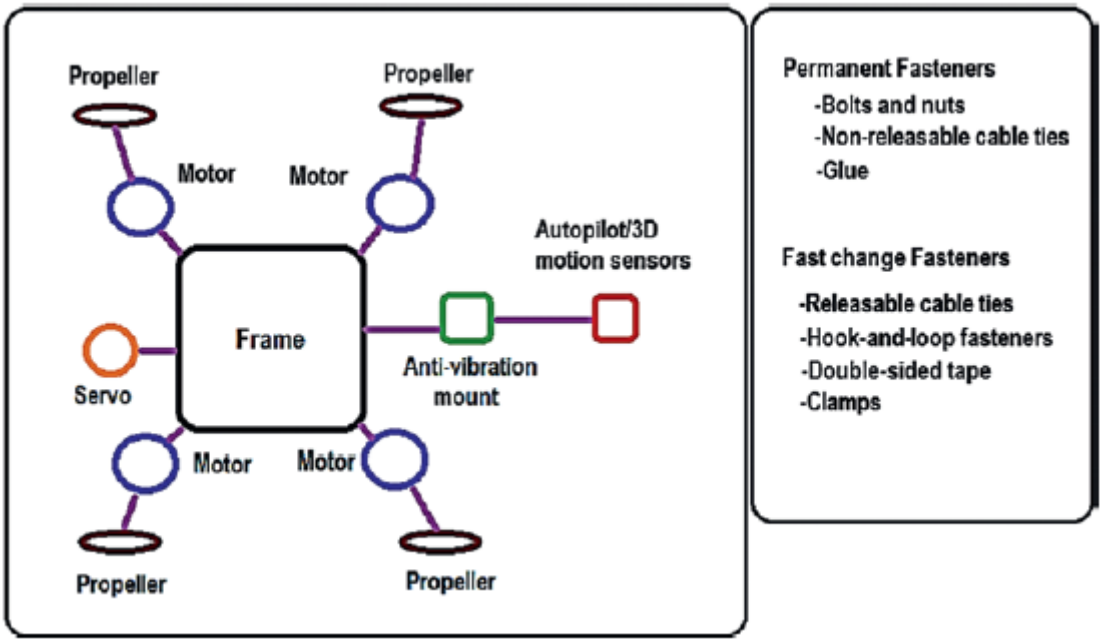
Zauważ, że przejście do przodu przez proces selekcji obejmuje również przejście od użytkownika hobbystycznego do naukowego, co wymaga głębszego zrozumienia programowania, sterowania i modelowania matematycznego, ale w zamian będziesz mieć możliwość całkowitego projektu, a nie tylko ograniczeń wyboru kilku parametrów. Dowiedziałeś się o komponentach i jak je dobierać. Czas nauczyć się je składać.

Połączenie komponentów

Ta sekcja ilustruje sposób łączenia komponentów drona. Pomoże to uniknąć bardzo typowych błędów montażowych na poziomie mechanicznym, elektrycznym i sterowania.

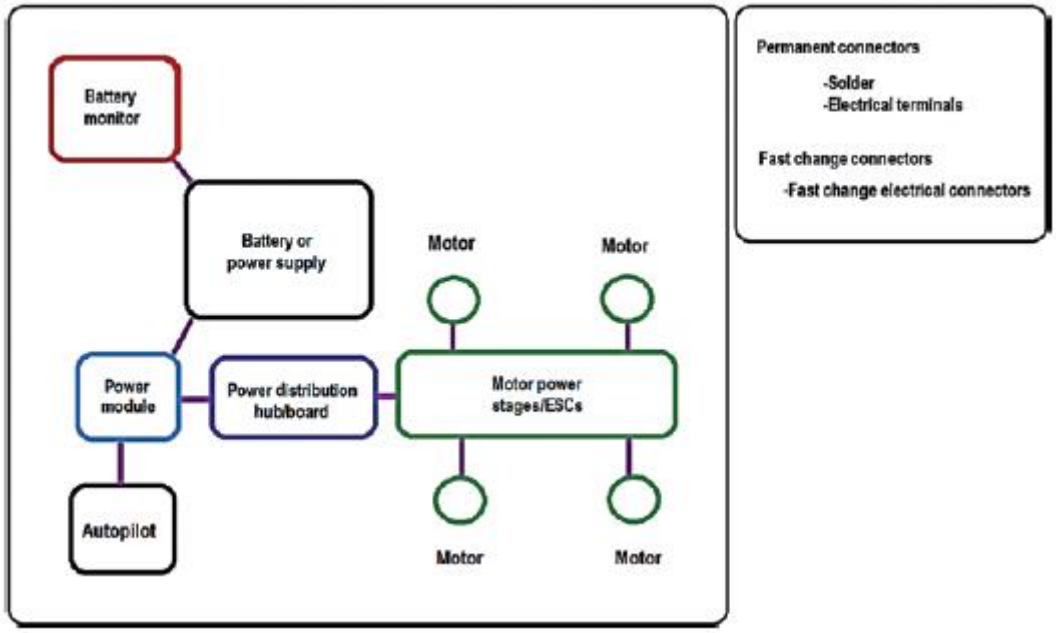
Połączenie mechaniczne

Głównym elementem połączenia mechanicznego jest rama, do której mocowane są pozostałe elementy. Zobacz rysunek



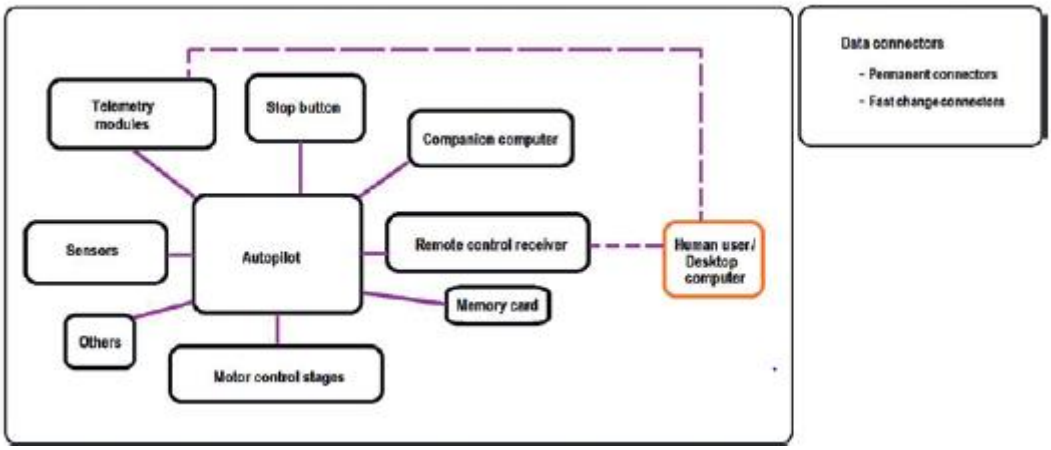
Połączenie elektryczne

Centralnym elementem jest tutaj bateria lub zasilacz. Zauważ, że silniki nie są bezpośrednio podłączone do tego zasilacza.



Połączenie kontrolne

Centralnym elementem jest tutaj autopilot. Na tym poziomie znajduje się również interfejs użytkownika lub interfejs z komputerem pomocniczym umieszczonym na ziemi.

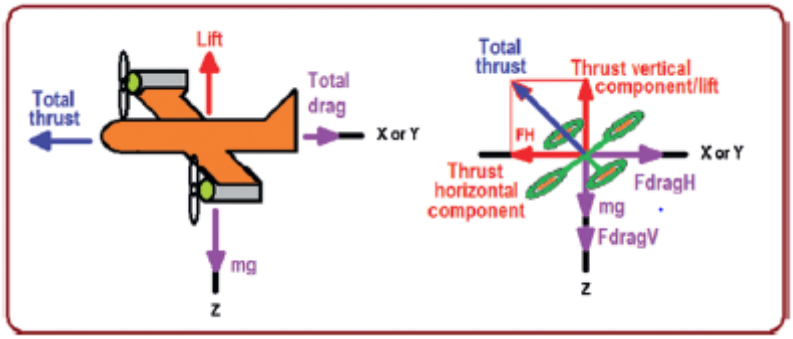


Streszczenie

Poznałeś komponenty do projektowania dronów ze szczególnym uwzględnieniem quadkopterów, ale z możliwością rozszerzenia na inne typy samolotów, w tym ich kontekst historyczny, etymologie i najczęstsze wyniki wyszukiwania, aby wybrać nazwę swojej pracy. Poznałeś również rodzaje dronów, które są przydatne w twoich działaniach, niektóre ogólne środki bezpieczeństwa, a także najczęściej używany międzynarodowy standard modelowania i projektowania drona i jego komponentów. Jeśli chodzi o komponenty, poznałeś kilka przydatnych klasyfikacji, ich najczęstsze cechy i procedury oraz ich połączenia i dobór.

Modelowanie

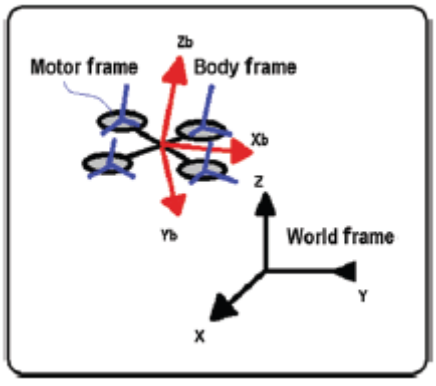
Ta część skupia się na modelowaniu dronów. Poznasz układy odniesienia, kinematykę, dynamikę, modele napędu lub alokacji oraz zagadnienia związane z modelowaniem liniowym. Dzięki temu będziesz mógł zamodelować drona i rozszerzyć tę wiedzę na inne typy pojazdów i samolotów. Jeśli chodzi o aerodynamikę, nie będziemy omawiać tych tematów, ponieważ konstrukcje pojazdów analizowanych to głównie pionowe starty i charakteryzują się małymi prędkościami, niewielką wagą i/lub małymi obszarami działania w porównaniu z innymi typami pojazdów (samoloty). i załogowych statków powietrznych, na przykład). Zobacz Rysunek.



Możesz zauważyć, że multikopter wymaga elementów ciągu, aby osiągnąć swój ruch (nazywa się to wektorowaniem ciągu). Z tego powodu w Dodatku podzieliliśmy analizę aerodynamiczną na dwie przydatne sekcje. Jedna z nich dotyczy obliczania maksymalnych prędkości dronów i może służyć do projektowania dronów wyścigowych. Drugi dotyczy tego, jak rozszerzyć wiedzę z tej książki na inne typy samolotów. W tych sekcjach wykorzystasz wiedzę zdobytą tu i będziesz rozszerzać go, wprowadzając efekty aerodynamiczne. Elementami, które ze względu na swoje prędkości obrotowe mają duży wpływ na aerodynamikę, są śmigła. Nie będą one jednak tutaj brane pod uwagę, ponieważ zakłada się, że nie wyprodukujesz własnych śmigieł, a raczej je kupisz(ale w razie potrzeby możesz przeczytać więcej o ich konstrukcji w sekcji z odnośnikami w Dodatku). Zakłada się również, że przez większość czasu będziesz latać swoim samolotem w znacznej odległości od podłogi lub ścian, a nawet innych obiektów i pojazdów; w przeciwnym razie możesz przeczytać powiązane informacje o ograniczeniach w Dodatku. Zacznijmy od opisu ram modelowania.

Ramy modelowania

Jak zaraz zobaczycie, z quadkopterem związane są trzy ramki modelowania. Modelowanie wspomnianego pojazdu jest opracowywane w ramach mieszanych i odbywa się z wygodą czujników i operacji matematycznych. Ramy te to rama światowa, rama pojazdu i rama silnika, które przedstawiono na rysunku.



• Rama globalna (lub podstawa, rama stała lub rama bezwładnościowa) wskazuje stały układ współrzędnych. Pojazd porusza się względem tego ustalonego odniesienia. Może to być globalne odniesienie geograficzne, takie jak biegun ziemi, prosty narożnik lub środek pokoju.

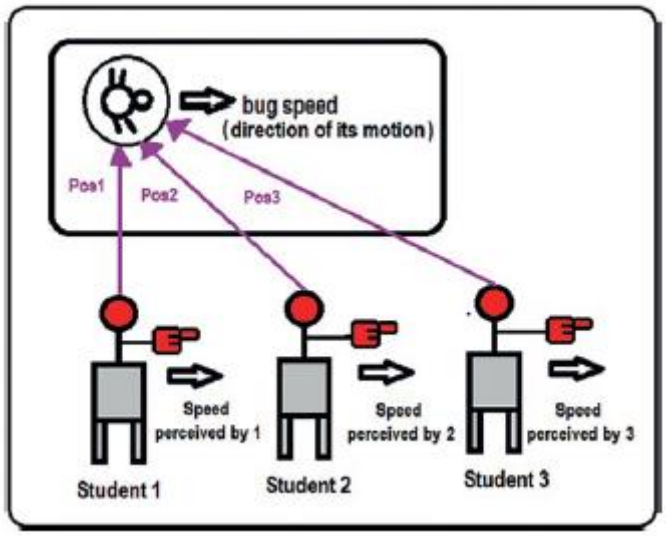
• Rama pojazdu (lub rama nadwozia lub rama ruchoma lub nieinercyjna) wskazuje układ współrzędnych, który jest zwykle umieszczany w interesującym punkcie na pokładzie pojazdu (środek geometryczny lub środek masy, lub grawitacja lub pływalność m.in. To w odniesieniu do tej ramy i ramy nieruchomej mierzone są przesunięcia i obroty.

• Wreszcie, rama silników (lub rama napędowa, siłownik lub rama alokacji) jest każdą z ram umieszczonych na każdym silniku. Pełnią funkcję rozdziału siły i momentu obrotowego każdego silnika na ramę odniesienia pojazdu.

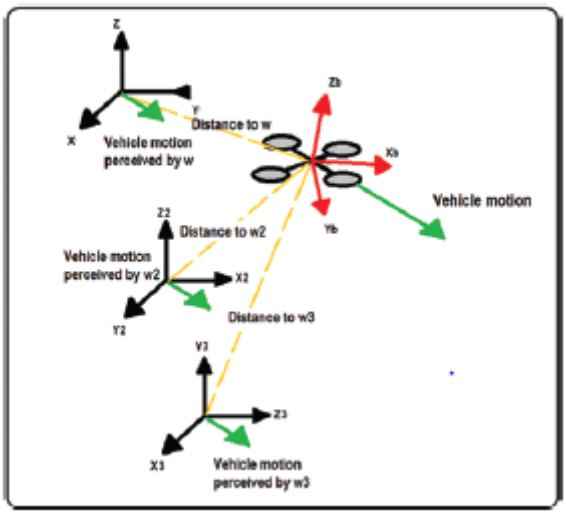
Zwróć uwagę, że aby maksymalnie ułatwić obliczenia, osie Z odpowiadające odpowiednio światu i ramom pojazdu są zwykle umieszczane równolegle. Dzieje się tak dlatego, że pozwala na to konstrukcja quadkoptera oraz, jak zobaczycie później, z uwagi na płynność lotu (takie uproszczenie w doborze wspomnianych osi nie jest możliwe w pojazdach o bardziej zaawansowanej mobilności, np. typu dookólnego). Czas przejść do kinematyki drona. Najpierw rozważymy kinematykę translacyjną.

Kinematyka translacyjna

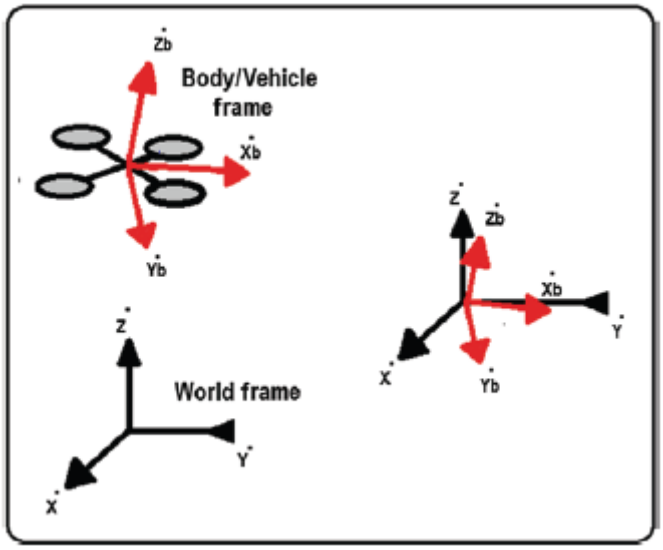
Załóżmy, że chcemy skojarzyć ramę świata z ramą pojazdu. Choć na pierwszy rzut oka wydaje się to proste, należy pamiętać, że wszelkie skojarzenia pojęć powinny odbywać się w bezstronnej przestrzeni. Aby to zrobić, wyobraź sobie, że uczniowie w klasie są proszeni o wskazanie pozycji owada chodzącego po tablicy w odniesieniu do ich pozycji, jak pokazano na rysunku.



Jak widać, każdy będzie miał inne pojęcie o insekcie i dadzą wynik zależny od miejsca, w którym zostaną umieszczone. Jednak każdy, bez względu na swoją pozycję, może wskazać, że owad porusza się we wskazanym kierunku. W ten sposób bezstronnym elementem dla ludzi w tej klasie jest ruch owada, a ten ruch jest powiązany z jego prędkością. Sugeruje to, że jeśli rama świata jest umieszczona w różnych pozycjach względem pojazdu, jak widać na Rysunku, niezmiennicza relacja między ramami nie byłaby na poziomie pozycji, ale na poziomie prędkości (jako wektor). , i dotyczy to obserwatora lub czujnika umieszczonego w każdej ramce świata.



Na poprzednich rysunkach widać, że percepcja ruchu pojazdu w każdej klatce nie jest modyfikowana przez odległość między klatkami. Jednak nakładając dwa z nich, jak na rysunku, można zauważyć, że istnieje wpływ orientacji w ich odpowiednich osiach.



Dlatego, aby je powiązać, konieczne jest znalezienie rzutu trygonometrycznego pomiędzy odpowiednimi osiami układów współrzędnych. To w trybie trójwymiarowym sprowadza się do znalezienia macierzy rotacji (zwróć uwagę na użycie notacji Newtona, która jest literą lub symbolem ozdobionym kropką; implikuje to pochodną pierwszego rzędu, a w naszym przypadku wskazuje, że jest to związek między ramkami prędkości lub prędkością, jak wspomniano wcześniej). Wśród rodziny macierzy rotacji i w celu spełnienia normy ISO 1151-2: 1985, w niniejszym tekście wykorzystana zostanie sekwencja kąta Eulera przechyłu, pochylenia i odchylenia, znana jako Tait-Bryan. To jest

$$R = R_{z_\psi} R_{x_\phi} R_{y_\theta}$$

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + s\phi s\psi c\theta \\ s\psi c\theta + s\phi c\psi s\theta & c\phi c\psi & s\psi s\theta - s\phi c\psi c\theta \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix}$$

Pamiętaj, że poszczególne obroty są zdefiniowane jako (zwróć uwagę, że oś obrotu jest osią obrotu)

$$R_{x_\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \quad R_{z_\psi} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{y_\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Dobre pytanie brzmi: Dlaczego ta sekwencja obrotów jest mnożeniem, a nie sumą? A odpowiedź, przynajmniej z punktu widzenia zgodności matematycznej, jest prosta. Każda macierz obrotu musi być ortonormalna, ponieważ każda z jej kolumn reprezentuje wektory jednostkowe, których składowe wahają się od 0 do 1 i opisują osie prostopadłe (aby zweryfikować tę właściwość, zapoznaj się z zalecanymi tekstami dotyczącymi algebry liniowej i macierzy obrotu), więc jeśli je dodamy, istnieje ryzyko, że otrzymane elementy przekroczą wartości jednostkowe, natomiast jeśli je pomnożymy, ponieważ są to liczby ułamkowe, zera lub jedynki, wynikiem będą nadal liczby ułamkowe, zera lub jedynki. Dlatego złożenie kolejnych sekwencji macierzy rotacji musi być multiplikatywne. Zauważ, że zestaw równań, którego używamy, jest ważny na poziomie prędkości, a także na poziomie siły. Stwierdzenie to zostanie użyte później, a jego demonstrację i podstawę dla najbardziej wymagających czytelników można znaleźć w wielu tekstach jako zastosowanie zasady D'Alemberta i zasady pracy wirtualnej, zwanej również zasadą najmniejszego działania. . Teraz poznasz kinematykę obrotową drona.

Kinematyka obrotowa

Uzyskanie kinematyki translacyjnej było stosunkowo łatwe, biorąc pod uwagę istnienie układów odniesienia bezpośrednio połączonych obrotem. Nie dotyczy to jednak zmiennych obrotowych (zauważ, że mamy informację kątową dotyczącą układu świata podaną przez kąty Eulera, ale nie mamy informacji kątowych bezpośrednio w nadwoziu lub ramie pojazdu). Aby wydedukować kinematykę, konieczne staje się użycie matematycznego własności macierzy rotacji. Macierze rotacji muszą spełniać zasadę ortonormalności. Oznacza to, co następuje:

$$RR^T = I$$

Jak wskazano, najbardziej obiektywna analiza w celu ustalenia związku między klatkami dotyczy poziomu prędkości. Prędkości iw konsekwencji prędkość są otrzymywane z pochodnych pozycyjnych, więc pierwszą intuicją jest wyprowadzenie poprzedniego równania. Należy zauważyć, że jest to

pochodna w odniesieniu do czasu, więc jeśli macierz rotacji zależy od kątów Eulera podczas wyprowadzania, pojawią się wartości kątowe wspomnianych kątów Eulera (reguła łańcucha). Zauważ też, że z regułami pochodnej iloczynu (iloczyn macierzy) i pochodnej stałej (macierz jednostkowa) otrzymujemy następujący wynik:

$$\dot{R}R^T + R\dot{R}^T = 0$$

Korzystając z następującej właściwości transponowanych macierzy w drugim członie poprzedniego równania

$$(AB)^T = B^T A^T$$
$$(\dot{R}R^T)^T = R\dot{R}^T$$

i zamieniając tę nieruchomość, otrzymujemy

$$\dot{R}R^T + R\dot{R}^T = \dot{R}R^T + (\dot{R}R^T)^T = 0$$

Z tego wynika specjalny typ macierzy nazywany antysymetryczną lub skośno-symetryczną. Spełniają następujące warunki:

$$S + S^T = 0$$

Dlatego prawdą jest, że następujący wyraz jest macierzą skośno-symetryczną:

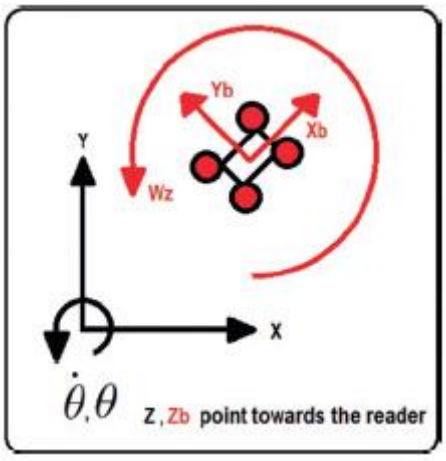
$$S = \dot{R}R^T$$

A do czego to jest przydatne? Macierz skośno-symetryczna jest powiązana z reprezentacją iloczynu krzyżowego. Załóżmy, że mamy dwa wektory:

$$a = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$$
$$b = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$
$$a \times b = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = S(a)b$$

W ten sposób macierz antysymetryczna (w tym przypadku skojarzona z wektorem a) zawiera informację o wektorze rozłożonym w swoich elementach we wskazany sposób. Pytanie o przydatność równania pozostaje otwarte i teraz odpowiemy na nie. Pamiętaj tylko o koncepcji macierzy skośno-symetrycznej i jej związku z iloczynem krzyżowym.

Załóżmy, że samochód na rysunku obraca się tylko wokół swojej osi Zb z prędkością kątową równą pochodnej teta. Zauważ, że oś Z ramy świata i ramy ciała (Zb) są równoległe (obie są skierowane w twoją stronę).



Dla tego przykładu jest oczywiste, że prędkość kątowa w osi Z ramy ciała jest równa pochodnej theta mierzonej w nieruchomej ramie (znowu, ponieważ są to osie równoległe).

$$\omega_z = \dot{\theta}$$

Tak więc możemy osiągnąć ten sam wynik z równaniem skośno-symetrycznym:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \dot{R}R^T = \begin{bmatrix} -\dot{\theta}\sin\theta & -\dot{\theta}\cos\theta & 0 \\ \dot{\theta}\cos\theta & -\dot{\theta}\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -\dot{\theta} & 0 \\ \dot{\theta} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Z definicji elementów macierzy skośno-symetrycznej możemy wywnioskować, że

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \dot{R}R^T$$

Następnie, używając złożonej macierzy rotacji Eulera i wykonując poprzednią procedurę, jak również odpowiadające jej uproszczenia, otrzymujemy następującą (ciekawe ćwiczenie, że rozwijasz pełną sekwencję tych równań; pamiętaj, że pochodna macierzy rotacji jest po czasie, a także pamiętaj o koncepcji różniczki całkowitej):

$$R = R_{z_\psi} R_{x_\phi} R_{y_\theta}$$

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + s\phi s\psi c\theta \\ s\psi c\theta + s\phi c\psi s\theta & c\phi c\psi & s\psi s\theta - s\phi c\psi c\theta \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

$$\omega = \begin{bmatrix} \omega_{xB} \\ \omega_{yB} \\ \omega_{zB} \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

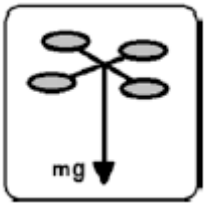
gdzie

$$\dot{R} = \frac{dR}{d\theta}\dot{\theta} + \frac{dR}{d\phi}\dot{\phi} + \frac{dR}{d\psi}\dot{\psi}$$

Dotyczy to prędkości kątowych ramy nadwozia ze stałą ramą prędkości kątowej. Pamiętaj, że rama stała jest wyrażona w kątach Eulera. Dla tych, którzy wolą geometrię od opracowywania równań, istnieje również zrozumiała i wizualna dedukcja dotycząca uzyskania tego związku między prędkościami w książce Beedforda. Należy również zauważyć, że nałożenie wspomnianych ograniczeń ortonormalności kątowej daje dronom coś, co jest znane jako właściwość nieholonomiczna (jest to niezauważalne w trybach lotu płynnego lub liniowego, jak zobaczycie wkrótce). Na koniec zauważmy, że wprowadzono nomenklaturę (pqr). Jest to często używane w samolotach i oczywiście multikopterach. Teraz, gdy znasz już kinematykę drona, nadszedł czas, aby przyjrzeć się dynamice, ale wcześniej przedstawiamy krótki opis sił działających na drona i jego śmigła.

Siły działające na multikopter i jego śmigła

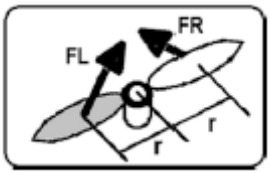
Przed opracowaniem równań dynamicznych wygodnie jest wydedukować podstawowe siły działające na multikopter. Można dodać efekty aerodynamiczne, ale ponieważ różnią się one między operacjami na zewnątrz i wewnątrz, nawet z bliskiej lub dużej odległości od ziemi, są uważane za materiał nieogólny, który można wykorzystać w innych pracach. Najbardziej oczywistą siłą działającą na drona jest grawitacja. Załóżmy, że pojazd jest dobrze wyważony, a wspomniana siła znajduje się bezpośrednio w środku ciężkości lub w środku geometrii, co w przypadku tego typu pojazdu jest pożądane, aby się pokrywać.



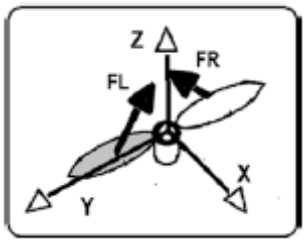
Siły działające na śmigło

Następnym krokiem jest analiza sił działających na pojedyncze śmigło. Należy zauważyć, że ta analiza jest przeprowadzana dla śmigieł dwułopatowych, ale można ją rozszerzyć o więcej łopat z odpowiednimi rzutami trygonometrycznymi. Pierwszą rzeczą, którą bierze się pod uwagę, jest to, że siła nacisku na każde ostrze może być równoważna sile przyłożonej do ostrza wymienionego ostrza. Na rysunku są one oznaczone FL (lub po lewej) i FR (lub po prawej). Drugą rzeczą, którą należy wziąć pod

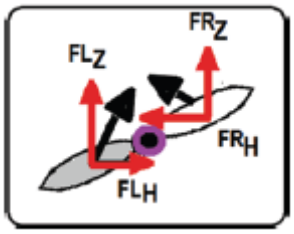
uwagę, jest to, że śmigło jest dobrze wyważone, co oznacza, że odległość między środkiem śmigła a punktem analizy na każdej łopatce jest taka sama (promień r).



Wykorzystamy również ramę śmigła pokazaną na rysunku



Następnym krokiem jest rozłożenie sił działających na każdą łopatę na ich składowe pionowe i poziome (etykiety Z i H). Patrz rysunek



Następnie wykonujemy sumę sił i momentów w trzech odpowiednich osiach (X, Y i Z). Ponieważ w każdej łopatce przyjęto założenia dotyczące symetrii i wyważenia, możliwe jest wykonanie następujących równości:

$$FR = FL$$

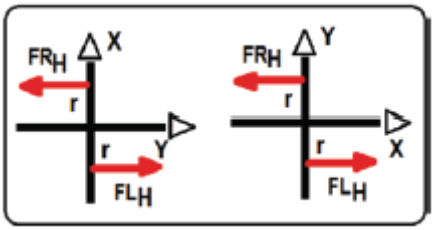
$$FR_Z = FL_Z = F_Z$$

$$FR_H = FL_H = F_H$$

Z poprzedniego rysunku pionowa analiza sił jest prosta. Zwróć uwagę na fakt, że użycie większej liczby lemieszy zwiększa ładowność pojazdu (zgodnie z oczekiwaniami każdy lemiesz działa jak oś podporowa).

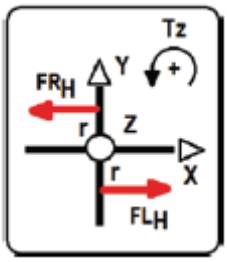
$$\sum F_z = FL_z + FR_z = 2F_z$$

Analiza w X Y jest taka sama, a do tego wykorzystamy symetrię promieniową śmigła (ta analiza z użyciem nieparzystych łopatek staje się bardzo interesująca, ponieważ wymaga rzutów trygonometrycznych, ale wynik końcowy musi być taki sam). Załóżmy, że śmigło znajduje się na chwilę na osi X lub Y, jak pokazano na rysunku



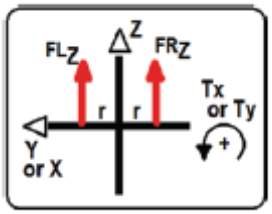
$$\sum F_X = \sum F_Y = FL_H + FR_H = F_H - F_H = 0$$

W związku z tym pokazano, że dopóki śmigło jest odpowiednio wyważone, siły poziome generowane przez każdą łopatę zostaną zniesione. Jednak, jak zauważyłeś, istnieje również obecność promieni działania, a co za tym idzie momentów obrotowych. Analizuje się to w następujący sposób. W związku z tym pokazano, że dopóki śmigło jest odpowiednio wyważone, siły poziome generowane przez każdą łopatę zostaną zniesione. Jednak, jak zauważyłeś, istnieje również obecność promieni działania, a co za tym idzie momentów obrotowych. Analizuje się to w następujący sposób. Wróćmy do poprzedniego rysunku i dodając fakt, że oś Z jest skierowana w Twoją stronę, moment obrotowy wokół tej osi jest taki, jak pokazano na rysunku



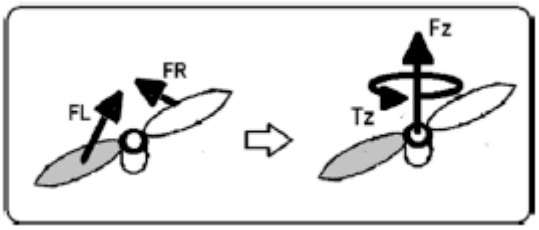
$$\sum \tau_Z = r(FR_H + FL_H) = 2rF_H$$

Należy zauważyć, że zwiększenie liczby łopatek zwiększa również zdolność przenoszenia momentu obrotowego śruby napędowej (każda łopatka działa jak ramię dźwigni). I na koniec określimy moment poziomy (wokół osi X lub Y; nie ma różnicy).



$$\sum \tau_X = \sum \tau_Y = r(FR_Z + FL_Z) = r(F_Z - F_Z) = 0$$

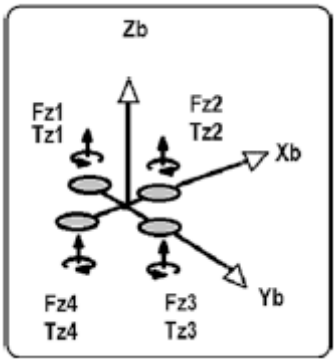
Najważniejszy wniosek z poprzedniej analizy jest następny. Chociaż śmigła są dobrze wyważone, mogą generować jedynie pionową siłę ciągu na swojej osi działania i moment obrotowy wokół tej samej osi.



Ciekawym ćwiczeniem dla ciebie jest wydedukowanie tego samego efektu dla śmigła z trzema łopatami rozmieszczonymi pod kątem 120 stopni każda. Ostateczny wynik musi być taki sam.

Siły działające na pojazd

Na podstawie poprzedniego wniosku przeniesiemy siły każdego śmigła na środek pojazdu.



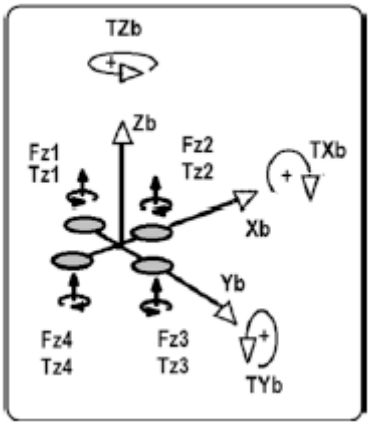
Dla tego przykładu i dla oczekiwanych rezultatów zrobimy to jak w przypadku rysunku. Zwróć uwagę na fakt, że oznakowanie i kierunek osi kartezjańskich, a także numeracja silników i ich kierunek obrotu, a nawet ich rozkład geometryczny, wpływają na to, co zobaczymy i jest znane jako macierz alokacji lub macierz napędu . Zauważ też, że połowa silników obraca się zgodnie z ruchem wskazówek zegara względem drugiej połowy, aby zapobiec autorotacji (co oznacza, że pojazd obraca się w niepożądany sposób wokół swojej głównej osi). Może się tak zdarzyć, jeśli wszystkie silniki lub większość z nich obraca się w jednym kierunku (zgodnie lub przeciwnie do ruchu wskazówek zegara). Analizując siły na osi Zb mamy

$$\sum F_{ZB} = F_{Z1} + F_{Z2} + F_{Z3} + F_{Z4}$$

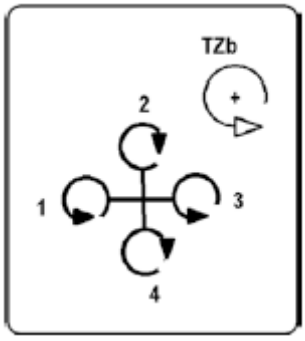
Biorąc pod uwagę dobrze wyważony pojazd z dobrze wyważonymi śmigłami, w osiach Yb i Xb nie występują siły działające bezpośrednio na nie, a jedynie momenty obrotowe,

$$\sum F_{XB} = \sum F_{YB} = 0$$

Zauważ, że istnieją warunki, w których te siły są obecne i nie można ich usunąć z modelu. Jednak efekty te zwykle pojawiają się w dużych samolotach i można je uprościć za pomocą przekształceń matematycznych lub kompensacji sterowania. Kontynuujmy z momentami obrotowymi. Pierwszą rzeczą do zrobienia jest ustalenie pozytywnego poczucia rotacji, jak pokazano na rysunku



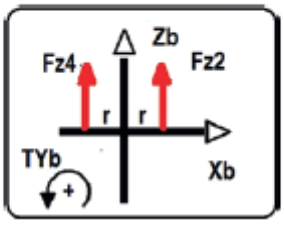
Należy zauważyć, że w przypadku momentu obrotowego wokół osi Zb jest on bezpośrednio indukowany przez poszczególne momenty obrotowe każdego silnika.

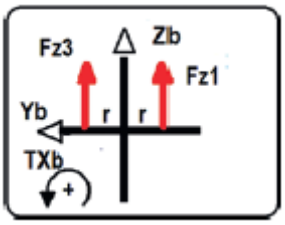


$$\sum T_{ZB} = T_{Z1} - T_{Z2} + T_{Z3} - T_{Z4}$$

W przypadku osi Yb i Xb momenty są indukowane jako efekt ramienia dźwigni wywołany siłami nacisku i ich odpowiednimi promieniami. Należy pamiętać, że pojazd musi być dobrze wyważony, aby zapewnić identyczne lub bardzo podobne promienie lub ramiona dźwigni

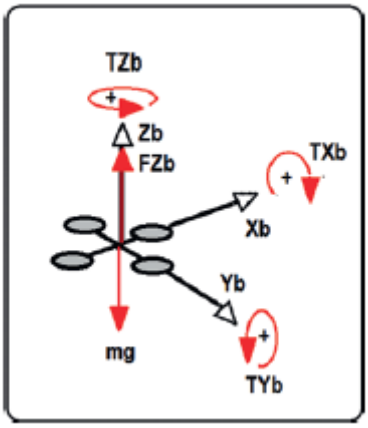
$$\sum T_{YB} = r(F_{Z2} - F_{Z4})$$





$$\sum T_{XB} = r(F_{Z1} - F_{Z3})$$

Ważnym wnioskiem jest to, że quadkopter ma w swoim środku ciężkości, zbieżnym z geometrycznym środkiem, jeden ciąg i trzy momenty obrotowe generowane przez jego silniki oraz działanie siły grawitacji. Zauważ, że siła grawitacji jest zawsze prostopadła do podłoża lub ramy świata, ale nie jest to przypadek FZb (ciąg drona), ponieważ FZb jest prostopadły do korpusu drona lub ramy pojazdu. Niedługo wykorzystamy odpowiednie relacje między tymi kadrami.



Po zapoznaniu się z siłami działającymi na pojazd nadszedł czas na zbadanie jego dynamiki.

Dynamika translacyjna

Po ustaleniu ruchu lub zależności kinematycznych między ramami oraz sił działających na quadkopter wyprowadzimy równania, które wytwarzają te ruchy. W tym przypadku zaczniemy od dynamiki translacyjnej. W tym celu wygodnie jest zadać sobie pytanie, gdzie i dlaczego przeprowadzać tę analizę. Dynamika translacyjna jest na ogół odejmowana w nieruchomej ramce. Czemu? Tylko dla matematycznego uproszczenia i dla kompatybilności z czujnikami (większość czujników translacyjnych wskazuje pozycję drona względem nieruchomej ramy jak GPS). Procedura wygląda następująco:

1. Wykonano założenia dotyczące symetrii i masy punktowej (dron typu quadkopter jest zaprojektowany w sposób zrównoważony i generalnie jego środek ciężkości pokrywa się z jego środkiem geometrycznym lub przynajmniej znajduje się jak najbliżej).

2. Równanie drugiego prawa Newtona jest stosowane dla samolotu, ponieważ była to cząstka o ruchu trójwymiarowym (to jest celem założeń kroku 1).

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

$$\vec{F} = m\vec{a}$$

3. Ponieważ interesuje nas sterowanie ruchem samolotu (modelowanego jako cząstka), przepisujemy poprzednie równanie względem położenia (przyspieszenie definiujemy jako drugą pochodną położenia).

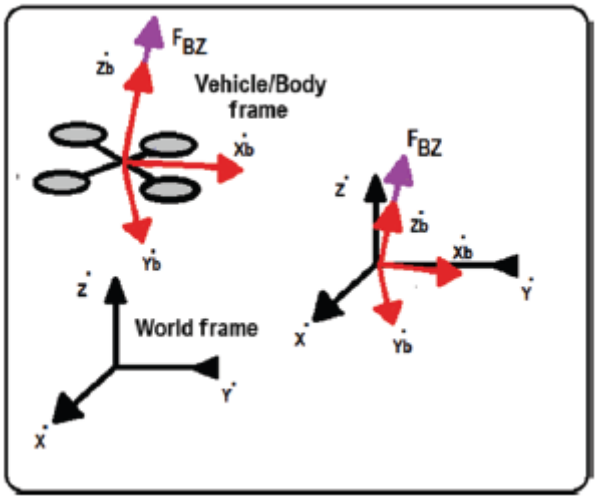
$$\vec{a} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \qquad \vec{F} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

4. Dodajemy efekt swobodnego spadania (grawitacji).

$$\vec{F} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

5. Ponieważ chcemy modelować drona w nieruchomej ramie, a siła ciągu drona działa na ramę ciała, musimy je powiązać. W tym przypadku, jako rozszerzenie zasady D'Alemberta, warto wskazać, co następuje. Pamiętaj, że macierz rotacji to kompozycja podana przez rotacje Taita-Bryana.

$$\vec{F} = R \begin{bmatrix} 0 \\ 0 \\ F_{BZ} \end{bmatrix}$$



6. Zatem układ równań modelujących dynamikę translacyjną to:

$$R\begin{bmatrix}0\\0\\F_{BZ}\end{bmatrix}+m\begin{bmatrix}0\\0\\-g\end{bmatrix}=m\begin{bmatrix}\ddot{x}\\\ddot{y}\\\ddot{z}\end{bmatrix}$$

Zwróć uwagę, że jest to uproszczony model, ale możesz dodać efekty, takie jak tarcie powietrza, turbulencje i inne, w zależności od potrzeb Twojej pracy. Teraz, gdy znasz już dynamikę translacyjną, przejdźmy do dynamiki rotacyjnej.

Dynamika rotacyjna

W tej części użyjemy równań Eulera dla ruchu obrotowego. Są one odpowiednikiem drugiego prawa Newtona dla ruchu obrotowego. W przeciwieństwie do dynamiki translacyjnej, analiza ta jest wyrażona w ramie ciała. Powodem jest ponownie uproszczenie równań i kompatybilność z czujnikami, które na ogół zapewniają pomiary obrotów na pokładzie. Procedura wygląda następująco:

1. Używamy rotacyjnego odpowiednika drugiego prawa Newtona lub równań Eulera. Zauważ, że w przeciwieństwie do poprzedniego przypadku, J nie jest stałą jak masa, ale jest macierzą 3x3, która po pomnożeniu przez przyspieszenia kątowe daje wektor 3x1.

$$\vec{F}=m\vec{a}\qquad\vec{\tau}=J\vec{\alpha}$$
$$\vec{F}_{3\times1}=m_{1\times1}\vec{a}_{3\times1}$$
$$\vec{\tau}_{3\times1}=J_{3\times3}\vec{\alpha}_{3\times1}$$

2. Przepisujemy w kategoriach czegoś znanego lub mierzalnego, takiego jak prędkość kątowa na pokładzie (zauważ, że analiza translacyjna jest wyrażona w postaci drugiej pochodnej pozycji). W tym przypadku pomocne jest, aby czujniki generalnie umożliwiały pomiar prędkości kątowych na pokładzie. Zauważ, że przyspieszenie jest również pierwszą pochodną prędkości.

$$\vec{\tau} = J\vec{\alpha} = J \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$$

3. Założenia dotyczące ciała sztywnego i symetrycznego zakładają, że macierz bezwładności ma składowe tylko na swojej przekątnej, co oznacza, że wszystko, co nie znajduje się na przekątnej, jest pomijalne.

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}$$

$$\vec{\tau} = \begin{bmatrix} J_x\dot{\omega}_x \\ J_y\dot{\omega}_y \\ J_z\dot{\omega}_z \end{bmatrix}$$

4. Pomija się efekt grawitacyjny, ponieważ biorąc pod uwagę, że dron jest symetryczny, sztywny i dobrze wyważony względem jego środka, momenty generowane przez siłę grawitacji znikają (promienie ramion ma tendencję do zera). Należy wziąć pod uwagę, że nie dzieje się tak w przypadku niewyważonego pojazdu lub drona o dużych gabarytach (np. drona przewożącego coś umieszczonego na krawędzi jego ramy jak ramię robota). Jednak efekty przyspieszeń odśrodkowych są zwykle ważne (w uproszczeniu płynnego lotu zauważysz, że te przyspieszenia są pomijane ze względu na zmniejszone prędkości kątowe).

$$\vec{\tau} = \begin{bmatrix} J_x\dot{\omega}_x \\ J_y\dot{\omega}_y \\ J_z\dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} J_x\omega_x \\ J_y\omega_y \\ J_z\omega_z \end{bmatrix}$$

Dlaczego te efekty odśrodkowe są pomijane podczas analizy translacyjnej? Ponieważ położenie siły ciągu jest prawie zerowe w stosunku do środka analizy pojazdu, a zatem są one pomijalne (ale nie w przypadku dużych pojazdów, gdzie coś podobnego dzieje się z obrotowym efektem grawitacji). Jak widać, w ruchu obrotowym są one brane pod uwagę, ponieważ nie zależą od promieni działania, ale od prędkości kątowych.

5. Rozwijając równania (iloczyn krzyżowy), dynamika obrotowa staje się

$$\begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix} = \begin{bmatrix} J_x\dot{\omega}_x \\ J_y\dot{\omega}_y \\ J_z\dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_y\omega_z J_z - \omega_y\omega_z J_y \\ \omega_x\omega_z J_x - \omega_x\omega_z J_z \\ \omega_x\omega_y J_y - \omega_x\omega_y J_x \end{bmatrix}$$

Ta dynamika zostanie uproszczona później dzięki płynnym warunkom lotu. Zostaną jednak ponownie wykorzystane w sekcji kontroli geometrycznej. Nauczyłeś się kinematyki i dynamiki. Aby zamknąć nasze modelowanie, porozmawiajmy o modelu alokacji.

Model alokacji

Model napędu lub alokacji określa związek między silnikami a środkiem analizy pojazdu (na przykład środkiem ciężkości). Jest to łącznik między teorią (modelowanie, symulacja i sterowanie) a praktyką (projektowanie i programowanie).

Kroki do uzyskania modelu alokacji

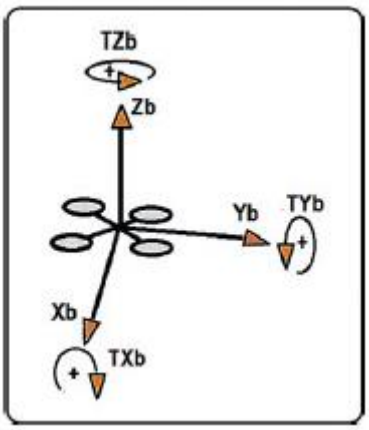
Procedura uzyskania tego modelu jest bardzo podobna do tej przeprowadzonej w części, w której zademonstrowano momenty i siły działające na quadkopter. Jednak powtarza się to w sposób systematyczny. Ogólnie oczekuje się następującej zależności (zakładając oczywiście liniowe zachowanie silników):

$$\begin{bmatrix} F_{XB} \\ F_{YB} \\ F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix}_{6\times 1} = [A]_{6\times m} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_m \end{bmatrix}_{m\times 1}$$

gdzie w, o indeksie od 1 do m, reprezentuje prędkość każdego z m silników pojazdu, a A jest macierzą alokacji, która wskazuje, w jaki sposób każdy silnik wpływa na zachowanie trzech możliwych momentów obrotowych i trzech możliwych sił w przestrzeni kartezjańskiej . Ta przestrzeń kartezjańska jest związana ze środkiem analizy wspomnianego samolotu (na przykład grawitacyjnym). W większości multikopterów macierz A ma elementy stałe, ale mogą się one zmieniać, jeśli na przykład każdy silnik ma system serw, który zmienia jego położenie i orientację.

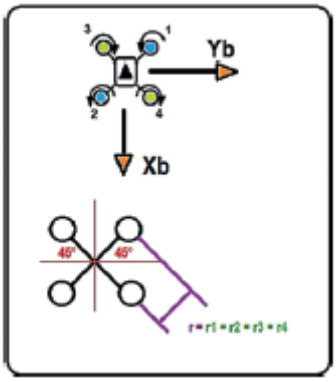
Zauważ, że wynik tego procesu modelowania zmienia się dla każdej konfiguracji drona, ale opisany proces jest ogólny i wygląda następująco:

1. Pierwszą rzeczą do zrobienia jest ustalenie konwencji osi, w tym dodatnich kierunków ruchu. Obejmuje to definiowanie zarówno tłumaczeń, jak i rotacji. W praktyce odbywa się to za pomocą drążków zdalnego sterowania, które często są oparte na wspomnianej normie ISO.



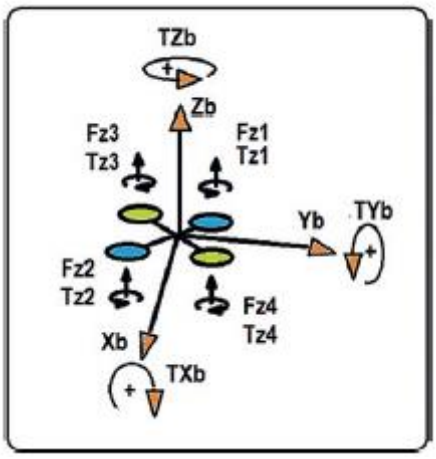
2. Kolejnym krokiem jest naklejenie etykiety na silniki. Wiąże się to z ich kierunkiem obrotu i rozkładem geometrycznym w stosunku do innych silników.

W tym przypadku, aby dać konkretny przykład, używamy płaskiej geometrii zwanej konfiguracją X, ze wskazanymi zwrotami, następującymi etykietami, a także z symetrią promieniową



Zauważ, że zwykle odbywa się to w odniesieniu do odniesienia autopilota (zwykle mała strzałka).

3. Teraz musimy powiązać ruchy ustalone w kroku 1 z efektami momentu obrotowego i ciągu każdego silnika. Pamiętaj, że istnieją momenty powstające w wyniku działania poszczególnych momentów obrotowych oraz momenty generowane przez ramiona dźwigni.

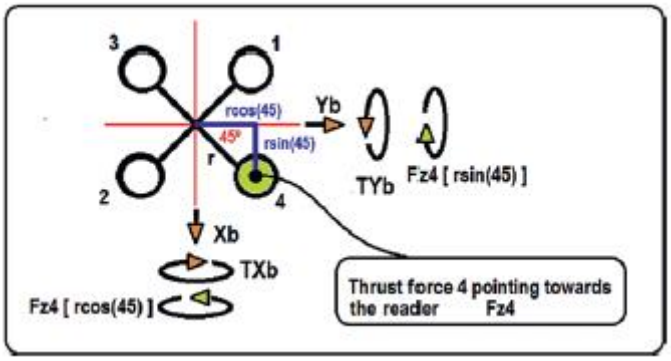


Model przydziału dla quadkoptera

Przeprowadzając procedurę podobną do tej z poprzedniej sekcji, dotyczącą sił działających na quadkopter, otrzymujemy następujące wyniki (pamiętaj, że każda konfiguracja samolotu jest inna, ale można ją przeanalizować na podstawie tego przykładu jako punktu wyjścia):

$$
\begin{aligned}
F_{ZB} &= F_{Z1} + F_{Z2} + F_{Z3} + F_{Z4} \\
F_{XB} &= F_{YB} = 0 \\
\tau_{ZB} &= \tau_{Z1} + \tau_{Z2} - \tau_{Z3} - \tau_{Z4} \\
\tau_{XB} &= (F_{Z2} + F_{Z3}) r \cos(45) - (F_{Z1} + F_{Z4}) r \cos(45) \\
\tau_{YB} &= (F_{Z1} + F_{Z3}) r \sin(45) - (F_{Z2} + F_{Z4}) r \sin(45)
\end{aligned}
$$

Zauważ, że momenty wokół osi Z ramy nadwozia są bezpośrednio wytwarzane przez inne momenty (silniki), podczas gdy momenty wokół osi X i Y ramy nadwozia są spowodowane działaniem ramion dźwigni



Dla uproszczenia jest to zilustrowane tylko dla silnika 4, ale można wydedukować pozostałe komponenty. Obserwuj kierunek ramion dźwigni w odniesieniu do osi współrzędnych, a tym samym ich komponentów. Należy również pamiętać o obrocie każdego silnika względem osi odniesienia w celu określenia odpowiednich znaków. W tym przypadku obie składowe w X i Y są ujemne i znajduje to odzwierciedlenie w równaniach, ponieważ obroty wytwarzane przez napór 4 są przeciwne do zadanego wcześniej dodatniego obrotu. Następnie poczynimy szereg założeń, aby maksymalnie uprościć powyższe równania. Musisz ocenić, czy takie założenia są wykonalne, czy nie we własnych projektach. Zakłada się następujące zależności między siłami i momentami obrotowymi a prędkością obrotową każdego silnika (założenia te są mniej więcej adekwatne, ponieważ producenci silników zwykle zapewniają ten stopień proporcji liniowej). Obserwuj stałe proporcjonalności, które wiążą moment obrotowy i siłę ciągu z prędkością silnika.

$$\tau = \kappa_\tau \omega$$
$$F = \kappa_F \omega$$

W związku z tym, a także zakładając, że wszystkie silniki są takie same (jeśli silniki są tym samym modelem, stałe proporcjonalności, według producenta, będą podobne lub w przybliżeniu takie same),

$$F_{ZB} = \kappa_F(\omega_1 + \omega_2 + \omega_3 + \omega_4)$$
$$F_{XB} = F_{YB} = 0$$
$$\tau_{ZB} = \kappa_\tau(\omega_1 + \omega_2 - \omega_3 - \omega_4)$$
$$\tau_{XB} = r\cos(45)\kappa_F(-\omega_1 + \omega_2 + \omega_3 - \omega_4)$$
$$\tau_{YB} = r\sin(45)\kappa_F(\omega_1 - \omega_2 + \omega_3 - \omega_4)$$

Kolejne założenie jest takie, że wyrazy kf, kt, rcos (45) kf i rsin (45) kf są stałymi wzmocnieniami i ponieważ występują we wszystkich wyrazach zawierających prędkość kątową w, mogą być faktoryzowane i absorbowane przez kompensację dla Fzb, Tzb, Txb i Tyb. W ten sposób często można znaleźć tę uproszczoną reprezentację. Należy pamiętać, że wspomniana kompensacja uwzględnia symetrie, a jeśli kąty ramion lub ich odległości były różne (asymetryczny dron), należy wziąć pod uwagę te terminy, a nie pominąć:

$$
\begin{aligned}
F_{ZB} &= \omega_1 + \omega_2 + \omega_3 + \omega_4 \\
F_{XB} &= F_{YB} = 0 \\
\tau_{ZB} &= \omega_1 + \omega_2 - \omega_3 - \omega_4 \\
\tau_{XB} &= -\omega_1 + \omega_2 + \omega_3 - \omega_4 \\
\tau_{YB} &= \omega_1 - \omega_2 + \omega_3 - \omega_4
\end{aligned}
$$

Te równania są prostsze niż oryginalne i dopuszczają tę wektorową reprezentację:

$$
\begin{bmatrix} F_{XB} \\ F_{YB} \\ F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}
$$

And thus the following reduction (to simplify matrix operations):

$$
\begin{bmatrix} F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}
$$

Dla tych, którzy zastanawiają się, w jaki sposób sterowane są ruchy X i Y, zostanie to opisane w dalszej części książki poświęconej kontroli, ale przewiduje się, że wynika to z ich zależności od zmiennych kątowych (sterowanie zagnieżdżone). Zwróć uwagę, że w przypadku heksarotorów lub tricopterów nieuchronnie będziemy mieć macierze niekwadratowe i konieczne będzie użycie pseudoodwrotności lub algorytmów optymalizacji numerycznej. Kontynuując naszą analizę, po zaprojektowaniu wartości kontrolnych tego wektora,

$$
\begin{bmatrix} F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix}
$$

jego dystrybucja do silników odbywa się w następujący sposób:

$$
\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix}
$$

Aby zredukować przetwarzanie i uniknąć ciągłego obliczania wspomnianej odwrotności i mnożenia macierzy, jest to zaprogramowane w opracowany sposób, to znaczy

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \end{bmatrix}$$

która, zgodnie z tym samym argumentem pochłaniania wartości stałych przez sterowanie momentami i siłami, jest równoważna w działaniu (jest to macierz alokacji)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}^{-1} \approx \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix}$$

i w końcu

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix} = \begin{bmatrix} F_{ZB} - \tau_{XB} + \tau_{YB} + \tau_{ZB} \\ F_{ZB} + \tau_{XB} - \tau_{YB} + \tau_{ZB} \\ F_{ZB} + \tau_{XB} + \tau_{YB} - \tau_{ZB} \\ F_{ZB} - \tau_{XB} - \tau_{YB} - \tau_{ZB} \end{bmatrix}$$

Format ten musi być włączony zarówno do sterowania automatycznego (zaprojektowanego i wprowadzonego przez teorię sterowania, jak zostanie pokazane w odpowiedniej sekcji), jak i do sterowania ręcznego (na przykład z pilota zdalnego sterowania). W przypadku korzystania z pilota, wektor, który należy włączyć do silników, jest następujący:

$$\begin{bmatrix} Throttle \\ Roll \\ Pitch \\ Rudder \end{bmatrix}$$

Półautomatyczny projekt (z operacjami ręcznymi i automatycznymi) mógłby wyglądać następująco (ta suma wektorów musi być pomnożona przez macierz alokacji, aby wprowadzić swój efekt do każdego silnika):

$$\begin{bmatrix} F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix} + \begin{bmatrix} Throttle \\ Roll \\ Pitch \\ Rudder \end{bmatrix}$$

Naturalnym pytaniem jest, czy poprzednie równania mają prostszy sposób na bycie użytym. Odpowiedź jest twierdząca, ale implikuje również pewne warunki pracy. Zobaczmy co się stanie.

Uproszczenia liniowe

Przed wprowadzeniem drugiego zestawu uproszczeń, pamiętajmy o tych już dokonanych, pamiętając, że powinieneś ponownie rozważyć użycie ich w swoich modelach matematycznych.

• Pojazdy o niewielkich rozmiarach, co implikuje bezwładnościowe uproszczenia

• Pojazdy dobrze wyważone w odniesieniu do ich środka ciężkości, co oznacza uproszczenie momentów i sił

• Wyrównanie środka ciężkości ze środkiem geometrycznym, co oznacza uproszczenia promieniowe lub ramiona dźwigni zmierzające do zera

• Dobrze wyważone śmigła, co oznacza, że każde śmigło zapewnia tylko ciąg i moment obrotowy

• Symetria pojazdów w celu uproszczenia macierzy bezwładności, a to oznacza również redukcję równań odśrodkowych

• Podstawowe modelowanie, które nie uwzględnia tarcia, turbulencji ani innych efektów modelowania aerodynamicznego. Zapewnia to wysoce uproszczony model, który można wykonać w prawie każdym oprogramowaniu do symulacji dynamicznej.

To powiedziawszy, przejdźmy do uproszczeń zadań. Płynny tryb lotu oznacza tutaj, że kąty przechyłu i pochylenia mają tendencję do zera, podczas gdy kąt odchylenia może przyjmować dowolną wartość dla przekierowania samolotu, jak kierownica w samochodzie. Płynny tryb lotu obejmuje również płynne prędkości kątowe. To daje do zrozumienia ze

• Kąty przechylenia i pochylenia będą zgodne z tymi przybliżeniami, gdy dążą do zera:

$$\sin(x) \approx x$$
$$\sin(x) \approx 0$$
$$\cos(x) \approx 1$$

Zwróć uwagę na przybliżenia sinusa; oba są używane zgodnie z najwygodniejszą z nich. Ta, która jest równa zero, jest granicą, a ta, która jest równa prostej, jest przybliżeniem liniowym (na przykład pierwszy element szeregu Taylora). Te podejścia nie są ze sobą sprzeczne i są komplementarne. Są też równie przydatne (w rzeczywistości mogą występować jednocześnie w tym samym problemie).

• Pomnożone przez nie powolne prędkości kątowe dają mniejszy wynik, który dąży do zera.

$$\omega_x\omega_y = \omega_y\omega_z = \omega_x\omega_z \approx 0$$

Wpłynie to również na dynamiczne zachowanie pojazdu w następujący sposób. Jeśli opracujemy translacyjny model dynamiczny,

$$R\begin{bmatrix} 0 \\ 0 \\ F_{BZ} \end{bmatrix} + m\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

i pamiętamy, że

$$\boldsymbol{R} = R_{z_\psi} R_{x_\phi} R_{y_\theta}$$

$$\boldsymbol{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + s\phi s\psi c\theta \\ s\psi c\theta + s\phi c\psi s\theta & c\phi c\psi & s\psi s\theta - s\phi c\psi c\theta \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

będziemy mieli

$$\begin{bmatrix} F_{BZ}\left(\sin\theta\cos\psi+\sin\phi\cos\theta\sin\psi\right) \\ F_{BZ}\left(\sin\theta\sin\psi-\sin\phi\cos\theta\cos\psi\right) \\ F_{BZ}\cos\phi\cos\theta \end{bmatrix}+m\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}=m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

Stosując płynne warunki lotu, dynamika translacyjna jest w ten sposób uproszczona (należy zauważyć, że uproszczenie kinematyki translacyjnej jest niejawne). Zauważ, że ten płynny lot dotyczy tylko kątów przechylenia i pochylenia, podczas gdy kąt odchylenia można ustawić dowolnie i ma żądaną wartość.

$$F_{BZ}\begin{bmatrix} \theta\cos\psi_d+\phi\sin\psi_d \\ \theta\sin\psi_d-\phi\cos\psi_d \\ 1 \end{bmatrix}+m\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}=m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

Zauważ, że te uproszczenia sinusoidalne dla małych kątów zostały zastosowane dla wygody. Gdybyśmy zastosowali inne uproszczenie, mielibyśmy tylko zera i nie byłoby równań, z którymi można by wchodzić w interakcje.

$$\sin(x)\approx x$$

Na poziomie rotacyjnym mamy następujący model dynamiczny:

$$\begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix}=\begin{bmatrix} J_x\dot{\omega}_x \\ J_y\dot{\omega}_y \\ J_z\dot{\omega}_z \end{bmatrix}+\begin{bmatrix} \omega_y\omega_zJ_z-\omega_y\omega_zJ_y \\ \omega_x\omega_zJ_x-\omega_x\omega_zJ_z \\ \omega_x\omega_yJ_y-\omega_x\omega_yJ_x \end{bmatrix}$$

A ze względu na płynny lot, powolne prędkości kątowe są uproszczone w ten sposób:

$$\begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix}=\begin{bmatrix} J_x\dot{\omega}_x \\ J_y\dot{\omega}_y \\ J_z\dot{\omega}_z \end{bmatrix}$$

Warto przeanalizować, czy możliwe jest wyrażenie tego równania bezpośrednio w ramce stałej, czy nie. W tym celu konieczne jest przeniesienie momentów obrotowych i prędkości kątowych z ramy pojazdu na ramę stałą. Wykorzystamy wcześniej wyprowadzoną zależność kinematyczną:

$$\boldsymbol{\omega}=\begin{bmatrix} \boldsymbol{\omega_{xB}} \\ \boldsymbol{\omega_{yB}} \\ \boldsymbol{\omega_{zB}} \end{bmatrix}=\begin{bmatrix} \boldsymbol{c\theta} & \mathbf{0} & \boldsymbol{-c\phi s\theta} \\ \mathbf{0} & \mathbf{1} & \boldsymbol{s\phi} \\ \boldsymbol{s\theta} & \mathbf{0} & \boldsymbol{c\phi c\theta} \end{bmatrix}\begin{bmatrix} \boldsymbol{\dot{\phi}} \\ \boldsymbol{\dot{\theta}} \\ \boldsymbol{\dot{\psi}} \end{bmatrix}$$

Jest to uproszczone w następujący sposób:

$$\begin{bmatrix} \omega_{xB} \\ \omega_{yB} \\ \omega_{zB} \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Tutaj dla wygody użyto tych przybliżeń dla małych kątów:

$$\sin(x) \approx 0$$

Dzieje się tak dlatego, że pożądane jest tutaj zniknięcie terminów i nie zachowanie ich do dalszej analizy. Wracając do dynamicznego modelu obrotowego opracowanego w ramie pojazdu, wykonujemy następujące czynności, aby sprowadzić go do ramy stałej:

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = J_w \begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix} = \begin{bmatrix} J_x\dot{\omega}_x \\ J_y\dot{\omega}_y \\ J_z\dot{\omega}_z \end{bmatrix}$$

$$J_w = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix}$$

Dla prędkości kątowych musimy wykonać następujące operacje:

$$\omega = J_w\dot{\Theta} \qquad \dot{\Theta} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\dot{\omega} = J_w\ddot{\Theta} + \dot{J}_w\dot{\Theta}$$

Ale pamiętaj o tym (i pamiętaj, że odwrotność macierzy jednostkowej jest również macierzą jednostkową):

$$J_w = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

W związku z tym,

$$\dot{J}_w \approx \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \dot{\omega} \approx J_w\ddot{\Theta} \approx \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}$$

W ten sposób dynamika obrotowa w ramie pojazdu staje się taki sam jak w ramie stałej.

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = J_w \begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix} = \begin{bmatrix} J_x \dot{\omega}_x \\ J_y \dot{\omega}_y \\ J_z \dot{\omega}_z \end{bmatrix}$$

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \approx \begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix} \approx \begin{bmatrix} J_x \ddot{\phi} \\ J_y \ddot{\theta} \\ J_z \ddot{\psi} \end{bmatrix}$$

Krótko mówiąc, uproszczenie płynnego lotu zmieniło pierwotny zestaw równań, jak pokazano na rysunkach

Translational kinematics

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix}$$

$\sin(\cdot) \rightarrow \cdot$

$\phi\theta \rightarrow 0$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} c\psi_d & -s\psi_d & \theta c\psi_d + \phi s\psi_d \\ s\psi_d & c\psi_d & \theta s\psi_d - \phi c\psi_d \\ -\theta & \phi & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix}$$

Rotational kinematics

$$\begin{bmatrix} \omega_{xB} \\ \omega_{yB} \\ \omega_{zB} \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$\sin(\cdot) \rightarrow 0$

$$\begin{bmatrix} \omega_{xB} \\ \omega_{yB} \\ \omega_{zB} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Translational dynamics

$$R\begin{bmatrix}0\\0\\F_{BZ}\end{bmatrix}+m\begin{bmatrix}0\\0\\-g\end{bmatrix}=m\begin{bmatrix}\ddot{x}\\\ddot{y}\\\ddot{z}\end{bmatrix}$$

$\sin(\bullet) \rightarrow \bullet$, $\phi\theta \rightarrow 0$

$$F_{BZ}\begin{bmatrix}\theta\cos\psi_d+\phi\sin\psi_d\\\theta\sin\psi_d-\phi\cos\psi_d\\1\end{bmatrix}+m\begin{bmatrix}0\\0\\-g\end{bmatrix}=m\begin{bmatrix}\ddot{x}\\\ddot{y}\\\ddot{z}\end{bmatrix}$$

Rotational dynamics

$$\begin{bmatrix}\tau_{BX}\\\tau_{BY}\\\tau_{BZ}\end{bmatrix}=\begin{bmatrix}J_x\dot{\omega}_x\\J_y\dot{\omega}_y\\J_z\dot{\omega}_z\end{bmatrix}+\begin{bmatrix}\omega_x\\\omega_y\\\omega_z\end{bmatrix}\times\begin{bmatrix}J_x\omega_x\\J_y\omega_y\\J_z\omega_z\end{bmatrix}$$

$\sin(\bullet) \rightarrow 0$

$$\begin{bmatrix}\tau_\phi\\\tau_\theta\\\tau_\psi\end{bmatrix}\approx\begin{bmatrix}\tau_{BX}\\\tau_{BY}\\\tau_{BZ}\end{bmatrix}\approx\begin{bmatrix}J_x\ddot{\phi}\\J_y\ddot{\theta}\\J_z\ddot{\psi}\end{bmatrix}$$

Allocation matrix

$$\begin{bmatrix}F_{XB}\\F_{YB}\\F_{ZB}\\\tau_{XB}\\\tau_{YB}\\\tau_{ZB}\end{bmatrix}_{6\times1}=[A]_{6\times m}\begin{bmatrix}\omega_1\\\omega_2\\\vdots\\\omega_m\end{bmatrix}_{m\times1}$$

$$\begin{bmatrix}F_{XB}\\F_{YB}\\F_{ZB}\\\tau_{XB}\\\tau_{YB}\\\tau_{ZB}\end{bmatrix}_{6\times1}=[A]_{6\times m}\begin{bmatrix}\omega_1\\\omega_2\\\vdots\\\omega_m\end{bmatrix}_{m\times1}$$

Zwróć uwagę, że zmiany zostały dokonane w zależnościach kinematycznych i oczywiście w dynamice, ale równania napędu pozostają takie same (pamiętaj, że jeśli twój pojazd ma system wektorowania ciągu, macierz alokacji również ulegnie zmianie).

Streszczenie

W tej części poznałeś ogólny model dronów ze szczególnym, ale nie wyłącznym podejściem do quadkopterów. Tu wyjaśniono ramy modelowania, równania kinematyczne i dynamiczne typów obrotowych i translacyjnych, ogólne modelowanie śmigła, model alokacji lub napędu, a na koniec kilka

uproszczeń liniowych, które będą przydatne w następnym rozdziale. Te tematy są bardzo popularne w różnych książkach i artykułach o dronach. Ponownie zapraszamy do zapoznania się z Dodatkiem, jeśli chcesz pogłębić swoją wiedzę na temat poznanych tematów. W kolejnym rozdziale poznasz tematy związane z projektowaniem sterowników postrzeganych jako dwa rodzaje rodzin: tryb pojazdu (lub widok pierwszoosobowy lub tryb sterowania pokładowego) i tryb robota (lub widok trzecioosobowy lub tryb sterowania zewnętrznego).

Kontrola dronów

W tej części o znaczeniu naukowym poznasz cztery z większości wykorzystywanych kontrolerów dronów w oparciu o dwie klasyfikacje, jedną, która wizualizuje samolot jako pojazd, a drugą jako robota. W tej części pokażemy również niektóre koncepcje sterowania i podstawowe pojęcia teorii stabilności Lapunowa. Na koniec dołączona jest sekcja poświęcona planowaniu trajektorii. W rezultacie będziesz w stanie zrozumieć i wdrożyć większość dostępnych kontrolerów dronów, w tym potężne sterowanie geometryczne. Mając to na uwadze, możesz rozszerzyć tę wiedzę na inne typy pojazdów i samolotów. Pamiętaj, że aby wstrzyknąć badane przez nas kontrolery do silników pojazdu (i ogólnie do każdego innego kontrolera), musisz to zrobić za pomocą macierzy alokacji lub macierzy napędów obliczonej dla Twoich pojazdów. Macierz alokacji dla konkretnego przypadku analizowanego w poprzedniej części była.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} F_{ZB} \\ \tau_{XB} \\ \tau_{YB} \\ \tau_{ZB} \end{bmatrix} = \begin{bmatrix} F_{ZB} - \tau_{XB} + \tau_{YB} + \tau_{ZB} \\ F_{ZB} + \tau_{XB} - \tau_{YB} + \tau_{ZB} \\ F_{ZB} + \tau_{XB} + \tau_{YB} - \tau_{ZB} \\ F_{ZB} - \tau_{XB} - \tau_{YB} - \tau_{ZB} \end{bmatrix}$$

Dlatego celem tej części jest poznanie najbardziej ogólnych metod projektowania siły FZB i pozostałych trzech momentów obrotowych. Należy również pamiętać, że jest to ważne, jeśli używany jest SDK typu 3, jak wskazano w sekcji dotyczącej sposobów programowania drona w części 5. W przypadku korzystania z SDK typu 2 wspomniana siła i momenty będą wstrzykiwane bezpośrednio do pojazd bez konieczności stosowania macierzy alokacji. Wreszcie, w przypadku korzystania z SDK typu 1 lub GUI, będziesz mógł tylko zdefiniować profile trajektorii (nie będziesz mógł zaprojektować własnych kontrolerów; możesz po prostu zmodyfikować wzmocnienia i ścieżki wstępnie załadowanych kontrolek). Zobaczysz, że quadkopter, jak każdy inny pojazd bezzałogowy, może być analizowany i postrzegany jako robot lub pojazd, a w obu przypadkach sterowanie może być automatyczne lub nie. Zanim zaczniemy mówić o tych kontrolerach, porozmawiajmy o użytecznej koncepcji zerowej.

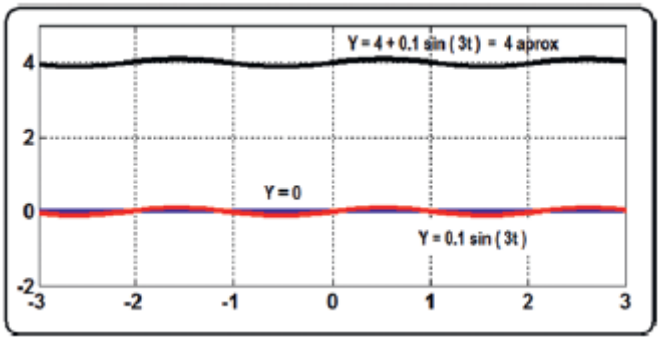
Przydatna koncepcja zera

Celem kontroli jest wprowadzenie użytecznych zer w celu opracowania zadania. W teorii sterowania zwykle znajduje się trzy rodzaje zera: zero z definicji, zero z dominacji i zero z przybliżenia. Nie myl tych pojęć z pojęciami biegunów i zer liniowej teorii sterowania.

• Zero z definicji jest idealną i stałą wartością, do której ma doprowadzić do błędu lub oszacowania. Jest to również równowaga statyczna lub dynamiczna, która zapewnia równowagę siły lub ruchu.

• Zero przez dominację występuje wtedy, gdy wyraz w równaniu jest zbyt duży w stosunku do innych wyrazów, tak że równanie po prostu zachowuje się jak wyraz dominujący, nie przypisując wagi

skutki pozostałych warunków. Te unieważnione warunki są traktowane jako zero przez dominację. Ten typ zera jest bardzo powszechny w analizie obwodów elektrycznych i elektronicznych, a także w teorii sterowania. Rysunek przedstawia wykres funkcji, w której efekt dominacji można uznać za w przybliżeniu równy 4.



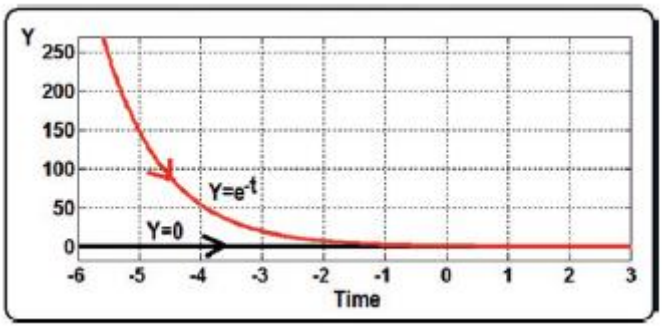
Zauważ, że zdominowany składnik sinusoidalny może być traktowany jako zero proporcjonalnie do 4.

Z drugiej strony to, co tutaj nazywa się użytecznym zerem, to to, które uzyskuje się na podstawie zachowania czasu. Matematycznie będzie to dowolna funkcja, która w miarę upływu czasu dąży do zera.

Funkcją szeroko stosowaną w teorii sterowania jako użyteczne zero jest ujemny wykładnik. To użyteczne zero można zaobserwować, zachowując się jak zero w miarę upływu czasu. Zwróć uwagę na fakt, że nie ma czasu ujemnego. Jest to po prostu wykreślone jako widok rozszerzony:

$$y = Ae^{-Bt}$$

Ta funkcja będzie zawsze zachowywać się tak, jak pokazano na rysunku , o ile B ma wartość dodatnią.



Zauważ, że użyteczne zero to sposób na zmuszenie czegoś, aby stało się stałym zerem. Ta funkcja jest również rozwiązaniem i zachowaniem związanym z tym równaniem różniczkowym:

$$\frac{dy}{dt} = -y$$

$$\int \frac{dy}{y} = \int -dt \Rightarrow \ln(y) = -t \Rightarrow y = e^{-t}$$

Używając notacji Newtona, jest to

$$\dot{y} = -y$$

Mówiąc bardziej ogólnie, ta rodzina równań różniczkowych zawsze skutkuje malejącą funkcją wykładniczą, która dąży do zera, o ile B i K są wartościami dodatnimi:

$$B\dot{y} = -Ky$$

Jeśli dokonamy następujących zmian zmiennych, otrzymamy równanie znane jako kontroler PD:

$$B\dot{y} + Ky = K_d\dot{E} + K_pE$$

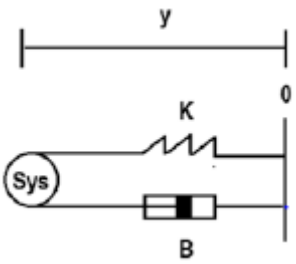
co oznacza, że wartość E związana z kontrolowanym błędem będzie dążyła do zera, a ponieważ E jest zdefiniowane jako

$$E = v_d - v$$

Innymi słowy, pożądana wartość minus rzeczywista wartość zmiennej, na przykład pozycja lub orientacja:

$$E \to 0$$
$$v_d - v \to 0$$
$$v \to v_d$$

Pożądana wartość będzie na ogół wartością rzeczywistą i dlatego wykonamy nasze zadanie! Użyjemy tej przydatnej koncepcji zerowej i jej aplikacji PD później. Należy pamiętać, że równanie różniczkowe pierwszego rzędu, a w konsekwencji wyładowania niezupełne, ma fizyczny odpowiednik zwany bezmasowym tłumionym oscylatorem harmonicznym lub układem sprężynowym. Dlatego człon proporcjonalny może być interpretowany jako sprężyna indukowana, a człon różnicowy jako tłumik indukowany;



Wspomniana indukowana sprężyna służy do dociągania rzeczywistego systemu do pożądanej wartości, a indukowany tłumik służy do zatrzymania rzeczywistego systemu przy tej wartości. Gdyby to była tylko indukowana sprężyna, a nie indukowany tłumik, rzeczywisty układ nie zatrzymywałby się i oscylowałby wokół wartości zadanej (zerowej wartości błędu). Innym użytecznym zerem jest równanie różniczkowe drugiego rzędu lub układ tłumienia drgań ze sprężyną masową, zwany także tłumionym oscylatorem harmonicznym, o ile wyrazy M, B i K są dodatnie. Dedukcja jest nieco mniej intuicyjna i jest kwestią książki o sterowaniu liniowym i równaniach różniczkowych. Jednak na potrzeby tej książki iw podanych warunkach po prostu założymy, że ten system jest również użytecznym zerem.

$$M\ddot{y} + B\dot{y} + Ky = 0$$

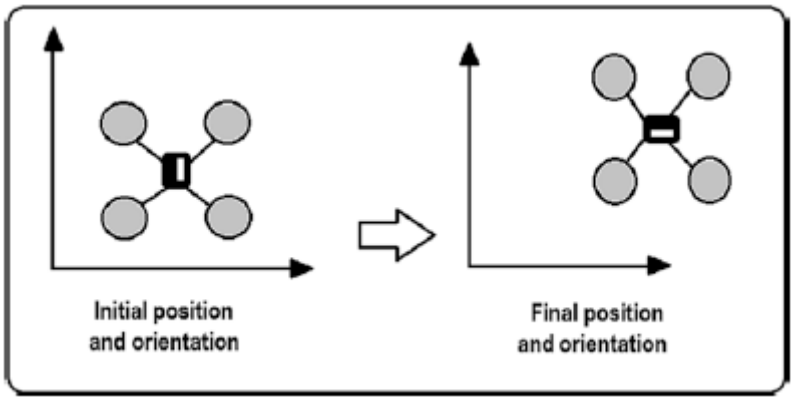
Dla celów kontrolnych podkreślono, że regulacja PID zachowuje się jak wspomniany układ amortyzatorów masowo-sprężynowych (przynajmniej matematycznie, ponieważ podczas implementacji istnieją względy fizyczne, jako efekt zastosowania różniczkujących zamiast integratorów).

$$\frac{d\,(K_d\dot{E} + K_pE + K_i\int E)}{dt} = K_d\ddot{E} + K_p\dot{E} + K_iE$$

Po zapoznaniu się z użyteczną koncepcją zerową przejdźmy do kontrolerów. Zaczniemy od kontrolerów trybu robota.

Sterowanie trybem robota

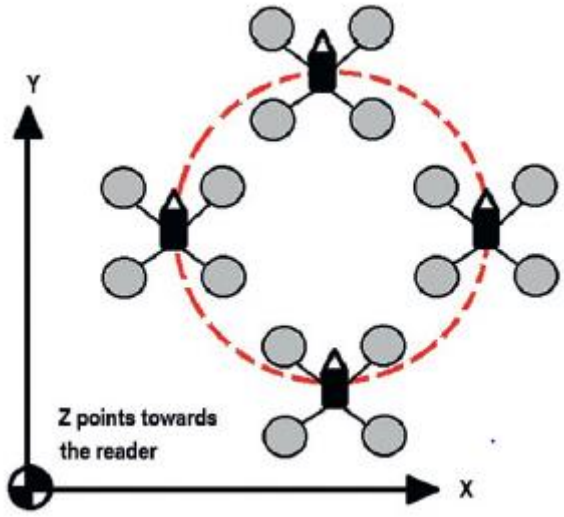
Jeśli dron jest analizowany jako robot, projekt sterowania będzie polegał na postrzeganiu go tak, jakby był kontrolowany przez osobę trzecią lub jako widok zewnętrzny. W tym celu celem sterowania będzie przesunięcie punktu odniesienia quadkoptera do danej trójwymiarowej lokalizacji z regulowaną orientacją lub bez niej.



Czas wyjaśnić sterowanie w trybie robota zwane liniowym sterowaniem kartezjańskim bez zmiany odchylenia.

W pełni liniowa kontrola kartezjańska bez zmienności odchylenia

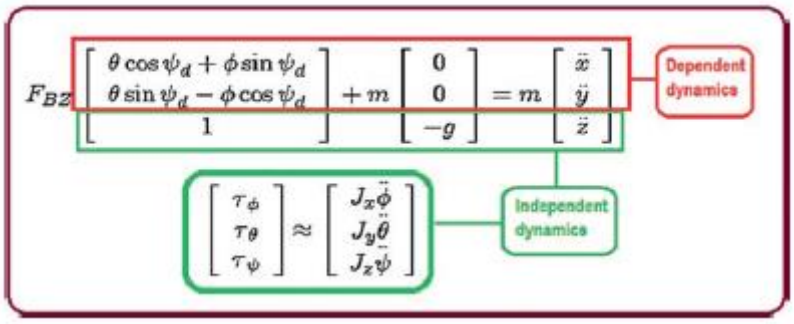
Jest to prosty i bardzo podstawowy kontroler, który umożliwia operację taką jak ta pokazana na rysunku



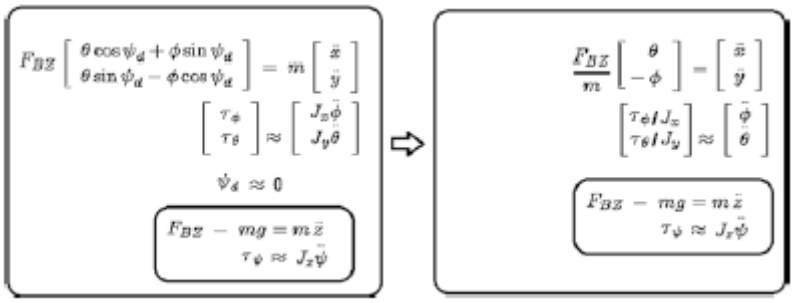
, która polega po prostu na podążaniu ścieżką bez zmiany orientacji drona. Przydaje się to na przykład do sadzenia lub malowania domów. Opiszmy jego niezależne i zależne moduły dynamiki.

Kontrola Niezależnej Dynamiki

Wróćmy do zlinearyzowanego dynamicznego modelu drona, oddzielając dynamikę niezależną od innych zmiennych (zauważmy, że odbywa się to w odniesieniu do zmiennych ruchu, w których można zidentyfikować trzy pozycje i trzy orientacje). Więc mamy



Zauważ, że zależność od dynamiki X i Y dotyczy dynamiki kątów theta i phi. Dlatego wyodrębniamy równania, które nie wchodzą w interakcję z innymi równaniami, tj. z równaniami psi i Z. Zauważ, że psi różni się od pożądanego psi ze względu na fakt, że to ostatnie jest wartością stałą lub trajektorią, a w naszym przykładzie pożądane psi będzie równe zero.



Jeżeli siła ciągu w Z jest proponowana w następujący sposób

$$F_{BZ} = m(g + PD_z + \ddot{z}_d)$$

gdzie

$$PD_z = K_{PZ}(z_d - z) + K_{DZ}(\dot{z}_d - \dot{z})$$

przy zamianie go na model dynamiczny, a szczególnie na równanie Z, otrzymamy:

$$m(g + PD_z + \ddot{z}_d) - mg = m\ddot{z}$$
$$m(PD_z + (\ddot{z}_d - \ddot{z})) = 0$$
$$K_{PZ}(z_d - z) + K_{DZ}(\dot{z}_d - \dot{z}) + (\ddot{z}_d - \ddot{z}) = 0$$

Staje się to użytecznym zerowym lub tłumionym oscylatorem harmonicznych.

$$E_z = z_d - z$$
$$K_{PZ}(E_z) + K_{DZ}(\dot{E}_z) + (\ddot{E}_z) = 0$$

I tak długo, jak zyski spełniają co najmniej pozytywne wyniki zgodnie z innymi kryteriami zawartymi w liniowych księgach kontrolnych, wiemy, że

$$E_z \to 0$$
$$z \to z_d$$
$$\dot{z} \to \dot{z}_d$$
$$\ddot{z} \to \ddot{z}_d$$

Ponadto teoria sterowania wykazuje, że jeśli pożądana wartość jest stałą lub funkcją, której maksymalna wartość bezwzględna może być ograniczona przez stałą (na przykład sinus lub cosinus),

$$z_d = K$$
$$z \to K$$
$$\dot{z} \to \dot{z}_d \to 0$$
$$\ddot{z} \to \ddot{z}_d \to 0$$

podobne rozumowanie można zastosować w przypadku regulatora PID. Demonstrację można opracować samodzielnie jako ćwiczenie.

Oznacza to również, że

$$F_{BZ} = m(g + PD_z + \ddot{z}_d) \to mg$$

i konsekwentnie

$$\frac{F_{BZ}}{m}\begin{bmatrix} \theta \\ -\phi \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \quad \to \quad g\begin{bmatrix} \theta \\ -\phi \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$$

Niedługo wykorzystamy ten wynik. Z drugiej strony w praktyce i symulacji są ludzie, którzy upraszczają proponowaną kontrolę ciągu w następujący sposób:

$$F_{BZ} = mg + PD_z$$

I zadziała, jeśli przyrosty różnicowe i proporcjonalne będą wystarczająco duże, biorąc pod uwagę następujące względy:

1. Jeśli przyspieszenia lotu są małe (co wynika z założenia, które przyjęliśmy o płynnym locie).

$$F_{BZ} = mg + mPD_z + m\ddot{z}_d$$

$$mg + mPD_z >> m\ddot{z}_d$$
$$F_{BZ} \approx mg + mPD_z$$

2. Jeśli masa jest stała lub ograniczona przez stałą, może zostać pochłonięta przez wzmocnienie proporcjonalne i różniczkowe (jedna stała pomnożona przez inną stałą daje inną stałą):

$$mPD_z = mK_{PZ}(z_d - z) + mK_{DZ}(\dot{z}_d - \dot{z})$$

$$mK_{PZ}(z_d - z) + mK_{DZ}(\dot{z}_d - \dot{z}) \approx K_{PZ}(z_d - z) + K_{DZ}(\dot{z}_d - \dot{z})$$
$$F_{BZ} \approx mg + PD_z$$

Kontynuujmy równanie związane z psi (zmienną odchylenia). W takim przypadku wartość kontrolną można zdefiniować jako

$$\tau_\psi = J_z(PD_\psi + \ddot{\psi}_d)$$

Kiedy ta kontrola zostanie zastąpiona modelem odchylenia, prowadzi to do następujących sytuacji:

$$J_z(PD_\psi + \ddot{\psi}_d) = J_z\ddot{\psi}$$
$$K_{P\psi}(\psi_d - \psi) + K_{D\psi}(\dot{\psi}_d - \dot{\psi}) + (\ddot{\psi}_d - \ddot{\psi}) = 0$$
$$K_{P\psi}(E_\psi) + K_{D\psi}(\dot{E}_\psi) + (\ddot{E}_\psi) = 0$$

I znowu, jeśli wzmocnienia regulatora są co najmniej dodatnie i spełniają pozostałe wytyczne opisane w podręcznikach automatyki, będziemy mieli użyteczny układ tłumienia zerowego lub masowo-sprężynowego, co implikuje, że

$$E_\psi = \psi_d - \psi$$
$$E_\psi \to 0$$
$$\psi \to \psi_d$$
$$\dot{\psi} \to \dot{\psi}_d$$
$$\ddot{\psi} \to \ddot{\psi}_d$$

ponownie biorąc pod uwagę płynne warunki lotu (małe przyspieszenia) i stałe wartości pochłaniania. Jak wywnioskowaliśmy dla zmiennej Z, otrzymamy:

$$\psi_d = K$$
$$\dot{\psi} \to K$$
$$\ddot{\psi} \to \dot{\psi}_d \to 0$$
$$\dddot{\psi} \to \ddot{\psi}_d \to 0$$

$$J_z(PD_\psi + \ddot{\psi}_d) \approx PD_\psi \approx \tau_\psi$$

Teraz przystępujemy do kontrolowania dynamiki zależnej.

Kontrola dynamiki zależnej

Z poprzednich uproszczeń możemy zauważyć, że pozostały system to

$$\begin{bmatrix} \tau_\phi / J_x \\ \tau_\theta / J_y \end{bmatrix} \approx \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix}$$

$$g \begin{bmatrix} \theta \\ -\phi \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$$

Podsystemy, które nie mają zależności od innych zmiennych to podsystemy rotacyjne, natomiast translacyjne zależne od nich zmienne rotacyjne. Z tego powodu zasadne jest rozpoczęcie projektowania regulatorów dynamiki obrotowej. Przy podobnym argumencie do przedstawionego dla zmiennej yaw i z wyjaśnionych już powodów płynnego lotu (małe przyspieszenia) i stałej absorpcji, proponuje się następujące regulatory przechyłu i pochylenia:

$$\tau_\phi = J_x(PD_\phi + \ddot{\phi}_d) \approx PD_\phi$$
$$\tau_\theta = J_y(PD_\theta + \ddot{\theta}_d) \approx PD_\theta$$

Oznacza to w podobny sposób do wcześniej analizowanych zmiennych, które:

$$E_\phi \to 0 \qquad E_\theta \to 0$$
$$\phi \to \phi_d \qquad \theta \to \theta_d$$
$$\dot{\phi} \to \dot{\phi}_d \qquad \dot{\theta} \to \dot{\theta}_d$$
$$\ddot{\phi} \to \ddot{\phi}_d \qquad \ddot{\theta} \to \ddot{\theta}_d$$

W konsekwencji dynamikę zależną można kontrolować poprzez pożądane wartości kątów, od których zależą. Jest to logiczne, ponieważ quadkopter jest w stanie poruszać się w osiach X i Y poprzez przechylanie się i przechylanie.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = g \begin{bmatrix} \theta \\ -\phi \end{bmatrix} \approx g \begin{bmatrix} \theta_d \\ -\phi_d \end{bmatrix}$$

Następnie żądane wartości theta i phi są wybierane w ten sposób (jest to znane jako zewnętrzna pętla sterowania lub sterowanie zagnieżdżone):

$$\theta_d = \frac{1}{g}(PD_x + \ddot{x}_d)$$

$$\phi_d = -\frac{1}{g}(PD_y + \ddot{y}_d)$$

Ta informacja zwrotna na temat translacyjnej dynamiki X i Y ma w konsekwencji:

$$E_{x,y} \to 0$$
$$x, y \to x_d, y_d$$
$$\dot{x}, \dot{y} \to \dot{x}_d, \dot{y}_d$$
$$\ddot{x}, \ddot{y} \to \ddot{x}_d, \ddot{y}_d$$

Przy założeniu płynnego lotu, ograniczonych trajektorii z małymi przyspieszeniami i absorpcji wcześniej stosowanych stałych, regulatory pomocnicze mogą być po prostu

$$\theta_d = PD_x$$
$$\phi_d = -PD_y$$

Podsumowując, kontrola, która ma być wstrzyknięta, jest następująca (patrz symulacja w następnej części, aby zaobserwować jej działanie):

$$F_{BZ} = mg + PD_z$$
$$\tau_\psi = PD_\psi$$

$$\theta_d = PD_x$$
$$\phi_d = -PD_y$$
$$\tau_\phi = PD_\phi$$
$$\tau_\theta = PD_\theta$$

który jest rozwijany w następujący sposób:

$$F_{BZ} = mg + K_{Pz}(z_d - z) + K_{Dz}(\dot{z}_d - \dot{z})$$
$$\tau_\psi = K_{P\psi}(\psi_d - \psi) + K_{D\psi}(\dot{\psi}_d - \dot{\psi})$$

$$\theta_d = K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x})$$
$$\phi_d = -[K_{Py}(y_d - y) + K_{Dy}(\dot{y}_d - \dot{y})]$$
$$\tau_\phi = K_{P\phi}(\phi_d - \phi) + K_{D\phi}(\dot{\phi}_d - \dot{\phi})$$
$$\tau_\theta = K_{P\theta}(\theta_d - \theta) + K_{D\theta}(\dot{\theta}_d - \dot{\theta})$$

Przypomnijmy sobie wstęp do normy ISO 1151-2: 1985 i zwróćmy uwagę, że pomiar theta jest wykonywany względem osi Y, ale jego zmienność wpływa na ruch w osi X. Podobnie pomiar phi jest wykonywany wokół osi X, ale jego zmienność wpływa na ruch na Y. Wreszcie, biorąc pod uwagę to

$$\psi_d = 0$$
$$\dot{\psi}_d = 0$$
$$\lim_{x \to x_d} \theta_d = 0$$
$$\dot{\theta}_d \approx 0$$
$$\lim_{y \to y_d} \phi_d = 0$$
$$\dot{\phi}_d \approx 0$$

$$\tau_\psi = -K_{P\psi}\psi - K_{D\psi}\dot{\psi}$$
$$\tau_\phi = K_{P\phi}(\phi_d - \phi) + K_{D\phi}(\dot{\phi}_d - \dot{\phi}) \approx K_{P\phi}(\phi_d - \phi) - K_{D\phi}\dot{\phi}$$
$$\tau_\theta = K_{P\theta}(\theta_d - \theta) + K_{D\theta}(\dot{\theta}_d - \dot{\theta}) \approx K_{P\theta}(\theta_d - \theta) - K_{D\theta}\dot{\theta}$$

powinieneś zauważyć, że pożądane prędkości theta i fi nie są równe zeru i że powinny one być odpowiednimi pochodnymi translacyjnych PD. Jednak zero jest dobrym przybliżeniem, biorąc pod uwagę przybliżenie gładkiego lotu i opisaną powyżej zbieżność zmienną. Jeśli chodzi o to płynne przybliżenie lotu, zobaczmy, co dzieje się w tym trybie lotu z kontrolerami.

Oddzielenie liniowe sterowników

Inną ważną kwestią jest fakt, że w tego typu płynnym locie ze stałym odchyleniem równym zeru, regulatory są odsprzęgane (przynajmniej regulatory liniowe jak PD, PID itp.), więc translacje planarne oraz obroty roll i pitch wydają się na przykład na samodzielne zachowanie

$$\tau_\theta \approx K_{P\theta}(\theta_d - \theta) - K_{D\theta}\dot{\theta} = K_{P\theta}\theta_d - K_{P\theta}\theta - K_{D\theta}\dot{\theta}$$
$$\theta_d = K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x})$$
$$\Downarrow$$
$$\tau_\theta \approx K_{P\theta}[K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x})] - K_{P\theta}\theta - K_{D\theta}\dot{\theta}$$

przyjmowanie wcześniejszych założeń stałej absorpcji

$$K_{P\theta}K_{Px} \approx K_{Px}$$
$$K_{P\theta}K_{Dx} \approx K_{Dx}$$

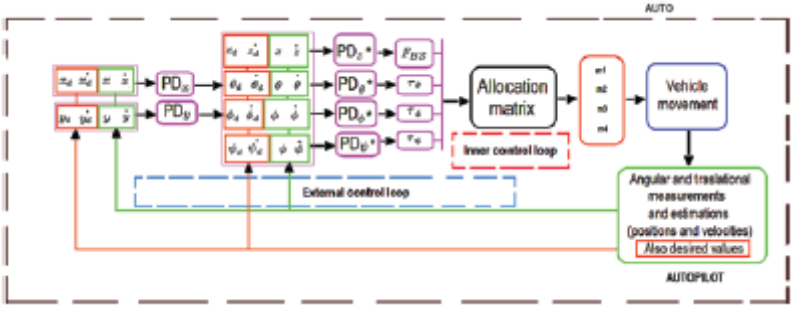
Uzyskuje się:

$$\tau_\theta \approx K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x}) + K_{P\theta}(0 - \theta) + K_{D\theta}(0 - \dot{\theta}) \approx PD_x + PD_\theta$$

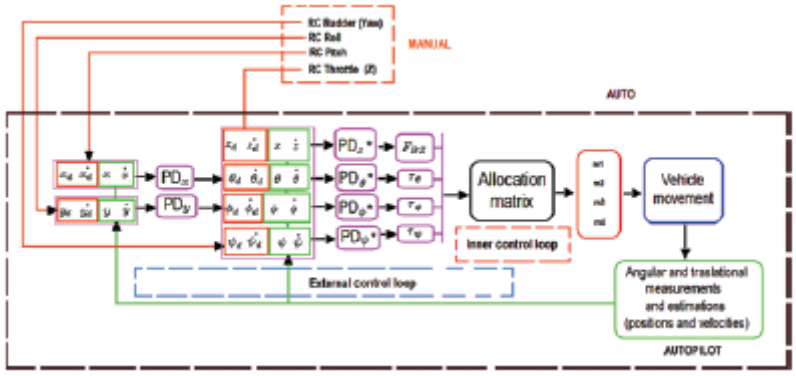
Podobny wynik uzyskano dla zmiennej phi. Zrób to jako ćwiczenie. Zanim zakończymy tę sekcję, porozmawiajmy o kilku uwagach.

Uwagi i interpretacje graficzne

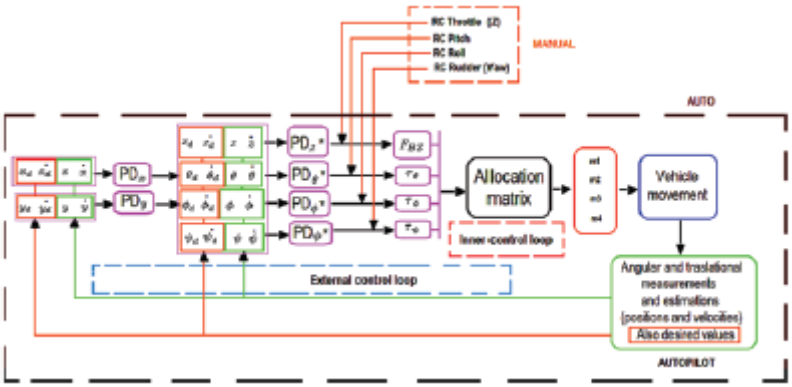
Graficznie implementacja tej kontroli ma sekwencję pokazaną na rysunku



Dwie wersje ze zdalnym sterowaniem (tryb półautomatyczny lub ręczny z przewodnikiem) są następujące. W tej pierwszej wersji pilot zmienia tylko żądane wartości;



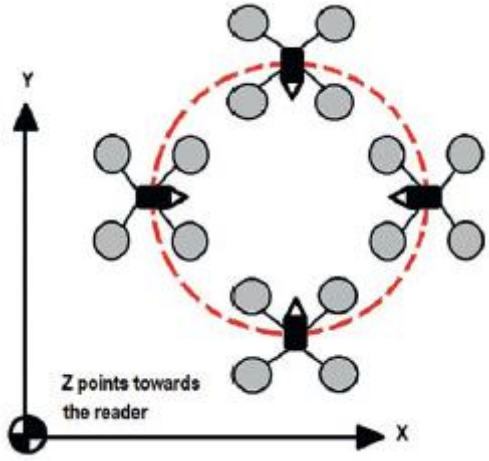
W drugiej wersji pilot modyfikuje momenty i siły.



Zauważ, że istnieje tyle kombinacji trybów ręcznych, ile chcesz. W opisanych przykładach druga może być niebezpieczna, ponieważ bezpośrednio wpływasz na kontrolery, ale ryzyko jest zmniejszone, jeśli wprowadzisz małe i ograniczone wartości. Dowiedziałeś się o sterowaniu w trybie robota bez zmiany odchylenia. Przejdźmy do tego ze zmiennym odchyleniem.

W pełni liniowa kontrola kartezjańska z odchyleniem odchylenia

Ten typ kontrolera jest modyfikacją poprzedniego. Teraz dron jest w stanie podążać po trajektorii, a także może modyfikować kąt odchylenia.



Ta operacja jest przydatna do nagrywania obiektu, osoby lub budynku znajdującego się wewnątrz ścieżki, tak jak w przypadku operacji nadzoru. W przypadku tego trybu lotu sterowanie ciągiem i momentem jest identyczne jak w poprzedniej sekcji:

$$F_{BZ} = mg + PD_z$$
$$\tau_{\psi} = PD_{\psi}$$
$$\tau_{\phi} = PD_{\phi}$$
$$\tau_{\theta} = PD_{\theta}$$

To, co się zmienia, to konstrukcja pożądanych sterowników pomocniczych theta i phi, jak następuje. Dynamiczne systemy XY nie mogą być uproszczone, ponieważ psi nie ma wartości zerowej:

$$\frac{F_{BZ}}{m}\begin{bmatrix} \theta\cos\psi_d + \phi\sin\psi_d \\ \theta\sin\psi_d - \phi\cos\psi_d \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \rightarrow g\begin{bmatrix} \theta\cos\psi_d + \phi\sin\psi_d \\ \theta\sin\psi_d - \phi\cos\psi_d \end{bmatrix}$$

Ale możesz przepisać je w formie macierzy w ten sposób:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = g\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}\begin{bmatrix} \theta \\ \phi \end{bmatrix}$$

Pamiętaj o tym z kontrolerów roll i pitch

$$\theta \rightarrow \theta_d$$
$$\phi \rightarrow \phi_d$$
$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = g\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}\begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix}$$

Można zaproponować następujące sterowniki pomocnicze:

$$\begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} = \frac{1}{g}\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}^{-1}\begin{bmatrix} PDx + \ddot{x}_d \\ PDy + \ddot{y}_d \end{bmatrix} = \frac{1}{g}\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}\begin{bmatrix} PDx + \ddot{x}_d \\ PDy + \ddot{y}_d \end{bmatrix}$$

Podstawienie regulatorów pomocniczych do zadanego modelu dynamicznego:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = g\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}\frac{1}{g}\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}\begin{bmatrix} PDx + \ddot{x}_d \\ PDy + \ddot{y}_d \end{bmatrix}$$

Uproszczenie:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} PDx + \ddot{x}_d \\ PDy + \ddot{y}_d \end{bmatrix}$$
$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} PDx + (\ddot{x}_d - \ddot{x}) \\ PDy + (\ddot{y}_d - \ddot{y}) \end{bmatrix}$$

A to oznacza, że

$$E_x = x_d - x \to 0$$
$$x \to x_d$$

$$E_y = y_d - y \to 0$$
$$y \to y_d$$

Przepisując proponowaną formę macierzową na równoważną, ale programowalną (co pozwala uniknąć wielu mnożeń macierzowych na komputerze), otrzymujemy znane równanie znalezione w badaniach dronów:

$$\begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} \approx \begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix} \begin{bmatrix} PDx \\ PDy \end{bmatrix} = \begin{bmatrix} PD_x \cos\psi_d + PD_y \sin\psi_d \\ PD_x \sin\psi_d - PD_y \cos\psi_d \end{bmatrix}$$

Na koniec zauważ, że macierz zależna od pożądanego psi jest macierzą rotacji (będzie to bardzo przydatne do interpretacji kontroli geometrycznej w nadchodzącej sekcji). Istnieją dwa sposoby sprawdzenia, czy jest to macierz rotacji:

1. Przekształcenie macierzy w znaną macierz rotacji za pomocą przekształceń:

$$\begin{bmatrix} \cos(x) & \sin x \\ \sin x & -\cos x \end{bmatrix} \quad x = \pi/2 - a$$
$$\begin{bmatrix} \cos(\pi/2 - a) & \sin(\pi/2 - a) \\ \sin(\pi/2 - a) & -\cos(\pi/2 - a) \end{bmatrix} = \begin{bmatrix} \sin a & \cos a \\ \cos a & -\sin a \end{bmatrix}$$
$$\begin{bmatrix} \sin a & \cos a \\ \cos a & -\sin a \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \cos a & \sin a \\ -\sin a & \cos a \end{bmatrix} = R_y(a)$$

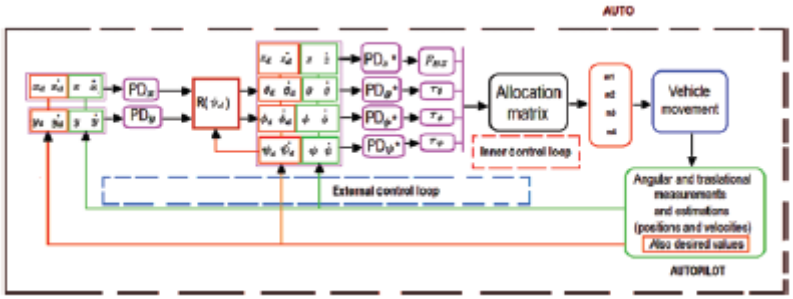
2. Korzystając z właściwości poprzednio stosowanych macierzy rotacji:

$$R^T R = I$$
$$\begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix}^T \begin{bmatrix} \cos\psi_d & \sin\psi_d \\ \sin\psi_d & -\cos\psi_d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Podsumowując, kontroler proponowany w tej sekcji jest następujący (patrz sekcja symulacji, aby obserwować działanie tego sterowania):

$$F_{BZ} = mg + PD_z$$
$$\tau_\psi = PD_\psi$$
$$\tau_\phi = PD_\phi$$
$$\tau_\theta = PD_\theta$$
$$\begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} \approx \begin{bmatrix} PD_x \cos\psi_d + PD_y \sin\psi_d \\ PD_x \sin\psi_d - PD_y \cos\psi_d \end{bmatrix}$$

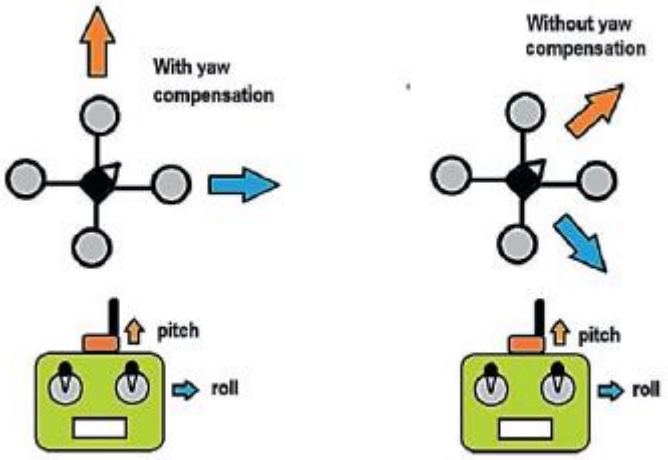
Sekwencja graficzna tych sterowników jest wyświetlana na rysunku



W przypadku wariantów zdalnego sterowania można je dostosować na dwa sposoby.

• Pierwszym z nich jest uwzględnienie działania zbaczania nad odpowiednimi pilotami zdalnego sterowania dla przechyłu i pochylenia (poprzez kompensację zbaczania, jak w poprzednim sterowaniu). W ten sposób ręczne zachowanie pojazdu będzie takie, jak po lewej stronie rysunku.

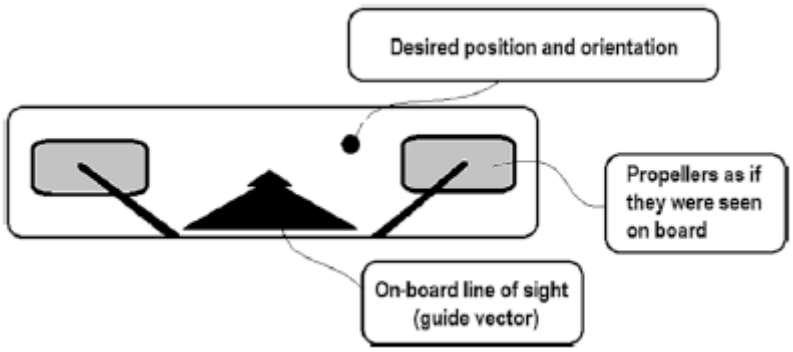
• Drugim jest nie uwzględnianie takiej zmienności, a ręczne zachowanie pojazdu będzie takie, jak po prawej stronie tego samego rysunku.



W tym momencie poznałeś sterowanie trybem robota, które pozwala na płynny lot. Teraz nadszedł czas, aby zobaczyć sterowanie trybem pojazdu, które pozwala na wyższy stopień mobilności i akrobatyczny lub agresywny ruch.

Kontrola trybu pojazdu

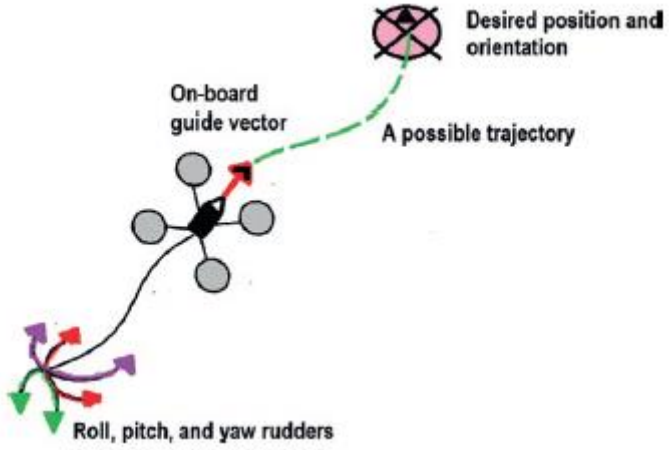
W przypadku analizowania go jako pojazdu, projekt sterowania drona oznacza postrzeganie go tak, jakby był to tryb pierwszej osoby, doświadczenie na pokładzie lub w „kokpicie”.



W tym celu linia widzenia samolotu, zwana również wektorem kierunkowym (wektorem, który miałby pilota na pokładzie), powinna być regulowana za pomocą „sterów” lub „sterów”. W przeciwieństwie do trybu robota lub trybu trzeciej osoby, sprzężenie między kontrolerami obrotowymi i translacyjnymi a dynamiką jest większe lub nawet całkowicie zależne. Jest to matematyczna wada, ale stanowi również zaletę w zakresie wszechstronności i zastosowania do operacji agresywnych lub akrobatycznych. Zacznijmy od sterowania sferycznego (opartego na sterach), a następnie geometrycznego (opartego na sterach strumieniowych).

Sterowanie sferyczne (stery i wektor prowadnic)

Poniżej przedstawiono podstawowe sterowanie trybem pojazdu. Należy pamiętać, że ten typ kontrolera jest szeroko stosowany w robotach kołowych.



W tego typu sterowaniu wygodnie jest przejść z liniowej przestrzeni kartezjańskiej na opartą na współrzędnych sferycznych, gdzie wektor prowadzenia jest powiązany z wypadkową składowych kartezjańskich, a stery z ruchem kątowym tej kuli. Kąty te są rzutami stycznymi składowych kartezjańskich.

Krok 1. Zdefiniuj wypadkową (wektor prowadzący) i kąty steru. Pamiętajmy, że model dynamiczny jest podawany przez

$$\begin{bmatrix} F_{BZ}\left(\sin\theta\cos\psi+\sin\phi\cos\theta\sin\psi\right) \\ F_{BZ}\left(\sin\theta\sin\psi-\sin\phi\cos\theta\cos\psi\right) \\ F_{BZ}\cos\phi\cos\theta \end{bmatrix} + m\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

$$\begin{bmatrix} \tau_\phi \\ \tau_\vartheta \\ \tau_\psi \end{bmatrix} \approx \begin{bmatrix} J_x\ddot{\phi} \\ J_y\ddot{\theta} \\ J_z\ddot{\psi} \end{bmatrix}$$

Zauważ, że dla naszych celów model ten nie zostanie uproszczony do warunków płynnego lotu (linearyzacja). Z poprzednich rozdziałów zauważ, że jest sześć kontrolerów: cztery z nich są sterowane siłą fizyczną lub momentem, a pozostałe dwa są zależne lub zależne od trajektorii:

$$\begin{bmatrix} F_{BZ} \\ \tau_\theta \\ \tau_\phi \\ \tau_\psi \\ \theta_d \\ \phi_d \end{bmatrix}.$$

W przypadku tej nowej konstrukcji regulatory momentu obrotowego w zakresie przechyłu, skoku i odchylenia zostaną pozostawione jak w poprzednich c zęściach:

$$\tau_\psi = PD_\psi$$
$$\tau_\phi = PD_\phi$$
$$\tau_\theta = PD_\theta$$

To daje do zrozumienia że

$$\theta \rightarrow \theta_d$$
$$\phi \rightarrow \phi_d$$
$$\psi \rightarrow \psi_d$$

Pozostałe trzy kontrolery do zdefiniowania:

$$\begin{bmatrix} F_{BZ} \\ \theta_d \\ \phi_d \end{bmatrix}$$

Ponieważ siła ciągu jest powiązana z wektorem kartezjańskim, skojarzymy z nim wektor prowadzący. W odniesieniu do pomocniczych regulatorów przechyłu i skoku (niezbędnych do poruszania się pojazdu w samolocie) skojarzymy je ze sterami, które będą wyznaczały kierunek wektora prowadzenia.

Krok 2. Znajdź wartość wektora prowadzącego i sterów. Przepisujemy translacyjne lub zależne równania dynamiczne pod kątem kontrolerów, które mają być zdefiniowane. Zauważ, że jedyną zmienną niezależną jest psi:

$$\begin{bmatrix} F_{BZ}\left(\sin\theta_d\cos\psi_d + \sin\phi_d\cos\theta_d\sin\psi_d\right) \\ F_{BZ}\left(\sin\theta_d\sin\psi_d - \sin\phi_d\cos\theta_d\cos\psi_d\right) \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix} + m\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = m\begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix}$$

Jak widać, FBZ, jak również pożądane wartości theta i phi, są dostępne w całym układzie równań (trzy niewiadome i trzy równania). Zauważ również, że głównym celem naszych kontrolerów jest to, aby zarówno zmienne translacyjne, jak i ich pochodne zbiegały się do pożądanych wartości. To jest

$$x \rightarrow x_d$$
$$y \rightarrow y_d$$
$$z \rightarrow z_d$$

Rozwiązanie tych równań dla pożądanych sterów i wektora prowadzenia nie jest proste. W ten sposób postępujemy krok po kroku, korzystając z relacji projekcyjnych i wynikowych. Początkowo wygodnie jest uzyskać FBZ za pomocą skalarnego odpowiednika tych równań wektorowych (norma wektorowa). W ten sposób zapisuje się translacyjne równania dynamiczne. Zauważ, że zmienne sterów i wektora prowadzenia zostały oddzielone od zmiennej niezależnej psi:

$$\begin{bmatrix} \cos\psi_d & \sin\psi_d & 0 \\ \sin\psi_d & -\cos\psi_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{BZ}\sin\theta_d \\ F_{BZ}\sin\phi_d\cos\theta_d \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix} = m \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d + g \end{bmatrix}$$

Rozwiązujemy dla sterów i wektora prowadzącego:

$$\begin{bmatrix} F_{BZ}\sin\theta_d \\ F_{BZ}\sin\phi_d\cos\theta_d \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix} = m \begin{bmatrix} \cos\psi_d & \sin\psi_d & 0 \\ \sin\psi_d & -\cos\psi_d & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d + g \end{bmatrix} = \begin{bmatrix} m\ddot{x}_d\cos\psi_d + m\ddot{y}_d\sin\psi_d \\ m\ddot{x}_d\sin\psi_d - m\ddot{y}_d\cos\psi_d \\ m(g+\ddot{z}_d) \end{bmatrix}$$

Jednym ze sposobów uzyskania skalara z wektora jest użycie iloczynu skalarnego:

$$\begin{bmatrix} F_{BZ}\sin\theta_d \\ F_{BZ}\sin\phi_d\cos\theta_d \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix}^T \begin{bmatrix} F_{BZ}\sin\theta_d \\ F_{BZ}\sin\phi_d\cos\theta_d \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix}$$

$$= \left(\cos^2\theta_d\cos^2\phi_d\right)F_{BZ}^2 + \left(\cos^2\theta_d\sin^2\phi_d\right)F_{BZ}^2 + \left(\sin^2\theta_d\right)F_{BZ}^2 = F_{BZ}^2$$

$$\begin{bmatrix} m\ddot{x}_d\cos\psi_d + m\ddot{y}_d\sin\psi_d \\ m\ddot{x}_d\sin\psi_d - m\ddot{y}_d\cos\psi_d \\ m(g+\ddot{z}_d) \end{bmatrix}^T \begin{bmatrix} m\ddot{x}_d\cos\psi_d + m\ddot{y}_d\sin\psi_d \\ m\ddot{x}_d\sin\psi_d - m\ddot{y}_d\cos\psi_d \\ m(g+\ddot{z}_d) \end{bmatrix}$$

$$= m^2(g+\ddot{z}_d)^2 + (m\ddot{x}_d\cos\psi_d + m\ddot{y}_d\sin\psi_d)^2 + (m\ddot{x}_d\sin\psi_d - m\ddot{y}_d\cos\psi_d)^2 = m^2([\ddot{z}_d+g]^2 + \ddot{y}_d^2 + \ddot{x}_d^2)$$

Zatem

$$F_{BZ} = m\sqrt{([\ddot{z}_d + g]^2 + \ddot{y}_d^2 + \ddot{x}_d^2)}$$

Zauważ, że FBZ jest skalarem, który zawiera informacje z naszego wektora przewodniego (jego odległość). Stery muszą być prezentowane w trybie arcus tangens (aby były kompatybilne ze współrzędnymi sferycznymi). Uzyskuje się je z następujących równań. Zauważ, że kąt odchylenia jest niezależnym sterem.

$$\begin{bmatrix} F_{BZ}\sin\theta_d \\ F_{BZ}\sin\phi_d\cos\theta_d \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix} = \begin{bmatrix} m\ddot{x}_d\cos\psi_d + m\ddot{y}_d\sin\psi_d \\ m\ddot{x}_d\sin\psi_d - m\ddot{y}_d\cos\psi_d \\ m(g+\ddot{z}_d) \end{bmatrix}$$

$$\frac{F_{BZ}\sin\phi_d\cos\theta_d}{F_{BZ}\cos\phi_d\cos\theta_d} = \tan\phi_d = \frac{\ddot{x}_d\sin\psi_d - \ddot{y}_d\cos\psi_d}{g+\ddot{z}_d}$$

$$\phi_d = \arctan\left(\frac{\ddot{x}_d\sin\psi_d - \ddot{y}_d\cos\psi_d}{g+\ddot{z}_d}\right)$$

$$(F_{BZ}\sin\phi_d\cos\theta_d)^2 + (F_{BZ}\cos\phi_d\cos\theta_d)^2 = F_{BZ}^2\cos^2\theta_d$$
$$= m^2([\ddot{x}_d\sin\psi_d - \ddot{y}_d\cos\psi_d]^2 + [g+\ddot{z}_d]^2)$$

$$F_{BZ}\cos\theta_d = m\sqrt{([\ddot{x}_d\sin\psi_d - \ddot{y}_d\cos\psi_d]^2 + [g+\ddot{z}_d]^2)}$$

$$\frac{F_{BZ}\sin\theta_d}{F_{BZ}\cos\theta_d} = \tan\theta_d = \frac{\ddot{z}_d\cos\psi_d + \ddot{y}_d\sin\psi_d}{\sqrt{([\ddot{x}_d\sin\psi_d - \ddot{y}_d\cos\psi_d]^2 + [g+\ddot{z}_d]^2)}}$$

$$\theta_d = \arctan\left(\frac{\ddot{x}_d\cos\psi_d + \ddot{y}_d\sin\psi_d}{\sqrt{([\ddot{x}_d\sin\psi_d - \ddot{y}_d\cos\psi_d]^2 + [g+\ddot{z}_d]^2)}}\right)$$

Postać tych równań implikuje współrzędne sferyczne wyrażone w układzie kartezjańskim, gdzie promień jest wielkością wektora prowadzącego FBZ, a kąty sferyczne są określone przez stery (pożądane wartości przechyłu, nachylenia i odchylenia).

Krok 3. Wstrzyknij kontroler. Odbywa się to poprzez zastąpienie znalezionych wartości wielkości wektora prowadzącego i sterów w oryginalnej dynamice translacyjnej następującymi tożsamościami trygonometrycznymi:

$$\cos(\arctan(A)) = \frac{1}{\sqrt{A^2+1}}$$

$$\sin(\arctan(A)) = \frac{A}{\sqrt{A^2+1}}$$

$$\begin{bmatrix} F_{BZ}\left(\sin\theta_d\cos\psi_d + \sin\phi_d\cos\theta_d\sin\psi_d\right) \\ F_{BZ}\left(\sin\theta_d\sin\psi_d - \sin\phi_d\cos\theta_d\cos\psi_d\right) \\ F_{BZ}\cos\phi_d\cos\theta_d \end{bmatrix} + m\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

Uzyskuje się:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix}$$

Na przykład (jest to żmudny, ale konieczny przykład, który można opracować za pomocą programu do obliczeń numerycznych, takiego jak MATLAB),

$$m\sqrt{\ddot{x}_d^2+\ddot{y}_d^2+(\ddot{z}_d+g)^2}\,\frac{1}{\sqrt{1+\left(\frac{\ddot{x}_d\sin\psi-\ddot{y}_d\cos\psi}{\ddot{z}_d+g}\right)^2}}\,\frac{1}{\sqrt{1+\left(\frac{\ddot{x}_d\cos\psi+\ddot{y}_d\sin\psi}{\sqrt{(\ddot{z}_d\sin\psi-\ddot{y}_d\cos\psi)^2+(\ddot{z}_d+g)^2}}\right)^2}}$$

$$= F_{BZ}\cos\theta_d\cos\phi_d$$

$$\frac{1}{1+\frac{\ddot{x}_d^2\sin^2\psi+\ddot{y}_d^2\cos^2\psi-2\ddot{x}_d\ddot{y}_d\sin\psi\cos\psi}{(\ddot{z}_d+g)^2}}\,\frac{1}{1+\frac{\ddot{x}_d^2\cos^2\psi+\ddot{y}_d^2\sin^2\psi+2\ddot{x}_d\ddot{y}_d\sin\psi\cos\psi}{(\ddot{x}_d\sin\psi-\ddot{y}_d\cos\psi)^2+(\ddot{z}_d+g)^2}} = \cos^2\theta_d\cos^2\phi_d$$

$$= \frac{(g+\ddot{z}_d)^2}{g^2+2g\ddot{z}_d+\ddot{x}_d^2+\ddot{y}_d^2+\ddot{z}_d^2}$$

$$F_{BZ}^2\cos^2\theta_d\cos^2\phi_d = \frac{(g+\ddot{z}_d)^2}{g^2+2g\ddot{z}_d+\ddot{x}_d^2+\ddot{y}_d^2+\ddot{z}_d^2}m^2\left(\ddot{x}_d^2+\ddot{y}_d^2+(\ddot{z}_d+g)^2\right) = m^2(g+\ddot{z}_d)^2$$

$$F_{BZ}\cos\theta_d\cos\phi_d = m(\ddot{z}_d+g)$$

W konsekwencji

$$m(\ddot{z}_d+g) - mg = m\ddot{z}$$
$$\ddot{z} = \ddot{z}_d$$

W ten sam sposób można uzyskać wyniki dla dynamiki X i Y. Oznacza to, że można zaproponować, co następuje:

$$\begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix} = \begin{bmatrix} PD_x + \ddot{x}_d \\ PD_y + \ddot{y}_d \\ PD_z + \ddot{z}_d \end{bmatrix}$$

z tym

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} - \begin{bmatrix} PD_x + \ddot{x}_d \\ PD_y + \ddot{y}_d \\ PD_z + \ddot{z}_d \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} PDx + (\ddot{x}_d - \ddot{x}) \\ PDy + (\ddot{y}_d - \ddot{y}) \\ PDz + (\ddot{z}_d - \ddot{z}) \end{bmatrix} = \begin{bmatrix} K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x}) + (\ddot{x}_d - \ddot{x}) \\ K_{Py}(y_d - y) + K_{Dy}(\dot{y}_d - \dot{y}) + (\ddot{y}_d - \ddot{y}) \\ K_{Pz}(z_d - z) + K_{Dz}(\dot{z}_d - \dot{z}) + (\ddot{z}_d - \ddot{z}) \end{bmatrix}$$

$$x \to x_d$$
$$y \to y_d$$
$$z \to z_d$$

W praktyce i we wcześniej opisanych warunkach płynnego lotu można po prostu powiedzieć, że

$$\begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix} = \begin{bmatrix} PD_x \\ PD_y \\ PD_z \end{bmatrix}$$

Podsumowując, wektor prowadnicy i stery są wstrzykiwane w następujący sposób:

$$\begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix} = \begin{bmatrix} PD_x \\ PD_y \\ PD_z \end{bmatrix}$$

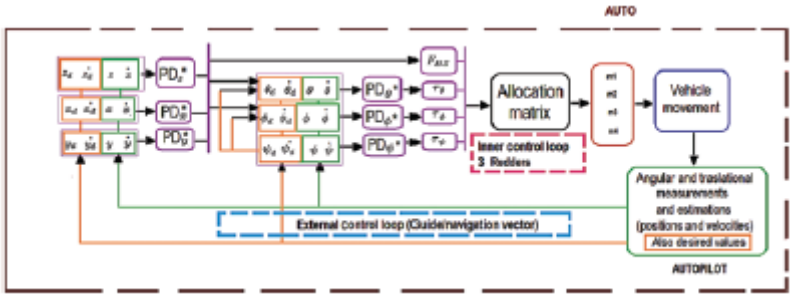
$$\phi_d = \arctan\left(\frac{\ddot{x}_d \sin\psi_d - \ddot{y}_d \cos\psi_d}{g + \ddot{z}_d}\right)$$

$$\theta_d = \arctan\left(\frac{\ddot{x}_d \cos\psi_d + \ddot{y}_d \sin\psi_d}{\sqrt{([\ddot{x}_d \sin\psi_d - \ddot{y}_d \cos\psi_d]^2 + [g + \ddot{z}_d]^2)}}\right)$$

$$F_{BZ} = m\sqrt{([\ddot{z}_d + g]^2 + \ddot{y}_d^2 + \ddot{x}_d^2)}$$

$$\tau_\psi = PD_\psi$$
$$\tau_\phi = PD_\phi$$
$$\tau_\theta = PD_\theta$$

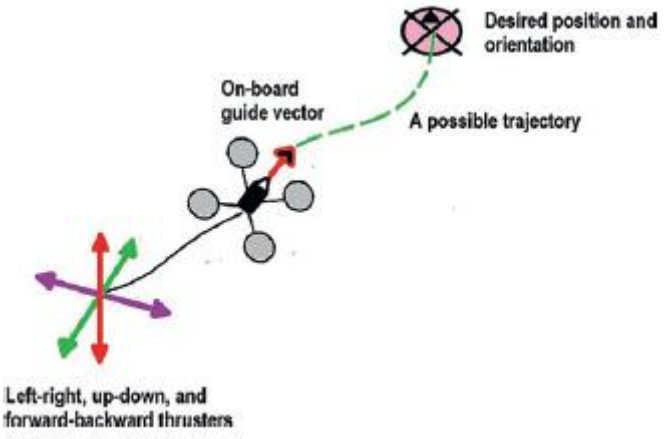
Mówimy, że jest to sterowanie trybem pojazdu, sterowanie trybem pierwszej osoby lub sterowanie trybem pokładowym, ponieważ tak zachowywałby się pilot w kokpicie. Dodatkowo ten rodzaj sterowania jest używany przez dziesięciolecia w robotach kołowych. Rysunek ilustruje ten kontroler jako schemat (pamiętaj, że proponowane tutaj kontrole mogą być dowolne inne niż PD, ale struktura jest ogólnie taka sama).



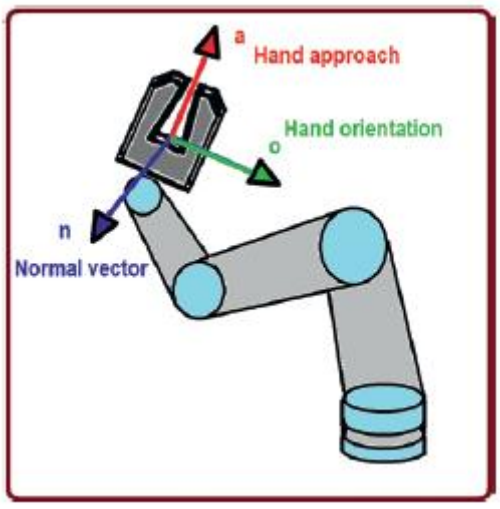
Włączenie pilota zdalnego sterowania można wykonać zgodnie z opisem w poprzednich sekcjach. Dowiedziałeś się o sterowaniu sferycznym dla dronów. Teraz nauczymy Cię jednego z najpotężniejszych dostępnych kontrolerów, geometrycznego.

Wprowadzenie do kontroli geometrycznej dla dronów (stery strumieniowe i wektor prowadzący)

Poprzedni przypadek opierał się na rozważaniach trygonometrycznych. W tej sekcji wyjaśniono bardziej zaawansowany sterownik oparty na teorii macierzy rotacji. Tym razem pojazd nie będzie miał wektora prowadzącego i trzech sterów (tak jak w poprzednim przypadku psi liczone jako niezależny ster, a także phi i theta). W przypadku tej sekcji pojazd będzie miał wektor kierunkowy i trzy stery strumieniowe.



Wyobraź sobie, że zmieniliśmy pokładowe „kierownice” lub „stery” za pomocą poleceń, które aktywują rodzaj „sterów strumieniowych” przód-tył, góra-dół i lewo-prawo. Jest to również szeroko stosowane w robotyce i jest związane z wektorami noa do korygowania lub opisywania orientacji ciała za pomocą trzech wektorów ortogonalnych; dwa z nich można zaprojektować, a tylko jeden jest niezależny od pozostałych. Zauważ, że wektory noa mają te nazwy ze względu na ich użycie w robotyce manipulatorów, a konkretnie w chwytaku lub ręce robota, gdzie a jest wektorem jednostkowym reprezentującym podejście chwytaka, n jest wektorem normalnym względem a i o jest również wektorem normalnym do a i n, który służy do orientacji chwytaka.



Ta sekcja wymaga od Ciebie dobrej znajomości następujących narzędzi (wszystkie przydatne właściwości są wskazywane w miarę ich używania, ale powinieneś przeczytać o nich trochę więcej): normy i odległości w przestrzeniach metrycznych (wartość bezwzględna, odległość kwadratowa itp. .); niezmienniki macierzy (ślad, wyznacznik itp.); macierze rotacyjne i ich własności, w tym ich własności pochodne; transpozycja macierzy i jej własności; ślad matrycy; oraz równoważny iloczyn macierzy

iloczynu krzyżowego znany jako iloczyn kapelusza i jego odwrotna operacja zwana iloczynem vee. Zacznijmy od części translacyjnej.

Kontrola translacji

Ta kontrola wykorzystuje alternatywną reprezentację kinematyczną opartą na macierzach obrotu, a także używa nielinearyzowanego ogólnego modelu dynamicznego. To jest

$$R\begin{bmatrix}0\\0\\F_{BZ}\end{bmatrix}+m\begin{bmatrix}0\\0\\-g\end{bmatrix}=m\begin{bmatrix}\ddot{x}\\\ddot{y}\\\ddot{z}\end{bmatrix}$$

$$\begin{bmatrix}\tau_{BX}\\\tau_{BY}\\\tau_{BZ}\end{bmatrix}=\begin{bmatrix}J_x\dot{\omega}_x\\J_y\dot{\omega}_y\\J_z\dot{\omega}_z\end{bmatrix}+\begin{bmatrix}\omega_y\omega_z J_z-\omega_y\omega_z J_y\\\omega_x\omega_z J_x-\omega_x\omega_z J_z\\\omega_x\omega_y J_y-\omega_x\omega_y J_x\end{bmatrix}$$

$$\omega=\begin{bmatrix}\omega_{xB}\\\omega_{yB}\\\omega_{zB}\end{bmatrix}=\begin{bmatrix}c\theta & 0 & -c\phi s\theta\\0 & 1 & s\phi\\s\theta & 0 & c\phi c\theta\end{bmatrix}\begin{bmatrix}\dot{\phi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix}\Rightarrow\boxed{\dot{R}=RS}$$

Zamiast używać kinematyki kątowej Taita-Bryana (wówczas sterowanie opierało się na trygonometrii), teraz model kinematyczny opiera się na macierzach obrotu, a sterowanie na właściwościach tych macierzy. Równoważność jest naprawdę prosta. Pamiętaj tylko, że z macierzy S można uzyskać elementy prędkości kątowej w układzie ciała i ich związek z prędkościami przechyłu, pochylenia i odchylenia w układzie świata:

$$S(\omega)=\begin{bmatrix}0 & -\omega_z & \omega_y\\\omega_z & 0 & -\omega_x\\-\omega_y & \omega_x & 0\end{bmatrix}=\dot{R}R^T$$

Możesz zastanawiać się nad zaletami zmiany modelu operacją opartą na sterach do tej opartej na sterach strumieniowych, a my przewidujemy, że jest to niezwykle przydatne w przypadkach, gdy kąty ciała nie są małe (jest to przeciwieństwo rozważania płynnego trybu lotu, który zakładaliśmy we wszystkich naszych poprzednie projekty). Innymi słowy, jest przydatny do ruchów agresywnych i akrobatycznych. Zgodnie z poprzednim akapitem zauważ, że nie możemy pracować z kątami Eulera w sposób bezpośredni, ale pośrednio, używając ich równoważnej macierzy rotacji. Sterowanie dynamiką translacyjną wykorzystuje pojęcie znane w teorii sterowania jako anihilator. Zauważ, że w artykułach Tae Young Lee, na których oparliśmy naszą analizę, osie mają przeciwny sens w stosunku do pozytywnych wartości, które tutaj rozważamy. Pamiętaj o tych odwróconych osiach, aby uzyskać korelację między tym tekstem a wyżej wymienionymi artykułami. Kontroler ciągu, który zawiera informacje o wektorze prowadzącym, wygląda następująco:

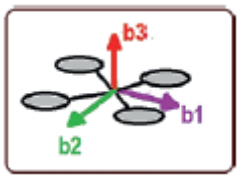
$$\begin{aligned}F_{BZ}&=\left(\begin{bmatrix}K_{Px}(x_d-x)+K_{Dx}(\dot{x}_d-\dot{x})+m\ddot{x}_d\\K_{Py}(y_d-y)+K_{Dy}(\dot{y}_d-\dot{y})+m\ddot{y}_d\\K_{Pz}(z_d-z)+K_{Dz}(\dot{z}_d-\dot{z})+m\ddot{z}_d\end{bmatrix}+\begin{bmatrix}0\\0\\mg\end{bmatrix}\right)\cdot\left[R\begin{bmatrix}0\\0\\1\end{bmatrix}\right]\\&=C_{xyz}\cdot\left(R\begin{bmatrix}0\\0\\1\end{bmatrix}\right)\end{aligned}$$

Obserwuj obecność iloczynu skalarnego, ponieważ FBZ jest wielkością skalarną. Zwróć także uwagę na obecność wektora jednostkowego [0 0 1]. To tutaj znajduje się FBZ na ramie drona. Ograniczeniem dla poprzedniej macierzy rotacji jest

$$R \rightarrow R_d$$

$$R_d = \begin{bmatrix} b_{1d} & b_{2d} & b_{3d} \end{bmatrix} = \begin{bmatrix} b_{1dx} & b_{2dx} & b_{3dx} \\ b_{1dy} & b_{2dy} & b_{3dy} \\ b_{1dz} & b_{2dz} & b_{3dz} \end{bmatrix}$$

Później zobaczysz, że te wektory b, które są składnikami pożądanej macierzy rotacji, są używane jako typ wektora noa, gdzie b3 jest równoważne wektorowi a, b2 do o i b1 do n



Zauważ, że

$$b_{3d} = R_d \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b_{1dx} & b_{2dx} & b_{3dx} \\ b_{1dy} & b_{2dy} & b_{3dy} \\ b_{1dz} & b_{2dz} & b_{3dz} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

gdzie nasze pierwsze ograniczenie projektowe jest następujące:

$$b_3 = R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \frac{C_{xyz}}{|C_{xyz}|}$$

Oznacza to trzy rzeczy:

• Po pierwsze, konieczne jest zaprojektowanie sterowania dla macierzy rotacji, tak aby zachowywała się jak pożądana macierz rotacji. Zostanie to zrobione w dalszej części tej sekcji.

• Po drugie, wektor prowadzący pojazdu jest powiązany z ciągiem, który jest powiązany z osią Z pojazdu lub b3.

• Po trzecie, wektor b3 jest znormalizowany, aby nie przekroczyć maksymalnych wartości dozwolonych w macierzy rotacji (ortonormalność).

Dzieją się w tym samym czasie, jak następuje. Korzystając z faktu, że FBZ jest skalarem, przepisujemy nasz model dynamiczny w następujący sposób:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R \begin{bmatrix} 0 \\ 0 \\ F_{BZ} \end{bmatrix} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = \boxed{F_{BZ}} R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$$

Zastępujemy ciąg, który zawiera wektor prowadzący w naszej dynamice translacyjnej:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = C_{xyz} \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$$

Wydawałoby się, że to zachowanie nieistotne, ale teraz dzielimy przez 1 wyraz zawierający macierz rotacji. Gdzie 1 jest równe (pamiętaj, że odwrotność macierzy rotacji pomnożona przez wspomnianą macierz rotacji jest macierzą jednostkową)

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T R^T R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 1$$

Tą drogą

$$C_{xyz} \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \frac{C_{xyz} \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T R^T R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}$$

Następnie przepisuję (pamiętaj o naszym pierwszym ograniczeniu)

$$C_{xyz} = R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} |C_{xyz}|$$

$$R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \frac{C_{xyz}}{|C_{xyz}|}$$

$$\frac{C_{xyz} \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T R^T R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}$$

I wreszcie (tu używamy relacji między iloczynem skalarnym a operatorem transpozycji)

$$\frac{R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} |C_{xyz}| \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)}{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T R^T R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}} \frac{C_{xyz}}{|C_{xyz}|} = \frac{\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T R^T \right) |C_{xyz}| \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)}{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T R^T R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}} \frac{C_{xyz}}{|C_{xyz}|} = C_{xyz}$$

Tak więc równanie pętli zamkniętej translacyjnego układu dynamicznego staje się

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = C_{xyz} \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = C_{xyz} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$$

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x}) + m\ddot{x}_d \\ K_{Py}(y_d - y) + K_{Dy}(\dot{y}_d - \dot{y}) + m\ddot{y}_d \\ K_{Pz}(z_d - z) + K_{Dz}(\dot{z}_d - \dot{z}) + m\ddot{z}_d \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + m \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$$

Prowadzi to do powstania trzech masowych systemów amortyzatorów sprężynowych, a odpowiedni dobór wartości proporcjonalnych i różnicowych implikuje, że

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x}) + m(\ddot{x}_d - \ddot{x}) \\ K_{Py}(y_d - y) + K_{Dy}(\dot{y}_d - \dot{y}) + m(\ddot{y}_d - \ddot{y}) \\ K_{Pz}(z_d - z) + K_{Dz}(\dot{z}_d - \dot{z}) + m(\ddot{z}_d - \ddot{z}) \end{bmatrix}$$

$$x \to x_d$$
$$y \to y_d$$
$$z \to z_d$$

Teraz przejdźmy do części obrotowej.

Kontrola obrotowa

Nadszedł czas na najtrudniejszą część tego kontrolera: zaprojektowanie sterowania orientacją poprzez regulację matrycy rotacji. W tym celu uważamy, że wszystkie sterowniki z zamkniętą pętlą są oparte na błędzie. Błąd to odległość między wartością pożądaną a wartością zmierzoną. W przypadku wielkości skalarnych odległość ta jest po prostu odejmowaniem. W podobny sposób zdefiniujemy pojęcie odległości między dwiema macierzami rotacji (mierzoną i pożądaną). Na początek wiemy już, że macierz R jest zdefiniowana przez kąty Eulera jako

$$\boldsymbol{R = R_{z_\phi} R_{x_\phi} R_{y_\theta}}$$

$$\boldsymbol{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + s\phi s\psi c\theta \\ s\psi c\theta + s\phi c\psi s\theta & c\phi c\psi & s\psi s\theta - s\phi c\psi c\theta \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

oraz fakt, że pożądana macierz zawiera trzy „silniki”. Dzieje się tak, ponieważ w macierzy rotacji każda z jej kolumn reprezentuje zachowanie osi kartezjańskiej. Ponadto, jak wspomniano, każda kolumna jest wektorem jednostkowym spełniającym warunek wektora normalnego, a warunek ortogonalny spełnia następujące definicje:

$$R_d = \begin{bmatrix} b_{1d} & b_{2d} & b_{3d} \end{bmatrix}$$

$$b_{1d} = b_{2d} \times b_{3d}$$
$$b_{2d} = \frac{b_{3d} \times B_{des}}{|b_{3d} \times B_{des}|}$$
$$b_{3d} = \frac{C_{xyz}}{|C_{xyz}|}$$

Jak wspomniano, wektory te są wektorami typu noa i dla ich definicji bardzo ważne jest, aby były względem siebie ortonormalne. To wyjaśnia obecność produktów krzyżowych i znormalizowanych ilości. Wreszcie, w przypadku b1d, ponieważ jest to iloczyn krzyżowy dwóch znormalizowanych lub unitarnych wektorów, jest on również znormalizowany lub unitarny. Zauważ, że jedynymi niezależnymi wektorami są Bdes i b3d, a jedynym, który można zaprojektować, jest Bdes, ponieważ b3d musi spełniać nasz warunek ograniczający, aby regulować FBZ. W przypadku b1d wymusza spełnienie ortonormalności dwoma pozostałymi wektorami. Przykładem Bdes może być

$$B_{des} = \begin{bmatrix} \cos\psi_d & \sin\psi_d & 0 \end{bmatrix}$$

Wektor ten jest powiązany z b1d i b2d i umożliwia np. obrót wokół osi ZB (wymuszona, ale funkcjonalna regulacja odchylenia, nie do końca przeprowadzona z powodu interakcji w b1d i b2d z b3d). Po zdefiniowaniu zmierzonej macierzy rotacji i pożądanej macierzy rotacji pamiętajmy o kilku pojęciach dotyczących odległości. Odległość między dwoma skalarami to odejmowanie. Odległość między dwoma wektorami jest m.in. kwadratową normą wektorów (pierwiastek kwadratowy z sumy jego odejmowań skalarnych w drugiej potędze). W przypadku macierzy istnieje również kilka definicji odległości. Posłużymy się tutaj normą Frobeniusa ze względu na jej niezmienne własności (ślad i wyznacznik są wartościami niezmiennymi w macierzy, a norma Frobeniusa opiera się na śladach). Jak widać, wszystkie odległości są albo funkcją skalarną, albo skalarną. Jeśli a i b są wartościami skalarnymi, odległość wynosi

$$a, b = C, C_d$$
$$d_{ab} = |a - b| = |C - C_d|$$

Jeśli są wektorami, bardzo powszechnym typem odległości jest

$$a = [a_x, a_y, a_n] = [C_x, C_y, C_n]$$
$$b = [b_x, b_y, b_n] = [C_{dx}, C_{dy}, C_{dn}]$$
$$d_{ab} = \sqrt{(C_x - C_{dx})^2 + (C_y - C_{dy})^2 + (C_z - C_{dz})^2} = \sqrt{(a-b)^T (a-b)}$$

A składowe wektora (indywidualne odejmowanie) można uzyskać w ten sposób (przydatne jest poznanie tych składowych w celu zaprojektowania kontroli trójwymiarowej):

$$\nabla f(d_{ab}) \rightarrow \nabla(\frac{d_{ab}^2}{2}) = \begin{bmatrix} C_x - C_{dx} \ , \ C_y - C_{dy} \ , \ C_z - C_{dz} \end{bmatrix}$$

gdzie użyty tutaj operator (trójkąt odwrócony znany również jako nabla lub del) jest znany jako gradient i jest rodzajem pochodnej kierunkowej, która uzyskuje składowe wektora z funkcji skalarnej zależnej od kilku zmiennych (w tym przypadku jest to funkcja odległość, która jest już skalarem zależnym od kilku zmiennych). Jeśli a i b są macierzami, odległość Frobeniusa wynosi

$$a = R$$
$$b = R_d$$
$$d_{ab} = Frob(R - R_d) = \sqrt{trace\left[(R - R_d)^T (R - R_d)\right]}$$

Zauważ, że sekwencja odejmowania jest rozszerzona wymiarowo. Wykonywanie następujących operacji (dotyczy macierzy rotacji i używania śladowych właściwości produktu)

$$\begin{aligned} trace((R^T - R_d^T)(R - R_d)) &= trace(R_d^T R_d + R^T R - R_d^T R - R^T R_d) \\ &= trace(2(I - R_d^T R)) = 2trace(I - R_d^T R) \end{aligned}$$

jest uzyskiwany

$$d_{ab} = \sqrt{2trace(I - R_d^T R)}$$

W tym przypadku, ponieważ mamy skalar i musimy zaprojektować trójwymiarową kontrolkę orientacji, potrzebujemy sposobu na uzyskanie trójwymiarowego wektora ze wspomnianej definicji odległości.

Pamiętajmy, że po określeniu odległości pomiędzy naszą zmierzoną matrycą a pożądaną matrycą, musimy wygenerować z niej trzy błędy, aby wygenerować trzy momenty obrotowe pojazdu. Jednym ze sposobów jest zastosowanie pojęcia pochodnej kierunkowej (tu użyjemy rodzaju pochodnej kierunkowej zwanej trywializowana prawą pochodną) i biorąc pod uwagę, że nasza odległość jest funkcją skalarną kilku zmiennych, gdy wyprowadzimy kierunkowo, otrzymamy wyrażenie wektorowe w podobnie jak wtedy, gdy zastosowaliśmy gradient (nabla) do odległości między dwoma wektorami. Zbanalizowana prawidłowa pochodna jest szeroko stosowana w tego typu kontrolerze, ale biorąc pod uwagę jej złożoność interpretacji, użyjemy jej tylko jako narzędzia. Więcej informacji na ten temat można znaleźć w dokumentach Bullo i Taeyoung Lee w Dodatku. Należy zauważyć, że wartość 1/4 jest tutaj użyta dla wygody w taki sposób, że wynik końcowy jest wyrażany jako 1/2 i jest to szczególnie przydatne w nieliniowej kontroli w celu wyeliminowania stałych wartości poprzez właściwości macierzy lub pochodne. Możesz jednak użyć dowolnego innego mnożnika. Wynik zmienia tylko wartości, które mają być przeciągane podczas procedury.

$$D_R f(d_{ab}) = D_R\left[\tfrac{1}{4}d_{ab}^2\right] = D_R\left[\tfrac{1}{2}trace(I - R_d^T R)\right] = -\tfrac{1}{2}trace(R_d^T D_R R) \\ = -\tfrac{1}{2}trace(R_d^T RS(n))$$

Zwróć uwagę na trzy rzeczy:

• W przypadku śladu, ponieważ jest to operacja liniowa, pochodną śladu jest ślad pochodnej.

• Wykorzystujemy również fakt, że pochodna generyczna macierzy rotacji daje macierz rotacji pomnożoną przez macierz skośno-symetryczną (S). Należy jednak zauważyć, że wspomniana macierz antysymetryczna nie jest taka sama jak poprzednia opisana w Części 2. Jest tak dlatego, że była pochodną względem czasu i jest pochodną względem kierunku zmian. Jednak generowana jest macierz antysymetryczna zależna od tempa zmian (n).

• Na koniec zauważ, że pochodna macierzy jednostkowej wynosi zero, a pozostała pochodna wpływa tylko na wyraz macierzy rotacji, gdzie Rd jest uważane za „stałą”, ponieważ jest niezależna od R (ta pochodna jest wykonywana z uwzględnieniem do R). Podsumowując, wykorzystano następujące właściwości:

$$D(trace\,[M]) = trace(D\,[M])$$
$$\dot{R} = RS(w) \neq D_R R = RS(n)$$

Termin S(n) jest reprezentacją wskazującą na obecność wektora n, który jest przekształcany w macierz skośną w celu uniknięcia użycia produktów krzyżowych i umożliwienia użycia produktów macierzy.

Od teraz będziemy używać notacji hat i vee. Notacją równoważną S(n) jest operator kapelusza, który implikuje uzyskanie macierzy skośnej z wektora.

$$n = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \qquad S(n) = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

$$S(n) = \hat{n}$$
$$D_R f(d_{ab}) = -\tfrac{1}{2}trace(R_d^T RS(n)) = -\tfrac{1}{2}trace(R_d^T R\hat{n})$$

Jeśli przyjrzysz się uważnie, zobaczysz, że nie mamy jeszcze wektora, którego moglibyśmy użyć, ponieważ ślad jest skalarem i nawet produkt, na który nakładany jest ślad, jest macierzą. Istnieją jednak trzy przydatne fakty. Po pierwsze, istnieje następująca tożsamość:

$$
\begin{aligned}
trace[\hat{r}\hat{c}] &= trace\left(\begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}\begin{bmatrix} 0 & -c_z & c_y \\ c_z & 0 & -c_x \\ -c_y & c_x & 0 \end{bmatrix}\right) \\
&= trace\begin{bmatrix} -c_y r_y - c_z r_z & c_x r_y & c_x r_z \\ c_y r_x & -c_x r_x - c_z r_z & c_y r_z \\ c_z r_x & c_z r_y & -c_x r_x - c_y r_y \end{bmatrix} \\
&= -2c_x r_x - 2c_y r_y - 2c_z r_z \\
&= -2(\vec{c}\cdot\vec{r}) = -2\vec{c}^T\vec{r}
\end{aligned}
$$

W tym celu używamy operatora vee, który jest odwrotnością operatora hat i implikuje uzyskanie wektora z macierzy skośnej:

$$\check{S}(n) = n = \check{\hat{n}}$$

Po drugie, następującą macierz i ogólnie każdą inną macierz można rozłożyć jako sumę części skośnej i części symetrycznej w następujący sposób:

$$R_d^T R = \frac{R_d^T R - (R_d^T R)^T}{2} + \frac{R_d^T R + (R_d^T R)^T}{2}$$

W ten sposób

$$-\tfrac{1}{2} trace(R_d^T R\hat{n}) = -\tfrac{1}{2} trace(\tfrac{R_d^T R - (R_d^T R)^T}{2}\hat{n} + \tfrac{R_d^T R + (R_d^T R)^T}{2}\hat{n})$$

Trzecim faktem jest to, że ślad iloczynu macierzy symetrycznej przez skośną symetryczną wynosi zero, więc przetrwa tylko ślad iloczynu macierzy skośnych:

$$-\tfrac{1}{2} trace(\tfrac{R_d^T R - (R_d^T R)^T}{2}\hat{n} + \tfrac{R_d^T R + (R_d^T R)^T}{2}\hat{n}) = -\tfrac{1}{4} trace(\left[R_d^T R - (R_d^T R)^T\right]\hat{n})$$

Używając równości, która wiąże ślad iloczynu dwóch macierzy antysymetrycznych z iloczynem skalarnym ich równoważnych wektorów vee, otrzymujemy, że

$$-\tfrac{1}{4} trace(\left[R_d^T R - (R_d^T R)^T\right]\hat{n}) = \frac{(R_d^T R - R^T R_d\check{)}\cdot n}{2}$$

Wreszcie wektor błędu, którego szukamy, to:

$$e_R = \frac{(R_d^T R - R^T R_d\check{)}}{2}$$

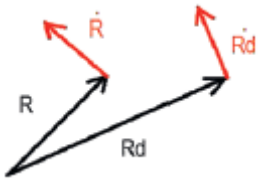
Aby sformułować naszą kontrolę orientacji (znaną również jako kontrola orientacji), oprócz składowej błędu orientacji (w tym przypadku podanej przez macierze obrotu, która służy do włączenia części proporcjonalnej), wymagana jest składowa prędkości kątowej (w celu włączenia części pochodnej). Odbywa się to w następujący sposób. Pierwszą próbą jest zaproponowanie

$$e_\omega = \omega - \omega_d$$

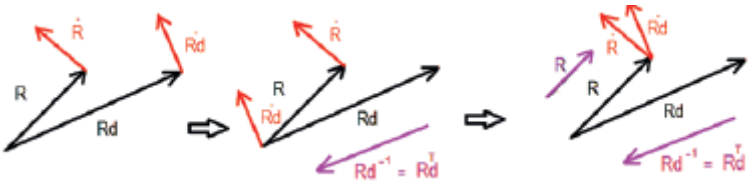
Prosta logika wskazuje, że te prędkości kątowe (mierzona i referencyjna) są wektorami, a to, co wyjaśniono w tym równaniu, można bezpośrednio zastosować:

$$\nabla f(d_{ab}) \rightarrow \nabla(\frac{d_{ab}^2}{2}) = \left[\; C_x - C_{dx} \; , \; C_y - C_{dy} \; , \; C_z - C_{dz} \; \right]$$

Jednak pożądana prędkość kątowa i zmierzona nie mogą być bezpośrednio odejmowane, ponieważ leżą w różnych przestrzeniach. W ten sposób zastosujemy dedukcję na podstawie pochodnych macierzy rotacji



W uproszczeniu wskazujemy, że prędkość macierzy rotacji wynosi styczną do siebie. Zauważ, że ani macierz rotacji, ani jej pochodna nie są wektorami, ale macierzami, chociaż możemy użyć ich wektorów własnych, aby je sobie wyobrazić i przedstawić graficznie. Matematyczny dowód prostopadłości pochodnej macierzy obrotu względem jej macierzy obrotu daje równanie pochodnej macierzy obrotu, w którym występuje macierz skośno-symetryczna zależna od prędkości kątowej, a ponieważ ta macierz antysymetryczna jest odpowiednikiem iloczynu krzyżowego, wynikiem jest macierz ortogonalna. Z faktu, że pożądana macierz rotacji nie leży w tej samej przestrzeni co zmierzona macierz rotacji, konieczne jest wykonanie procesu pokazanego na rysunku



Wynik tej sekwencji graficznej daje następujące równanie:

$$\dot{R} - \dot{R}_d(R_d^T R)$$

Ponieważ wymagana jest różnica między prędkościami kątowymi, musimy przepisać poprzednie równanie w ten sposób:

$$\dot{R} - \dot{R}_d(R_d^T R) = RS(\omega) - R_d S(\omega_d) R_d^T R$$

Jeśli drugi wyraz jest pomnożony przez macierz jednostkową (pamiętaj, że jest to równoważne pomnożeniu przez 1),

$$RR^T = I$$

$$RS(\omega) - R_d S(\omega_d) R_d^T R = RS(\omega) - RR^T R_d S(\omega_d) R_d^T R$$

Równanie zostało przepisane w postaci R:

$$\dot{R} - \dot{R}_d(R_d^T R) = R\left[S(\omega) - R^T R_d S(\omega_d) R_d^T R\right]$$

Można to przepisać w następujący sposób (właściwość transpozycji produktu):

$$R\left[S(\omega) - R^T R_d S(\omega_d) R_d^T R\right] = R\left[S(\omega) - R^T R_d S(\omega_d)\left(R^T R_d\right)^T\right]$$

Istnieją dwie przydatne tożsamości. Zauważ, że aby uniknąć użycia ogromnego symbolu kapelusza, został on umieszczony na końcu nawiasów, a to oznacza, że należy go zastosować do wyniku operacji wewnątrz nawiasów:

$$R(x \times y) = Rx \times Ry = RS(x)y$$
$$R\hat{x}R^T = RS(x)R^T = R(x \times R^T) = Rx \times I = S(Rx)I = S(Rx) = (Rx)\hat{}$$

Wykorzystując te tożsamości w rozwijanym przez nas równaniu (konkretnie w jego drugim członie), mamy

$$R\left[S(\omega) - R^T R_d S(\omega_d)\left(R^T R_d\right)^T\right] = R\left[\hat{\omega} - \left(R^T R_d \omega_d\right)\hat{}\right]$$

Ponieważ suma dwóch macierzy antysymetrycznych to kolejna macierz skośno-symetryczna

$$R\left[\hat{\omega} - \left(R^T R_d \omega_d\right)\hat{}\right] = R(\omega - R^T R_d \omega_d)\hat{}$$

szukany błąd prędkości kątowej to po prostu wektor

$$e_\omega = \omega - R^T R_d \omega_d$$

Zauważ, że w_d nie jest wartością arbitralną i należy ją uzyskać jako zależność od R_d w następujący sposób:

$$S(\omega_d) = R_d^T \dot{R}_d$$

Zauważ, że w modelu naszej dynamiki rotacyjnej mamy wyraz

$$\dot{\omega}$$

Jego pośrednie wykorzystanie (zostanie to wkrótce skompensowane) uzyskuje się w ten sposób:

$$e_\omega = \omega - R^T R_d \omega_d$$
$$\dot{e}_\omega = \dot{\omega} - \dot{R}^T R_d \omega_d - R^T \dot{R}_d \omega_d - R^T R_d \dot{\omega}_d$$
$$\dot{e}_\omega = \dot{\omega} - (RS(\omega))^T R_d \omega_d - R^T R_d S(\omega_d)\omega_d - R^T R_d \dot{\omega}_d$$

Należy zauważyć, że

$$S(\omega_d)\omega_d = \omega_d \times \omega_d = 0$$

Tą drogą

$$\dot{e}_\omega = \dot{\omega} - (RS(\omega))^T R_d \omega_d - R^T R_d \dot{\omega}_d$$
$$\dot{e}_\omega = \dot{\omega} - (S^T(\omega) R^T) R_d \omega_d - R^T R_d \dot{\omega}_d$$

Stosując definicję macierzy skośno-symetrycznej:

$$S^T = -S$$
$$\dot{e}_\omega = \dot{\omega} + S(\omega) R^T R_d \omega_d - R^T R_d \dot{\omega}_d$$

Niedługo wykorzystamy ten wynik. Wracając do naszej pierwotnej dynamiki obrotowej, można ją przepisać w następujący sposób:

$$\begin{bmatrix} \tau_{BX} \\ \tau_{BY} \\ \tau_{BZ} \end{bmatrix} = \begin{bmatrix} J_x \dot{\omega}_x \\ J_y \dot{\omega}_y \\ J_z \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_y \omega_z J_z - \omega_y \omega_z J_y \\ \omega_x \omega_z J_x - \omega_x \omega_z J_z \\ \omega_x \omega_y J_y - \omega_x \omega_y J_x \end{bmatrix}$$
$$\tau = J\dot{\omega} + \omega \times J\omega$$

Korzystając z definicji błędu obrotu i błędu prędkości kątowej, które wydedukowaliśmy wcześniej, możemy zaproponować następującą kontrolę:

$$\tau = -K_R e_R - K_\omega e_\omega + \omega \times J\omega$$

gdzie KR i Kw są skalarne (pomysł polega na identycznym przeskalowaniu odpowiednich wektorów błędu, aby zachować ich ortonormalne proporcje). Jeśli zamienimy ten kontroler w oryginalne równanie, otrzymamy

$$-K_R e_R - K_\omega e_\omega + \omega \times J\omega = J\dot{\omega} + \omega \times J\omega$$
$$-K_R e_R - K_\omega e_\omega = \boxed{J\dot{\omega}}$$

To równanie wymaga dynamicznej kompensacji zawartego terminu. Nauczymy Cię, jak to zrobić w następnej sekcji.

Kompensacja dynamiki i dodatkowe uwagi

Jak widać, człon pochodny prędkości kątowej pozostaje w równaniu. Jak już wcześniej zrobiliśmy, PD może być wystarczająco duże, aby przezwyciężyć ten warunek, lub można je skompensować. W tym celu pamiętaj, że wydedukowaliśmy to równanie:

$$\dot{e}_\omega = \dot{\omega} + S(\omega) R^T R_d \omega_d - R^T R_d \dot{\omega}_d$$

W ten sposób możemy zmodyfikować proponowaną kontrolę, aby zawierała następujące elementy:

$$\tau = -K_R e_R - K_\omega e_\omega + \omega \times J\omega - J(S(\omega) R^T R_d \omega_d - R^T R_d \dot{\omega}_d)$$

Zastępując zmodyfikowaną kontrolę w dynamice obrotowej, którą mamy

$$\tau = J\dot{\omega} + \omega \times J\omega$$
$$-K_R e_R - K_\omega e_\omega + \omega \times J\omega - J(S(\omega)R^T R_d \omega_d - R^T R_d \dot{\omega}_d) = J\dot{\omega} + \omega \times J\omega$$
$$-K_R e_R - K_\omega e_\omega - J(\dot{\omega} + S(\omega)R^T R_d \omega_d - R^T R_d \dot{\omega}_d) = 0$$
$$-K_R e_R - K_\omega e_\omega - J\dot{e}_\omega = 0$$

Ten system nie jest dokładnie systemem masowo-sprężynowych amortyzatorów, ale jego podobieństwem, wraz z niezbędnymi warunkami, które zostały opracowane za pomocą metody Lyapunova stosowanej przez Lee, Leok i McClamroch w kontroli złożonych manewrów dla UAV Quadrotor przy użyciu metod geometrycznych na SE (3) oraz w Geometry Tracking Control of Quadrotor UAV na SE (3) przez tych samych autorów, z którymi można zapoznać się w załączniku, oznacza, że

$$e_R \to 0$$
$$R \to R_d$$
$$e_\omega \to 0$$
$$\omega \to \omega_d$$

Tutaj pomijamy te dedukcje, ponieważ są one nawet dłuższe do przeprowadzenia krok po kroku niż wyjaśnienie definicji błędów i ich interakcji z dynamiką poczynioną w tym rozdziale. Pierwszym krokiem jest znalezienie związku między błędem obrotu a błędem prędkości kątowej, który również rozwijają wspomniani autorzy. Drugim krokiem jest przejście do następnej sekcji.

Podsumowując, nieliniowy system związany z dronem w niepłynnym locie jest kontrolowany w następujący sposób:

$$F_{BE} = \left(\begin{bmatrix} K_{Px}(x_d - x) + K_{Dx}(\dot{x}_d - \dot{x}) + m\ddot{x}_d \\ K_{Py}(y_d - y) + K_{Dy}(\dot{y}_d - \dot{y}) + m\ddot{y}_d \\ K_{Pz}(z_d - z) + K_{Dz}(\dot{z}_d - \dot{z}) + m\ddot{z}_d \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \right) \cdot \left[R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]$$

$$F_{BE} = C_{BE} \cdot \left(R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)$$

$$B_{des} = \begin{bmatrix} \cos\psi_d & \sin\psi_d & 0 \end{bmatrix}$$

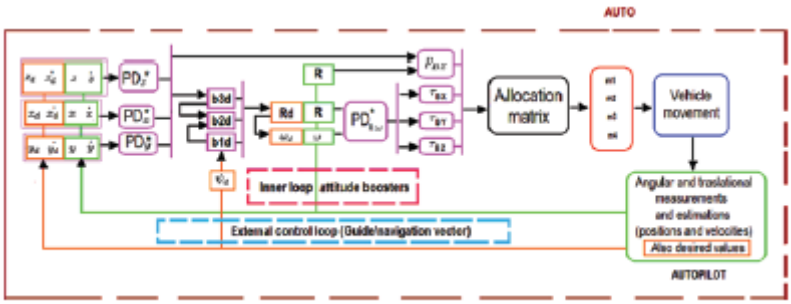
$$b_{3d} = \frac{C_{BE}}{|C_{BE}|}$$
$$b_{1d} = b_{2d} \times b_{3d}$$
$$b_{2d} = \frac{b_{3d} \times B_{des}}{|b_{3d} \times B_{des}|}$$

$$R_d = \begin{bmatrix} b_{1d} & b_{2d} & b_{3d} \end{bmatrix}$$

$$S(\omega_d) = R_d^T \dot{R}_d$$
$$e_\omega = \omega - R^T R_d \omega_d$$
$$e_R = \frac{(R_d^T R - R^T R_d)^\vee}{2}$$

$$\tau = -K_R e_R - K_\omega e_\omega + \omega \times J\omega - \underbrace{J(S(\omega)R^T R_d \omega_d - R^T R_d \dot{\omega}_d)}_{\text{Optional, it could be overwhelmed by PD terms}}$$

Należy zauważyć, że sterowanie opiera się na wektorze prowadzącym regulowanym przez trzy silniki odrzutowe (składniki pożądanej macierzy rotacji), które są zależne od tego samego wektora prowadzącego. Graficznie wygląda to tak, jak na rysunku (*zwróć uwagę, że wszystkie operacje są opisane w poprzednich równaniach i, jak już widziałeś, można dodać efekty dynamiczne, aby uzyskać dokładniejszy kontroler).



Jeśli chcesz dodać piloty zdalnego sterowania, mogą one zmienić żądane wartości X, Y, Z i odchylenia lub momenty i siłę, jak w przypadku poprzednich projektów. Poniższa część zawiera wprowadzenie do metody Lapunowa w celu wykazania stabilności systemów. To, wraz z wcześniej wyprowadzonymi równaniami, pozwoli ci z mniejszym trudem zanurzyć się w poprzednich odniesieniach.

Wprowadzenie do stabilności Lapunowa

Ta sekcja jest przeznaczona dla tych, którzy chcą dowiedzieć się więcej o rozszerzonych i formalnych dowodach dostępnych w wyżej wymienionych artykułach, które opierają się na teorii stabilności Lapunowa i które uzupełniają poprzednią sekcję. Istnieją dwie metody Lapunowa, jedna dla układów liniowych, a druga dla układów nieliniowych. Biorąc pod uwagę zakres drugiej metody oraz fakt, że liniowa jest jej szczególnym przypadkiem, pokrótce wyjaśnimy jej zastosowanie na przykładach, a także jej interpretację graficzną.

Krok 1

Biorąc pod uwagę system, na przykład system pierwszego rzędu i tłumiony oscylator harmoniczny bez wejścia sterującego, pamiętaj, że dochodzimy do tych systemów po zastosowaniu sterowników pętli zamkniętej w naszych systemach dynamicznych, jak widać w poprzednich sekcjach.

$$\dot{e} = -e$$
$$\ddot{e} + \dot{e} + e = 0$$

Można zaproponować wirtualną energię. Jest to funkcja skalarna kilku zmiennych, która zawsze jest dodatnia, z wyjątkiem wartości minimalnej, gdzie wynosi zero lub stan całkowitej bezruchu lub stan stabilny. Generalnie funkcja ta zależy od zmiennych, które analizujemy.

$$V(e, e, ..e^n) > 0$$
$$V(0) = 0$$

Dla pierwszego systemu funkcja spełniająca poprzednie warunki jest następująca (zauważ, że jej graficzna reprezentacja jest wklęsła parabola):

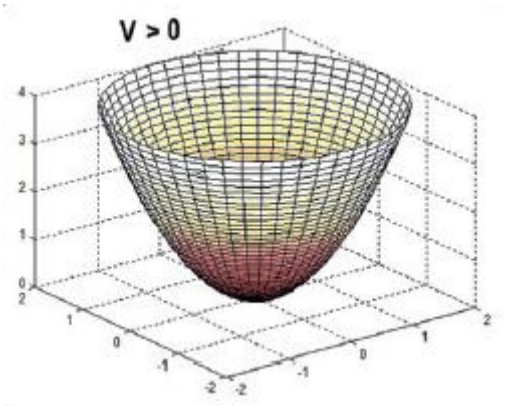
$$V = \frac{e^2}{2}$$

W przypadku drugiego systemu innym przykładem funkcji spełniającej oba warunki jest ten (jego graficzną reprezentacją jest paraboloida eliptyczna, która jest również wklęsła):

$$V = \frac{e^2}{2} + \frac{\dot{e}^2}{2}$$

Od teraz zauważ, że funkcja Lapunowa jest zwykle oznaczana literą V (możliwe, że dzieje się tak, ponieważ V wydaje się być graficzną reprezentacją funkcji wklęsłej). Należy również zauważyć, że często są one dzielone przez 2, aby nie przeciągać współczynnika 2 podczas wyprowadzania funkcji kwadratu Lapunowa.

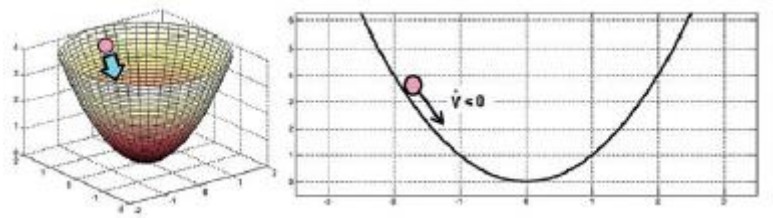
Zauważ, że funkcja Lapunowa zwykle ma kształt wyświetlany na Rysunek (przynajmniej tak, jak można go tak przedstawić, gdy zależy tylko od jednej lub dwóch zmiennych).



Wygląda jak naczynia wklęsłe, niekoniecznie zakrzywione. Niektóre z nich wyglądają jak odwrócone piramidy (pomysł można rozszerzyć o więcej niż dwie zmienne).

Krok 2

Pochodna funkcji Lapunowa po czasie musi być ujemna lub przynajmniej ujemna półokreślona i musi zależeć od jednej lub więcej interesujących nas zmiennych. Ma to wiele wspólnego z analogią graficzną. Pamiętajmy, że funkcje Lapunowa to pojemniki wklęsłe i zamknięte, a ich nachylenie widziane z dowolnego punktu na ich powierzchni jest ujemne.



Oznacza to, że stany systemu „umieszczonego” na wspomnianej powierzchni energetycznej „przesuną się” w kierunku minimalnego punktu powierzchni lub stanu stabilności lub energii zerowej, lub po prostu, że system zatrzyma się na minimalnej wartości energii.

$$\dot{V}(e, e, ..e^n) = \frac{dV(e,e,..e^n)}{dt} = \frac{dV(e,e,..e^n)}{d(e,e,..e^n)} \frac{d(e,e,..e^n)}{dt} =\leq 0$$

Dla systemu pierwszego rzędu:

$$V = \frac{e^2}{2}$$
$$\dot{V} = e\dot{e}$$

I zastąpienie systemu, dla którego zaprojektowano wspomnianą funkcję Lapunowa:

$$\dot{e} = -e$$
$$\dot{V} = e\dot{e} = -e^2 \leq 0$$

Ta pochodna jest ujemna częściowo zdefiniowana i zależy całkowicie lub częściowo od interesujących nas stanów. W przypadku tłumionego układu oscylatora harmonicznego mamy

$$V = \frac{e^2}{2} + \frac{\dot{e}^2}{2}$$
$$\dot{V} = e\dot{e} + \dot{e}\ddot{e}$$

I zastąpienie systemu, dla którego zaprojektowano wspomnianą funkcję Lapunowa:

$$\ddot{e} + \dot{e} + e = 0$$
$$\ddot{e} = -\dot{e} - e$$

$$\dot{V} = e\dot{e} + \dot{e}\ddot{e} = e\dot{e} - \dot{e}^2 - \dot{e}e = -\dot{e}^2 \leq 0$$

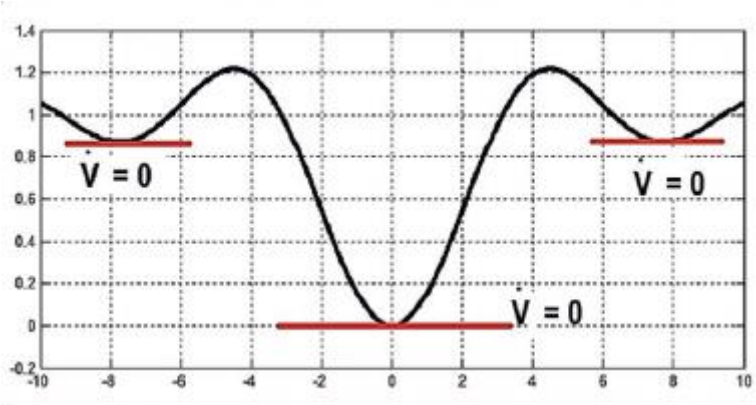
Ta pochodna jest ujemna częściowo zdefiniowana i zależy całkowicie lub częściowo od interesujących nas stanów. (W tym przypadku nie zależy to od błędu, ale od jego pochodnej. Niedługo zobaczymy, jak poradzić sobie z tym problemem.) Jeśli dla systemu pierwszego rzędu zaproponowaliśmy

$$V = \frac{K_p e^2}{2}$$
$$\dot{V} = K_p e\dot{e} = -K_p e^2$$
$$K_p > 0$$

pochodna funkcji Lapunowa nie zmieniłaby jej ujemności, a funkcja Lapunowa jej dodatniości, podczas gdy stała Kp byłaby dodatnia. Niedługo zapamiętasz ten fakt. Zauważmy, że możemy tylko powiedzieć, że stany, od których zależy pochodna Lapunowa, mają tendencję do zachowania minimalnej energii funkcji Lapunowa; oznacza to, że mają tendencję do 0. Jeśli reszta stanów nie pojawiają się, nie możemy niczego zapewnić o nich i musimy zaprojektować i wypróbować nową funkcję Lapunowa.

Krok 3

Używając postulatów pomocniczych, znanych jako lemat Barbalata lub zasada Lasalle'a (są one używane zgodnie z typem posiadanego systemu, więc musisz je zbadać), należy ocenić, kiedy V i jego pochodne są zerowe, jeśli stany systemu również dążą do zera. Dzieje się tak, ponieważ mogą występować lokalne doliny.



W tych dolinach wartość energii wynosi zero, ale system nie osiągnie minimalnej wartości funkcji Lapunowa, ale lokalną pułapkę. W pierwszym przykładzie zobaczmy, co się dzieje, gdy pochodna V jest równa zero:

$$\dot{V} = e\dot{e} = -e^2 = 0$$

$$-e^2 = 0$$
$$e = 0$$

i również

$$e\dot{e} = 0$$
$$\dot{e} = 0$$

Ten wynik można również osiągnąć za pomocą równania systemowego:

$$\dot{e} = -e$$

Zatem,

$$e \to 0$$
$$\dot{e} \to 0$$

W przypadku drugiego systemu postępujemy w ten sam sposób:

$$\dot{V} = e\dot{e} + \dot{e}\ddot{e} = e\dot{e} - \dot{e}^2 - \dot{e}e = -\dot{e}^2 = 0$$
$$-\dot{e}^2 = 0$$
$$\dot{e} = 0$$

i również

$$V = \frac{e^2}{2} + \frac{\dot{e}^2}{2} = 0$$
$$e = -\dot{e} = 0$$

Rozwiązywanie podwójnej pochodnej e przez zastąpienie znalezionych wartości e i jej pierwszej pochodnej:

$$\dot{V} = e\dot{e} + \dot{e}\ddot{e} = 0$$
$$\ddot{e} = 0$$

Ten wynik można również osiągnąć za pomocą równania systemowego:

$$\ddot{e} = -\dot{e} - e$$

Zatem,

$$\ddot{e} \to 0$$
$$\dot{e} \to 0$$
$$e \to 0$$

Dodatkowo bardzo powszechny sposób pracy z funkcjami Lapunowa opiera się na równaniach kwadratowych. Na przykład funkcję Lapunowa tłumionego układu harmonicznego można przepisać w następujący sposób:

$$V = \frac{e^2}{2} + \frac{\dot{e}^2}{2}$$
$$V = \begin{bmatrix} e & \dot{e} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} e \\ \dot{e} \end{bmatrix}$$

$$\dot{V} = \begin{bmatrix} \dot{e} & \ddot{e} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} e & \dot{e} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix}$$
$$= \begin{bmatrix} e & \dot{e} \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} e \\ \dot{e} \end{bmatrix}$$

Jeśli spojrzysz na

$$\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$
$$\begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix}$$

Bezpośrednim celem jest wykazanie, że te macierze są odpowiednio odpowiednikami dodatnich i ujemnych lub liniowej algebrya jest znana jako dodatnia macierz określona, ujemna macierz określona lub macierze półokreślone. Stosując teorię algebry liniowej, za pomocą kryterium wartości własnych uzyskuje się, że pierwsza macierz jest dodatnio określona, a druga ujemna półokreślona, co odpowiada oryginalnym wynikom. Ta macierzowa reprezentacja znajduje się w wielu tekstach dotyczących systemów sterowania, w tym w demonstracjach Taeyoung Lee we wcześniej wskazanych artykułach. Zaproponowanie funkcji Lapunowa, a także wykazanie, że ich pochodne (gdzie układy i ich kontrolery zwykle występują) są zdefiniowane jako negatywne, jest rozległą gałęzią nauki zwaną nieliniową teorią sterowania. Każdy układ dynamiczny stanowi wyzwanie w znalezieniu stabilności za pomocą funkcji Lapunowa; nawet każdy indywidualny wynik reprezentuje książkę lub nową naukę. To, co bardzo pomaga w tych badaniach, to umiejętność wykonywania operacji algebraicznych za pomocą kontrolera. Jako przykład w Części 4 znajduje się sekcja poświęcona reprezentacji w przestrzeni

stanów, w której wskazano, że nieliniowy układ równań różniczkowych (w tym multikopter) można przedstawić w następujący sposób:

$$\dot{x} = f(x, u)$$

gdzie x to stany układu, a u ich sygnały sterujące. W ten sposób, zgodnie z metodą Lapunowa

$$V(x) > 0$$
$$V(0) = 0$$
$$\dot{V}(x) = \nabla V(x)\dot{x} = \nabla V(x) f(x, u)$$

i zgodnie z oczekiwaniami:

$$\dot{V}(x) = \nabla V(x) f(x, u) \leq 0$$

Sposobem na osiągnięcie tego jest odpowiednia modyfikacja kontrolki u. Na przykład przeanalizujmy przypadek układu drugiego rzędu (tłumiony oscylator harmoniczny) z wejściem niewolnym (z obecnością siły u zamiast prostego 0). Celem jest, aby zmienna Y dążyła do pożądanej wartości znanej jako Yd. Będziemy mieli dwa równania: równanie układu rzeczywistego i równanie sterowania.

Mamy więc dwie opcje:

1. Kontroler jest testowany i jeśli działa poprawnie, bezpośrednio staramy się wykazać stabilność systemu pętli zamkniętej. Nie jest to całkowicie zalecane, ale większość ludzi robi to w ten sposób, zwłaszcza, że prawie wszystko działa z PD lub PID i ludzie chcą tylko odpowiedzieć na powody. Jak powiedziałem, mamy system i kontrolę, która jest już przydatna:

$$\ddot{Y} + \dot{Y} + Y = u$$
$$u = \ddot{Y}_d + K_d(\dot{Y}_d - \dot{Y}) + K_p(Y_d - Y)$$

Zamknięta pętla jest analizowana:

$$\ddot{Y} + \dot{Y} + Y = \ddot{Y}_d + K_d(\dot{Y}_d - \dot{Y}) + K_p(Y_d - Y)$$
$$\ddot{Y} = \ddot{Y}_d + K_d(\dot{Y}_d - \dot{Y}) + K_p(Y_d - Y) - \dot{Y} - Y$$

Poczyniono pewne dominujące założenia. Poniższe informacje dotyczą dodatnich wartości Kp i Kd, a także większych niż 1:

$$-K_d\dot{Y} \gg -\dot{Y}$$
$$-K_pY \gg -Y$$

$$\ddot{Y} = \ddot{Y}_d + K_d(\dot{Y}_d - \dot{Y}) + K_p(Y_d - Y)$$
$$0 = (\ddot{Y}_d - \ddot{Y}) + K_d(\dot{Y}_d - \dot{Y}) + K_p(Y_d - Y)$$

Zmiana nazwy:

$$Y_d - Y = e$$

A także zmieniając nazwy jego pochodnych, mamy ten system:

$$0 = \ddot{e} + K_d\dot{e} + K_p e$$

I wreszcie, jego stabilność w pętli zamkniętej jest demonstrowana w podobny sposób, jak wcześniej analizowany system drugiego rzędu:

$$V(e) = \frac{K_p e^2}{2} + \frac{\dot{e}^2}{2}$$
$$\dot{V}(e) = K_p e\dot{e} + \dot{e}\ddot{e} = K_p e\dot{e} + \dot{e}(-K_d\dot{e} - K_p e) = -K_d\dot{e}^2 \leq 0$$

2. Wykazuje się stabilność oryginalnego systemu i projektuje się kontrolę procesu. Na przykład proponowana jest ta funkcja Lapunowa, a następnie jej pochodna

$$V(Y_d - Y, \dot{Y}_d - \dot{Y}) = \frac{(K_p[Y_d - Y])^2}{2} + \frac{(\dot{Y}_d - \dot{Y})^2}{2}$$
$$\dot{V} = (K_p[Y_d - Y])(\dot{Y}_d - \dot{Y}) + (\dot{Y}_d - \dot{Y})(\ddot{Y}_d - \ddot{Y})$$

Wprowadzamy dynamikę układu do pochodnej Lapunowa:

$$\ddot{Y} + \dot{Y} + Y = u$$
$$\ddot{Y} = u - \dot{Y} - Y$$
$$\dot{V} = (K_p[Y_d - Y])(\dot{Y}_d - \dot{Y}) + (\dot{Y}_d - \dot{Y})(\ddot{Y}_d - \left[u - \dot{Y} - Y\right])$$

W tej chwili nie można tego powiedzieć

$$\dot{V} \leq 0$$

Ale można zaprojektować u, na przykład:

$$u = \ddot{Y}_d + K_d(\dot{Y}_d - \dot{Y}) + K_p(Y_d - Y)$$

Wtedy termin

$$\begin{aligned}\ddot{Y}_d - \left[u - \dot{Y} - Y\right] &= \ddot{Y}_d + \dot{Y} + Y - \ddot{Y}_d - K_d(\dot{Y}_d - \dot{Y}) - K_p(Y_d - Y) \\ &= \dot{Y} + Y - K_d(\dot{Y}_d - \dot{Y}) - K_p(Y_d - Y)\end{aligned}$$

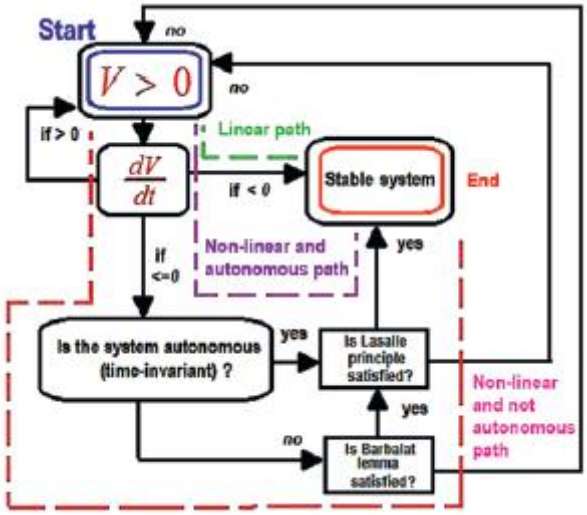
Ponownie możemy przyjąć te założenia

$$K_d\dot{Y} \gg \dot{Y}$$
$$K_p Y \gg Y$$
$$\begin{aligned}\ddot{Y}_d - \left[u - \dot{Y} - Y\right] &= \dot{Y} + Y - K_d(\dot{Y}_d - \dot{Y}) - K_p(Y_d - Y) \\ &\approx -K_d(\dot{Y}_d - \dot{Y}) - K_p(Y_d - Y)\end{aligned}$$

pod tymi rozważaniami

$$\dot{V} = (K_p[Y_d - Y])(\dot{Y}_d - \dot{Y}) + (\dot{Y}_d - \dot{Y})(\ddot{Y}_d - \left[u - \dot{Y} - Y\right])$$
$$\approx (K_p[Y_d - Y])(\dot{Y}_d - \dot{Y}) + (\dot{Y}_d - \dot{Y})\left[-K_d(\dot{Y}_d - \dot{Y}) - K_p(Y_d - Y)\right]$$
$$\dot{V} = -K_d(\dot{Y}_d - \dot{Y})^2 + K_p\left[(Y_d - Y)(\dot{Y}_d - \dot{Y}) - (Y_d - Y)(\dot{Y}_d - \dot{Y})\right]$$
$$= -K_d(\dot{Y}_d - \dot{Y})^2 \leq 0$$

Zauważ, że tak jak w przypadku wspomnianych artykułów Taeyoung Lee, czasami nie jest specjalnie udowodnione, że funkcje Lapunowa i ich pochodne są ściśle określone dodatnie i ujemne, ale po prostu, że niektóre z ich członów równań są ograniczone (oznacza to, że ich maksimum wartości minimalne nie mogą przekroczyć pewnej wartości), a ze względu na to, że są ograniczone, mogą być zdominowane w swoich skutkach przez inne negatywne określenia określone lub pozytywnie zdefiniowane. Ograniczenia te pozwalają również znaleźć fizyczne ograniczenia pojazdu i jego zakresy działania. Na koniec należy wspomnieć, że analiza stateczności metodami Lapunowa jest opcjonalna w układach liniowych, ponieważ mają one wiele innych narzędzi. Ale w układach nieliniowych, tak jak w przypadku kontroli geometrycznej, metoda Lapunowa jest najpopularniejszą opcją i to nie ze względu na prostotę (która nie istnieje), ale dlatego, że ma określoną metodę. Wspomniany sposób przedstawiono na rysunku.

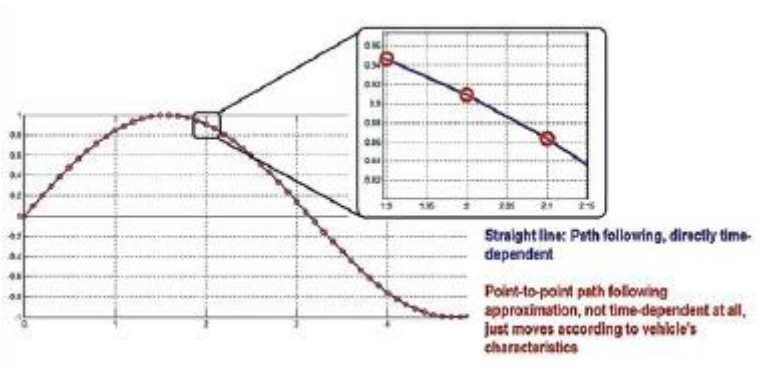


Poznałeś kontrolery trybu robota i pojazdu, a także podstawy teorii Lapunowa. Zakończmy sekcją o pożądanych wartościach (pamiętajmy, że są one składową błędu, który definiuje nasze kontrolery).

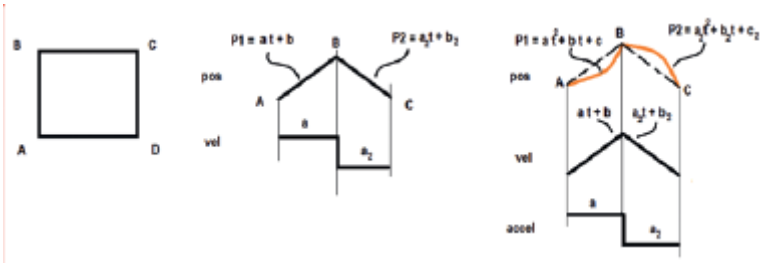
Definicja pożądanych wartości: regulacja, trajektoria lub punkt-punkt

W części 4 zweryfikujemy większość opisanych tutaj regulatorów i zwrócimy uwagę (zwłaszcza w przypadku geometrycznym), że rodzaj toru zmienia sposób strojenia regulatora (i to nie tylko w doborze wzmocnień, ale także w włączenie lub pominięcie określonych terminów). W ten sposób możemy zadać sobie pytanie, czy istnieje sposób na śledzenie trajektorii bez uzależnienia bezpośrednio od czasu wykonania, ale bezpośrednio od mobilności pojazdu, na przykład jego prędkości, przyspieszenia, szarpnięcia (pierwsza pochodna przyspieszenia), a nawet jego skok (druga pochodna przyspieszenia). Zacznijmy od zdefiniowania pewnych pojęć związanych z pożądaną wartością. Inne pojęcia można znaleźć w artykule Kelly'ego w Dodatku (na przykład w celu zdefiniowania błędu dotykowego). Regulacja: Zadanie regulacji ma miejsce, gdy system ma stałą wartość. W tym przypadku dobrym przykładem jest zadanie multikoptera w zawisie, co oznacza, że pojazd dociera do punktu w przestrzeni i tam utrzymuje „statyczną” pozycję. Śledzenie trajektorii: ma miejsce, gdy pojazd otrzymuje polecenie podążania za odniesieniem zależnym od czasu (okrąg, cosinus itp.). Podstawowym

zadaniem jest podążanie za punktem odniesienia w krytycznych momentach i zostanie to osiągnięte, o ile silniki i system same będą w stanie poruszać się z prędkością, przyspieszeniem itp. Tutaj istotne jest powtórzenie punktu odniesienia i jego ruchu cechy, na przykład pościg pojazdu. Śledzenie od punktu do punktu: Jest to szczególny przypadek poprzedniego, w którym brane są pod uwagę maksymalne cechy drona (prędkość, przyspieszenie itp.). Celem jest tutaj, aby nasz dron podążał trasą bez przekraczania własnych wartości granicznych, unikając forsowania silników i uszkadzania samolotu z powodu nagłych zmian bezwładności. Polega na podzieleniu danej trajektorii na określone odcinki i wykonaniu tej trajektorii jako ruchu pomiędzy punktami.



W ten sposób projektant może kontrolować profile prędkości i przyspieszenia (lub więcej pochodnych) pomiędzy punktami docelowymi, tak aby te profile znajdowały się poniżej maksymalnych wartości pojazdu. Można to zrobić na wiele sposobów, takich jak splajny, Hermite, wielomianowa interpolacja Catmull-Rom n stopni, interpolacja z użyciem trygonometrycznych lub wykładniczy szereg Fouriera. Głównym problemem przy definiowaniu wartości punkt-punkt jest uzyskanie płynnej pracy silników. Nie jest istotne, aby powtórzyć czas wykonania odniesienia, ale tylko odniesienie i jest przydatne, na przykład, w zadaniach wyszukiwania obiektów lub wykonywaniu predefiniowanych ruchów w celu rehabilitacji ciała. Zauważ, że im bliżej są punkty, tym śledzenie od punktu do punktu jest również bliższe zachowaniu śledzenia trajektorii. Innymi słowy, im większy rząd interpolacji, pojazd będzie zachowywał się w mniej „ruchu robotów", a wszystko to za cenę uszkodzenia elementów wykonawczych pojazdu. Stanowi to kompromis między wydajnością mobilności a bezpieczeństwem silników. Pamiętaj też, że im bliżej punktów, tym łatwiej mogą wystąpić błędy oversamplingu, a co za tym idzie, możesz uszkodzić swój pojazd. I odwrotnie, gdy odległość między punktami jest większa, może wystąpić śledzenie podpróbkowania. Na podanych przykładach zobaczysz, że wartości pośrednie między punktami niekoniecznie podążają za pierwotnym profilem trajektorii. Należy również zauważyć, że jeśli ścieżka, którą należy podążać, sama w sobie ma powolne i płynne działanie w odniesieniu do granic pojazdu (na przykład podążanie za sygnałem cosinusowym o niskiej częstotliwości), śledzenie od punktu do punktu można zastąpić trajektorią śledzenie (aby poeksperymentować, przejdź do sekcji symulacji w części kontroli geometrycznej i zmodyfikuj kod, aby śledzić funkcje trygonometryczne od niższych do wyższych częstotliwości). Oczywiście śledzenie od punktu do punktu jest przydatne, gdy trajektorii nie można zdefiniować za pomocą standardowych funkcji lub gdy funkcje, którymi należy podążać, nie są płynne (na przykład kwadraty, trójkąty itp.). Planowanie trajektorii może samo w sobie generować kompletną książkę, jeśli rozważamy trajektorie z minimalnym czasem, przyspieszeniem lub innymi pochodnymi profilu ruchu, także trajektorie z wieloma punktami lub trajektorie punkt-punkt, trajektorie z jedną zmienną lub z wieloma zmienne, trajektorie translacyjne, orientacyjne lub kombinowane, trajektorie oparte na wielomianach lub funkcjach trygonometrycznych i tak dalej. Kontynuujmy. Załóżmy, że chcemy podążać ścieżką kwadratową pokazaną na rysunku (jak zobaczysz, śledzenie figur lub ścieżek z wierzchołkami stanowi wyzwanie dla każdego kontrolera).



Część tej ścieżki sugeruje, że musimy przejść przez punkty ABC. Intuicja oparta na najkrótszej odległości podpowiada nam, że możemy to zrobić w linii prostej (używając dwóch wielomianów pierwszego stopnia). Problem polega na tym, że pochodna trajektorii prostych przybiera kształt urwany, gdy występują punkty przegięcia (pochodna jest funkcją skokową). Aby tego uniknąć, można by użyć wielomianu drugiego rzędu (ale teraz trajektoria kwadratowa staje się przybliżeniem, gdzie osiągane są tylko punkty ABCD). Idąc tym samym tokiem rozumowania, wielomian drugiego rzędu nie przedstawia już nagłych zmian na poziomie prędkości, ale na poziomie przyspieszenia (ponownie, gdzie następuje przegięcie lub punkt przejścia między dwoma wielomianami), co również uszkadza silniki. Wówczas rozwiązaniem jest po prostu użycie trajektorii wielomianowych trzeciego rzędu. Głębszą demonstrację, która opiera się również na wcześniejszych rozumowaniach fizycznych i minimalizacji funkcji kosztów równań Eulera Lagrange'a, można znaleźć w artykułach MOOC, artykułach i tezach Kumara i jego zespołu. Zauważ, że nawet w takim przypadku pochodna przyspieszenia będzie nadal przedstawiać gwałtowne zmiany, a zatem w nieskończoność na każdym poziomie pochodnej przy wprowadzaniu nowych stopni wielomianu. Nie jest to jednak do końca istotne, ponieważ interesują nas miękkie wartości prędkości, dla których przyspieszenia są małe (dążąc do zera) i nawet przy tych nagłych zmianach ich efekty nie będą istotne. Jeśli jednak interesuje nas ruch agresywny, musimy wziąć pod uwagę wielomiany wyższego rzędu, jak pokazuje Kumar, które wykonują planowanie od punktu do punktu na poziomie przyciągania (druga pochodna przyspieszenia). Od tego momentu możemy wprowadzać wartości prędkości zgodne z tymi, które pojazd może sam wygenerować w następujący sposób (zawsze z uwzględnieniem punktu początkowego i końcowego, chociaż istnieją algorytmy wielopunktowe). Najpierw obserwujemy, że mamy pożądany wielomian trzeciego rzędu i że będziemy mieć wielomian tego typu dla każdej pożądanej wartości. Oznacza to, że jeśli interesuje nas zdefiniowanie pożądanych wartości zmiennych X, Y i Z, musimy powtórzyć procedurę dla każdej z tych zmiennych:

$$x_d, y_d, z_d = val_d = P_d = C_0 + C_1 t + C_2 t^2 + C_3 t^3$$

Oznacza to, że do obliczenia czterech współczynników potrzebujemy czterech równań. Aby uniknąć problemów i pomyłek z literami, będziemy kontynuować przykład ze zmienną X, ale należy to powtórzyć dla wszystkich pozostałych zmiennych. Mamy dwa równania, jedno, które zależy od końcowej pozycji, a drugie, które zależy od początkowej pozycji. Należy pamiętać, że konieczne jest dokonanie oceny w momencie początkowym i końcowym, więc tutaj rozważana jest maksymalna prędkość pojazdu.

$$
\begin{aligned}
x_0 &= x(t_0) = P_d(t_0) \\
x_f &= x(t_f) = P_d(t_f) \\
x_0 &= C_0 + C_1 t_0 + C_2 t_0^2 + C_3 t_0^3 \\
x_f &= C_0 + C_1 t_f + C_2 t_f^2 + C_3 t_f^3.
\end{aligned}
$$

Pozostałe dwa równania są generowane z prędkością początkową i końcową. W tym celu widzimy, że pochodna lub profil prędkości naszego wielomianu to

$$\dot{P}_d = 3C_3 t^2 + 2C_2 t + C_1$$

I dlatego równania z prędkością początkową i końcową to

$$
\begin{aligned}
\dot{x}_0 &= \dot{x}(t_0) = \dot{P}_d(t_0) \\
\dot{x}_f &= \dot{x}(t_f) = \dot{P}_d(t_f) \\
\dot{x}_0 &= C_1 + 2C_2 t_0 + 3C_3 t_0^2 \\
\dot{x}_f &= C_1 + 2C_2 t_f + 3C_3 t_f^2
\end{aligned}
$$

Należy zauważyć, że współczynniki tych wielomianów muszą być przeliczane w miarę zmiany punktu początkowego i końcowego. Wygodnie jest przepisać nasze cztery równania z czterema niewiadomymi w trybie macierzowym:

$$
\begin{aligned}
x_0 &= C_0 + C_1 t_0 + C_2 t_0^2 + C_3 t_0^3 \\
x_f &= C_0 + C_1 t_f + C_2 t_f^2 + C_3 t_f^3 \\
\dot{x}_0 &= C_1 + 2C_2 t_0 + 3C_3 t_0^2 \\
\dot{x}_f &= C_1 + 2C_2 t_f + 3C_3 t_f^2
\end{aligned}
$$

$$
\begin{bmatrix} x_0 \\ x_f \\ \dot{x}_0 \\ \dot{x}_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix}
$$

A obliczenie wielomianu zostanie zaktualizowane w ten sposób:

$$
\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix}^{-1} \begin{bmatrix} x_0 \\ x_f \\ \dot{x}_0 \\ \dot{x}_f \end{bmatrix}
$$

$$P_d = C_0 + C_1 t + C_2 t^2 + C_3 t^3$$

W każdym nowym obliczeniu wartość pozycji początkowej, prędkość początkowa i czas początkowy muszą być wzięte z poprzednich wartości końcowych. Jeśli chcemy również uwzględnić przyspieszenia początkowe i końcowe, wielomian musi być piątym rzędem.

Gdzie jest wliczona maksymalna prędkość pojazdu? Zauważ, że maksymalna wartość prędkości na osi X może być aproksymowana przez

$$V_x \max = \frac{x_f - x_0}{t_f - t_0}$$

Wartości pozycji są niezmienne, ponieważ określają nasz punkt początkowy i końcowy. Kolejną rzeczą, której nie możemy zmienić, jest czas początkowy, więc możemy zmodyfikować tylko czas końcowy zgodnie z naszą maksymalną prędkością:

$$t_f = \frac{x_f - x_0}{V_x \max} + t_0$$

W ten sposób nasz projekt może wykonać następujące czynności:

$$t_f \geq \frac{x_f - x_0}{V_x \max} + t_0$$
$$\dot{x}_0 \leq V_x \max$$
$$\dot{x}_f \leq V_x \max$$

Ponieważ pojazd nie generuje większej prędkości niż prędkość wytwarzana przez jego silniki, może to mieć wpływ na ostateczny czas, co zniekształca pożądaną trajektorię wielomianu. Aby przeciwdziałać temu efektowi, można zastosować wielomian wyższego rzędu do wprowadzenia profili przyspieszenia lub można uniknąć tych obliczeń, pracując z wartościami przekroczonymi w ten sposób:

$$t_f \gg \frac{x_f - x_0}{V_x \max} + t_0$$
$$\dot{x}_0 \ll V_x \max$$
$$\dot{x}_f \ll V_x \max$$

Przy definiowaniu ścieżek orientacji, w zależności od tego, czy używanymi narzędziami są bezpośrednio kąty Eulera, kwaterniony, macierze obrotu itp., należy wziąć pod uwagę więcej. Na koniec wspomnimy, że PD samo w sobie reprezentuje gładki wielomian nieskończonego rzędu na poziomie błędu (śledzenie od punktu do punktu, gdzie punktem początkowym jest wartość zmierzona, a końcowym wartość pożądana), więc PD kontrola może być wykonywana bez definiowania ścieżek wielomianowych. Dzieje się tak, ponieważ równanie wyładowań niezupełnych wiąże się z zachowaniem wykładniczym

$$PD(Error) \rightarrow Error \approx e^{-t}$$

oraz funkcję wykładniczą, która zgodnie z szeregiem Taylora jest wielomianem n-tego rzędu (znacznie większym niż przybliżenie trzeciego, piątego lub wyższego rzędu)

$$e^{-t} = 1 - t + \tfrac{1}{2}t^2 - \tfrac{1}{6}t^3 + \tfrac{1}{24}t^4 + ...(-1)^n \tfrac{t^n}{n!}$$
$$Error \approx 1 - t + \tfrac{1}{2}t^2 - \tfrac{1}{6}t^3 + \tfrac{1}{24}t^4 + ...(-1)^n \tfrac{t^n}{n!}$$

Jednak nie ma sposobu na regulowanie czasu końcowego (można to zrobić za pomocą trajektorii wielomianowych), chyba że istnieje coś dodatkowego kontrolującego proporcjonalne i różniczkowe wzmocnienia PD w oparciu o czas końcowy. Można to osiągnąć dzięki inteligentnym algorytmom lub sterownikowi TBG (generator podstawy czasu). Zobacz koncepcję TBG w artykule Parry w załączniku. Dodatkowo, jak już wspomniano, PD opiera się na błędzie zależnym od dwóch wartości. Z tego powodu może być stosowana tylko przy śledzeniu punkt-punkt, podczas gdy planowanie trajektorii, zgodnie z algorytmem, pozwala również na wygenerowanie trajektorii obejmującej więcej niż dwa punkty.

Rodzaj żądanej wartości: Użycie

Stała : Reguluj zadania, w których pożądane jest pozostawanie w nieskończoność ze stałą wartością (na przykład zadania z zawisem drona)

Trajektoria zależna od czasu : Zadania, w których trajektoria zmienia się powoli i płynnie w odniesieniu do maksymalnych prędkości lub przyspieszeń pojazdu

Trajektoria punkt-punkt lub wielopunktowa przy użyciu wielomianów n-tego rzędu (lub innych metod opartych na funkcjach trygonometrycznych: Zadania, w których trajektorii nie można zdefiniować za pomocą znanych funkcji, lub w których następuje nagła zmiana toru powodująca nagłe zmiany prędkości lub przyspieszenia pojazdu, lub w których projektant chce ograniczyć prędkość, przyspieszenie lub ich pochodne za pomocą w odniesieniu do maksymalnych wartości, jakie może osiągnąć pojazd

Streszczenie

W tym rozdziale poznałeś przydatne pojęcia z teorii sterowania związane z dronami. W tym przypadku koncepcje zostały również zastosowane do quadkopterów, ale są one szeroko stosowane w innych typach samolotów. Zbadałeś dwie rodziny kontrolerów, te typu pojazdu i te typu robota. W obu przypadkach przedstawiliśmy cztery z najczęściej stosowanych w artykułach i książkach regulatorów: regulator liniowy, regulator zmiennego odchylenia, regulator sferycznej kompensacji dynamicznej i regulator geometryczny. Dzięki nim będziesz mógł projektować od płynnych operacji po akrobatyczne tryby lotu. Rozdział zakończył się pojęciami dotyczącymi planowania trajektorii. W kolejnym rozdziale poznasz szczegóły dotyczące symulacji drona.

Symulacja

Ta Część ma wpływ zarówno na twórców, jak i na naukę. Pokaże ci temat porzucony praktycznie we wszystkich książkach i dostępnych odniesieniach, a który składa się z ogólnej symulacji dronów. W tym konkretnym przypadku skorzystamy z narzędzi MATLAB/Simulink ze względu na ich popularność i uproszczenie użytkowania w środowisku naukowym, ale wiedzę można rozszerzyć na dowolne inne środowisko programistyczne, w tym symulatory open source. Podejście opiera się na wykorzystaniu zmiennych z przestrzeni stanów oraz grafiki reprezentującej poprzez bloki przetwarzania. Dzięki temu będziesz mógł projektować własne pojazdy za pomocą narzędzi programowych, a także rozszerzać tego typu wiedzę na inne roboty.

Rodzaje symulatorów

Mamy cztery rodzaje symulatorów:

• Numeryczne z symbolami: Wymagają tylko równań, w szczególności modelu matematycznego systemu i jego kontrolera. Ten rodzaj symulatora to dowolny program zdolny do rozwiązywania tych równań. W zależności od złożoności modelu program będzie w stanie go rozwiązać lub nie. W przypadku drona można to zrobić za pomocą standardowego języka programowania, w którym można zakodować algorytm do numerycznej rozdzielczości równań różniczkowych zwyczajnych, takich jak Euler, RK4 itp. Ten symulator wymaga co najmniej dwóch sekcji kodu od solvera i system do rozwiązania.

• Numeryczne z blokami: Podobnie jak w poprzednim przypadku, ale zamiast symbolicznej reprezentacji równań i kontrolera mają one reprezentację graficzną opartą na blokach. Jest to znacznie bardziej wizualna forma programowania niż symboliczna i ułatwia łączenie komponentów (z wyjątkiem sytuacji, gdy jest wiele elementów). Znane przykłady to LabView, Simulink lub Scilab/Scicos. W przypadku korzystania z bloków solwer jest sam w sobie interfejsem poleceń, a układem do rozwiązania jest plik z diagramami.

• Animowane: w tym przypadku są to interfejsy z trójwymiarową grafiką, które umożliwiają symulację systemu. Przykładami są V-rep, Gazebo lub GUI Mission Planner i jego wirtualny symulator SITL (oprogramowanie w pętli). Generalnie składają się z dwóch części: animowanego obiektu i jego świata oraz pliku modelowania. Wiele z tych symulatorów ma silnik fizyczny, który obejmuje efekty otoczenia, takie jak wiatr lub ciśnienie.

• Interaktywny: Tutaj znajdujemy możliwość interakcji pomiędzy symulatorami dowolnego z poprzednich typów lub interakcji ze sprzętem. Przykładami są X-Plane lub Mission planner z trybem HITL (sprzęt w pętli).

W przypadku tego tekstu skupimy się na dwóch najprostszych w obsłudze (z punktu widzenia instalacji): numerycznych. Oczywiście wymagają one matematyki, ale użytkownicy mogą je uprościć lub skomplikować W przypadku tego tekstu skupimy się na dwóch najprostszych w obsłudze (z punktu widzenia instalacji): numerycznych. Oczywiście wymagają matematyki, ale użytkownicy mogą uczynić je tak prostymi, jak i skomplikowanymi.

Reprezentacja Przestrzeni Państwowej

Jeśli masz równanie różniczkowe rzędu n, można je przekształcić w n równań różniczkowych rzędu pierwszego. Jest to przydatne w przypadku używania pojedynczego zbiorczego bloku integratora w symulatorze lub obsługi operacji wektorowych i macierzowych zamiast skalarnych.

$$(1x1)$$
$$y^{(n)} + a_1 y^{(n-1)} + \dots a_{n-1}\dot{y} + a_n y = u \quad \Rightarrow \quad (nx1)$$
$$\dot{x} = f(x,u)$$

Na przykład pamiętaj, że system amortyzatorów masowo-sprężystych jest systemem drugiego rzędu:

$$m\ddot{x} + b\dot{x} + kx = u$$
$$n = 2$$

W ten sposób będzie wymagało dwóch zmiennych przestrzeni stanów. W takich przypadkach najwygodniejszym wyborem jest przyjęcie jako zmiennych przestrzeni stanów tych, które mają niższe rzędy w stosunku do porządku systemowego. W tym równaniu pozycja i prędkość są wybrane jako zmienne w przestrzeni stanów:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

Otrzymując pochodną dochodzimy do wymaganego wyrażenia:

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix}$$

Ostatnie równanie można rozwiązać z oryginalnego równania układu:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{u-kx-b\dot{x}}{m} \end{bmatrix}$$

Zastępując zmienne przestrzeni stanów, skutkuje to

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{u-kx-b\dot{x}}{m} \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{u-kx_1-bx_2}{m} \end{bmatrix}$$

pozostałe tylko dwa równania pierwszego rzędu zgodnie z oczekiwaniami. Quadkopter może mieć, zgodnie z jego modelem dynamicznym, dwanaście zmiennych w przestrzeni stanów (sześć równań drugiego rzędu; pamiętaj, że trzy z tych równań są związane z dynamiką translacyjną, a pozostałe z dynamiką obrotową):

$$F_{BZ}\begin{bmatrix}\theta\cos\psi_d+\phi\sin\psi_d\\ \theta\sin\psi_d-\phi\cos\psi_d\\ 1\end{bmatrix}+m\begin{bmatrix}0\\0\\-g\end{bmatrix}=m\begin{bmatrix}\ddot{x}\\\ddot{y}\\\ddot{z}\end{bmatrix}$$

$$\begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix}\approx\begin{bmatrix}J_x\ddot{\phi}\\ J_y\ddot{\theta}\\ J_z\ddot{\psi}\end{bmatrix}$$

Jeśli wybierzemy zmienne przestrzeni stanów w następujący sposób (jest to możliwy wybór naprzemiennych pozycji i prędkości), mamy

$$\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\\x_8\\x_9\\x_{10}\\x_{11}\\x_{12}\end{bmatrix}=\begin{bmatrix}x\\\dot{x}\\y\\\dot{y}\\z\\\dot{z}\\\theta\\\dot{\theta}\\\phi\\\dot{\phi}\\\psi\\\dot{\psi}\end{bmatrix}\qquad\begin{bmatrix}\dot{x}_1\\\dot{x}_2\\\dot{x}_3\\\dot{x}_4\\\dot{x}_5\\\dot{x}_6\\\dot{x}_7\\\dot{x}_8\\\dot{x}_9\\\dot{x}_{10}\\\dot{x}_{11}\\\dot{x}_{12}\end{bmatrix}=\begin{bmatrix}\dot{x}\\\ddot{x}\\\dot{y}\\\ddot{y}\\\dot{z}\\\ddot{z}\\\dot{\theta}\\\ddot{\theta}\\\dot{\phi}\\\ddot{\phi}\\\dot{\psi}\\\ddot{\psi}\end{bmatrix}=\begin{bmatrix}x_2\\\frac{F_{BZ}}{m}(x_7\cos x_{11d}+x_9\sin x_{11d})\\x_4\\\frac{F_{BZ}}{m}(x_7\sin x_{11d}-x_9\cos x_{11d})\\x_6\\\frac{F_{BZ}}{m}-g\\x_8\\\tau_\theta/J_y\\x_{10}\\\tau_\phi/J_x\\x_{12}\\\tau_\psi/J_z\end{bmatrix}$$

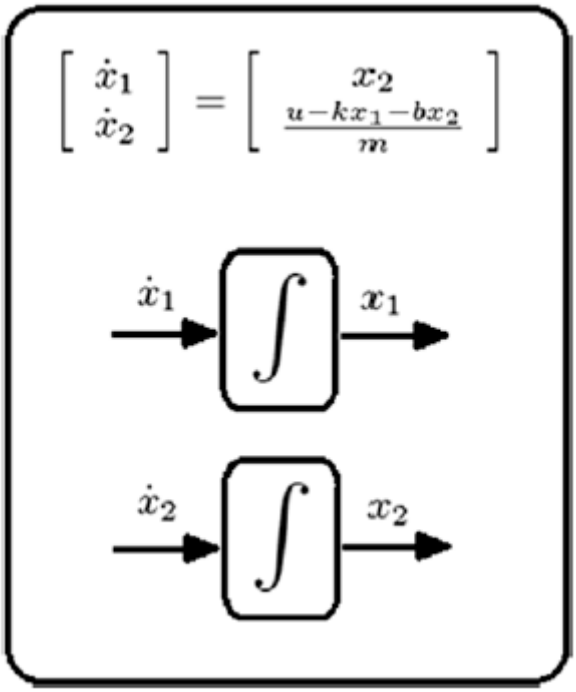
Należy również zauważyć, że wejście sterujące u ma cztery wartości. Są to siła ciągu i trzy momenty obrotowe:

$$\dot{x}=f(x,u)=f(x,[F_{BZ},\tau_\theta,\tau_\phi,\tau_\psi])$$

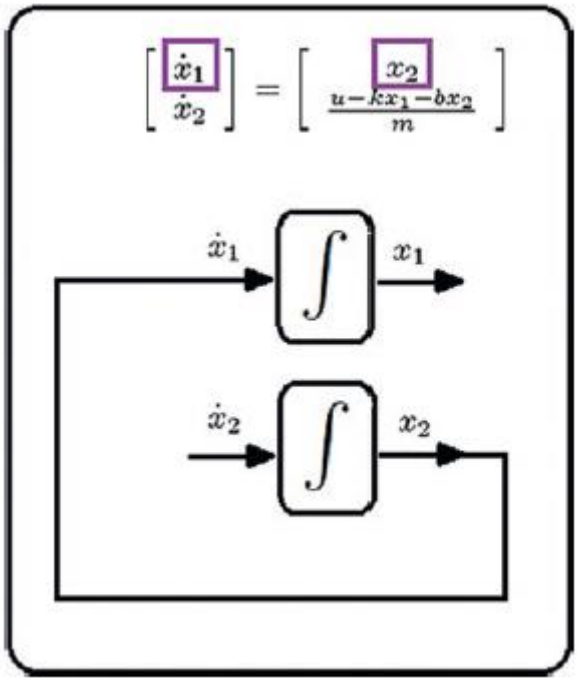
Innego możliwego wyboru można dokonać, wybierając najpierw wszystkie pozycje, a następnie wszystkie prędkości, lub po prostu inną sekwencję od tej przedstawionej tutaj. Niezależnie od tego, którą wybierzesz, należy ją uszanować do późniejszego wykorzystania. Drugim narzędziem przydatnym w naszych symulacjach jest reprezentacja bloków. Porozmawiajmy o tym.

Reprezentacja blokowa

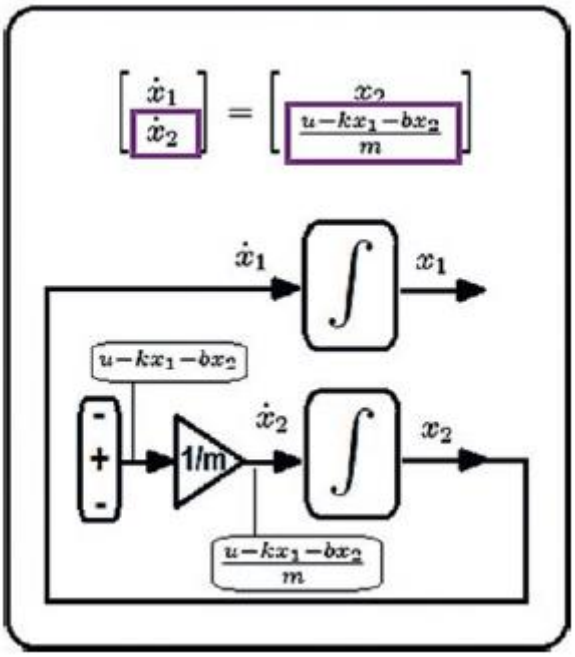
W tym przypadku bloki są graficzną reprezentacją zmiennych przestrzeni stanów. Są przydatne do tworzenia diagramów lub symulacji za pomocą języków wizualnych, takich jak Simulink lub Labview, które są zaprojektowane do korzystania z bloków. Wracając do przykładu masowego amortyzatora sprężynowego, a konkretnie jego reprezentacji w zmiennych w przestrzeni stanów, możemy wykonać następujące czynności, aby uzyskać jego reprezentację blokową. Po pierwsze, zauważ, że istnieją dwa równania w przestrzeni stanów, więc wymagane są dwa główne bloki. Te bloki reprezentują integratorów;



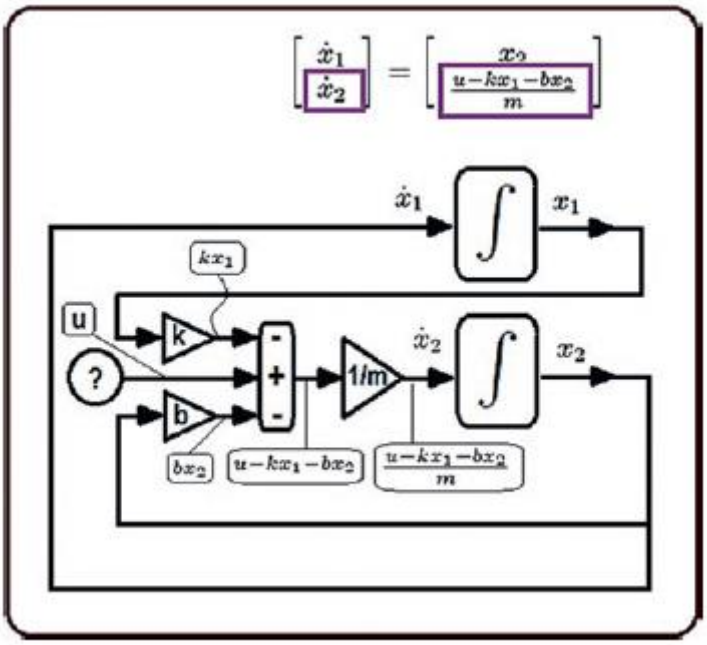
W tym przykładzie dodaliśmy etykiety do stanów, co jest dobrą praktyką przy programowaniu i rozumieniu, ale takie etykietowanie nie jest często wykonywane.Norma implikuje, że po prawej stronie bloku jest wyjście, a po lewej wejście. Zaczniemy od definicji pochodnej x1. Wykonujemy połączenie pokazane na rysunku , które definiuje pochodną x1.



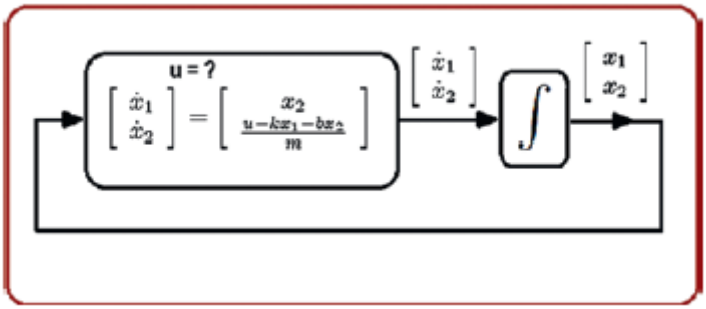
W ten sam sposób możesz zauważyć, że pochodna x2 jest sumą. Z tego powodu będziemy potrzebować kolejnego bloku. Należy również zauważyć, że na wyjściu wspomnianego bloku musi być wzmocnienie;



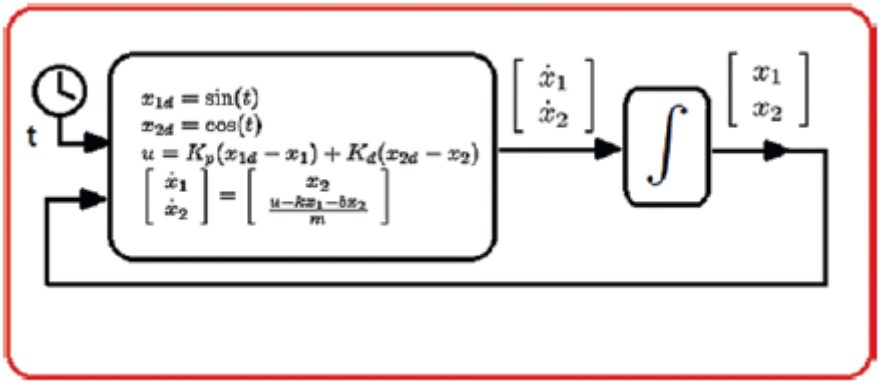
I wreszcie, zgodnie z poprzednią logiką, dochodzimy do rysunku



Mimo że ten system ma tylko kilka równań i terminów, jego reprezentacja blokowa wymagała wielu ikon i łączników (należy również zauważyć, że wartość u nie została zdefiniowana, co może zwiększyć liczbę bloków). Jednym ze sposobów uproszczenia tego jest praca z macierzami i wektorami, sprowadzając równania do jednego bloku, jak pokazano na rysunku



W ten sposób agregat taki jak u można zredukować do kilku linii kodu zamiast dużej liczby bloków. Również zmienna u, na przykład, może być kontrolerem PD, który zależy od trajektorii, która również zależy od czasu. Zauważ, że potrzebujesz bloku zegarowego.



Zauważ, że jeśli system ma wiele równań i każde z nich ma kilka terminów, podobnie jak w przypadku quadkoptera, ten uproszczony model klocków jest bardzo wygodny. Dokładnie to zrobimy w następnej sekcji, używając kombinacji Simulink-MATLAB.

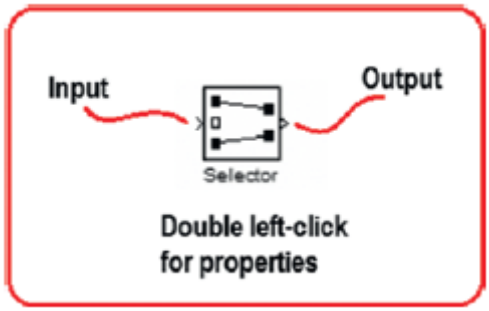
Symulacja za pomocą Simulinka i zinterpretowanych funkcji MATLAB

Jeśli chcesz używać wolnego oprogramowania, takiego jak Scilab lub Python, zaleca się przeczytanie tej sekcji. W oparciu o to, czego się tutaj dowiesz i odniesienia do tej części (które można znaleźć w dodatku), możesz szukać równoważnych poleceń lub bloków we wspomnianych programach i językach programowania.

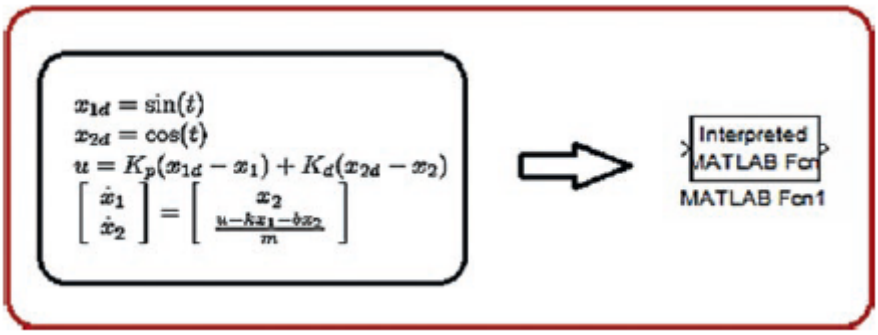
Pamiętaj, że dzięki wielu bibliotekom, które ma ta kombinacja (MATLAB/Simulink), możesz wchodzić w interakcje z animowanym lub interaktywnym oprogramowaniem symulacyjnym, takim jak Gazebo, X-plane, biblioteki Ardupilot, autopilot Pixhawk lub oprogramowanie CAD z silnikami fizycznymi. Kontynuujmy. Jest to specjalny tryb spośród wielu, w których Simulink musi wykonywać symulacje. Oferuje praktyczność bloków w jednej z najbardziej uproszczonych postaci, w której plik tekstowy służy wyłącznie do zapisywania równań układu i jego sterownika. W ten sposób zestaw równań z wieloma wyrażeniami jest przenoszony do pliku tekstowego, a ogólny szablon jest umieszczany w pliku blokowym. Plik blokowy zawiera wstępnie zaprogramowane programy do rozwiązywania równań, takie jak Euler lub RK4 i został opracowany w celu łatwego łączenia i przetwarzania. Ta metoda jest ważna dla wersji MATLAB 2012b lub nowszych. Najlepszą rzeczą w tego typu reprezentacji i symulacji jest to, że można dokonać wielu modyfikacji i zastosowań sygnałów wejściowych i wyjściowych (filtrowanie, edycja wykresu, wprowadzanie opóźnień itp.). Nie jest to możliwe w symulatorach, które nie wykorzystują bloków.

Wymagane bloki

Opisane tutaj bloki Simulinka mają co najmniej jedną z trzech następujących właściwości: sygnał wejściowy, sygnał wyjściowy i sposób ustawiania własnych parametrów.



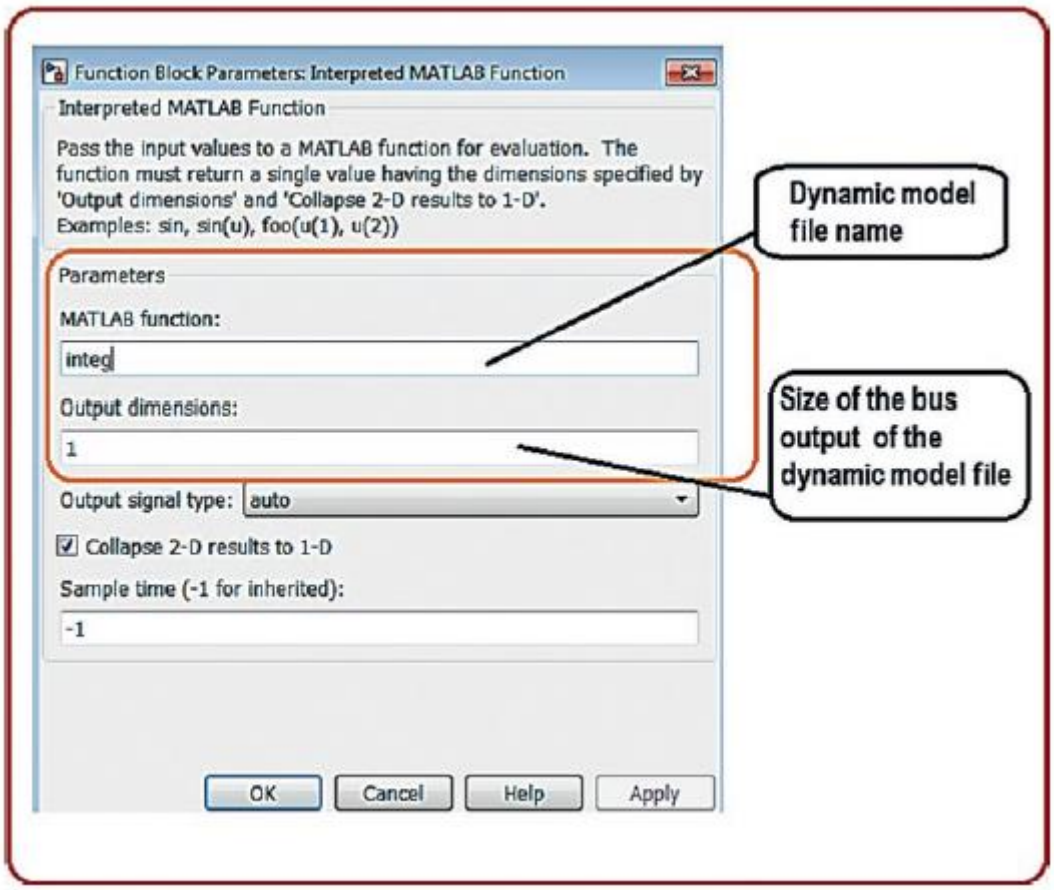
Oto siedem najbardziej użytecznych bloków dla naszych celów. Zinterpretowana funkcja MATLAB: Ten blok służy do łączenia pliku tekstowego zawierającego dynamiczny model systemu i jego kontrolera z plikiem bloku.



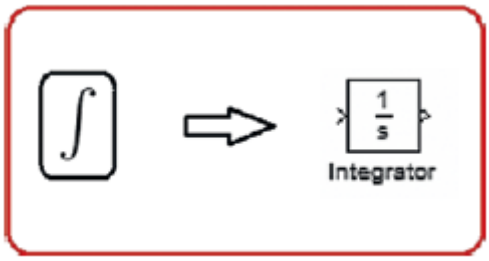
Wejściem tego bloku jest magistrala lub kanał danych z n sygnałami (stany, czas, wartości pomocnicze itp.). Wyjściem tego bloku jest również magistrala lub kanał danych z m sygnałów (stany, wartości pomocnicze, sygnały do wykreślenia itp.). Ważnymi właściwościami tego bloku, którymi jesteśmy zainteresowani są (nie modyfikuj innych właściwości):

1. Nazwa pliku zawierającego dynamiczny model systemu i jego kontrolera. Ta nazwa musi być identyczna z plikiem tekstowym, który zawiera wspomniany model i bez rozszerzenia pliku.

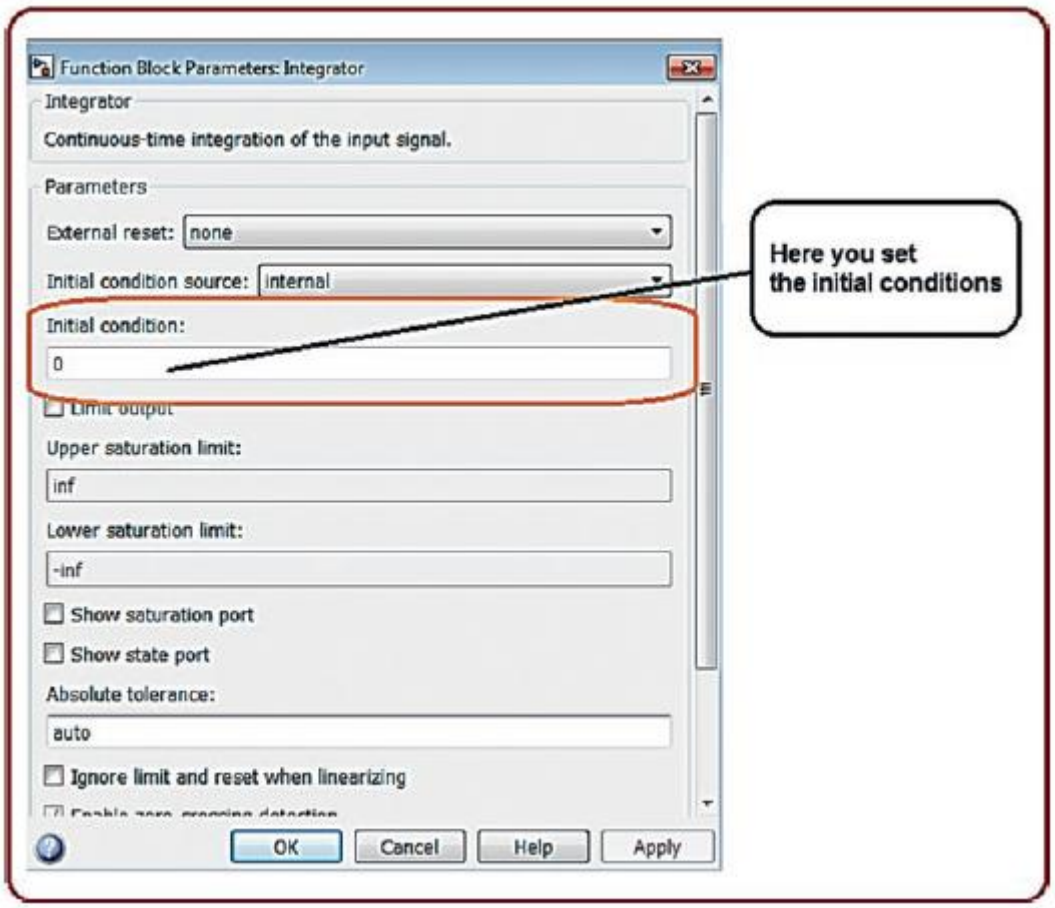
2. Wielkość magistrali wyjściowej (mierzona liczbą zmiennych). Po zmodyfikowaniu tych parametrów wystarczy kliknąć OK i zamknąć okno dialogowe bloku.



Zauważ, że używamy poprzedniego bloku ze względów praktycznych, ale powinieneś również zbadać funkcję MATLAB i bloki funkcji S. Integrator: Umożliwia integrację magistrali sygnałów.



Ten blok służy do integracji sygnałów. W Simulinku istnieją różne typy integratorów, ale dla naszych celów używamy tego właśnie. Wejściem tego bloku jest magistrala lub kanał danych z n sygnałami (na przykład pochodne stanów systemu). Wyjściem tego bloku jest również magistrala lub kanał danych z n sygnałami (np. stany systemu). Jedynym parametrem, który nas interesuje modyfikowanie, są warunki początkowe lub punkt początkowy symulacji naszego systemu. Jeśli jest to pojedyncze równanie, po prostu napisz skalar. Jeżeli istnieje więcej niż jedno równanie, wartości początkowe należy wprowadzić w następujący sposób: [val1 val2 val3 valN]. Po zmodyfikowaniu tego parametru wystarczy kliknąć OK i zamknąć okno dialogowe właściwości.



Zakres: Ten blok pozwala zobaczyć wykresy naszej symulacji bezpośrednio w pliku blokowym, zarówno podczas symulacji, jak i na jej końcu.

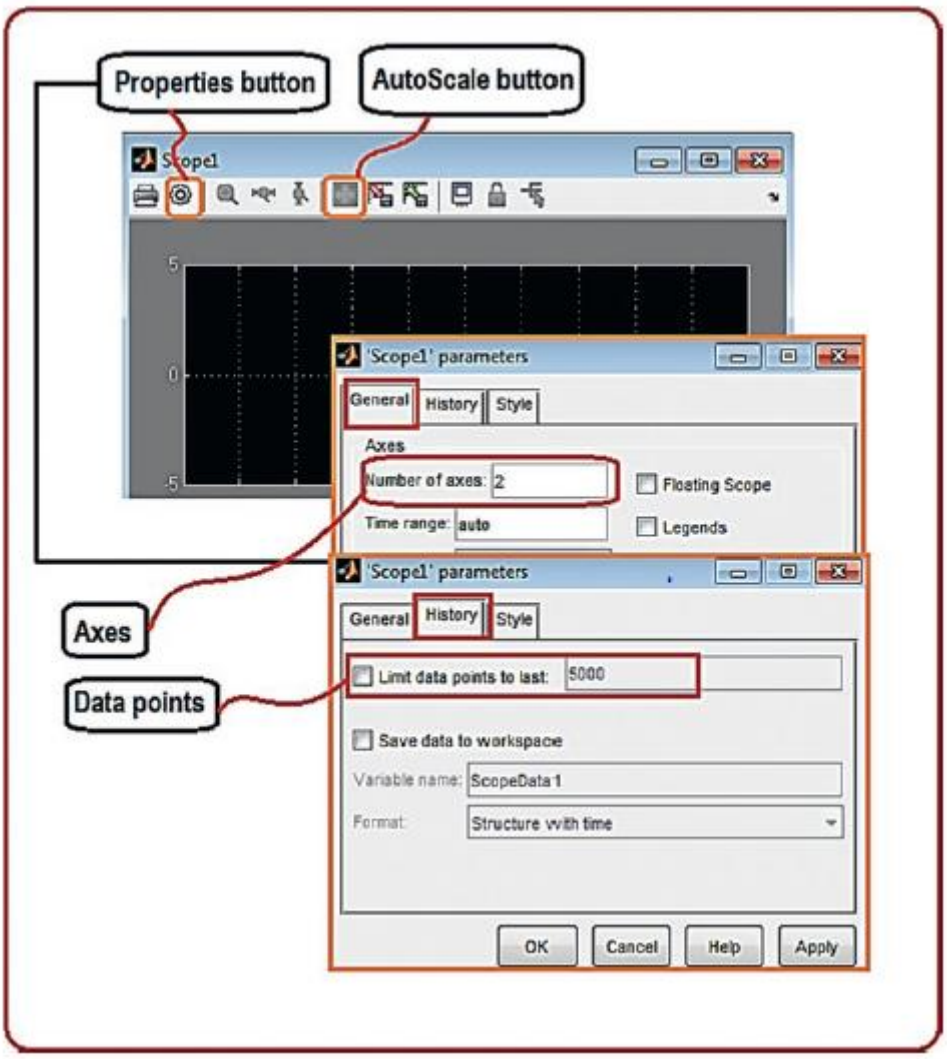
Ten blok nie posiada wyjść. Jego wejścia to magistrale z n sygnałami do wykreślenia. Jego istotne parametry są następujące:

1. Przycisk autoskalowania: Przydaje się do dostosowania wykresu na końcu lub podczas symulacji.

2. Przycisk właściwości: Po naciśnięciu tego przycisku pojawi się menu pomocnicze. Interesują nas tylko dwie opcje:

• Liczba osi wejściowych: Określa, ile niezależnych wykresów będzie w zakresie. Każde wejście do zakresu będzie niezależną szyną n zmiennych. Dostęp do tej opcji można uzyskać z menu Ogólne.

• Pole wyboru limitu danych: jest zwykle aktywne i wyświetla tylko ostatnie 5000 punktów danych. Jeśli chcesz zobaczyć obszerną symulację, musisz dezaktywować to pole wyboru. Ta opcja jest dostępna z menu Historia.



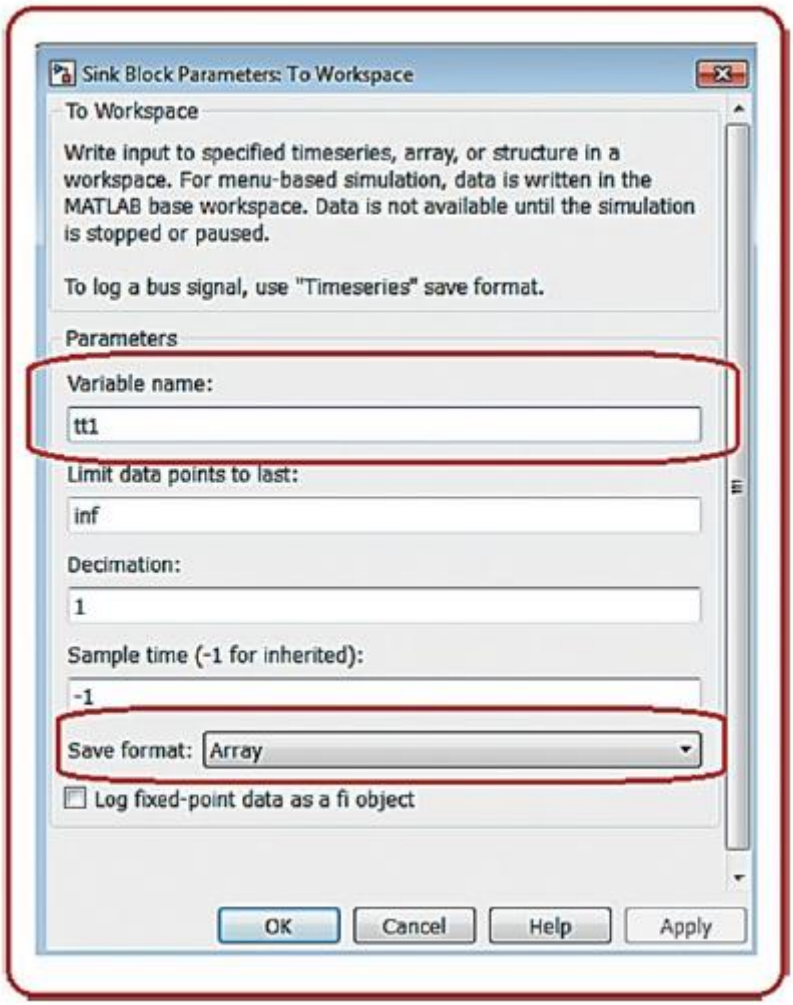
Podobnie jak w poprzednich przypadkach, aby zapisać zmiany, należy kliknąć OK i zamknąć menu pomocnicze. Możesz także poeksperymentować z menu Styl parametrów Scope, jeśli chcesz zmodyfikować prezentację i kolory wykresu.

Blok Workspace: Blok ten umożliwia wykorzystanie określonych danych poza interfejsem Simulink w celu dalszego przetwarzania lub edycji (na przykład eksportowania wykresów w celu edycji ich wykresów do wykorzystania w publikacji).

Ten blok ma tylko wejście, którym jest magistrala danych. Jego interesujące dla naszych celów parametry to tylko dwa:

1. Nazwa zmiennej, która będzie używana z konsoli MATLAB do edycji informacji

2. Typ danych wyjściowych. W naszym przypadku i dla uproszczenia dostępu do danych musi to być zawsze tablica.

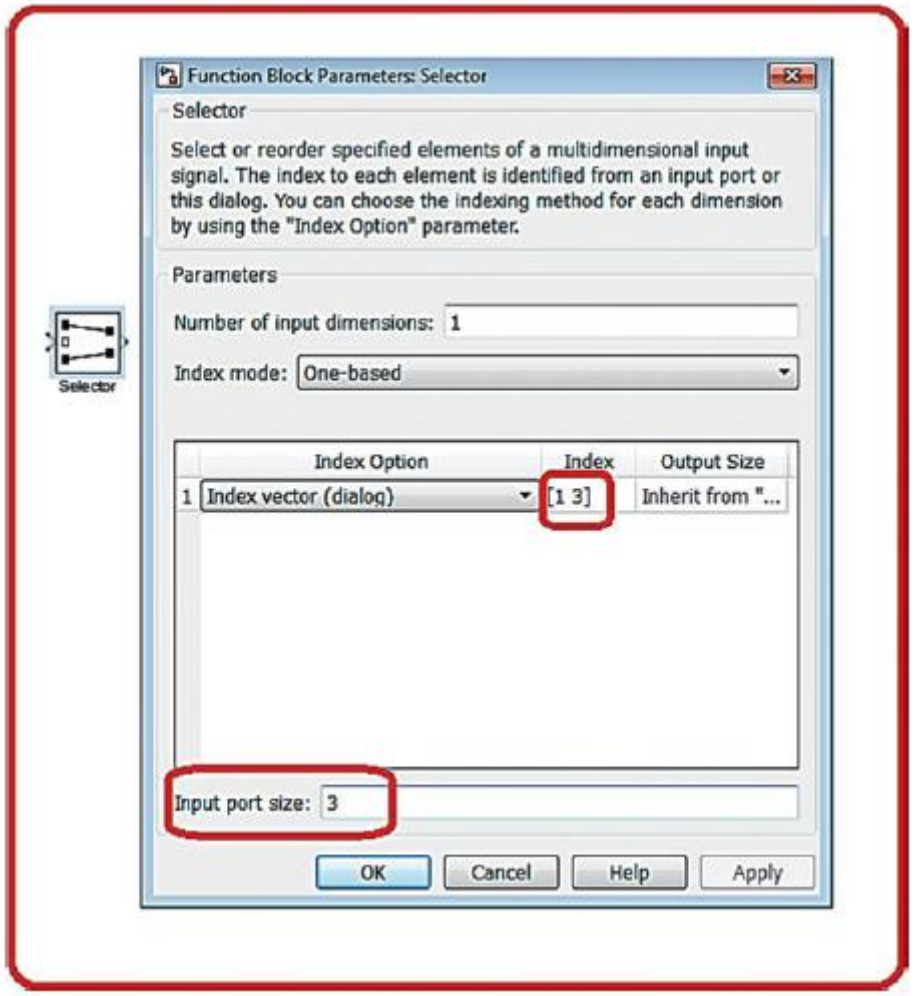


Selektor: Ten blok jest używany do wyboru jednego lub więcej sygnałów z magistrali.

Wejściem do bloku jest magistrala danych z n zmiennymi. Jego wyjściem jest kolejna magistrala z p zmiennych, gdzie p przechodzi od jednego sygnału do n sygnałów. Rozmiar p będzie po prostu zależał od sygnałów, które chcesz wyodrębnić. Jego interesujące parametry to

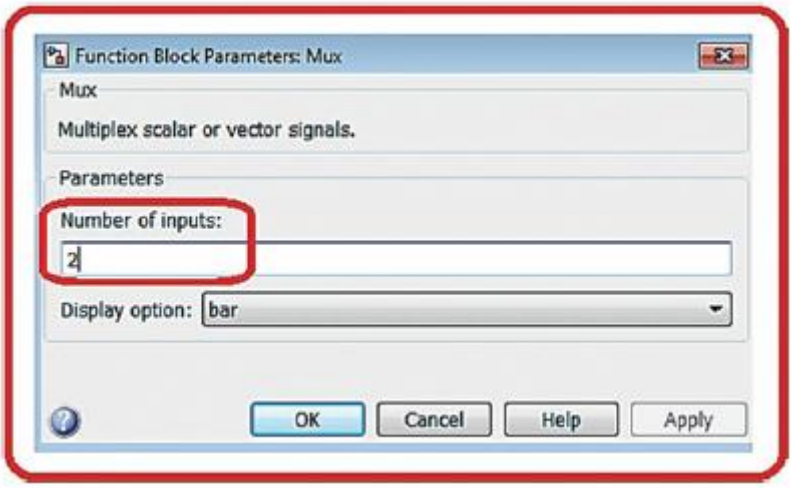
1. Indeks sygnału wyjściowego w formacie skalarnym, jeśli wybrany jest tylko jeden sygnał, lub w formacie wektorowym [val1 val2 val3 valN], jeśli wymagany jest więcej niż jeden sygnał.

2. Wielkość szyny wejściowej.



Przeanalizujmy podany przykład. Zauważ, że jest to graficznie intuicyjne. Blok na rysunku odbiera magistralę danych z trzema zmiennymi, a jej wyjściem jest kolejna magistrala z dwiema zmiennymi, w szczególności pierwszą i trzecią wartością magistrali wejściowej. Mux: Jest to multiplekser, a jego funkcją jest generowanie pojedynczej magistrali z wejścia dwóch lub więcej magistral danych. Przeciwny blok nazywa się Demux. Ten blok jest przydatny do kreślenia w pojedynczym kanale oscyloskopu. Po prostu wybieramy wszystkie interesujące zmienne z różnych bloków w celu wysłania ich do zakresu za pomocą Muxa.

Jego wejścia to kilka magistral z n zmiennych każda, a wyjściem jest pojedyncza magistrala ze wszystkimi zebranymi zmiennymi ze wszystkich magistral wejściowych. Jedynym interesującym parametrem dla naszych aplikacji jest liczba szyn wejściowych.



Zegar: Jest to bardzo przydatne przy definiowaniu trajektorii (funkcje zależne od czasu). Blok ten nie posiada wejść, a jego jedynym wyjściem jest czas symulacji (nie jest to czas rzeczywisty).

Na potrzeby tej książki blok ten nie ma parametrów, które należy zmodyfikować. Inne bloki, które mogą być przydatne: Więcej informacji można znaleźć między innymi w bloku wzmocnienia, bloku pochodnej, bloku sumy, bloku stałej, bloku funkcji matematycznych, etykietach goto i bloku nasycenia. Teraz, gdy już wiesz, jak pracować z blokami, zmiennymi przestrzeni stanów i podstawowymi blokami, których będziemy używać w programie MATLAB, możemy zacząć od naszych symulacji.

Przykład quadkoptera, plik tekstowy

Zwróć uwagę, że ten plik wystarczyłby, gdybyśmy chcieli symulować naszego drona i mieliśmy tylko podstawowy język programowania, taki jak C++ lub Python, oczywiście poprzez włączenie solvera ODE. Pamiętaj, że model będzie tak precyzyjny, jak chcesz, dodając bardziej dynamiczne efekty. W tym przypadku wykorzystamy poprzednie modele, które nie mają wpływu na aerodynamikę, ale służą do sprawdzenia pewnych szczegółów w projekcie sterowania, takich jak fakt, że płaskie wnz nie mogą przekraczać wartości 2Pi (pamiętaj, że te wnz reprezentują pożądane kąty i nie mogą osiągnąć wartości większych niż 2Pi). Jednym ze sposobów na ulepszenie modelu jest dodanie większej liczby efektów lub wywołanie szumu w stanach za pomocą funkcji losowych, a nawet stworzenie opóźnienia sygnałów w celu symulacji zachowania czujników. Plik tekstowy o nazwie quadmodel.m jest wyświetlany na liście 1.

```
function outp=quadmodel(inpt)

% system inputs: states and time

% x1-x, x2-xdot, x3-y, x4-ydot, x5-z, x6-zdot

% x7-theta, x8-thetadot, x9-phi, x10-phidot, x11-psi, x12-psidot

x1=inpt(1);
```

```
x2=inpt(2);
x3=inpt(3);
x4=inpt(4);
x5=inpt(5);
x6=inpt(6);
x7=inpt(7);
x8=inpt(8);
x9=inpt(9);
x10=inpt(10);
x11=inpt(11);
x12=inpt(12);
t=inpt(13);
% constant parameters: mass, inertial values, gravity
m=1;
g=9.8;
Jx=0.2;
Jy=0.2;
Jz=0.2;
% control gains: notice the small values at x and y gains
because they
% define an angle
kpx=0.1;
kdx=0.09;
kpy=0.1;
kdy=0.09;
kpz=1;
kdz=0.7;
kptet=3;
kdtet=1;
kpfi=3;
kdfi=1;
```

```
kppsi=3;
kdpsi=1;
% planar desired values
x1d=5;
x2d=0;
x3d=-6;
x4d=0;
% planar PD: remember that these PD define desired angular
values
PDx=(kpx*(x1d-x1))+(kdx*(x2d-x2));
PDy=(kpy*(x3d-x3))+(kdy*(x4d-x4));
% altitude and attitude desired values
x5d=8+sin(t);
x6d=cos(t);
x7d=PDx;
x8d=0;
x9d=-PDy;
x10d=0;
x11d=0;
x12d=0;
% altitude and attitude PDs
PDz=(kpz*(x5d-x5))+(kdz*(x6d-x6));
PDteta=(kptet*(x7d-x7))+(kdtet*(x8d-x8));
PDfi=(kpfi*(x9d-x9))+(kdfi*(x10d-x10));
PDpsi=(kppsi*(x11d-x11))+(kdpsi*(x12d-x12));
% vehicle's forces and torques
Fbz=PDz+(m*g);
taoteta=PDteta;
taofi=PDfi;
taopsi=PDpsi;
% vehicle's dynamic model xp notation implies xdot or xpoint or x
```

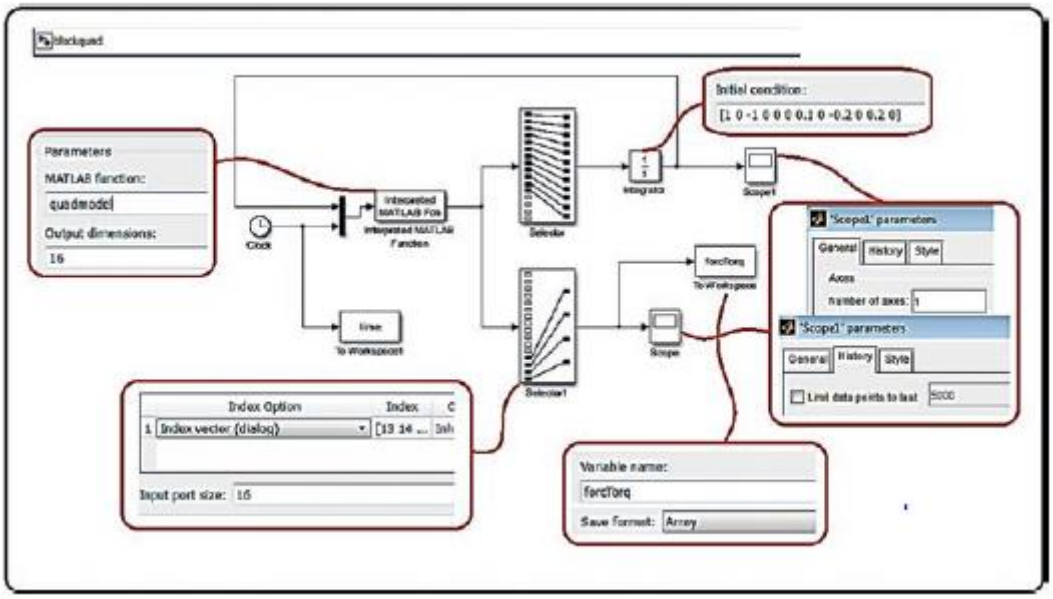
```
% derivative

x1p=x2;

x2p=(Fbz/m)*((x7*cos(x11d))+(x9*sin(x11d)));

x3p=x4;

x4p=(Fbz/m)*((x7*sin(x11d))-(x9*cos(x11d)));

x5p=x6;

x6p=(Fbz/m)-g;

x7p=x8;

x8p=taoteta/Jy;

x9p=x10;

x10p=taofi/Jx;

x11p=x12;

x12p=taopsi/Jz;

% output: derivatives of the system states and vehicle's force

and

% torques

outp=[x1p;x2p;x3p;x4p;x5p;x6p;x7p;x8p;x9p;x10p;x11p;x12p;Fbz;ta

oteta;taofi;taopsi];

end
```

Uwaga: Nazwa pliku tekstowego zawierającego model dynamiczny systemu (ten z rozszerzeniem .m) musi być identyczna ze słowem, które napisaliśmy w wierszu funkcji.

Pamiętaj, że jako alternatywę dla MATLAB i Simulink możesz użyćj Scilab, który ma symulator bloków o nazwie Scicos lub Xcos w zależności od wersji. Scilab to bezpłatne narzędzie programowe i musisz wyszukać odpowiednik siedmiu pokazanych tutaj bloków (na przykład funkcje zdefiniowane przez użytkownika Scicos)

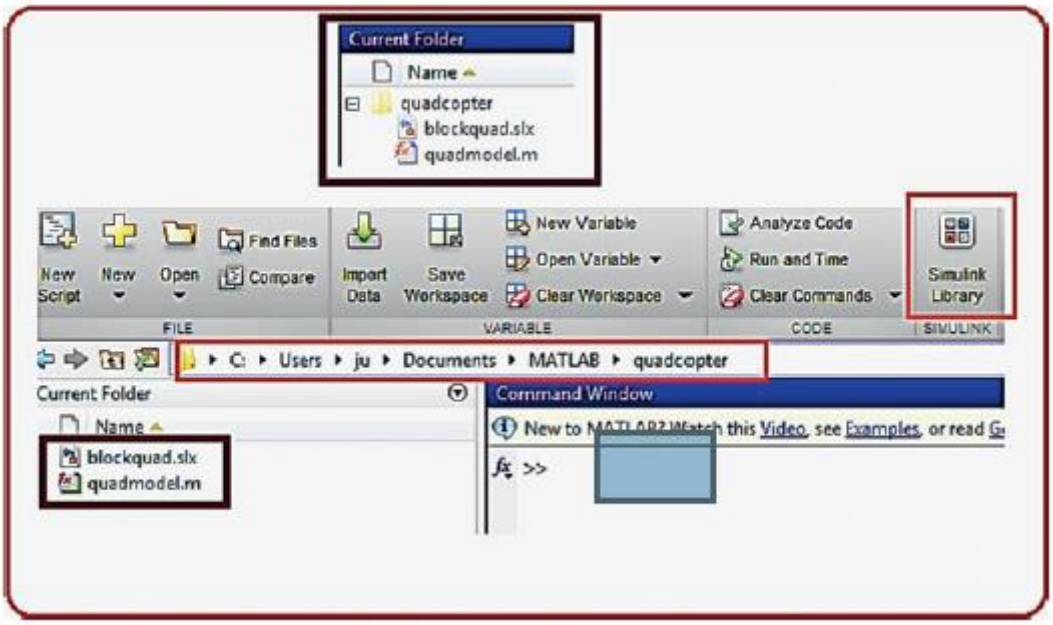
Przykład quadkoptera, plik bloku

W poniższym pliku bloku o nazwie blockquad.slx, wykorzystaliśmy wszystkie z siedmiu wspomnianych wcześniej podstawowych bloków. Zauważ, że złożoność modelu pozostaje w pliku tekstowym. Z tego powodu liczba bloków jest zmniejszona. Ten plik bloku może być użyty jako szablon dla dowolnego innego modelu. Zmienia tylko rozmiar autobusów i plik tekstowy, który zawiera system. Na rysunku wyświetlane jest powiększenie parametrów, które mają być modyfikowane.

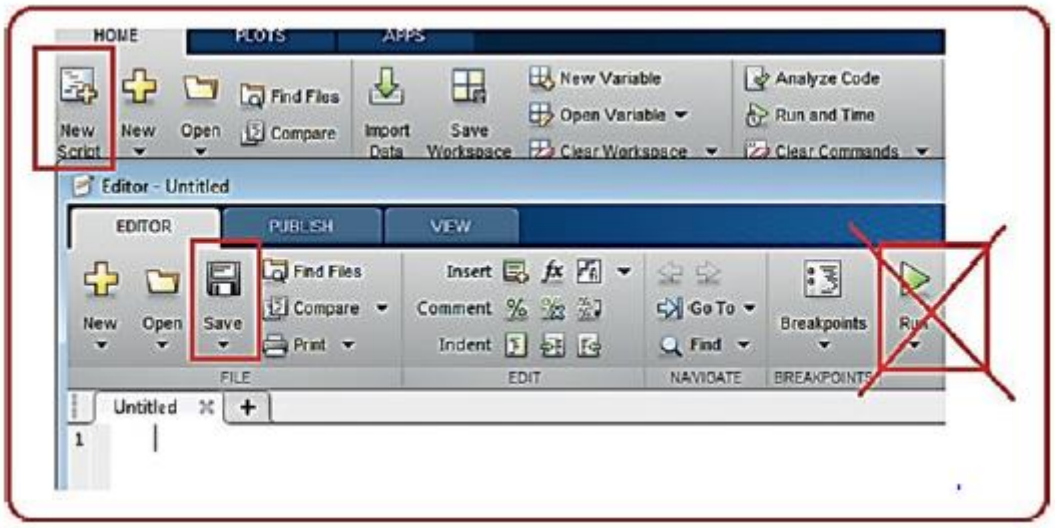


Korzystanie z symulatorów

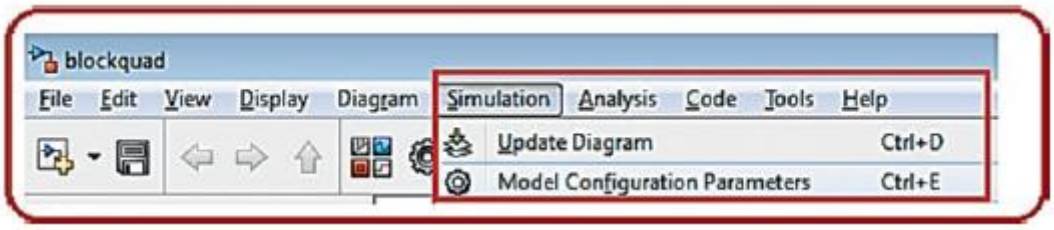
Jak wskazano, aby rozwiązać równanie różniczkowe, wymagany jest solver lub metoda jego rozwiązania, wraz z plikiem zawierającym model lub równania różniczkowe skojarzone z tym modelem. Podczas korzystania z proponowanej metody Simulink oba pliki muszą znajdować się w tym samym folderze, jak pokazano na rysunku . Jeśli plik blockquad.slx nie istnieje, kliknij ikonę biblioteki Simulink i utwórz ten plik, jak pokazano na rysunku.



Jeśli plik quadmodel.m nie istnieje, należy go wygenerować za pomocą przycisku NewScript, jak pokazano na rysunku. Nigdy nie próbuj uruchamiać symulacji z pliku z rozszerzeniem .m. Należy to zrobić z pliku schematu blokowego.



W pliku blokowym i pliku tekstowym wypełnionym dostarczonym kodem w oknie Simulink zauważysz kilka rozwijanych menu. Jeśli przejdziesz do menu Symulacja, a następnie do opcji Parametry konfiguracji modelu, możesz wybierać spośród wielu solwerów.



Funkcje, które chcemy zmodyfikować, to:

• Typ kroku: Tutaj wybierasz, czy krok jest stały czy zmienny. Oznacza to, że czas symulacji jest wykonywany bez zmian w ustalonej jednostce czasu lub jest dostosowywany w miarę rozwoju symulacji.

• Czas trwania kroku: W przypadku wybrania stałych kroków, czas trwania kroku określi jak dokładna iw konsekwencji jak wolna będzie symulacja. Mniejszy krok oznacza bardziej szczegółową symulację (zostanie to odzwierciedlone na wykresach i jakości wyników), ale jej wykonanie zajmie więcej czasu.

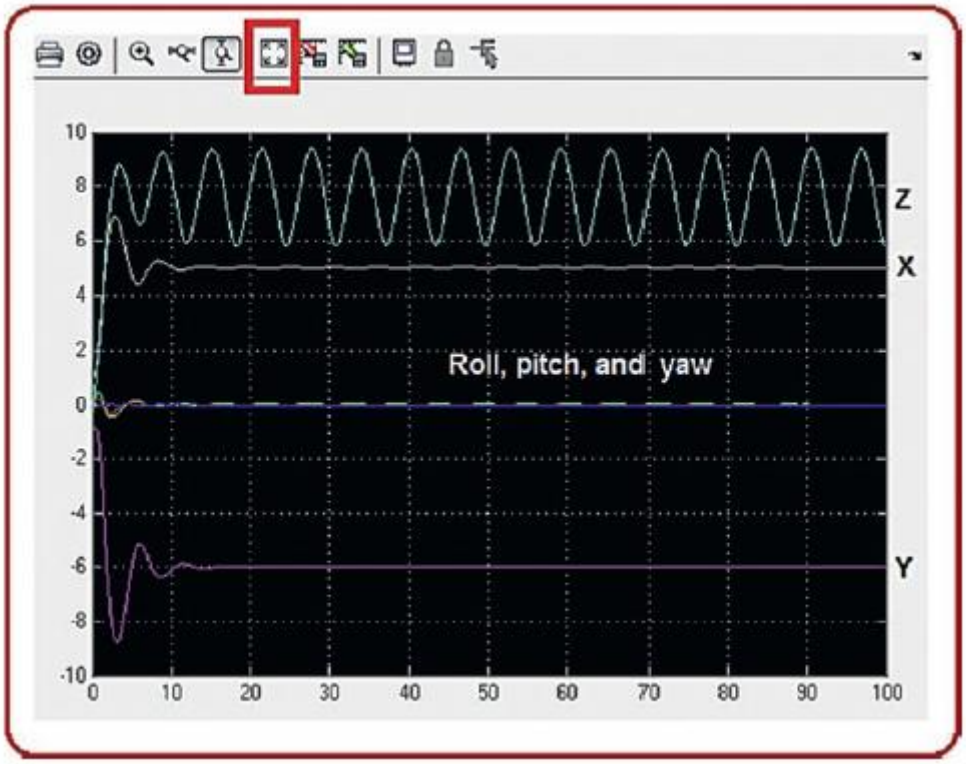
• Metoda: Tutaj wybierasz rodzaj solwera, który ma być użyty. Najczęstsze to Euler i RK4.

Połączenie metody i czasu trwania kroku wpływa na wynik i czas wykonania. Zaleca się stosowanie prostej metody o stosunkowo skróconym kroku. Na przykład metoda Eulera ze stałymi krokami 0,001 sekundy jest dopuszczalną opcją dla quadkoptera;

Ogólnie ta kombinacja będzie używana z resztą naszych symulacji. Po wybraniu solwera i ustawieniu wyżej wymienionych cech, należy zdefiniować całkowity czas symulacji. Kliknij OK, zamknij okno pomocnicze i zapisz zmiany.

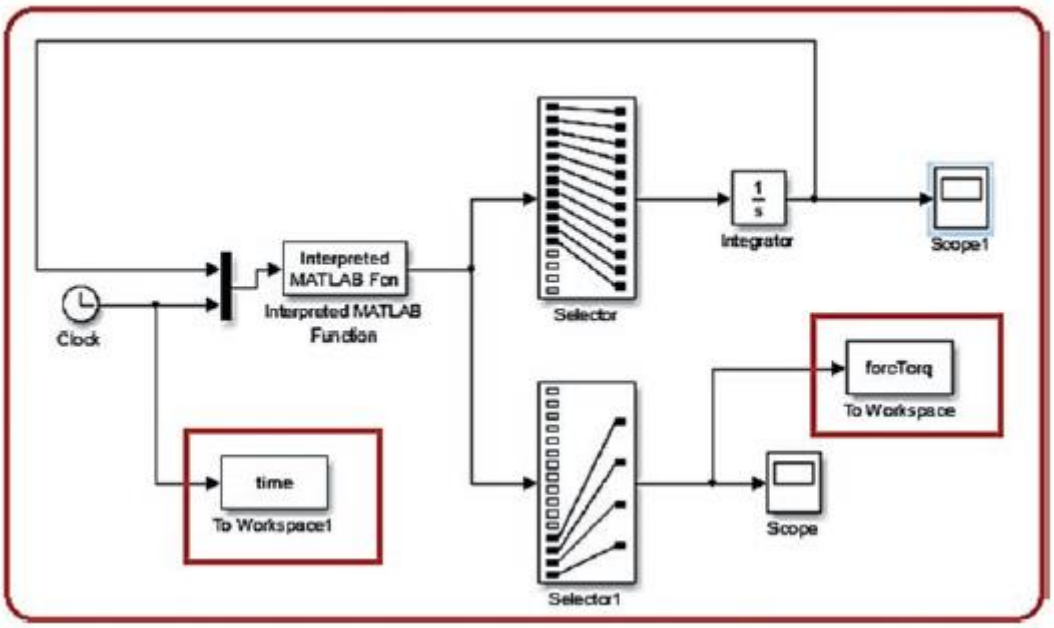
Naciśnij przycisk odtwarzania i poczekaj na wynik; patrz Rysunek 4-27. Zauważ, że czas symulacji i czas rzeczywisty nie są takie same. Czas symulacji będzie zależał od procesora maszyny i pożądane jest, aby symulacja została wykonana tak szybko, jak to możliwe. Gdyby tak nie było, czas symulacji wynoszący 10 000 sekund oznaczałby czekanie prawie 3 godziny.

Po naciśnięciu dowolnego zakresu w pliku bloku, pojawią się wykresy symulacji.

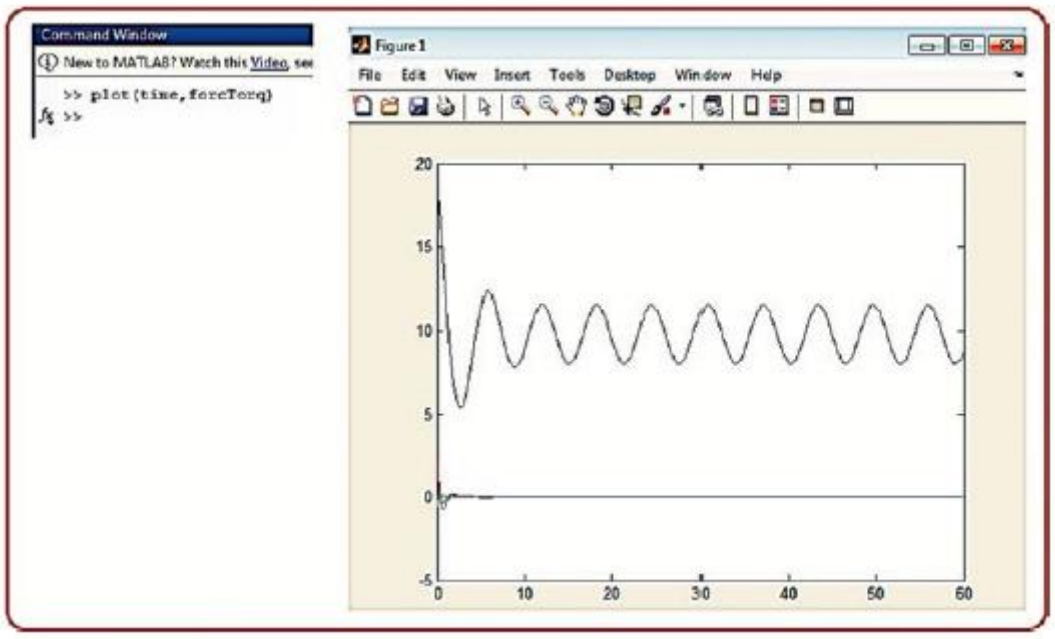


W tym przykładzie stany zbiegają się do żądanych wartości z danym kontrolerem i odniesieniami ustawionymi w pliku quadmodel.m. Wygodnie jest nacisnąć przycisk autoskalowania w menu scope.

Jeśli chcemy używać wykresów poza zakresami symulatora, używamy bloków To Workspace i okna poleceń MATLAB.



W podanym przykładzie do obszaru roboczego zostały wysłane dwie magistrale danych: czas symulacji zwany czasem oraz siły i momenty zwane forcTorq. Po uruchomieniu symulacji wystarczy przejść do okna poleceń MATLAB i wpisać to, aby użyć wyżej wymienionych danych: wykres (czas, forcTorq). Pojawi się wykres przedstawiony na rysunku.



Zwróć uwagę, że opracowane tutaj symulacje pominą efekt silników, ponieważ skupiają się na kontrolowaniu środka ciężkości pojazdu. Zakłada się, że macierz alokacji ma jedynie liniowy wpływ na przenoszenie sterowania w kierunku silników ze względu na wcześniej wskazane fakty symetrii, równowagi i punktowego rozkładu masy. Jeśli chcesz przeanalizować działanie silników, wystarczyłoby wziąć z symulacji sygnały siły ciągu i momentu obrotowego i pomnożyć te wartości przez odpowiednią macierz alokacji (oczywiście konieczne byłoby uwzględnienie współczynników skalujących, aby była kompatybilna z prawdziwe silniki). Odbywa się to zgodnie z opisem w Listingu 2

function outp=allocquadmodel(inpt)

```
% system inputs: states and time
% x1-x, x2-xdot, x3-y, x4-ydot, x5-z, x6-zdot
% x7-theta, x8-thetadot, x9-phi, x10-phidot, x11-psi, x12-psidot
x1=inpt(1);
x2=inpt(2);
x3=inpt(3);
x4=inpt(4);
x5=inpt(5);
x6=inpt(6);
x7=inpt(7);
x8=inpt(8);
x9=inpt(9);
x10=inpt(10);
x11=inpt(11);
x12=inpt(12);
t=inpt(13);
% constant parameters: mass, inertial values, gravity
m=1;
g=9.8;
Jx=0.2;
Jy=0.2;
Jz=0.2;
% control gains notice that in order to make visible the motor
effects
% we reduce the original differential gains
kpx=0.1;
kdx=0.03;
kpy=0.1;
kdy=0.03;
kpz=1;
kdz=0.7;
```

```
kptet=3;
kdtet=0.5;
kpfi=3;
kdfi=0.5;
kppsi=3;
kdpsi=1;
% planar desired values
x1d=5;
x2d=0;
x3d=-6;
x4d=0;
% planar PD: remember that these PD define desired angular values
PDx=(kpx*(x1d-x1))+(kdx*(x2d-x2));
PDy=(kpy*(x3d-x3))+(kdy*(x4d-x4));
% altitude and attitude desired values
x5d=8+sin(t);
x6d=cos(t);
x7d=PDx;
x8d=0;
x9d=-PDy;
x10d=0;
x11d=0;
x12d=0;
% altitude and attitude PDs
PDz=(kpz*(x5d-x5))+(kdz*(x6d-x6));
PDteta=(kptet*(x7d-x7))+(kdtet*(x8d-x8));
PDfi=(kpfi*(x9d-x9))+(kdfi*(x10d-x10));
PDpsi=(kppsi*(x11d-x11))+(kdpsi*(x12d-x12));
% vehicle's forces and torques
Fbz=PDz+(m*g);
%Fbz=20;
```

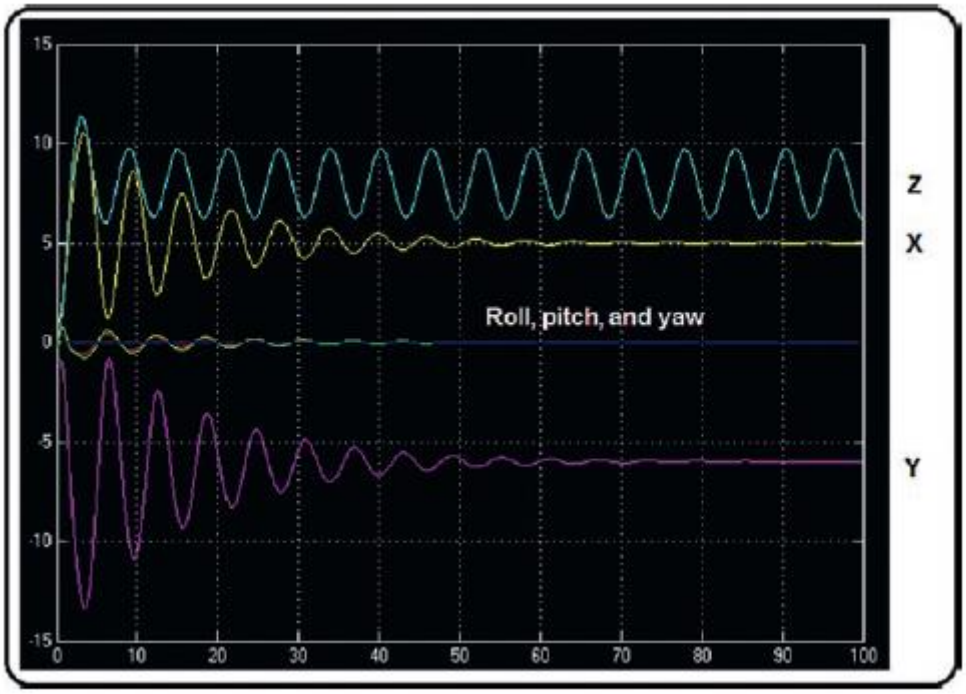
```
taoteta=PDteta;
taofi=PDfi;
taopsi=PDpsi;
% vehicle's dynamic model
x1p=x2;
x2p=(Fbz/m)*((x7*cos(x11d))+(x9*sin(x11d)));
x3p=x4;
x4p=(Fbz/m)*((x7*sin(x11d))-(x9*cos(x11d)));
x5p=x6;
x6p=(Fbz/m)-g;
x7p=x8;
x8p=taoteta/Jy;
x9p=x10;
x10p=taofi/Jx;
x11p=x12;
x12p=taopsi/Jz;
% Here goes the motor effects throught the allocation matrix
W=[1,-1,1,1;1,1,-1,1;1,1,1,-1;1,-1,-1,-
1]*[Fbz;taoteta;taofi;taopsi];
w1=W(1);
w2=W(2);
w3=W(3);
w4=W(4);
% output: derivatives of system states and vehicle force and
torques
%outp=[x1p;x2p;x3p;x4p;x5p;x6p;x7p;x8p;x9p;x10p;x11p;x12p;Fbz;
taoteta
%taofi;taopsi];
% here we change the end of the line above by replacing forces and
% torques by the motor velocities, the Simulink® file does not need
% modifications
```

outp=[x1p;x2p;x3p;x4p;x5p;x6p;x7p;x8p;x9p;x10p;x11p;x12p;w1;w2;

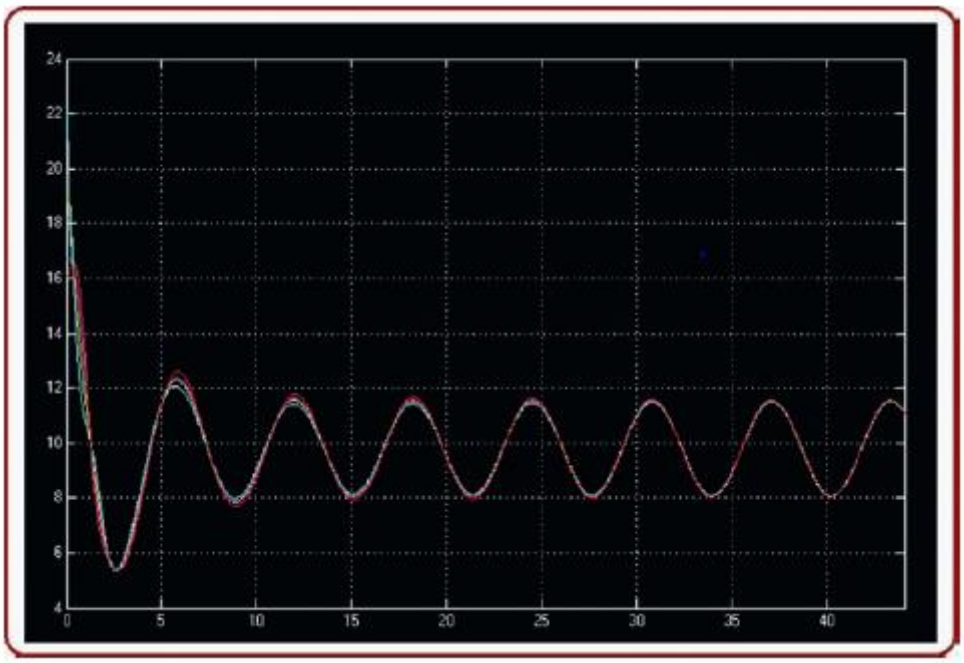
w3;w4];

end

Jeśli przeanalizujemy nowe wykresy, efekt stabilizacji osiąga się teraz około 30 sekund w zmiennych kątowych i przy dużej obecności oscylacji w X i Y .



Wpłynie to na zachowanie każdego silnika przez co najmniej około 30 sekund, a następnie będą działać identycznie;



(dron pozostanie nieruchomy i nie zmieni swojego położenia, tylko jego wysokość; w ten sposób silniki będą zachowywać się identycznie).

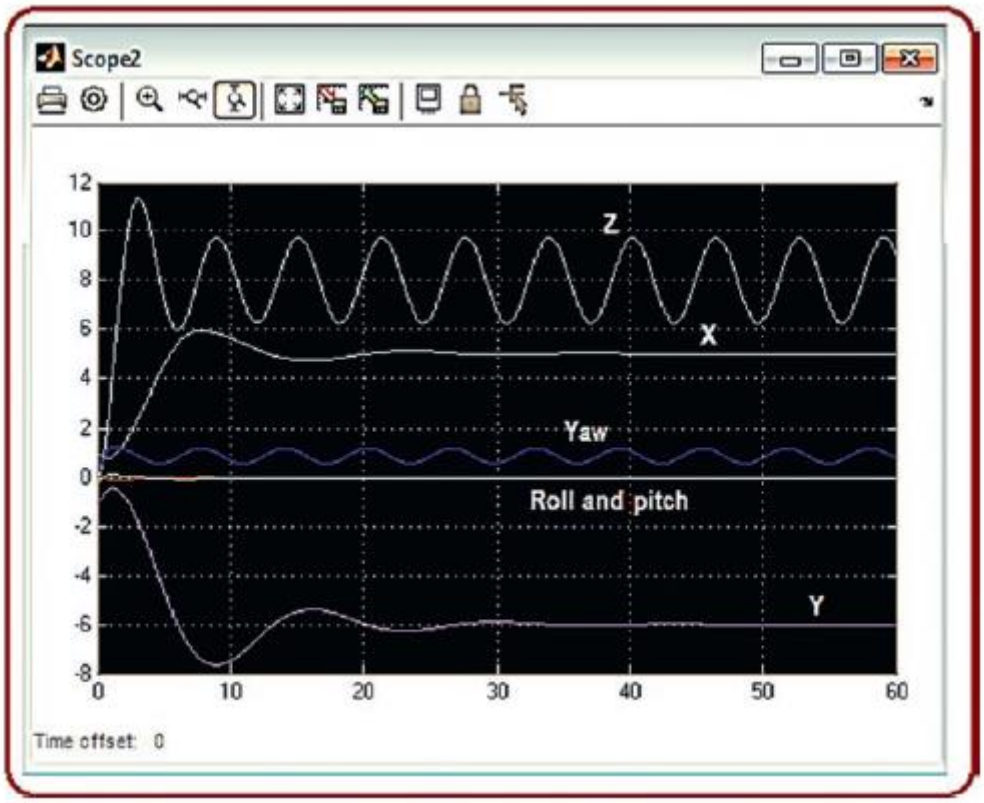
Teraz plik quadmodel.m zostaje zmodyfikowany, jak pokazano na Listingu 3, w celu wykonania zmiennej kontroli odchylenia.

```
function outp=quadmodel2(inpt)

% system inputs: states and time

% x1-x, x2-xdot, x3-y, x4-ydot, x5-z, x6-zdot

% x7-theta, x8-thetadot, x9-phi, x10-phidot, x11-psi, x12-psidot

x1=inpt(1);

x2=inpt(2);

x3=inpt(3);

x4=inpt(4);

x5=inpt(5);

x6=inpt(6);

x7=inpt(7);

x8=inpt(8);

x9=inpt(9);

x10=inpt(10);

x11=inpt(11);

x12=inpt(12);

t=inpt(13);

% constant parameters: mass, inertial values, gravity

m=1;

g=9.8;

Jx=0.2;

Jy=0.2;

Jz=0.2;

% control gains

kpx=0.2;

% sometimes because of the noise or just the control

interaction

% differential gains could be greater than proportional ones
```

```
% try the opposite and see the resulting behavior
kdx=0.35;
kpy=0.2;
kdy=0.35;
kpz=1;
kdz=0.7;
kptet=5;
kdtet=2;
kpfi=5;
kdfi=2;
kppsi=3;
kdpsi=1;
% planar desired values
x1d=5;
x2d=0;
x3d=-6;
x4d=0;
% planar PDs: remember that these PDs define desired angular
values
PDx=(kpx*(x1d-x1))+(kdx*(x2d-x2));
PDy=(kpy*(x3d-x3))+(kdy*(x4d-x4));
% altitude desired values
x5d=8+sin(t);
x6d=cos(t);
% yaw desired values
x11d=0.3*sin(t)+0.9;
x12d= 0.3*cos(t);
% roll and pitch desired values, instead of x11d, try changing
to x11
x7d=(1/g)*((PDx*cos(x11d))+(PDy*sin(x11d)));
% instead of cero try to change x8d to the x7d derivative
```

```
x8d=0;
x9d=(1/g)*((PDx*sin(x11d))-(PDy*cos(x11d)));
% instead of cero try to change x10d to the x9d derivative
x10d=0;
% altitude and attitude PDs
PDz=(kpz*(x5d-x5))+(kdz*(x6d-x6));
PDteta=(kptet*(x7d-x7))+(kdtet*(x8d-x8));
PDfi=(kpfi*(x9d-x9))+(kdfi*(x10d-x10));
PDpsi=(kppsi*(x11d-x11))+(kdpsi*(x12d-x12));
% vehicle's force and torques
Fbz=PDz+(m*g);
taoteta=PDteta;
taofi=PDfi;
taopsi=PDpsi;
% vehicle's dynamic model
x1p=x2;
x2p=(Fbz/m)*((x7*cos(x11d))+(x9*sin(x11d)));
x3p=x4;
x4p=(Fbz/m)*((x7*sin(x11d))-(x9*cos(x11d)));
x5p=x6;
x6p=(Fbz/m)-g;
x7p=x8;
x8p=taoteta/Jy;
x9p=x10;
x10p=taofi/Jx;
x11p=x12;
x12p=taopsi/Jz;
%output: derivatives of the system states and vehicle's force and
%torques
outp=[x1p;x2p;x3p;x4p;x5p;x6p;x7p;x8p;x9p;x10p;x11p;x12p;Fbz;
taoteta;taofi;taopsi];
```

end



Plik blockquad pozostaje nienaruszony. Na koniec pokazana jest symulacja sterowania geometrycznego. Posługujemy się zestawem trajektorii podobnym do tego, który znajdujemy we wcześniej wskazanych publikacjach Taeyoung Lee, ale o profilu wolniejszego ruchu, aby użyć prostego kontrolera PD. Jeśli chcesz pójść dalej, możesz zmodyfikować następujący kod, dodając wymagane warunki, aż osiągniesz wysoką wydajność, jak we wspomnianych artykułach.

```
function outp=geoquad(inpt)

% system inputs: states

% x xderivative y yder z zder

x1=inpt(1);

x2=inpt(2);

x3=inpt(3);

x4=inpt(4);

x5=inpt(5);

x6=inpt(6);

% w and R (angular velocity and orientation)

w1=inpt(7);

w2=inpt(8);
```

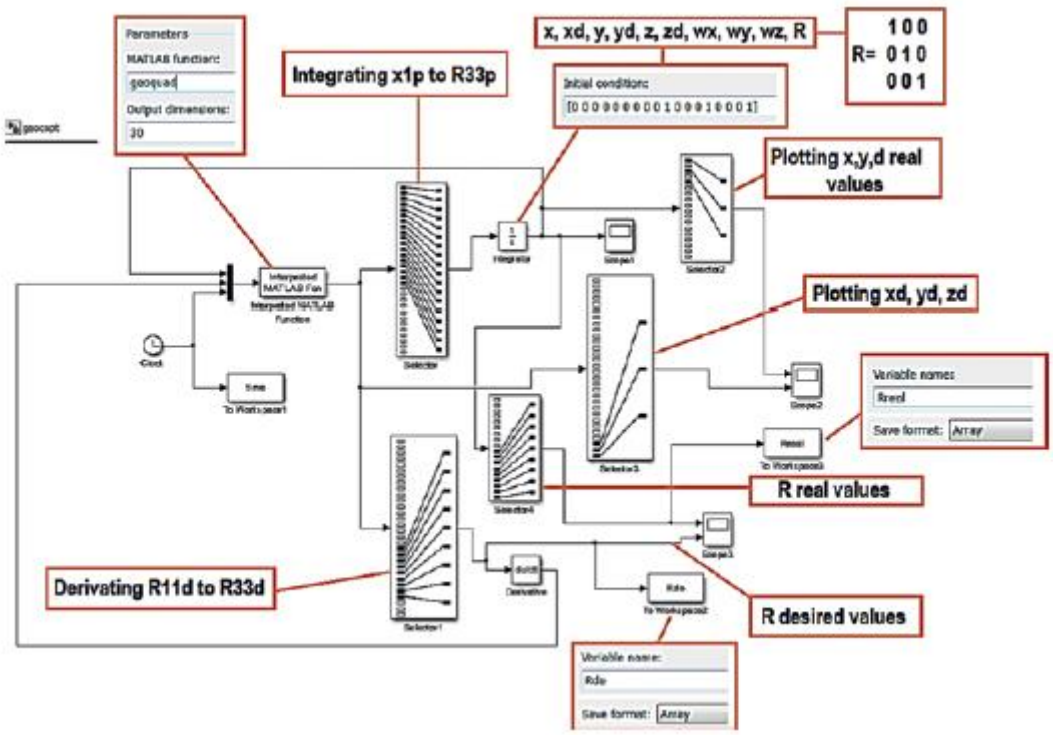
```
w3=inpt(9);
r11=inpt(10);
r12=inpt(11);
r13=inpt(12);
r21=inpt(13);
r22=inpt(14);
r23=inpt(15);
r31=inpt(16);
r32=inpt(17);
r33=inpt(18);
% we need also Rdesired derivative
r11dp=inpt(19);
r12dp=inpt(20);
r13dp=inpt(21);
r21dp=inpt(22);
r22dp=inpt(23);
r23dp=inpt(24);
r31dp=inpt(25);
r32dp=inpt(26);
r33dp=inpt(27);
% time
t=inpt(28);
% constant parameters: mass, inertial values, gravity
m=1;
g=9.8;
Jx=0.2;
Jy=0.2;
Jz=0.2;
% building the desired Rotation matrix from its components
Rdp=[r11dp,r12dp,r13dp;r21dp,r22dp,r23dp;r31dp,r32dp,r33dp];
% building also the rotation matriz and the skew angular
```

```
velocity
% matrix
R=[r11,r12,r13;r21,r22,r23;r31,r32,r33];
S=[0,-w3,w2;w3,0,-w1;-w2,w1,0];
% control gains
kpx=0.3;
% sometimes because of the noise or due to control interaction
% differential gains could be greater tham propotional ones
% try the opposite and see the resulting behavior
kdx=0.5;
kpy=0.3;
kdy=0.5;
kpz=0.7;
kdz=0.5;
KR=50;
Kw=30;
% cartesian translational desired values
% this is an helical trajectory
x1d=0.4*t;
x2d=0.4;
x3d=0.4*sin(t/2);
x4d=0.2*cos(t/2);
x5d=0.6*cos(t/2);
x6d=-0.3*sin(t/2);
% planar PD
PDx=(kpx*(x1d-x1))+(kdx*(x2d-x2));
PDy=(kpy*(x3d-x3))+(kdy*(x4d-x4));
PDz=(kpz*(x5d-x5))+(kdz*(x6d-x6));
% Main thrust
Fbz=dot(([PDx;PDy;PDz]+[0;0;m*g]),(R*[0;0;1]));
% try to increase the angular frequency of these motion profile in
```

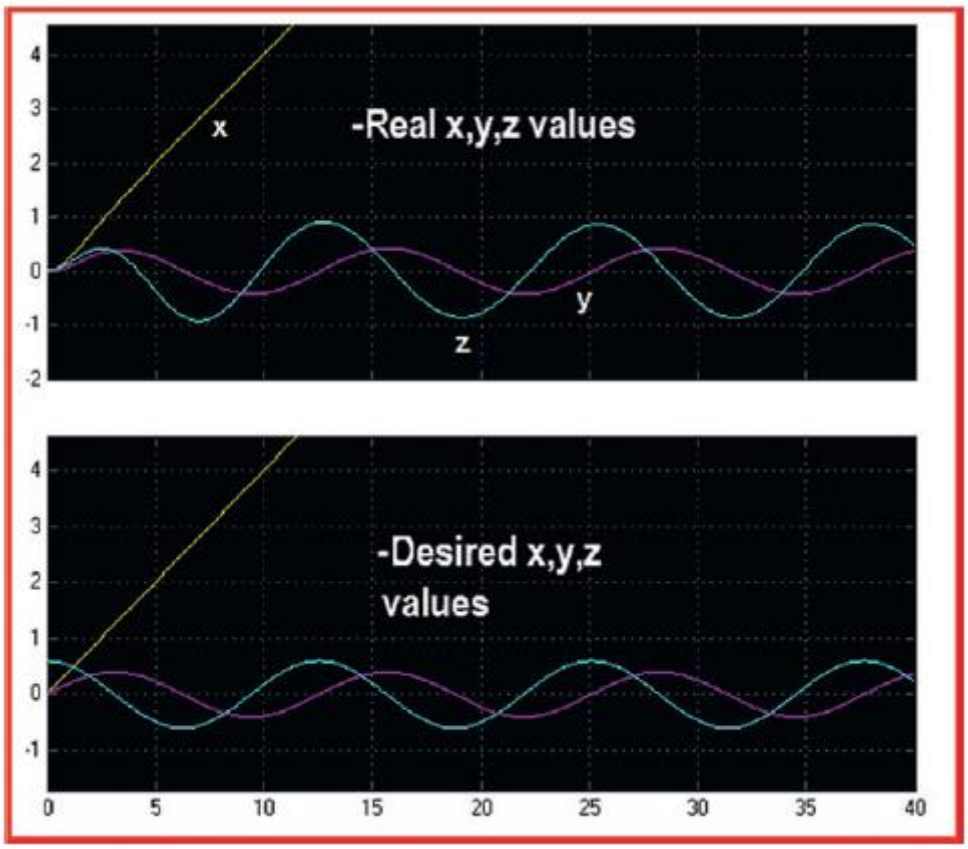
```
% order to disrupt the control loop by increasing velocities and
% accelerations as a consequence, in this case add the full
controller
% above described, or an improvement, because here we are
simulating
% just a simple PD
% Rd definition notice the indirect regulation of the yaw by means
% of the Bdes vector (a circular motion)
psid=t/3;
Bdes=[cos(psid);sin(psid);0];
b3d=[PDx;PDy;PDz]+[0;0;m*g]/norm([PDx;PDy;PDz]+[0;0;m*g]);
b2d=cross(b3d,Bdes)/norm(cross(b3d,Bdes));
b1d=cross(b2d,b3d);
Rd=[b1d,b2d,b3d];
% here the desired rotation matrix is decomposed into its
% components in order to be sent as a vector in the output of
% this model,we need to send them in this order to calculate
% their derivatives
r11d=Rd(1,1);
r12d=Rd(1,2);
r13d=Rd(1,3);
r21d=Rd(2,1);
r22d=Rd(2,2);
r23d=Rd(2,3);
r31d=Rd(3,1);
r32d=Rd(3,2);
r33d=Rd(3,3);
% definition of the rotation and angular velocities errors
Rmul=transpose(Rd)*Rdp;
eRM=(transpose(Rd)*R)-(transpose(R)*Rd);
w=[w1;w2;w3];
```

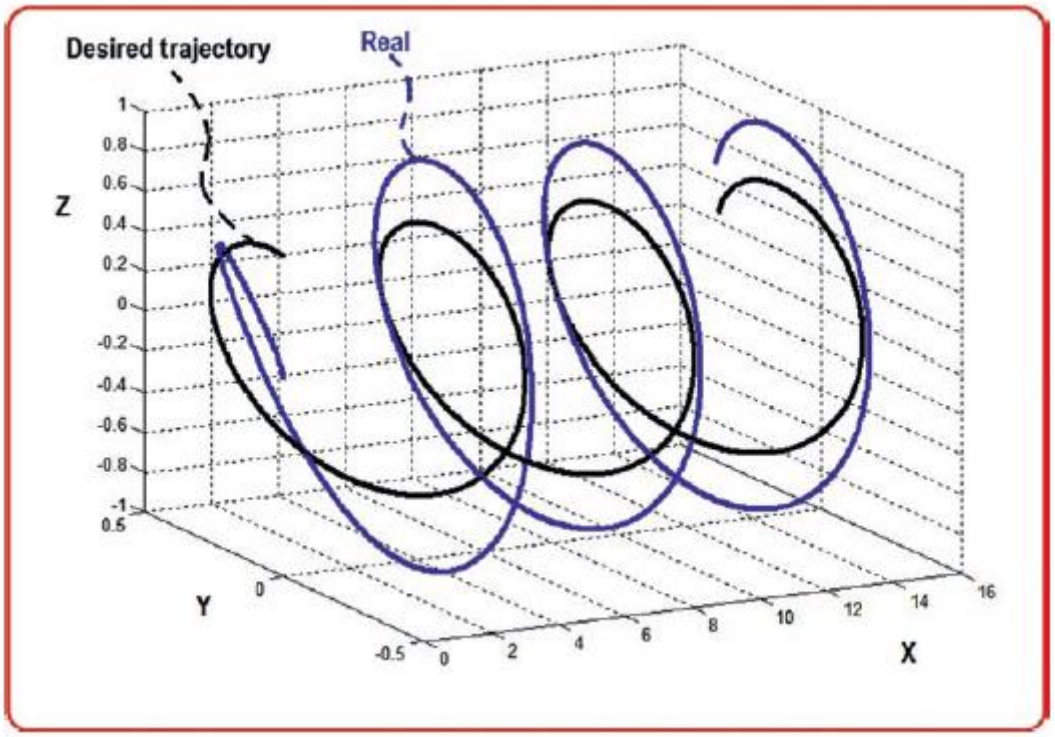
```
wd=[Rmul(3,2);Rmul(1,3);Rmul(2,1)];
ew=w-(transpose(R)*Rd*wd);
eR=[eRM(3,2);eRM(1,3);eRM(2,1)]/2;
% simple rotational PD
PDrw=-KR*eR-Kw*ew;
% here PDrw is decomposed in order to be injected into the 3 body
% torques
Tbx=PDrw(1);
Tby=PDrw(2);
Tbz=PDrw(3);
% vehicle's dynamic model traslational
% velocities
xp=[x2;x4;x6];
% accelerations
xpp=(R*[0;0;Fbz])-[0;0;m*g];
% we do this in order to send state variables for integration
% xdot xdoubledot ydot ydoubledot zdot zdoubledot
x1p=xp(1);
x2p=xpp(1);
x3p=xp(2);
x4p=xpp(2);
x5p=xp(3);
x6p=xpp(3);
% vehicle's rotational dynamics
w1p=(Tbx-(w2*w3*Jz)+(w2*w3*Jy))/Jx;
w2p=(Tby-(w1*w3*Jx)+(w1*w3*Jz))/Jy;
w3p=(Tbz-(w1*w2*Jy)+(w1*w2*Jx))/Jz;
% auxiliar kinematics of rotation
Rp=R*(S);
% we decompose Rderivative into its components for sending to the
% output the integrating and obteining R elements, notice that
```

```
we need
% R initial values at the Simulink® solver
r11p=Rp(1,1);
r12p=Rp(1,2);
r13p=Rp(1,3);
r21p=Rp(2,1);
r22p=Rp(2,2);
r23p=Rp(2,3);
r31p=Rp(3,1);
r32p=Rp(3,2);
r33p=Rp(3,3);
% output:
% translational system states, elements 1-6
% angular velocities, elements 7-9
% derivative of rotation matrix, elements 10-18
% desired rotation matrix for obtaining its derivative,
elements 19-27
% desired translational values just for plotting, elements 28-30
outp=[x1p;x2p;x3p;x4p;x5p;x6p;w1p;w2p;w3p;r11p;r12p;r13p;r21p;
r22p;r23p;r31p;r32p;r33p;r11d;r12d;r13d;r21d;r22d;r23d;r31d;
r32d;r33d;x1d;x3d;x5d];
end
```

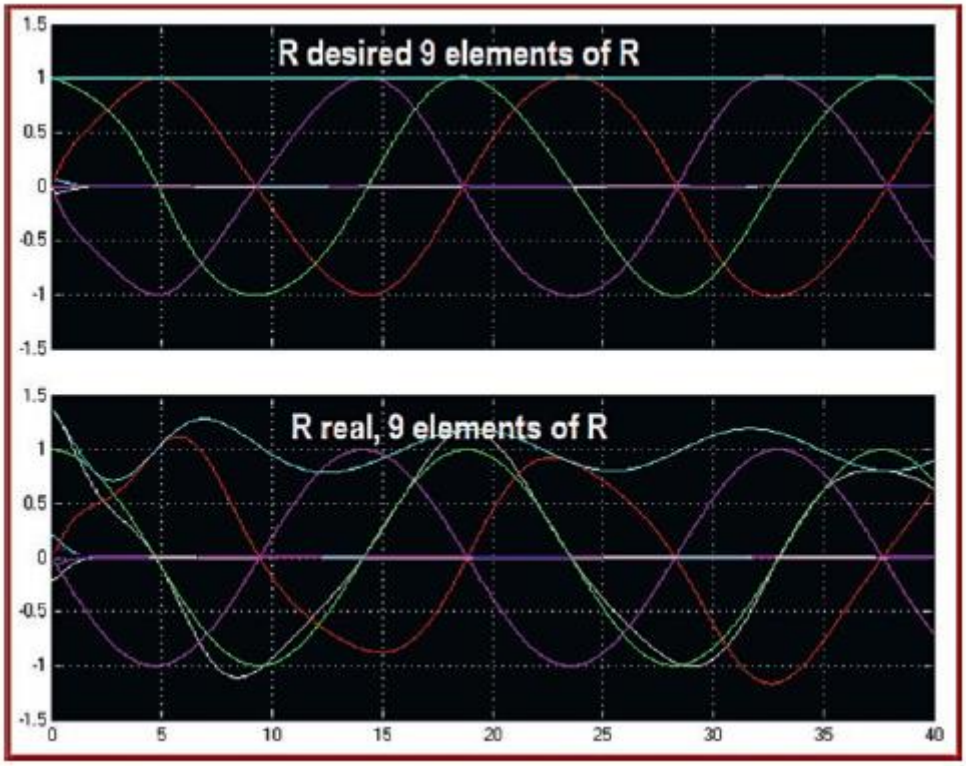


przedstawia plik bloku Simulink o nazwie geocopt.slx z wymaganymi modyfikacjami. Do zarządzania danymi macierz rotacji była traktowana jako wektor (dekonstruowany element po elemencie). Na rysunkach





przedstawiono wartości translacyjne, a na rysunku



wartości rotacyjne wyświetlane jako składowe macierzy rotacji. Oprócz trajektorii translacyjnych pokazane są macierze żądanej i zmierzonej rotacji. Dla lepszej wizualizacji tych wyników postaw możesz pamiętać, że

$$R = R_{z_\psi} R_{x_\phi} R_{y_\theta}$$

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + s\phi s\psi c\theta \\ s\psi c\theta + s\phi c\psi s\theta & c\phi c\psi & s\psi s\theta - s\phi c\psi c\theta \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

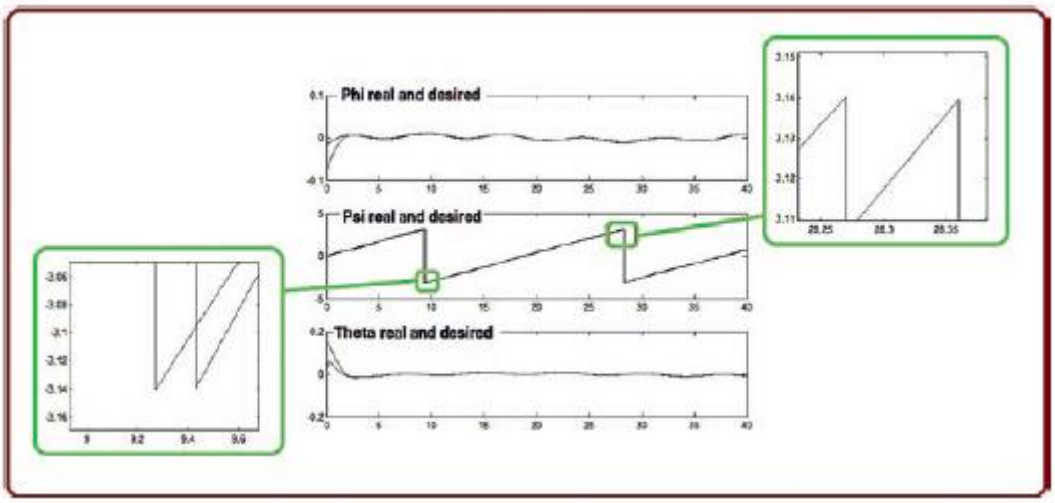
Czyli wykonując następujące operacje (jeśli wybierzesz inną macierz rotacji, musisz określić jej odpowiednią analizę):

$$\psi = \arctan 2(-R_{1,2}/R_{2,2})$$
$$\theta = \arctan 2(-R_{3,1}/R_{3,3})$$
$$\phi = \arctan 2(R_{3,2}/(\frac{R_{3,3}}{\cos\theta})) = \arctan 2(\sqrt{\frac{R_{3,2}^2}{R_{1,2}^2+R_{2,2}^2}})$$

W przypadku obrotów o szerokim zakresie, takich jak kąt odchylenia, należy użyć funkcji atan2, aby uniknąć nadmiarowości lub niespójności, ale w przypadku małych ruchów kątowych można zastosować coś bardziej bezpośredniego, jak w przypadku

$$\phi \approx \arcsin(R_{3,2})$$

W ten sposób poniższy kod, który wykorzystuje informacje uzyskane z bloków To Workspace (których należy użyć na końcu symulacji), służy jako przewodnik do określenia, co dzieje się z odpowiednimi kątami Eulera;



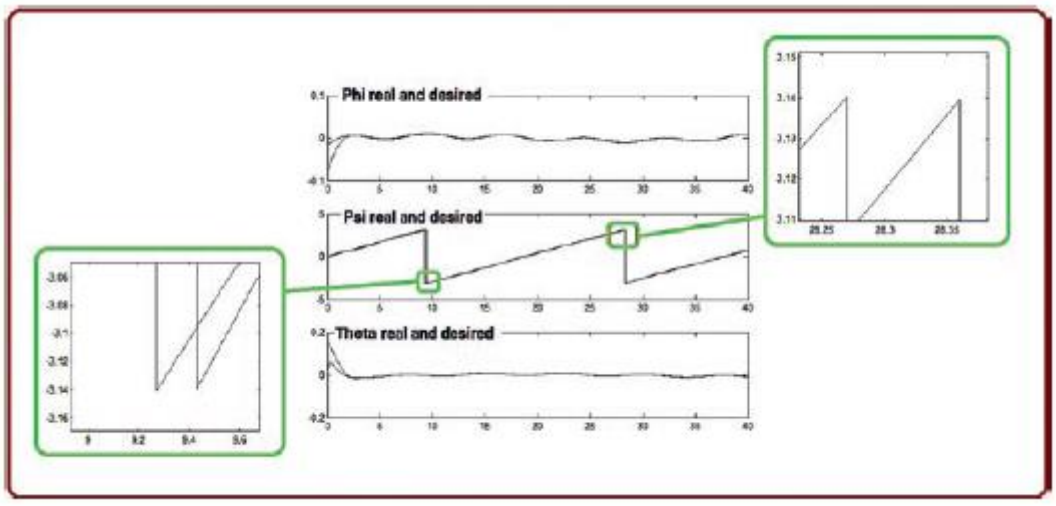
(można to wykonać bezpośrednio w oknie poleceń lub używając pliku m, jak pokazano na Listingu).

```
subplot(3,1,1)
% phi
plot(time,asin(Rde(:,8)))
hold on
plot(time,asin(Rreal(:,8)),'m')
subplot(3,1,2)
% psi
plot(time,atan2(-Rde(:,2),Rde(:,5)))
hold on
plot(time,atan2(-Rreal(:,2),Rreal(:,5)),'m')
subplot(3,1,3)
% teta
plot(time,atan2(-Rde(:,7),Rde(:,9)))
hold on
plot(time,atan2(-Rreal(:,7),Rreal(:,9)),'m')
```

Zauważ, że indeksy mają następujący format, którego używamy do pracy z blokiem To Workspace:

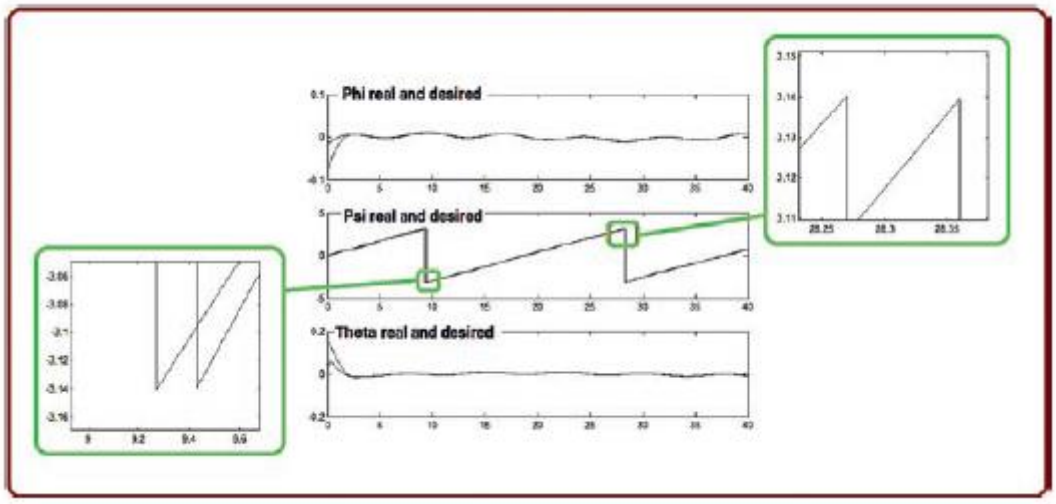
$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \rightarrow \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{21} & R_{22} & R_{23} & R_{31} & R_{32} & R_{33} \end{bmatrix}$$

Zauważ, że ten przykład jest poprawny, ponieważ kąty (z z wyjątkiem psi) miał małe wartości podczas symulacji.

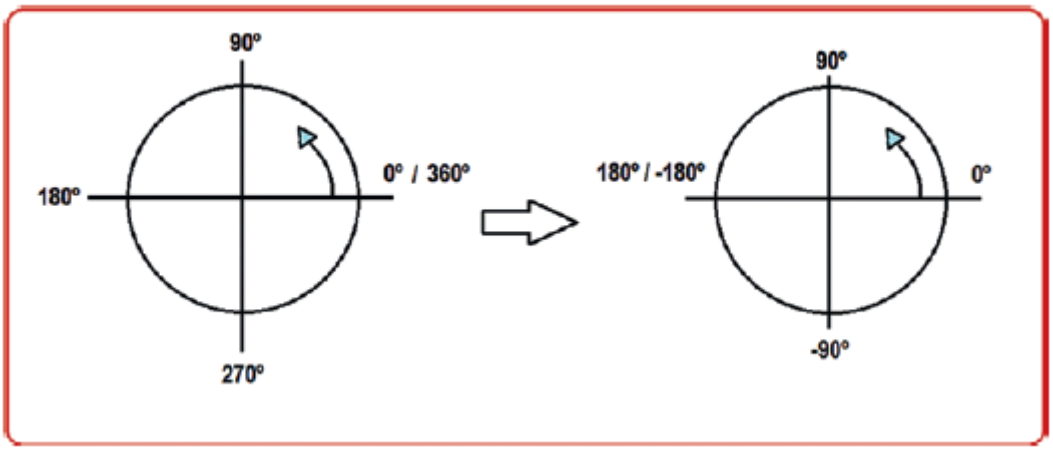


Zwróć uwagę na małe opóźnienie między żądanym sygnałem a sygnałem sterującym. Jest to oczywiste, ponieważ pożądany sygnał wymaga przetworzenia. Polega to na wygenerowaniu błędu, zdefiniowaniu

sterowania i wprowadzeniu tego sterownika do silników. W przypadku wolnych komputerów opóźnienie jest jeszcze bardziej znane. W tym przypadku moglibyśmy powiedzieć, że nie ma to znaczenia. Zgodnie z życzeniem, oprócz podążania po spiralnej trajektorii, dron wykonuje go podczas ciągłego obracania się wokół swojej osi Zb. Na rysunku



wygląda to jak ruch od -180 stopni lub -pi radianów do 180 stopni lub pi radianów, ale jest identyczny z obrotem o 0-360 stopni;



Interpretacja jest następująca. Wektor sterujący obrotem w osi nadwozia pojazdu jest zdefiniowany jako

$$B_{des} = \begin{bmatrix} \cos\psi_d & \sin\psi_d & 0 \end{bmatrix}$$

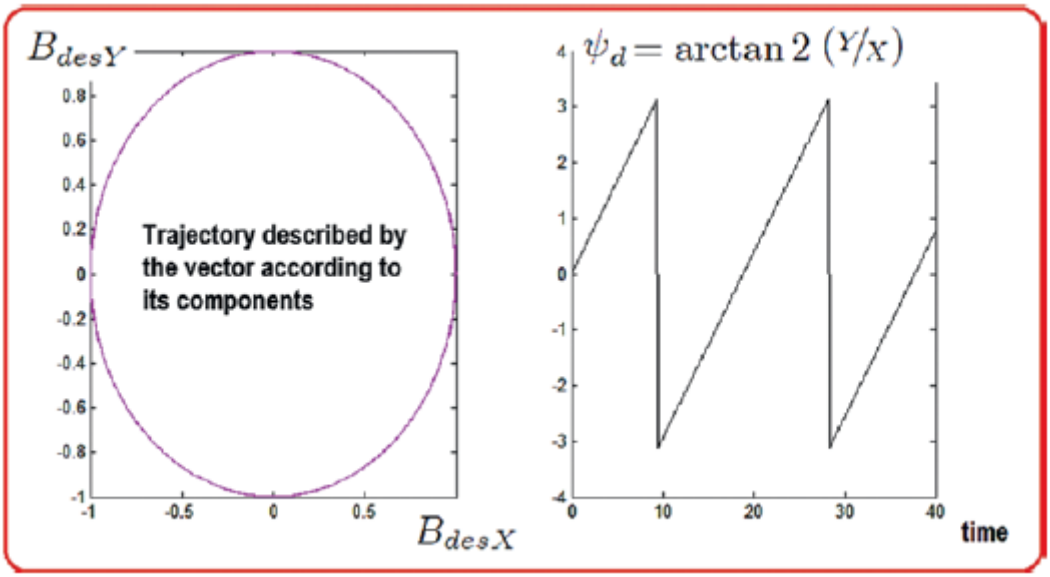
W tym przypadku pożądane Psi zależy od czasu:

$$\psi_d = \frac{t}{3}$$

$$B_{desX} = \cos(t/3)$$
$$B_{desY} = \sin(t/3)$$

$$\psi_d = \arctan 2(\frac{\sin(t/3)}{\cos(t/3)})$$

Można to przedstawić graficznie jak na rysunku



Rysunek został wykreślony za pomocą kodu dostępnego na listingu;

```
t=[0:0.1:40];

x=cos(t/3);

y=sin(t/3);

% trayectory around body axes

subplot(1,2,1)

plot(x,y,'m')

% angle between this axes (psi/yaw)

subplot(1,2,2)

plot(t,atan2(y,x),'k')
```

Zanim zamkniemy ten rozdział, porozmawiajmy o niektórych alternatywach dla symulacji, dostępnych również w środowisku Simulink i MATLAB.

Alternatywy symulacji ze środowiskiem Simulink i MATLAB

W tej sekcji przedstawiono alternatywne metody symulacji wykorzystania MATLAB/Simulink. W tym celu i aby procedur zrozumiałych i zwartych, zostanie zasymulowane rozwiązanie ogólne układu masowo-sprężynowego amortyzatora (przy wejściu siły równej zero i przy niezerowych warunkach

początkowych układ ten powinien powrócić do zera). Należy zauważyć, że jest to zachowanie w pętli zamkniętej regulatora PD z kompensacją przyspieszenia, jak widzieliśmy wcześniej, lub PID. Na podstawie tych przykładów możesz je dowolnie modyfikować, aby symulować quadkopter lub dowolny inny system lub samolot. Konkretny system, który zostanie przedstawiony w tej sekcji, jest następujący:

$$0.7\ddot{x} + 0.5\dot{x} + 0.5x = 0$$

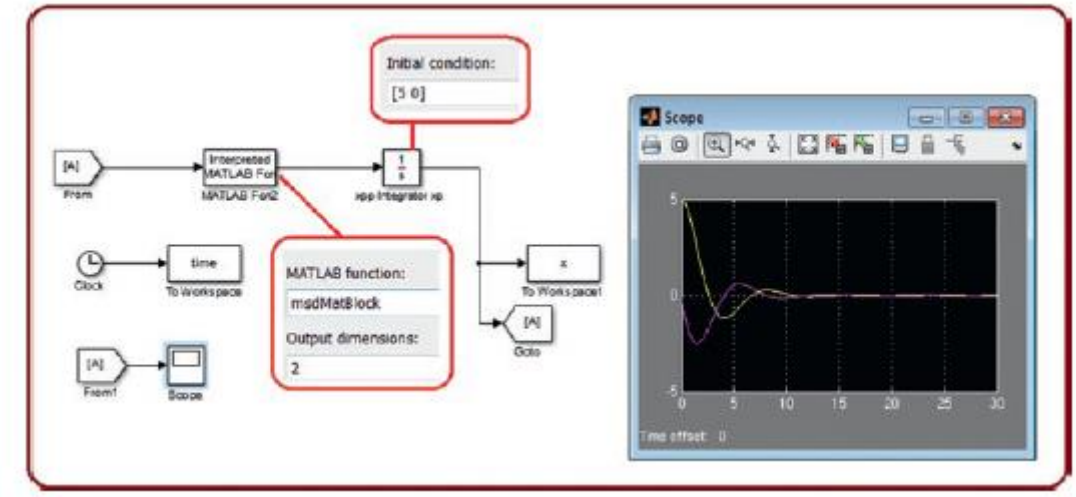
z warunkami początkowymi podanymi przez

$$x_1 = x = 5$$
$$x_2 = \dot{x} = 0$$

Istnieje więcej sposobów na symulację systemu za pomocą MATLAB i Simulink, ale to pięć, które uważamy za najbardziej przydatne.

Bloki i interpretowane funkcje MATLAB

Tak było już we wszystkich poprzednich symulacjach, ale tym razem ze wspomnianym wcześniej zredukowanym systemem, dzięki czemu można go porównać z innymi dostępnymi technikami. To jest ten, który polecamy ze względu na jego praktyczność i łatwość użytkowania. Ogólnie ta metoda jest najbardziej kompatybilna z innymi blokami Simulinka i funkcjami MATLAB w sposób intuicyjny i nie tak pracochłonny w porównaniu z innymi, które zostaną zaprezentowane później. Rysunek



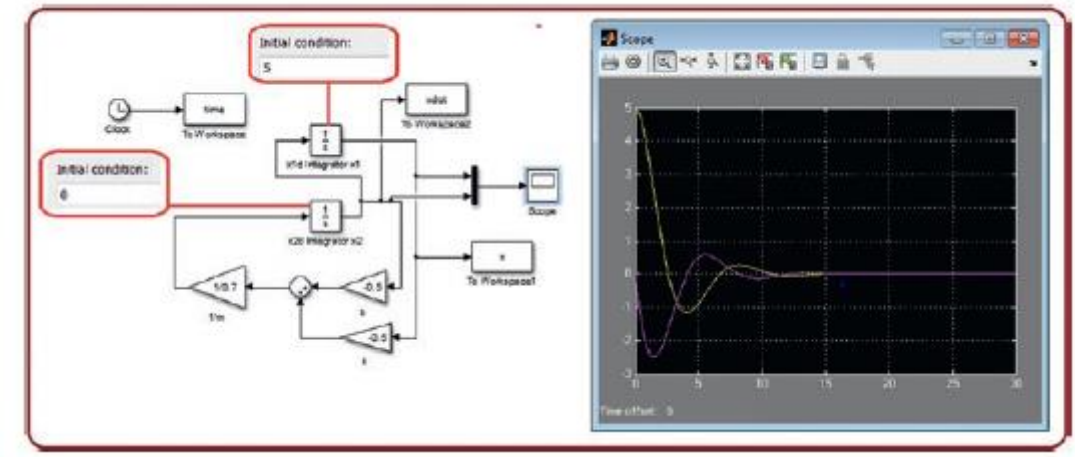
ilustruje plik bloku o nazwie msdmatBlockSim.slx. Zauważ, że dołączamy dwa bloki, które ułatwiają wzajemne połączenie pod względem rozstawu i estetyki (znaczniki Goto i From). Bez nich użytkownik po prostu nawiązałby bezpośrednie i splątane połączenia. Kod dynamiki i kontroli (w tym przypadku kontrolka wynosi zero) znajduje się w pliku o nazwie msdMatBlock.m, który jest wyświetlany na listingu

```
function outp=msdMatBlock(X)

% as you know you can modify the following dynamics with the

dynamics

% and control that you want, for example a quadcopter
```

```
x1=X(1);

x2=X(2);

B=0.5;

K=0.5;

M=0.7;

x1d=x2;

x2d=(1/M)*((-B*x2)-(K*x1));

outp=[x1d;x2d];

end
```

Tylko bloki

Rysunek



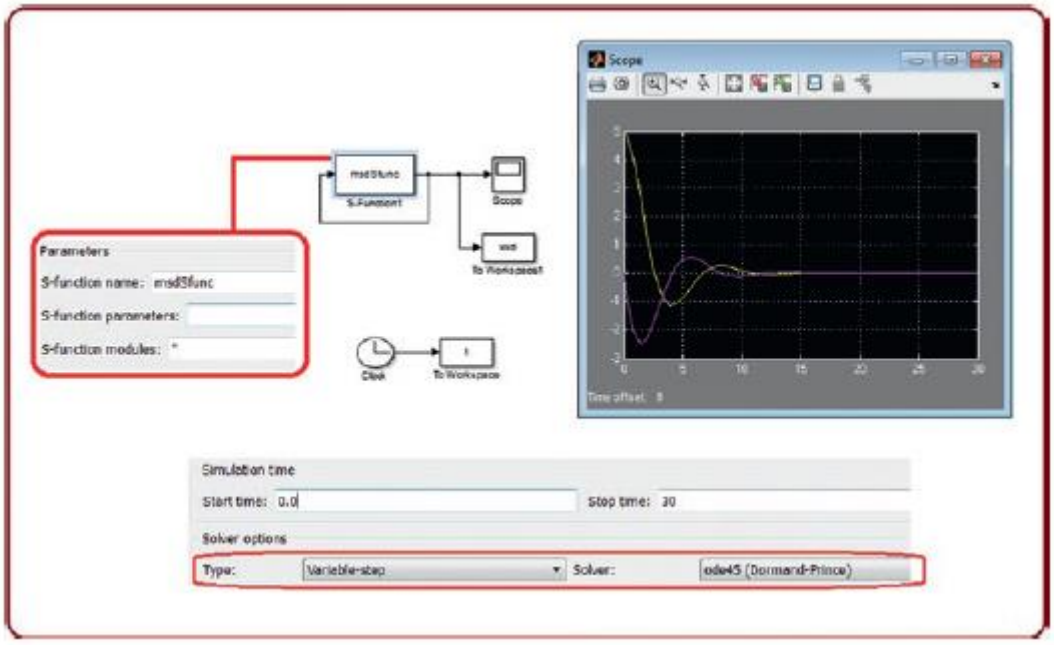
ilustruje również poprzedni przypadek, ale tylko przy użyciu zbioru bloków. W tym przykładzie nazywa się msdBlock.slx.

Ten typ symulacji jest nadal przydatny, ponieważ istnieją pewne urządzenia sprzętowe, które są połączone z Simulink iz jakiegoś powodu nie pozwalają na interakcję z funkcjami zdefiniowanymi przez użytkownika. Największą wadą tej techniki, jak już wskazano, jest to, że w większym systemie bloki również stają się liczne. Pamiętaj, że każde równanie wymaga bloków dla wszystkich symboli i operatorów (sumy, iloczyny, wykładniki, macierze itp.). Gdy masz więcej równań, wygodnie jest przeczytać o użyciu podsystemów, a zwłaszcza o tym, jak utworzyć podsystem z Wyboru.

Bloki i funkcje S

Funkcję S można uznać za ewolucję interpretowanej funkcji MATLAB, która zawiera dynamikę różniczkową w pliku tekstowym (nie wymaga użycia integratora, potrzebuje jedynie sprzężenia zwrotnego). Ma kilka zalet, takich jak możliwość używania ze sprzętem i oprogramowaniem zewnętrznym względem MATLAB i Simulink. Największy problem z tymi funkcjami związany jest z zarządzaniem czasem. W poniższym przykładzie symulacja jest użyteczna z solverami ODE ze zmiennym krokiem, ale nie z solverami ODE o stałym kroku (najbardziej wymagającym zadaniem

podczas pracy z funkcjami S jest to, że musisz uważać na zgodność czasów symulacji). Rysunek przedstawia plik bloku o nazwie msdSfuncBlock.slx.



Listing to odpowiedni plik tekstowy o nazwie msdSfunc.m. Jego opis jest zarezerwowany dla Ciebie do zbadania, ponieważ jak wskazano, nie są one tak intuicyjne i mają różne konfiguracje. Możesz rozpocząć swoje badania na ten temat od książki o sterowaniu w trybie przesuwnym autorstwa Jinkun Liu wskazanej w załączniku. Tutaj jest wiele przykładów wykorzystujących podejście S-Function.

```
function [sys,x0,str,ts]=msdSfunc(t,x,u,flag)

switch flag,

case 0,

[sys,x0,str,ts]=mdlInitializeSizes;

case 1,

sys=mdlDerivatives(t,x,u);

case 3,

sys=mdlOutputs(t,x,u);

case {2, 4, 9 }

sys = [];

otherwise

error(['Unhandled flag = ',num2str(flag)]);

end

function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
```

```
sizes.NumContStates = 2;

sizes.NumDiscStates = 0;

sizes.NumOutputs = 2;

sizes.NumInputs = 2;

sizes.DirFeedthrough = 0;

sizes.NumSampleTimes = 0;

sys=simsizes(sizes);

x0=[5 0];

str=[];

ts=[];

function sys=mdlDerivatives(t,x,u)

B=0.5;

K=0.5;

M=0.7;

sys(1)=x(2);

sys(2)=(1/M)*((-B*x(2))-(K*x(1)));

function sys=mdlOutputs(t,x,u)

sys(1)=x(1);

sys(2)=x(2);
```

Tryb tekstowy z predefiniowanymi poleceniami

W takim przypadku wymagany jest tylko plik tekstowy i plik pomocniczy do kreślenia (opcjonalnie) (plik pomocniczy można zastąpić wpisując bezpośrednio w oknie poleceń). Jak sama nazwa wskazuje, używane są predefiniowane funkcje (w poniższym przykładzie polecenie ode45). Wadą tej metody jest to, że jej interakcja z Simulinkiem jest niezwykle trudna, a każde dodatkowe przetwarzanie sygnału wiąże się z koniecznością napisania kodu pomocniczego (staje się to bardzo trudne, gdy wymagane jest sprzężenie zwrotne). Z drugiej strony zauważ, że aby przekonwertować ten kod na zinterpretowaną funkcję MATLAB, musisz dokonać kilku prostych korekt (plik dynamiki ma już strukturę podobną do zinterpretowanej funkcji MATLAB). Listing przedstawia kod pliku msdPredefinedFuncSys.m oraz sposób przekonwertowania tego pliku na zinterpretowany kod funkcji MATLAB.

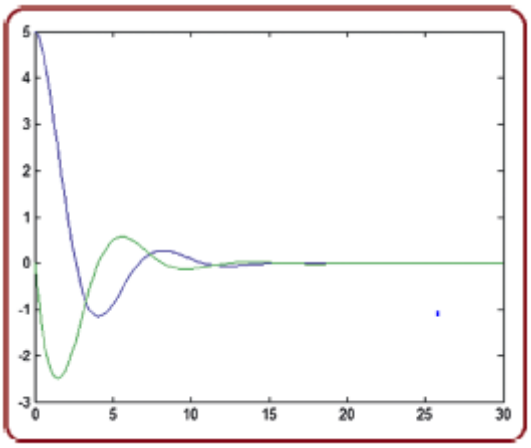
```
function out=msdPredefinedFuncSys(t,inpt)

% for converting this file into an Interpreted MATLAB® function

% just delete the letter t from the function arguments and add the

% word end after the output line

x1=inpt(1);
```

```
x2=inpt(2);

B=0.5;

K=0.5;

M=0.7;

x1d=x2;

x2d=(1/M)*((-B*x2)-(K*x1));

out=[x1d;x2d];
```

Korzystanie z tego pliku jest opisane w wierszach Listingu (zwróć uwagę na część, w której ustawiony jest czas i warunki początkowe)

```
[t,X]=ode45(@msdPredefinedFuncSys,[0 30],[5;0]);

plot(t,X)
```

Poprzednie polecenia można wprowadzić bezpośrednio do okna poleceń lub za pomocą pliku pomocniczego. W przypadku tej książki jest to plik msdPredefinedFuncPlot.m. Wynik tego rodzaju symulacji przedstawiono na rysunku



Tryb tekstowy z niestandardowymi poleceniami

Jest to identyczne jak w poprzednim przypadku, ale użytkownik może generować własne solwery ODE (funkcje nie zaprojektowane wcześniej). Oznacza to bezpośredni sposób migracji do innych języków programowania. Pamiętaj tylko, że zmieni się składnia, ale nie algorytmy czy równania. Listing przedstawia kod systemowy (msdCustomSys.m). Pamiętaj, że w tym przypadku kontroler to zero. W tym momencie powinieneś już zauważyć wzorzec symulacji (plik z dynamiką systemu, inny plik z solverem ODE i linią, segmentem ody lub interfejsem wykonawczym do uruchomienia symulacji).

```
function h=msdCustomSys(t,X)

x1=X(1);

x2=X(2);

B=0.5;

K=0.5;
```

```
M=0.7;

x1p=x2;

x2p=(-(B*x2)-(K*x1))/M;

h=[x1p;x2p];
```

Linie wykonania są takie, jak wskazano na listingu i są zawarte w pliku msdCustomPlot.m (co jest opcjonalne, ponieważ można je wykonać z okna poleceń).

```
X0=[5;0];

t = linspace(0,30,5000);

X=ode3('msdCustomSys',t,X0);

plot(t,X)
```

Wykres jest praktycznie identyczny z rysunkiem powyżej

Zwróć uwagę, że kod udostępnia w obszerny sposób własną metodę numeryczną.

Streszczenie

Poznałeś różne metody symulacji zachowania i sterowania dronami za pomocą narzędzi MATLAB i Simulink. Jest on jednak przedstawiony w taki sposób, abyś mógł wykorzystać zdobytą wiedzę za pomocą innych narzędzi i języków programowania. Konkluzja tego rozdziału jest taka, że aby symulować samolot i ogólnie dowolny system, będziesz potrzebować co najmniej trzech programów: modelu dynamicznego z kodem sterującym, rozwiązania numerycznego dla twojego układu równań oraz środowiska wykonawczego. Jeśli chcesz dołączyć grafikę lub interakcję z użytkownikiem, wspomniano o niektórych opcjach, ale ostatecznie i w każdym razie będziesz musiał znać podstawowe pojęcia trzech wyżej wymienionych programów.

Realizacja

Ta część ma wpływ naukowy i twórczy, a dowiesz się o tematach, które są również pomijane w wielu publikacjach, związanych z podstawowymi szczegółami implementacji, takimi jak przetwarzanie sygnałów dla czujników i siłowników, transmisja danych, sposoby programowania drona oraz słowniczek z główne polecenia używane z najpopularniejszymi interfejsami programistycznymi. Dzięki temu będziesz w stanie zwięźle zrozumieć, jak zaprogramować drona lub inny pojazd zrobotyzowany.

Zadania drona

Istnieje sześć podstawowych zadań drona:

• Silnik: To jest najbardziej podstawowe i najważniejsze zadanie. Ogólnie składa się z automatycznego sterowania w pętli otwartej, które jest funkcją innych sterowników w pętli zamkniętej pojazdu. Prędkość każdego silnika jest modyfikowana, co ma wpływ na zachowanie drona. To zadanie jest tym, które przedstawia najszybszą zmienność i w konsekwencji wymaga aktualizacji z wysoką częstotliwością. Silniki muszą otrzymywać macierz napędu lub alokacji w sposób automatyczny.

• Postawa lub orientacja: Jest to drugie najważniejsze zadanie (również pod względem częstotliwości wykonywania). Jest to ważniejsze niż zadanie wysokościowe, ponieważ oznacza, że samolot jest odpowiednio wyważony. Gdyby tak nie było, dron mógłby losowo poruszać się po samolocie i w najgorszym wypadku spaść. Ze względu na znaczenie w locie pojazdu jest to zwykle w pełni automatyczna pętla zamknięta.

• Wysokość lub wzniesienie: jest to trzecia istotna kwestia, ponieważ każdy samolot oznacza zdolność do unoszenia się na wodzie. Z tego powodu osoby z wiedzą wstępną uważają, że to zadanie jest najważniejsze, ale jak już wspomnieliśmy, brak równowagi ma tendencję do destabilizacji drona. Zwykle jest to automatyczna pętla zamknięta, ale możesz nadać jej zachowanie półautomatyczne za pomocą pilota.

• Planarny: jest to czwarty pod względem znaczenia. Polega ona na kontrolowaniu pozycji drona, która jest prostopadła do osi wzniesienia (pozycja, którą dron utrzymuje przy ziemi). To zadanie może być również automatyczne lub półautomatyczne.

• Planowanie trajektorii: w tym przypadku jest to jednoczesna zmiana położenia, wysokości i zadań planarnych. Ta zmiana odbywa się w zależności od czasu lub określonej ścieżki od punktu do punktu. Może być automatyczny lub półautomatyczny. Zasadniczo jest to nawigacja po zbiorze punktów.

• Zdalne sterowanie: Jest to ręczny element zadań lotu. Jest to kontrola człowieka w zamkniętej pętli, ponieważ to pilot reguluje błąd zgodnie z pomiarem zapewnianym przez jego wzrok i działanie przycisków sterujących. Ponieważ akcja jest zależna od człowieka, zadanie to jest tak powolne i nieprecyzyjne, jak możliwości pilota. Jego użycie z innymi zadaniami daje im tryb półautomatyczny. W niektórych przypadkach pilot zmienia tylko żądane wartości, aw innych modyfikuje bezpośrednio efekt sterowania. Nie zaleca się, aby pilot bezpośrednio modyfikował zadania orientacyjne (przyciski i drążki poruszają się wolniej i gwałtowniej niż wymagane automatyczne ustawianie pozycji).

Teraz, gdy już mówisz o zadaniach drona, porozmawiajmy o rodzaju kontrolerów, których używają.

Pętle i rodzaje kontrolerów dla drona

Quadkopter to system, który służy jako dobry przykład do opisu różnych typów kontrolerów i pętli sterowania. Zostały one omówione w kolejnych sekcjach.

Pętle kontrolne

Otwarte i zamknięte pętle

Ogólnie wysokość, położenie planarne i orientacja drona są zamkniętymi pętlami, co oznacza, że ich korekcja błędów opiera się na sprzężeniu zwrotnym. Jest jednak zadanie, które zależy od otwartej pętli. Jest to kontrola prędkości silników. Otwarta pętla to taka, w której zadanie jest wykonywane bardziej w wyniku proporcjonalności niż w wyniku korekcji błędów. W przypadku silników wysyłane jest napięcie i obserwujemy proporcjonalną prędkość silnika bez zamkniętej pętli, aby zweryfikować i skorygować tę prędkość.

Pętle wewnętrzne i zewnętrzne

Quadkopter ma cztery silniki, a dzięki swojej geometrycznej konfiguracji może bezpośrednio sterować czterema zmiennymi, którymi są wysokość i położenie (trzy orientacje). Nazywa się to pętlą wewnętrzną. Ponieważ jednak wymagane jest, aby pojazd mógł poruszać się również w płaszczyźnie XY, odbywa się to w sposób zależny od orientacji przechyłu i pochylenia w zewnętrznej pętli sterowania.

Pętle zależne i niezależne

Niezależne pętle występują we wszystkich zmiennych, których sterowanie nie wymaga monitorowania dodatkowej zmiennej; przykładami są kontrola wysokości i kontrola odchylenia. Z drugiej strony pętle zależne wymagają informacji o jednej lub więcej dodatkowych zmiennych; przykładami są elementy sterujące X i Y, które zależą od orientacji pojazdu.

Pętle pozycji i prędkości

Quadkopter to pojazd, którego pętle pozycji (orientacja i pozycja) zależą od otwartej pętli prędkości każdego z jego silników. Serwomotory są siłownikami pozycji, a bezszczotkowe silniki prądu stałego są zwykle siłownikami prędkości.

Rodzaje kontrolerów

Kontrolery wolne od modelu i inne niż wolne od modelu

Niektóre zmienne, takie jak przechylenie, pochylenie i orientacja odchylenia, można regulować za pomocą „prostego” kontrolera PD. Oznacza to, że nie jest wymagana kompensacja ich częściowej lub pełnej dynamiki. Z drugiej strony przy kontroli wysokości konieczne jest wprowadzenie elementu kompensacji masy pojazdu (częściowy sterownik bezmodelowy).

Solidne i adaptacyjne kontrolery

W przypadku zmiennych regulowanych przy użyciu PD bez jakiejkolwiek kompensacji zakłada się, że PD są wystarczająco duże w odniesieniu do dynamiki pojazdu. W tym przypadku sterowanie nazywa się dominującym lub solidnym (o ile silniki pozwalają na tę odporność). Innym przykładem solidnych sterowników jest rodzina trybów przesuwnych. W zasadzie nie wymagają modelu systemu, tylko pętli błędów. W przypadku wysokości, gdy waga jest kompensowana, jest to bardzo szczątkowy przykład kontroli adaptacyjnej. Jeśli jednak wzmocnienia PD są wystarczająco duże, a silniki mogą same przeciwdziałać temu efektowi, człon kompensacyjny można pominąć. Ogólnie rzecz biorąc, kontrolery adaptacyjne są zaprojektowane do kompensacji częściowej lub pełnej dynamiki lub niemodelowanych perturbacji.

Nieograniczona i ograniczona kontrola

Nieograniczona kontrola to taka, która może używać nieograniczonego zakresu wartości, na przykład wszystkich wartości dodatnich i ujemnych. Ten rodzaj kontroli nie istnieje, ale możesz pomyśleć, że jest to opłacalna opcja ze względu na symulatory. Z drugiej strony, ograniczone kontrolery wprowadzają ograniczenia operacyjne. Pierwsze ograniczone sterowanie znalezione w quadkopterze jest podawane przez silniki, ponieważ mają one maksymalne i minimalne wartości prędkości i momentu obrotowego. Druga ograniczona kontrola jest bardzo zauważalna przy użyciu silników bezszczotkowych (zakładając, że nie są one dwukierunkowe). Jest to obecność wyłącznie pozytywnych wartości operacyjnych. A ostatnia ograniczona kontrola, nie tak widoczna w dronach, jest determinowana przez efekty fizyczne, takie jak czas reakcji. W pierwszym ograniczeniu stosuje się nasycenia. W drugim wymagane są współczynniki skalowania. W trzecim wykorzystywane są techniki kontroli opóźnień. Ten ostatni efekt nie jest tak widoczny na zwykłych dronach ze względu na szybką reakcję autopilotów i komputerów i zwykle jest pomijany, ale w samolotach, które używają śmigieł dwukierunkowych, należy wziąć pod uwagę te opóźnienia, ponieważ śmigła nie zmieniają kierunku obrotu momentalnie.

Sterowanie liniowe i nieliniowe

Ten rodzaj kontroli zależy od warunków pracy. Widzieliście kilka trybów lotu, w których ruch jest płynny, a kąty są ograniczone do małego obszaru wokół zera, więc słuszną decyzją było zlinearyzowanie zarówno modelu, jak i kontrolera. Opisaliśmy jednak również niektóre tryby lotu, w których kąty nie są ograniczone, a pojazd może wykonywać agresywne ruchy; dlatego sterowanie i model mają nieliniowości.

Praca ciągła i oparta na zdarzeniach

W tym przypadku kontrolerem, który jest zawsze stały i który musi być obecny podczas pełnego działania statku powietrznego, jest kontroler położenia. Tym, który zależy od zdarzeń, takich jak sygnały czasowe lub interakcja z pilotem, jest sterowanie trajektorią.

Sterowniki ciągłe i dyskretne

Tu pojawia się druga definicja ciągłości, która jest związana z tym, jak mały jest czas próbkowania. Zależy to od dwóch czynników: pierwszy to komputer, a drugi siłowniki i czujniki. W przypadku komputerów, dziś prawie wszystkie są na tyle szybkie, aby traktować ich działanie jako modele ciągłe lub układy oparte na równaniach różniczkowych. Jednak zmieniając technologię siłowników od elektrycznych silników bezszczotkowych do silników spalinowych, zauważysz, że ten drugi rodzaj silników ma wolniejsze czasy reakcji w porównaniu z silnikami elektrycznymi (mają dużo elementów, tarcie i cykle pracy) i wymagają do analizy równań różnicowych (ich modelowanie za pomocą równań różniczkowych nie jest już możliwe).

Sterowanie elektryczne i mechaniczne

Nadużywamy niektórych technologii, ponieważ podzespoły elektryczne i elektroniczne są gwarantowane przez producenta do prawidłowego działania, więc na przykład nie martwimy się o działanie ESC. Polegamy również na tym, że urządzenia elektryczne są szybsze w stosunku do elementów mechanicznych. Z tych powodów nasz projekt ignoruje komponenty elektryczne i elektroniczne i traktujemy drona jako system z czysto mechanicznym modelowaniem i sterowaniem. Jednak nowoczesne projekty, które wprowadzają sterowanie w pętli zamkniętej prędkości silników, mogą być postrzegane jako sterowanie pośrednie lub elektromechaniczne, gdzie prędkość mechaniczna silników zależy od zmiennych elektrycznych, takich jak fazy elektryczne ESC. Patrz Załącznik, aby uzyskać informacje na temat sterowania silnikiem w pętli zamkniętej. Poznałeś już

zadania drona i większość kontrolerów, z których może korzystać ten pojazd/robot. Kontynuujmy podstawowe przetwarzanie sygnału, którego wymaga dron i przykład zastosowania.

Przetwarzanie sygnału dronów

Może kusi Cię wdrożenie poprzednich kontrolerów. Jednak przed tą implementacją wymagane są pewne kroki. Najczęstsze to:

• Filtrowanie sygnału, które jest stosowane w czujnikach

• Mapowanie sygnału, które jest używane w czujnikach, drążkach zdalnego sterowania, podczas transmisji danych itp.

• Odlewanie sygnału lub konwersja danych, która jest wykorzystywana do wprowadzania wartości sterującej do silników lub podczas szeregowej transmisji danych

• Nasycenie sygnału, które służy do ograniczania wartości wysyłanych do silników lub do ograniczania wartości odczytywanych z drążka zdalnego sterowania

• Normalizacja sygnału, która służy do narzucenia zakresu pracy w czujnikach

Filtrowanie sygnału

Jak widać w poprzednich rozdziałach, głównym składnikiem regulatorów z zamkniętą pętlą jest błąd, który również zależy od mierzonych zmiennych i jest to osiągane za pomocą czujników. Jednak te czujniki zwykle mają szum elektryczny z powodu własnego zachowania lub z powodu ich interakcji z otoczeniem. Dlatego wymagany jest etap filtrowania w celu zmniejszenia lub stłumienia szumu elektrycznego. Ponieważ implikuje się przetwarzanie obliczeniowe lub elektroniczne, tłumienie szumu jest zgodnością między wygładzeniem sygnału szumu a opóźnieniem wygładzonego sygnału. Można to osiągnąć na kilka sposobów. Oto kilka przykładów:

• Pasywny: Tutaj stosowane są dodatkowe urządzenia, które z natury pochłaniają część hałasu otoczenia. Przykładami są gąbki rozpraszające lub płyny do ruchu mechanicznego, soczewki pochłaniające częstotliwość światła lub pasywne urządzenia elektroniczne, które mogą być używane jako filtry.

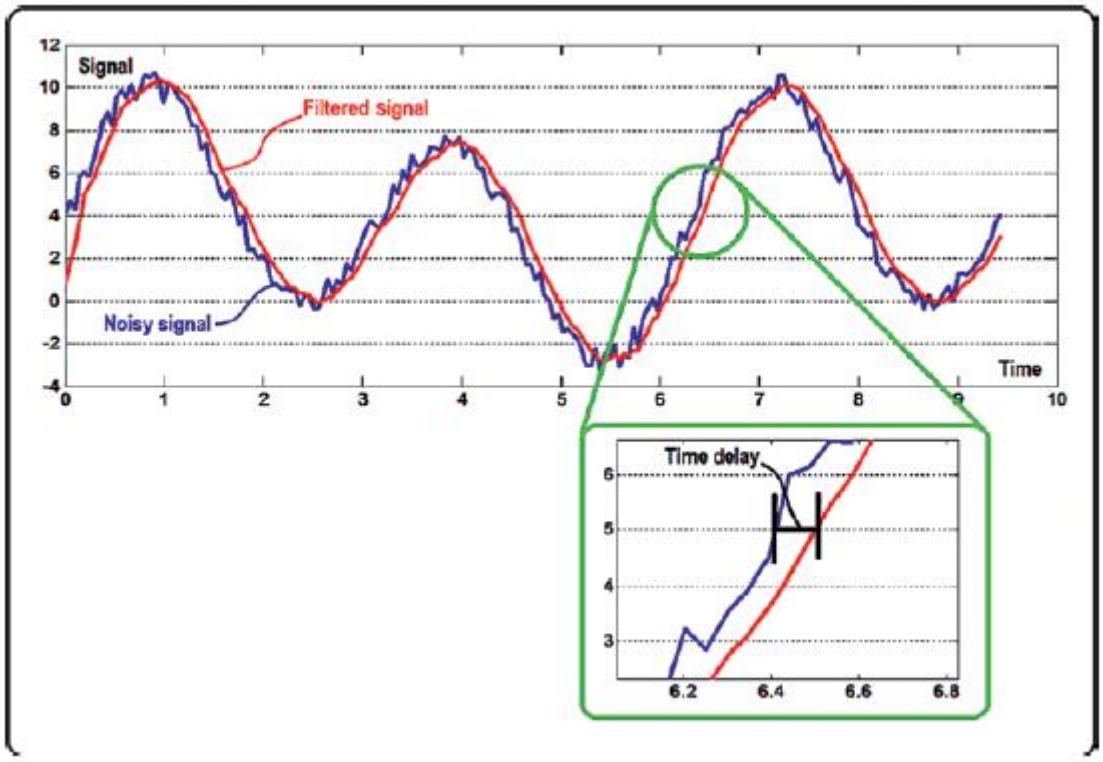
• Aktywne przez filtrowanie elektroniczne: dobrze znane przykłady to obwody dolnoprzepustowe, górnoprzepustowe, pasmowoprzepustowe i pasmowo-odrzucające.

• Aktywny przez filtrowanie matematyczne: sygnał jest wysyłany do filtru matematycznego. Wśród najbardziej znanych filtrów znajdują się filtr Kalmana i obserwator Luenbenger.

• Aktywny przez transformację przestrzeni: sygnał jest konwertowany do innej przestrzeni w celu wykonania prostego filtrowania w nowej przestrzeni, takiego jak filtrowanie na poziomie częstotliwości zamiast na poziomie czasu. Ta metoda może być również uważana za filtr matematyczny, ale ma swój własny kierunek badań. Przykładami są transformacje Fouriera i Wavelets.

• Aktywny dzięki inteligentnemu filtrowaniu: odbywa się to za pomocą dowolnego algorytmu wykorzystującego sztuczną inteligencję

Kompromis między redukcją szumów a opóźnieniem sygnału można zobaczyć na rysunku (użyto kodu ogólnego filtra dolnoprzepustowego).



Na listingu można zobaczyć ogólny kod wysokiego poziomu do używania tego typu filtrów (w tym przypadku MATLAB).

filteredSignal=lowpassFilter(noisySignal)

Nasycenie

W takim przypadku konieczne jest nałożenie wartości granicznych na sterowniki, aby maksymalne i minimalne wartości, które mają być wstrzykiwane, nie przekraczały granic silnika. Innym zastosowaniem jest ograniczenie wartości, które użytkownik może osiągnąć w drążku pilota. Nasycenie to można wykonać skokowo (na przykład za pomocą funkcji znaku, która jest równoważna czynności włączania i wyłączania) lub w sposób ciągły (za pomocą funkcji zwanej nasyceniem). Nawet funkcja ciągła może być gładka lub ostra (można użyć wielu funkcji z rodziny saturatorów zwanych funkcjami sigmoidalnymi). Zazwyczaj nieograniczona część saturatora kopiuje oryginalny sygnał jako standardową funkcję liniową, jak pokazano w kodzie z Listingu .

Mini=0

Maxi=7

if(signal<=Mini)

satsignal=Mini;

elseif(signal>=Maxi)

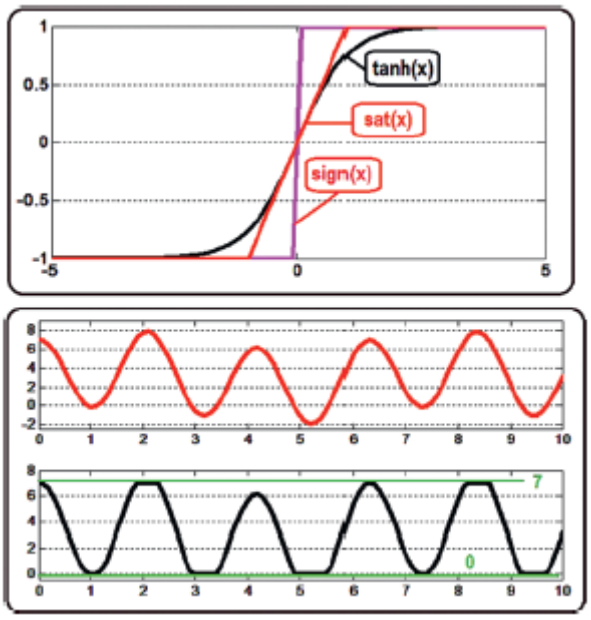
satsignal=Maxi;

else

satsignal=signal;

end

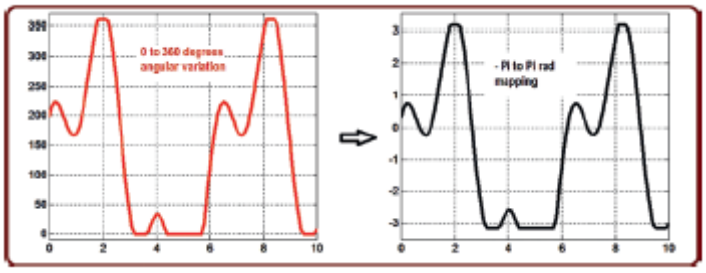
Jednak, aby przeciwdziałać niepożądanemu działaniu (na przykład nagłemu zachowaniu drążka zdalnego sterowania), czasami stosuje się inne typy sigmoidów. Rysunek



ilustruje dwie rzeczy. Górna część to rodzina saturatorów, w tym przypadku funkcja znaku, funkcja saturacji i funkcja tangensa hiperbolicznego. Oczywiście projektant musi je tak zmodyfikować, aby osiągnęły wartości minimalne i maksymalne zgodnie ze swoimi zastosowaniami, a nie tylko -1 i 1. Dolna część to zastosowanie funkcji nasycenia przy danym sygnale, w tym przypadku dopuszczalne wartości minimalne i maksymalne przejdź od 0 do 7. Kod użyty na rysunku jest naprawdę prosty i jest wyświetlany na listingu .

Stronniczość i mapowanie

Kolejnym ważnym zadaniem jest przekształcenie sygnału sterującego (sygnał o wartościach dodatnich i ujemnych) w jeden sygnał kompatybilny z silnikami (np. w zakresie wartości 1000-2000 w bezszczotkowych silnikach RC). Innym przykładem jest zasięg czujników, który może być dowolny i musi zostać przekonwertowany na standardowy zakres roboczy (0 do 360 stopni, -180 do 180 stopni, -1 do 1 metra itd.). Zobacz rysunek .



W tym przypadku wykorzystywane są funkcje skalowania i tłumaczenia. Jedną z najczęściej używanych technik odchylania i mapowania jest równanie linii między dwoma punktami znane jako mapowanie liniowe, które dostosowuje wartość wejściową do równania linii prostej (wartość wyjściowa). W tym celu sygnał wejściowy musi być wcześniej ograniczony w taki sposób, aby nie przekraczał

maksymalnych i minimalnych limitów wejściowych. Kod jest również prosty i jest pokazany na listingu (zauważ, że ten kod mapowania opisuje linię między dwoma punktami).

```
mininput=0;

minoutput=-pi;

maxinput=360;

maxoutput=pi;

mapsignal=(signal - mininput) * (maxoutput - minoutput) /

(maxinput - mininput) + minoutput;
```

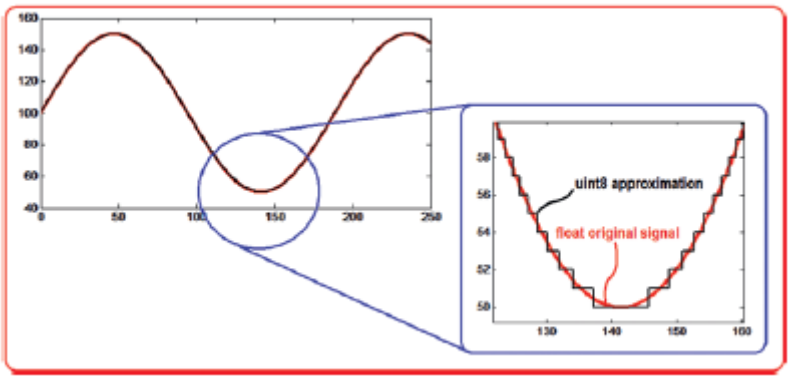
Przesyłanie danych

Jest to kolejna ważna adaptacja sygnału, którą należy przeprowadzić, gdy polecenie dopuszcza tylko jeden typ danych i otrzymuje dane innego typu. Na przykład polecenia zapisu do silnika RC typu PWM zwykle wymagają danych typu integer, a podane wartości są liczbami zmiennoprzecinkowymi lub ułamkami. W tym przypadku używamy tego, co w programowaniu nazywa się rzutowaniem, co jest transformacją z jednego typu danych na inny. Może to obejmować dwie rzeczy:

• Casting nie istnieje i użytkownik musi zaprojektować ten program.

• Odlewanie generuje aliasing, który wpływa na jakość pomiarów lub wykonania (np. sterownik może przejść z płynnego i naturalnego ruchu do gwałtownego i zrobotyzowanego lub z obrazu o wysokiej rozdzielczości do obrazu o niskiej jakości).

Rysunek przedstawia typowy przykład.



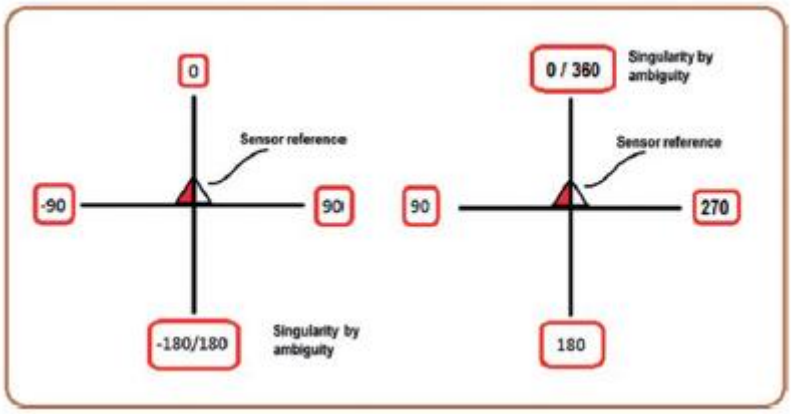
Sygnał czerwony reprezentuje wyliczoną wartość sterującą (z dużą liczbą miejsc po przecinku) i w momencie podania tego sygnału do silników, ponieważ polecenie zapisu dopuszcza inny typ danych (np. uint8), dokładność wartości pierwotnej jest zmniejszona. Z wyglądu wyglądają podobnie, ale jeśli przyjrzymy się szczegółowo, zaprezentowany zostanie efekt schodkowy zwany aliasingiem. Ten efekt może sugerować mniej płynne zachowanie w ruchu drona. W zależności od języka programowania polecenia rzutowania mogą, ale nie muszą istnieć (musi je utworzyć użytkownik), ale generalnie przyjmują postać wyświetlaną na listingu

```
signal=100+50*sin(t/30);

castSignal = uint8(signal);
```

Zauważ, że jeśli rzutowanie istnieje, oznacza to, że istnieją funkcje, które wymuszają zmianę typu danych sygnału lub danych wejściowych (w tym przykładzie uint8).

Normalizacja redundancji i osobliwości

Jest to częsty problem spotykany w czujnikach kątowych i polega on na tym, że mają one przeskok od wartości maksymalnej do wartości minimalnej np. z 359 stopni na 0 stopni lub nadmiarowe wartości znane jako osobliwości takie jak 0 stopni i 360 stopni (tak dzieje się również przy -180 stopniach i 180 stopniach, jeśli czujnik działa w ten sposób).



Aby poradzić sobie z takimi warunkami osobliwości, niejednoznaczności lub nadmiarowości, istnieje wiele metod normalizacji. Jeden z nich (który dotyczy przypadku z lewej strony rysunku) opiera się na symetrycznym określeniu błędu kątowego względem wartości mierzonej.

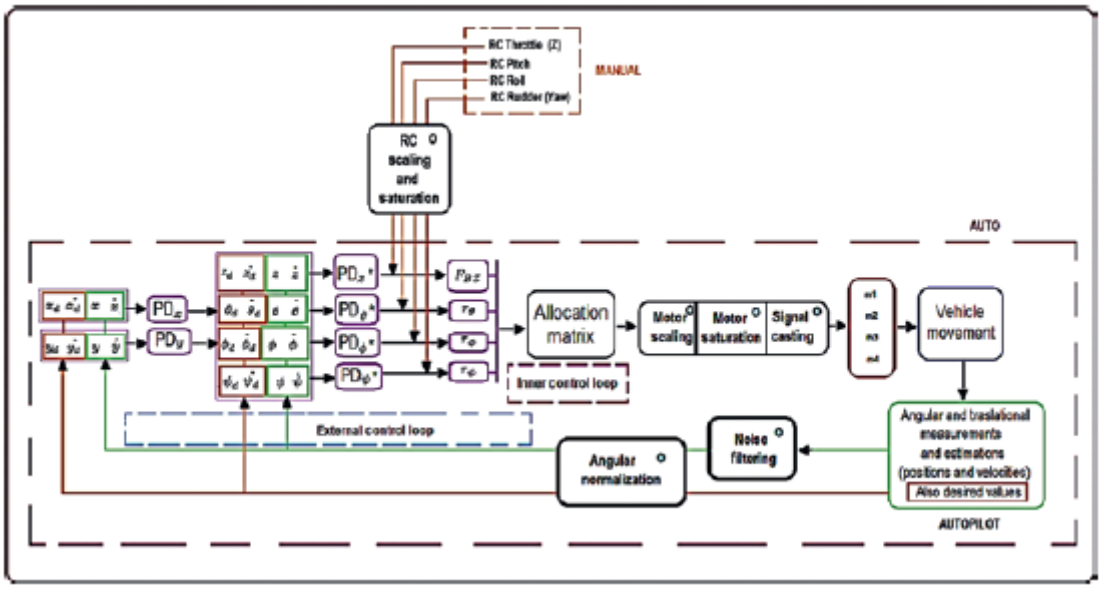
$$e_\psi = \psi_d - \psi$$

$$e_{\psi\,\mathrm{Norm}} = \begin{cases} e_\psi - 360 \text{ if } e_\psi > 180 \\ e_\psi + 360 \text{ if } e_\psi < -180 \\ e_\psi \text{ in another case} \end{cases}$$

Jest to normalizacja dynamiczna, której dodatkowym celem jest zmniejszenie obrotu pojazdu w sposób implikujący krótszą odległość. Aby osiągnąć pożądaną orientację, pojazd może to zrobić, obracając się w lewo lub w prawo. Ten algorytm, oprócz zapobiegania osobliwościom, wybiera, czy lepiej poruszać się zgodnie z ruchem wskazówek zegara, czy przeciwnie do ruchu wskazówek zegara. Należy również opracować lub zbadać alternatywne metody normalizacji, które są oparte na wartości pożądanej, a nie zmierzonej, lub oparte na bezwzględnym odniesieniu zamiast odniesienia pojazdu, lub metody oparte na wielu obrotach i metodach pomiarów kątowych od 0 do 360 stopni itp.

Przykład użycia

Rysunek



przedstawia modyfikację sterownika półautomatycznego bez ruchu odchylającego (lub bardzo małego), jak opisano wcześniej. Ta odmiana obejmuje w ten sposób pięć podstawowych metod przetwarzania sygnału: mapowanie i nasycenie pilota zdalnego sterowania jest stosowane w celu dostosowania jego wartości w celu uzyskania akceptowalnych poleceń kątowych. Filtrowanie jest stosowane do czujników pozycji w celu zmniejszenia ich poziomu hałasu. Błąd kątowy jest znormalizowany. Błąd kątowy jest znormalizowany. Przed wstrzyknięciem macierzy alokacji do silników stosuje się skalowanie, aby ich sygnały były dodatnie. Ogólnie rzecz biorąc, silniki dronów mają stały kierunek obrotu, a ich prędkość może być zmieniana od zera lub wartości początkowej do maksymalnej prędkości, dlatego w sygnałach silnika stosuje się nasycenie, aby nie przekroczyć ich wartości początkowej i maksymalnej. Wreszcie, rzutowanie jest wykonywane, ponieważ generalnie polecenia zapisu do silników żądają typu liczby całkowitej, a sygnały sterujące są wartościami zmiennoprzecinkowymi. Poznałeś podstawy przetwarzania sygnałów dla dronów. Kontynuujmy podstawy teorii transmisji danych, która jest wykorzystywana w większości środowisk programowania dronów, które zwykle bazują na transmisji szeregowej typu UART.

Teoria transmisji danych

Teoria transmisji danych jest uważana za temat o dużym znaczeniu, ponieważ umożliwia komunikację między zespołem dronów lub między pojedynczym dronem a odległą bazą. Wśród dostępnych prezentacji opisujemy tutaj typ UART (Universal Asynchronous Receiver-Transmitter), ponieważ jest to jeden z najbardziej podstawowych i wykorzystywanych protokołów z dziesięcioleciami użytkowania i jest naprawdę powszechny w SDK dronów.

Typy i podtypy danych

Zanim zaczniemy mówić o tym standardzie, wygodnie jest porozmawiać o typach i podtypach danych. Ten temat jest związany z wprowadzoną wcześniej konwersją danych. W poniższych akapitach będziemy używać pojęć C/C++, ponieważ jest to najpopularniejszy język programowania występujący w różnych pakietach SDK do tworzenia aplikacji dronów. Jest to jednak bardzo częsty temat wśród innych języków programowania i każdy z nich ma następujące podstawowe typy (lub bardzo podobne).

Typ: opis

int : Używany do liczb całkowitych

float : Używany do liczb z wartościami dziesiętnymi

double : zwiększa dokładność pływaków i jest przydatny do obliczeń naukowych lub inżynierskich

char : Pozwala na użycie znaków

bool : Pozwala na użycie wartości logicznych

Jednak w zależności od przeprowadzanych operacji, takich jak zarządzanie czasem, zarządzanie pamięcią i protokoły komunikacyjne, często można znaleźć warianty tych podstawowych typów danych. Warianty te są identyfikowane przez włączenie przedrostków i przyrostków. W programowaniu dronów bardzo często stosuje się warianty typu integer. Na przykład uint8 ma przedrostek u i przyrostek 8. Ogólnie rzecz biorąc, przedrostek wskazuje ograniczenie znaku, a przyrostek ograniczenie zakresu na podstawie bitów. Istnieje 8, 16, 32 i 64-bitowe liczby całkowite, które mogą przechowywać do 255 (255 = 2 podniesione do ósmej potęgi minus 1, aby zacząć od elementu 0), odpowiednio 65535, 4294967295 i 18446744073709551615. Prefiks u wskazuje, że możliwe do użycia wartości mają tylko znak dodatni i 0. Czyli w przypadku uint8 jego użyteczny zakres wynosi od 0 do 255. Zamiast tego, jeśli użyjemy opcji int8, będziemy mieli również 255 wartości, ale w zakresie od -128 do 127. Przydatność przedrostków i przyrostków jest opisana na przykładach. Wyobraźmy sobie proces, który musi zostać wykonany w mikrosekundach. Jeśli użyjemy podtypu danych uint8, będziemy mieli tylko 255 mikrosekund, co nie jest nawet milisekundą czasu. Jeśli użyjemy uint16, będziemy mieli tylko 65 milisekund. Wreszcie, wybierając podtyp uint32, odpowiednik w sekundach wynosi 4294 lub około 71 minut. Przejdźmy nieco dalej do wykorzystania UART, aby wskazać, że jest to komunikacja, która obsługuje tylko 8-bitowe bloki bez znaku. W ten sposób użycie podtypów 16 i 32 nie jest możliwe, a tym bardziej podtypu z wartościami ujemnymi. Z tego powodu tutaj znajduje się użyteczność podtypów uint8. Wreszcie, w wielu środowiskach programistycznych można znaleźć agregat t w przyrostku liczbowym (na przykład uint16t). Ma to tylko właściwość bycia kompatybilnymi danymi między różnymi platformami sprzętowymi. Kiedy opracowano typy danych 8, 16 i 32, zużycie pamięci różniło się między różnymi platformami obliczeniowymi. W ten sposób producenci doszli do porozumienia w sprawie przenoszenia kodu, tworząc standard oznaczony literą t.

Wprowadzenie do UART

Zauważ, że zakłada się zestaw SDK z poleceniami UART wysokiego poziomu, takimi jak read(), write(), begin() lub available(), jak ten prezentowany w Arduino lub Ardupilot. Użycie UART niskiego poziomu jest zarezerwowane dla książek o języku maszynowym, języku asemblera lub mikrokontrolerach. Transmisja UART jest opisana, ponieważ jest jedną z najczęściej stosowanych w obecnych autopilotach i ich odpowiednich pakietach SDK. Jest dwustronna, ale asynchroniczna (bez udziału timera, a co za tym idzie bez zapewnienia czasu transmisji i odbioru). Wskazaliśmy również, że standard ten opiera się na zarządzaniu 8-bitowymi pakietami danych dodatnich i całkowitych. Aby kontynuować, zadajemy następujące pytania. Biorąc pod uwagę wskazane ograniczenia,

• Jak pracujesz z danymi negatywnymi?

• Jak pracujesz z danymi dziesiętnymi (liczba z miejscami dziesiętnymi)?

• W jaki sposób gwarantowane są czynności czytania i pisania, jeśli nie ma licznika czasu do czytania lub pisania?

• A jak pracujesz z danymi większymi niż wartość 255?

Aby odpowiedzieć na te pytania, zaczynamy od koncepcji licznika cyklicznego. Łatwo to zrozumieć, jeśli odniesiemy się do ruchu obrotowego. W ruchu obrotowym ciało może być zorientowane od 0 do 360 stopni lub od 0 do 2 pi radianów, w zależności od zastosowanego standardu. Wartości wyższe niż ten zakres oznaczają wykonanie wielokrotnych obrotów lub umieszczenie w operacji modulo wskazanej wartości z 360 lub 2pi. Zatem 800 stopni odpowiada dwóm obrotom o 360 stopni i nadwyżce 80 stopni. Zauważ, że dowolny kąt jest funkcją modulo lub reszty o podstawie 360 lub 2pi oraz ilorazu tego kąta o tej samej podstawie (zauważ, że interesują nas tylko części całkowite modulo i ilorazu).

$$N = N \bmod(360) + 360(quotient(N/360))$$

W poprzednim przykładzie

$$800 = 800 \bmod(360) + 360(quotient(800/360))$$
$$800 = 80 + 360(2)$$

Korzystając z tej logiki transmisja szeregowa implikuje, że naszą bazą jest liczba 256 (od 0 do 255 jest 256 wartości), a dowolną liczbę do wysłania można rozłożyć na modulo o tej podstawie i iloraz o tej samej podstawie:

$$N = N \bmod(256) + 256(quotient(N/256))$$

Na przykład,

$$7515 = 7515 \bmod(256) + 256(quotient(7515/256))$$
$$7515 = 91 + 256(29)$$

Krótko mówiąc, aby pracować z wartościami większymi niż 255, liczba musi zostać rozłożona na dwie części, jedną związaną z modulo tej liczby, a drugą z jej ilorazem:

$$7515 = \boxed{7515 \bmod(256)} + 256(\boxed{quotient(7515/256)})$$

Kolejnym pytaniem do rozwiązania jest sposób wysyłania liczb ujemnych. Odpowiedź brzmi, że istnieją dwie opcje, a każda z nich będzie zależeć od zakresu danych, który ma być użyty (na przykład zakres czujnika). Pierwszym z nich jest w zasadzie zmarnowanie jednego z 8 bitów do wysłania na wskazanie, że wspomniany bit ma wartość 0, jeśli liczba do wysłania jest dodatnia lub 0, a ma wartość 1, jeśli liczba jest ujemna. Jednak naszą bazą komunikacyjną nie będzie już 256, ale 128 (z pozostałych 7 bitów możemy utworzyć tylko 128 różnych liczb). Drugim jest wykonanie nasycenia i mapowania (tłumaczenia i skalowania) wartości (operacje opisane w poprzedniej sekcji). Podczas wykonywania nasycenia ustalamy limity dla naszego sygnału i te limity są wymagane do zdefiniowania minimalnych i maksymalnych wartości wejściowych w naszej funkcji mapowania. Tłumaczenie umożliwia nam przejście w kierunku pozytywnego odniesienia, na przykład przejście z zakresu wartości od [-100 100] do jednej z [0 200]. Skalowanie jest opcjonalne, jeśli wejście ma duże wartości w stosunku do wyjścia, na przykład przechodząc od [-1000 1000] do [0 200]. Zauważ, że funkcja mapowania ma już oba efekty. Efektem nadmiernego skalowania (przy dużych liczbach staje się to bardziej zauważalne) jest obecność liczb z częścią ułamkową. Teraz odpowiemy, jak z nimi pracować. Pierwszą rzeczą do zrobienia jest ustalenie pożądanego stopnia precyzji. Musimy ocenić, czy interesuje nas jedno, dwa lub więcej miejsc

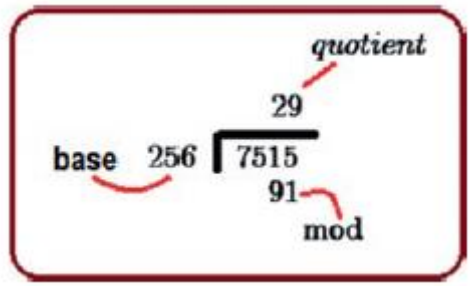
po przecinku. Na przykład, jeśli w naszym zakresie [0 200] interesuje nas praca z liczbami takimi jak 199,3 lub liczbami takimi jak 199,15. Aby komunikować się z tymi numerami, mamy też dwie podstawowe opcje. Pierwszym z nich jest oddzielenie części całkowitej od części ułamkowej i przesłanie ich jako dwóch różnych liczb. W przykładzie numeru 199.15 osobno zostanie wysłana 199, a następnie 15. Problem w tym podejściu polega na tym, że jeden z dwóch numerów może zgubić się po drodze. Inną opcją jest zastosowanie skalowania. W ten sposób, jeśli interesuje nas tylko jedno miejsce po przecinku, wystarczy pomnożyć nasz początkowy zakres przez 10, więc korzystając z podanych przykładów, 199,3 stanie się 1993, a 199,15 stanie się 1991. Zamiast tego, jeśli chcemy mieć dwa miejsca po przecinku miejsc, współczynnik wynosiłby 100, wysyłając odpowiednio 1930 i 19915. W ten sposób wysyłany jest przeskalowany numer. W tym celu wymagane jest, aby zarówno odbiornik, jak i nadajnik znali współczynnik skali. Uzyskanie wysokiego poziomu precyzji przy takim podejściu jest problemem, ponieważ całkiem możliwe, że liczba do wysłania musi zostać podzielona na dwie lub więcej części. Aplikacja jest prosta. Na przykład, jeśli chcemy mieć dwa miejsca po przecinku, zamiast mapować od [-1000 1000] do [0 200], powinniśmy to zrobić od [-1000 1000] do [0 20000]. Zwróć uwagę, że w nadawcy i odbiorniku można wykonać prawie każdą operację, w tym operacje zmiennoprzecinkowe lub podwójne. Ale transmisja danych dotyczy wyłącznie dodatnich wartości całkowitych. Z tego powodu musimy wprowadzić wyżej wymienione poprawki (istnieją interfejsy programistyczne, takie jak Arduino, które zawierają funkcje, które automatycznie wykonują te konwersje; z tego powodu wydaje się, że obsługują nie tylko liczby całkowite dodatnie). Na koniec, aby odpowiedzieć na pozostałe pytanie, jak gwarantowana jest komunikacja, wskażemy, w jaki sposób zaimplementowane są komponenty komunikacji UART. Podstawą tej procedury jest zrozumienie operacji binarnych, które są równoważne z tymi wcześniej przedstawionymi (modulo i iloraz) oraz niektórych innych binarnych operacji sprawdzania. Dzieje się tak, ponieważ wygodnie jest uprościć zadania przetwarzania danych na komputerze w języku, który zna procesor. Te operacje to wysyłanie, odbieranie, sprawdzanie i komunikowanie się.

Wysyłanie UART

Punktem wyjścia jest to równanie:

$$N = N \bmod(256) + 256(quotient(N/256))$$

W formacie dziesiętnym, operacje, aby uzyskać modulo i iloraz są uzyskiwane z podziału, jak pokazano na rysunku



W formacie binarnym iloraz można otrzymać z niecyklicznej operacji przesunięcia w prawo (zastępując liczby przesuwane zerami). Jest to stosowane do ostatnich 8 bitów binarnego odpowiednika liczby, która ma zostać podzielona. Ta operacja jest reprezentowana jako >> 8 w C ++ jako odniesienie dla innych języków programowania. Z drugiej strony, moduł uzyskuje się z operacją AND zaaplikowaną z

255 lub 1111 1111 w bazie binarnej. Jest to reprezentowane jako & w C ++ jako odniesienie dla innych języków programowania.

Na tym samym przykładzie:

7515=111001011011011 (bin)

11101 0101 1011(bin) >>8 = 0000 0000 11101(bin) = 29(dec)

Z drugiej strony:

11101 0101 1011 (bin)

& 00000 1111 1111 (bin)

= 00000 0101 1011 (bin) = 91 (dec)

W ten sposób numer (w tym przypadku 7515) jest wysyłany jako dwie grupy po 8 bitów:

0101 1011 (bin) = 91 (dec)

0001 1101(bin) = 29(dec)

Oczywiście przed wysłaniem tych numerów zostały one uzyskane za pomocą opisanych wcześniej operacji (>>8 i & 1111 1111).

Odbieranie UART

Odbiór polega na zastosowaniu następującego równania:

$$N = N \bmod(256) + 256(quotient(N/256))$$

Bezpośrednio w niektórych językach programowania lub za pomocą następującego odpowiednika binarnego: 8-bitowe niecykliczne przesunięcie w lewo i operacje OR (<< 8 i | w C ++):

Wartość binarna = Część binarna ilorazu << 8 | Część modułu binarnego

Korzystając z poprzedniego przykładu:

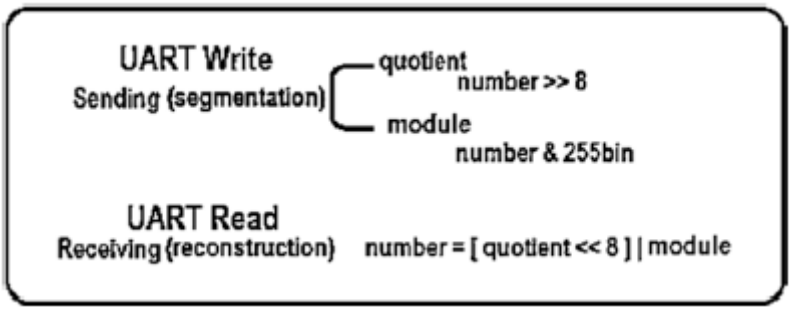
0001 1101(bin) = 29 (dec) << 8 = 0000 11101 0000 0000 = 7424 (dec)

Mając taki wynik, kontynuujemy operację OR. Zauważ, że jest więcej niż 8 bitów, ale pamiętaj, że jest to przetwarzane przez komputer odbierający po odebraniu danych. Nie dotyczy to kanału komunikacyjnego, który wymaga wiadomości 8-bitowych.

11101 0000 0000| 00000 0101 1011 (bin) = 91 (dec)

= 11101 0101 1011(bin) = 7515(dec)

Procesy czytania i pisania podsumowano na rysunku



Teraz wiesz, jak wysyłać i odbierać informacje. Jak jednak weryfikujesz dane? A skąd wiesz, czy potrafisz się komunikować? Odpowiedzią na to są następujące operacje.

Sprawdzanie UART

W takim przypadku błędna informacja jest odrzucana, najprostszym algorytmem jest suma kontrolna, a jego ulepszeniem jest suma kontrolna XOR. Proste wyjaśnienie odbywa się z podstawy dziesiętnej.

Załóżmy, że wysyłane są następujące dane:

val1Send=50

val2Sendj=30

Trzecią wartością do wysłania może być suma poprzednich ilości (suma części algorytmu sumy kontrolnej):

wartSend=wart1+wart2=80

Te same dane powinny dotrzeć do odbiorcy:

Val1Rec=50

Val2Rec=30

valSumRec=80

Następnie budowana jest część kontrolna algorytmu:

Check=Val1Rec+Val2Rec=80

W ten sposób Check i valSumRec są takie same i jest to prawidłowy zbiór danych. Załóżmy teraz, że odbiornik odczytuje błędną wartość:

Val1Rec=20

Val2Rec=30

valSumRec=80

Ponownie budujemy część kontrolną:

Check=Val1Rec+Val2Rec=50

W ten sposób Check różni się od valSumRec i gromadzenie danych jest odrzucane lub informacje są ponownie żądane.

Jak widać, algorytm sumy kontrolnej ma kilka ważnych problemów:

1. Suma może przekroczyć wartość 255 i wymagane będzie przesłanie liczby w postaci dwóch lub więcej grup danych.

2. Suma błędnych danych może skutkować poprawną kontrolą. Na przykład nadawca pisze

val1Send=50

val2Send=30

valsumSend=val1+val2=80

Odbiorca czyta

Val1Rec=60

Val2Rec=20

valSumRec=80

A ponieważ kontrola pozostaje równa 80, ten niepoprawny pakiet danych nie zostanie odrzucony.

3. Dane docierają poprawnie, ale suma jest błędna.

Nadawca pisze

val1Send=50

val2Send=30

valsumSend=val1+val2=80

A odbiorca czyta

Val1Rec=50

Val2Rec=30

valSumRec=10

Ponieważ wartość check=80 różni się od wartości valSumRec, informacje są odrzucane, mimo że dotarły poprawnie.

Aby zmniejszyć te problemy (przynajmniej problem przepełnienia), stosuje się binarne sumy kontrolne. Jednym z nich jest suma kontrolna XOR (operator ^ w C++). Zauważ, że pozostałe problemy są nadal obecne. Jeśli chcesz wysłać wartość sumy przekraczającą 255, na przykład 520, pierwszą rzeczą do zrobienia jest przekształcenie binarne:

1000001000 (bin) = 520 (dec)

Następnie dodaje się XOR element po elemencie:

(1)xor(0)xor(0)xor(0)xor(0)xor(0)xor(1)xor(0)xor(0)xor(0)=0

Tak więc dodatkowe dane do wysłania to po prostu 0.

Na przykład:

Val1Send=250

Val2Send=250

Val3Send=20

ValSum=520

Nie można wysłać wartości Valsum, ale zamiast tego wysyłana jest suma XOR jej składników binarnych:

XORSend=(1)xor(0)xor(0)xor(0)xor(0)xor(1)xor(0)xor(0)xor(0)=0

Odbiorca czyta

Val1Rec=250

Val2Rec=10

Val3Rec=20

XORRec=0

Następnie buduje się kotrolę:

Check=Val1Rec+Val2Rec+Val3Rec=280

Suma XOR składa się z binarnych składników czeku:

100011000 (bin) = 280 (dec)

XORCheck=(1)xor(0)xor(0)xor(0)xor(1)xor(1)xor(0)xor(0)xor(0)=1

Ponieważ wynik różni się od XORRec, dane są odrzucane lub ponownie żądane.

Zgoda na obrót

W takim przypadku przesyłane są dodatkowe informacje wskazujące, które urządzenia są odbiornikami, a które nadajnikami podczas procesu komunikacji. Algorytm polega na wysłaniu i ocenie wartości logicznej. Jest to posiadanie dwóch lub więcej dronów, które zawierają urządzenia komunikacyjne; wszystkie mają zmienną, która działa jak sygnalizacja świetlna lub flaga. Jedna z nich ma tę zmienną w stanie ON, a pozostałe w stanie OFF. Tylko ten, który zaczyna się od stanu ON, może pisać, podczas gdy inne czytają. Jak tylko skończy się ten, który zaczął się w stanie ON, zmienia swoją flagę lub semafor na OFF. Kiedy inni członkowie zespołu przeczytają, że flaga początkowego nadajnika zmienia się na WYŁ., inicjują sekwencję, aby określić, który z nich jako następny zmieni flagę na WŁ. Powtarza się to cyklicznie, na przemian lub tylko raz. Zasadniczo jest to uprzejma rozmowa, w której każdy uczestnik prosi lub obraca swoją kolejkę, aby się porozumieć, czekając na zakończenie rozmowy.

Ogólny algorytm UART

Ogólny algorytm komunikacji UART z wykorzystaniem wyżej wymienionych operacji jest następujący. (Podkreślono specyficzny wygląd każdej operacji; zauważ, że suma kontrolna jest operacją współdzieloną. Aby ulepszyć tę sekwencję, każdy proces może zależeć od timera, który przyspiesza lub zapewnia jego wykonanie lub po prostu w celu uniknięcia opóźnień.)

1. Sprawdzana jest flaga lub obrót urządzenia komunikacyjnego.

2. Jeśli bieżąca tura polega na pisaniu,

• Informacje do wysłania są podzielone na 8-bitowe segmenty. Informacje te są podzielone na dwie części: modulo i iloraz.

• Przeprowadzana jest suma wszystkich informacji do wysłania, az wyniku otrzymuje się sumę XOR jej składowych binarnych.

• Dostępność kanału jest weryfikowana.

• Informacja zostaje wysłana.

• Zmienne używane do pisania są czyszczone do późniejszego wykorzystania.

• Stan flagi zostaje zmieniony na odczyt.

3. Jeśli kolejka oznacza czytanie,

• Odebrane dane są odczytywane.

• Przeprowadzana jest kontrola XOR, a następnie weryfikowana z otrzymaną sumą.

• Jeśli się nie pokrywają, projektant określi procedurę do wykonania, która może obejmować od odrzucenia informacji do ponownego zażądania informacji.

• Jeśli pokrywają się, informacja jest rekonstruowana ze wskazanym przesunięciem bitowym i operacją OR.

• Zmienne wykorzystywane do odczytu są czyszczone do późniejszego wykorzystania.

• Stan flagi zostaje zmieniony na zapis.

4. Proces trwa w nieskończoność do momentu przerwania komunikacji przez użytkownika, kod lub zasilanie.

Teraz, gdy wiesz już o sterowaniu, przetwarzaniu sygnału i komunikacji UART , kolejne sekcje wskażą Ci sposoby kodowania drona. Najpierw pokażemy Ci dostępne środowiska do programowania Twojego samolotu.

Dostępne sposoby programowania drona

Obecnie istnieją dwa sposoby kodowania dronów: GUI i SDK.

GUI

GUI to uproszczony i wysoce wizualny interfejs zaprojektowany dla użytkownika z ograniczonymi lub zerowymi umiejętnościami programowania. W tym przypadku jest to graficzny interfejs użytkownika. Wiele zamkniętych architektur do programowania dronów ma jedną z nich. Te zamknięte architektury charakteryzują się tym, że umożliwiają użytkownikowi modyfikowanie podstawowych parametrów, takich jak ścieżka do przebycia, wzmocnienia wstępnie załadowanych kontrolerów oraz podstawowa kalibracja czujników i elementów wykonawczych. Bardziej specjalistyczne zadania są zamknięte dla użytkownika. Niektóre z najpopularniejszych GUI to Mission Planner i LibrePilot.

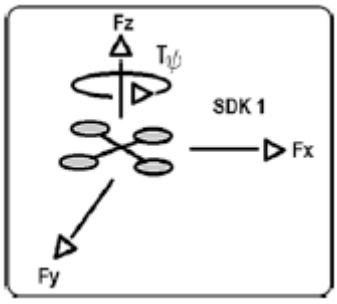
SDK

SDK to zestaw programistyczny. We wszystkich przypadkach użytkownik musi znać język programowania.

Poziom 1 pakietu SDK

W przypadku SDK Level 1 użytkownicy niewyspecjalizowani w sterowaniu dronem lub robotyce mogą zaprogramować drona do celów innych niż rekreacyjne. Architektura nie jest całkowicie otwarta i

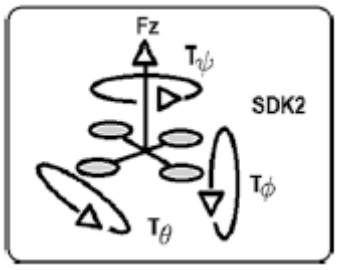
pozwala jedynie użytkownikowi zaprojektować niektóre kontrolery wysokiego poziomu w taki sposób, jakby pojazd był cząsteczką zdolną do poruszania się w osi X, Y, Z i obracania się wokół własnej osi Z. Zobacz rysunek



Jednym z najbardziej znanych przykładów jest SDK Dronekit oparty na Pythonie. W tego typu SDK użytkownik może modyfikować tylko żądane odniesienia X, Y, Z i odchylenia lub ich prędkości. Nic nie da się zrobić z siłą ciągu i momentami obrotowymi drona i oczywiście z macierzą alokacji.

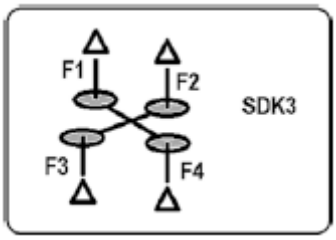
Poziom 2 pakietu SDK

Architektura w SDK Level 2 jest nieco bardziej otwarta i umożliwia sterowanie pojazdem na poziomie siły i momentu. Oznacza to, że pozwala użytkownikom, którzy są zaznajomieni ze sterowaniem dronami i robotyką, na włączenie własnych algorytmów i kontrolerów do wstępnie zaprojektowanych pojazdów, które są kompatybilne z SDK. Jednym z najbardziej znanych przykładów jest Parrot Bebop-autonomy SDK oparty na C++, a także tryb lotu bibliotek PX4. W tego typu SDK użytkownik może jedynie modyfikować siłę ciągu i momenty obrotowe drona. Nic nie można zrobić z macierzą alokacji. Ponieważ siła ciągu i momenty obrotowe drona są funkcją odniesienia przestrzennego (X, Y, Z i odchylenia), ten pakiet SDK ma wyższy poziom niż pakiety SDK typu 1.



SDK poziom 3

SDK Level 3 jest całkowicie otwarty i pozwala użytkownikowi sterować pojazdem na poziomie silnika. Umożliwia także programowanie własnych protokołów komunikacyjnych, włączanie własnych czujników, monitorowanie interesujących ich parametrów i nie tylko. Krótko mówiąc, pozwala użytkownikowi zaprojektować własne pojazdy, nawet jeśli pojazdy te nie mają precedensu. Najbardziej znanymi przykładami są biblioteki Ardupilot oraz biblioteki PX4 oparte na C++. W tego typu SDK użytkownik musi podać macierz alokacji, która jest funkcją momentów ciągu i sił drona, które z kolei są funkcją żądanej pozycji i wartości orientacji. Dlatego ten rodzaj SDK ma wyższy poziom niż zestawy SDK typu 2 i 1.



Nauczyłeś się, gdzie kodować, ale co z kodowaniem? To jest wprowadzone w następnej sekcji.

Niektóre przydatne polecenia dostępne w większości zestawów SDK

Poniżej znajdują się najczęściej używane polecenia do programowania dronów powietrznych i ogólnie każdego innego rodzaju pojazdu. Można je znaleźć w trzech najczęściej używanych pakietach SDK: Ardupilot, PX4 i Dronekit. Polecenia te są przedstawione w sposób tematyczny i są kompatybilne odpowiednio z C++, C++ dla ROS i Pythonem (chociaż te SDK mają warianty dla innych języków programowania, systemów operacyjnych i architektur obliczeniowych). W odniesieniu do wyżej wymienionych bibliotek należy wskazać, że jedyną, która pozwala użytkownikowi na włączanie zadań czasu rzeczywistego jest Ardupilot, najprostszym w obsłudze jest Dronekit, a ta, która ma wiele kompatybilności, ponieważ bazuje na ROS to PX4, co jako wadę ma ograniczoną i mało zrozumiałą dokumentację (przynajmniej do tej pory).

Streszczenie

Poznałeś zadania drona, a także podstawowe rodzaje przetwarzania sygnału, które są wymagane do jego użycia. Wyjaśnione przetwarzanie sygnału obejmowało filtrowanie sygnału, nasycenie sygnału, mapowanie sygnału, rzutowanie danych oraz normalizację nadmiarowości i osobliwości. Wszystkie mogą być używane z czujnikami, siłownikami oraz ze stopniami komunikacji lub sterowania. W przypadku zadania komunikacyjnego poznałeś zwykłe części UART wysokiego poziomu. Na koniec dowiedziałeś się o czterech sposobach programowania drona, w tym o tematycznej liście poleceń drona z podejściem opartym na trzech najczęściej używanych pakietach SDK.