Mechanika Nieba

Misja na Jowisza

z asystą grawitacyjną Marsa

Andrzej Odziemkowski

2018

**Spis treści**

s[1. Wstęp 3](#_Toc503932604)

[1.1. Cel projektu 3](#_Toc503932605)

[1.2. Misja Galileo i Juno 3](#_Toc503932606)

[2. Narzędzia do realizacji programu 5](#_Toc503932607)

[2.1. Python 5](#_Toc503932608)

[2.2. Poliastro. 5](#_Toc503932609)

[2.3. Astropy 5](#_Toc503932610)

[3. Podstawy mechaniki orbitalnej 6](#_Toc503932611)

[3.1. Asysta grawitacyjna 6](#_Toc503932612)

[3.2. Zagadnienie dwóch ciał 6](#_Toc503932613)

[3.3. Równanie Ciołkowskiego 7](#_Toc503932614)

[4. Program 8](#_Toc503932615)

[4.1. Parametry wejściowe 8](#_Toc503932616)

[4.2. Funkcja optymalizująca transfer pomiędzy planetami 8](#_Toc503932617)

[4.3. Funkcja optymalizująca datę startu 9](#_Toc503932618)

[4.4. Funkcja sprawdzająca osiągniętą orbitę 11](#_Toc503932619)

[4.5. Główna część programu 11](#_Toc503932620)

[5. Obliczenia 11](#_Toc503932621)

[6. Wnioski. 11](#_Toc503932622)

[7. Bibliografia 11](#_Toc503932623)

# 1. Wstęp

## Cel projektu

Celem poniższego projektu było stworzenie programu, który przeprowadzi obliczenia misji międzyplanetarnej. Podstawowym założeniem było przeprowadzenie misji polegającej na transferze satelity z orbity Ziemi na orbitę Jowisza. Program stworzono w języku *Python* używając bibliotek wspierających mechanikę orbitalną i astronomię, np.: *Poliastro, astropy.*

Wynikami obliczeń są sumaryczne przyrosty prędkości niezbędne do udanego przeprowadzenia zaplanowanych manewrów. Dodatkowo zakładając impuls właściwy napędu, używając równania Ciołkowskiego jest możliwe określenie jaka ilość materiału pędnego zostanie użyta podczas misji.

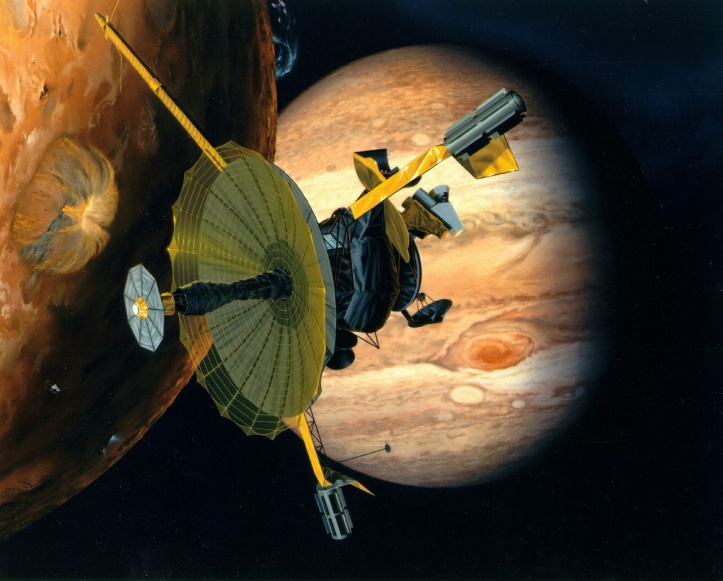
Podczas misji międzyplanetarnych manewr zwany asystą grawitacyjną, pozwala wykorzystać ciała niebieskie w celu zwiększenie prędkości satelity bez zużywania materiału pędnego. Możliwe asysty są determinowane przez pozycje ciał niebieskich w układzie słonecznym w związku z czym zależnie od daty manewr transferu może być mniej lub bardziej kosztowny pod względem zużycia paliwa. Bardzo ważnym zagadnieniem jest określenie tych furtek, gdy transfer jest najbardziej optymalny.

Aby zrealizować wszystkie wspomniane wcześniej cele, opisywanemu satelicie nadano następujące parametry:

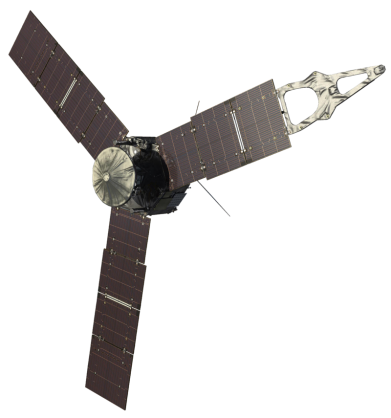
* masa własna satelity
* impuls właściwy napędu
* wysokość początkowej orbity kołowej dookoła Ziemi
* data startu, zakończenia misji

## Misja Galileo i Juno

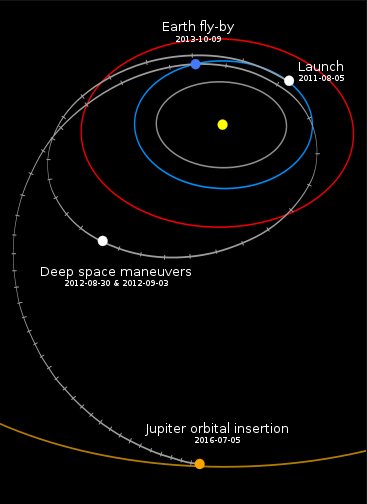
Jowisz od czasów starożytnych był interesującym obiektem na nieboskłonie. Dzięki użyciu teleskopu, włoski astronom i filozof Galileusz, zaobserwował 4 największe księżyce Jowisza zwane obecnie galileuszowymi: Io, Europa, Ganimedes i Kallisto. Największa planeta naszego układu słonecznego charakteryzuje się również układem barwnych pasów i 'oczek' opasających planetę. Najbardziej znanym z nich jest Wielka Czerwona Plama, ogromny antycyklon zaobserwowany już w XXVII wieku, od wieków pobudzał ludzką wyobraźnię.

  
**Rys 1 - satelita Galileo**

Wraz z nadejściem czasów nowożytnych, Jowisz stał się naturalnym celem agencji kosmicznych. Galileo - bezzałogowa sonda kosmiczna została wystrzelona w 1989 roku przez agencję kosmiczną NASA w celu wykonania badań Jowisza, jego księżyców i pierścieni. W grudniu 1995 r. sonda stała się pierwszym sztucznym satelitą Jowisza oraz wprowadziła w jego atmosferę próbnik z aparaturą pomiarową.

  
**Rys 2 - satelita Juno**

Przesyłając ogromną ilość danych na Ziemię Galileo przyczynił się w ogromnym stopniu do zrozumienia gazowych olbrzymów oraz ich księżyców. W 2002 roku nastąpił koniec misji. Na następcę przyszło czekać do 2016 roku kiedy satelita Juno weszła na orbitę Jowisza. W celu zmniejszenia masy materiały pędnego wykonała ona asystę grawitacyjną Ziemi. Obie te, niezwykle ważne misje stanowiły inspirację dla tematu mojego projektu z Mechaniki Nieba.

  
**Rys 3 - przebieg misji Juno**

# Narzędzia do realizacji programu

## Python

Język programowania wysokiego poziomu ogólnego przeznaczenia, o rozbudowanym pakiecie bibliotek standardowych, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością. Python wspiera różne paradygmaty programowania: obiektowy, imperatywny oraz w mniejszym stopniu funkcyjny. Posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią.

Ze względu na to, że manualne odwzorowanie w Pythonie układu słonecznego, manewrów orbitalnych itd. byłoby niezwykle problematyczne, zastosowano pewne biblioteki posiadające już takie funkcje.

## Poliastro.

Poliastro jest open-sourcowym pakietem dedykowanym problemom Astrodynamiki i Mechaniki orbitalnej takim jak propagacja na orbicie, rozwiązanie problemu Lambert'a, uzyskanie wektorów prędkości i położenia.

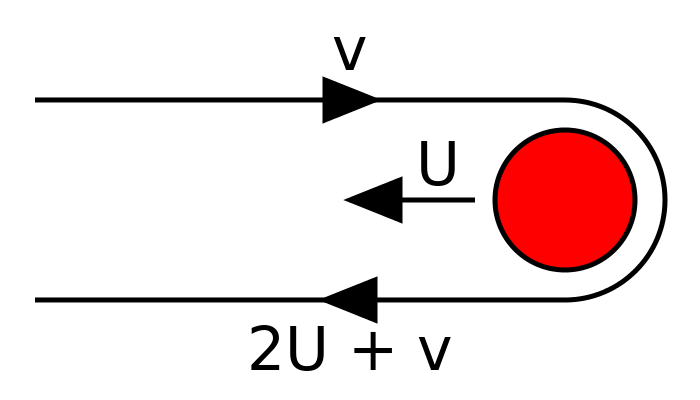
## Astropy

Astropy to open-soursowy pakiet, który zawiela podstawowe narzędzia pozwalające na pisanie programów związanych z astronomią i astrofizyką w Pythonie.

# Podstawy mechaniki orbitalnej

## Asysta grawitacyjna

Jest to zmiana prędkości i kierunku lotu kosmicznego przy użyciu pola grawitacyjnego planety lub innego dużego ciała niebieskiego. Jest to obecnie powszechnie używana metoda uzyskiwania prędkości pozwalających osiągnąć zewnętrzne planety Układu Słonecznego. Asysta grawitacyjna zmienia kierunek, w którym porusza się pojazd, nie zmieniając jego prędkości względem planety. Umożliwia to zwiększenie prędkości względem Słońca maksymalnie o dwukrotność prędkości orbitalnej planety. Manewrowanie w przestrzeni międzyplanetarnej wymaga brania pod uwagę grawitacji Słońca. Pojazdy wysyłane w kierunku wewnętrznych planet – Wenus i Merkurego, zbliżając się do Słońca nabierają prędkości i aby wejść na ich orbitę muszą ją jakoś zmniejszyć. Z kolei pojazdy wysyłane w kierunku zewnętrznych planet muszą nabrać odpowiedniej prędkości, aby móc oddalić się na wystarczającą odległość od Słońca. Realizacja tego przy pomocy napędu rakietowego wymaga dużych ilości paliwa – dlatego poszukuje się innych metod.

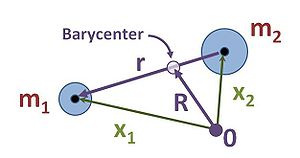
  
**Rys 4 - Uproszczony schemat asysty grawitacyjnej**

## Zagadnienie dwóch ciał

Zagadnienie to opisuje ruch dwóch ciał względem wspólnego środka ciężkości. Ruch układu składającego się z dwóch ciał wygodnie jest rozpatrywać w układzie odniesienia, w którym środek masy układu jest w spoczynku. W tym układzie każde z nich porusza się po trajektorii, która jest krzywą stożkową, a środek masy znajduje się w jednym z jej ognisk.

Jeśli krzywe, po których poruszają się ciała, są zamknięte, to są one elipsami. Energia potencjalna układu ciał (jest ona ujemna), przewyższa co do wartości bezwzględnej sumaryczną energię kinetyczną układu, a całkowita energia układu jest ujemna (pomijamy tutaj energię kinetyczną obrotu ciał wokół swych osi).

Jeżeli orbity są otwarte, ruch ciał odbywa się po hiperboli lub paraboli.

  
**Rys 5 - Schemat zagadnienia dwóch ciał**

Rozwiązanie tego zagadnienia zawiera zię w pakiecie poliastro.twobody. Zdefiniowanie orbit i obliczenia przeprowadzane są na obiekcie klasy nazwanym *Orbit*. W poniższym projekcie analizowany układ składa się ze Słońca, Ziemi, Marsa i Jowisza. Zakładamy, że początkowo orbita dookoła Ziemi jest kołowa.

## Równanie Ciołkowskiego

Celem optymalizacji jest uzyskanie minimalnego zużycia paliwa podczas manewrów. W praktyce oznacza to minimalizację wektora zmiany prędkości tzw. Δv. W połączeniu ze znanym impulsem właściwym napędu możemy obliczyć jaka masa paliwa jest niezbędna do wykonania zadanego manewru. Używając równania Ciołkowskiego:

gdzie:

Isp – impuls właściwy napędu [m/s]

m – masa własna satelity [kg]

m0 – początkowa masa satelity = masa własna + masa paliwa [kg]

Przekształcając wzór otrzymujemy:

gdzie:

mp – masa paliwa [kg]

Otrzymana masa to masa paliwa zużyta podczas manewru. Znając masy zużyte podczas kolejnych manewrów jesteśmy w stanie optymalizować je tak, by sprostać założeniom misji.

Weryfikując prawdziwość wyników uzyskanych za pomocą powyższego wzoru należy mieć w pamięci, że nie uwzględnia on czasu zmiany prędkości, wszelkie jej zmiany odbywają się w sposób natychmiastowy. W rzeczywistości manewry takie odbywają się w pewnym skończonym czasie, który dla napędów takich jak np. jonowe, może być bardzo długi.

# Program

Program działa bazując na głównych założeniach projektu, wyniki wyświetlane są w formie tekstowej i wykresów.

Schemat jego działania można streścić w następujący sposób:

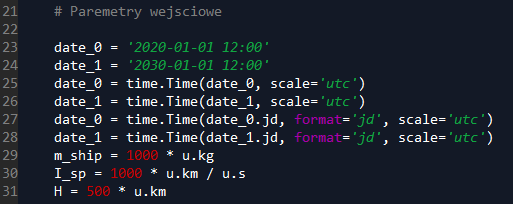
* zadanie parametrów wejściowych
* optymalizacja daty wylotu
* optymalizacja transferów między planetarnych
* osiągnięcie orbity Jowisz
* wyświetlenie wyników w postaci tekstowej i wykresów

## Parametry wejściowe

Parametry zadane zostały w sposób bezpośredni, warunki początkowe należy zmienić bezpośrednio w kodzie programu.

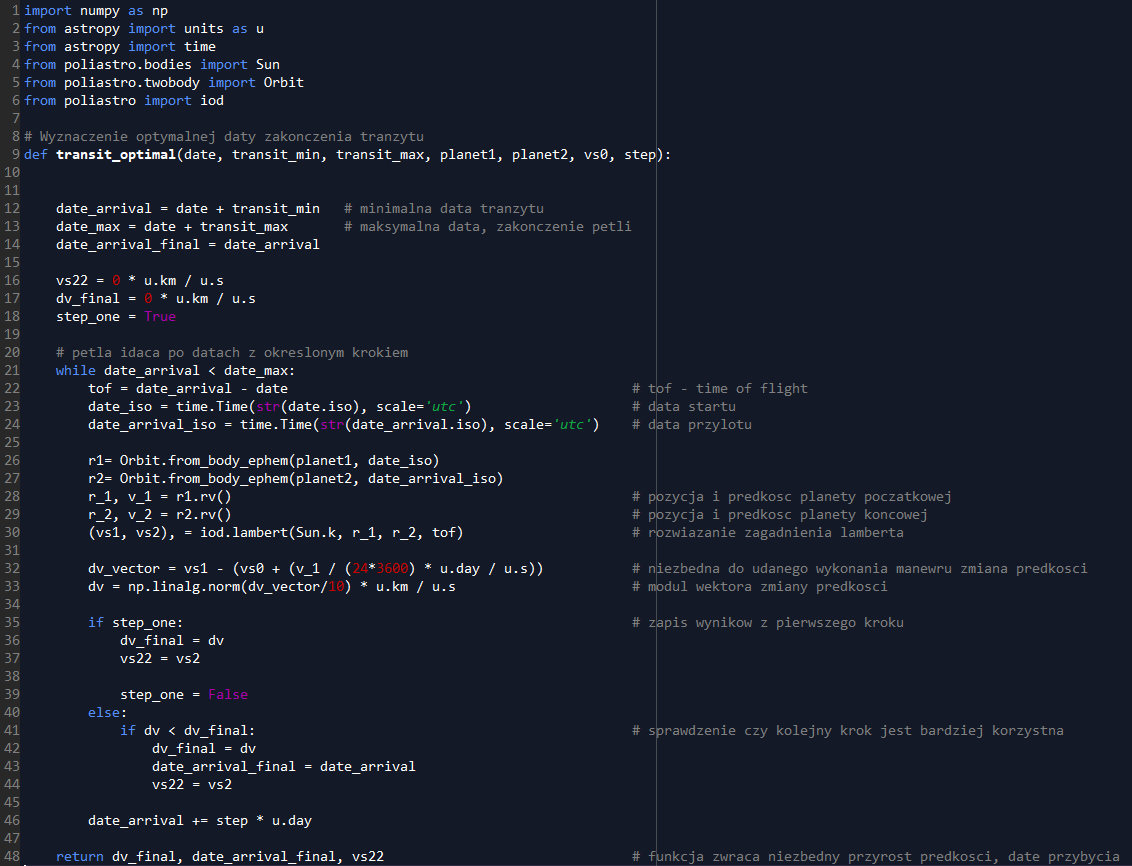
Na rysunku 6 przedstawione zostały zadane parametry wejściowe symulacji. Zmiennym *date\_0* i *date\_1*, które symbolizują początek i koniec okresu, w którym będzie przeprowadzana analiza możliwych transferów, przypisuje się daty, za pomocą klasy *Time*. Daty te są następnie zamieniane na format juliański.

Następnie zmiennym *m\_ship*, *I\_sp* oraz *H* przypisywane są wartości: masy własnej satelity, impulsu właściwego i wysokości początkowej orbity kołowej.

  
**Rys 6 - Przykładowe parametry początkowe sytuacji**

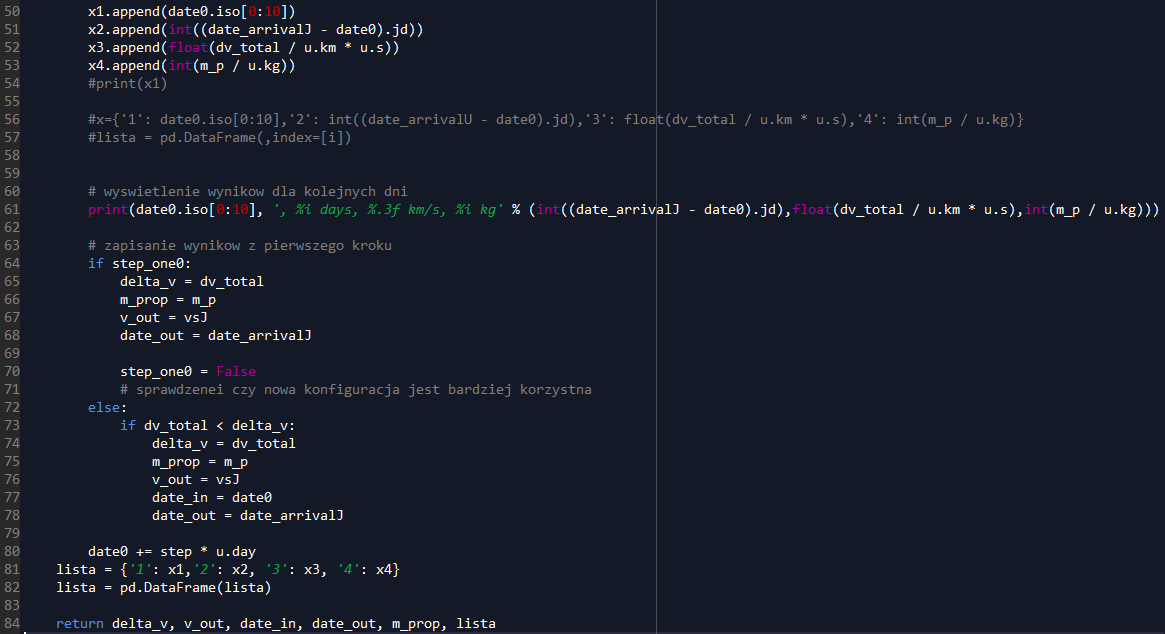
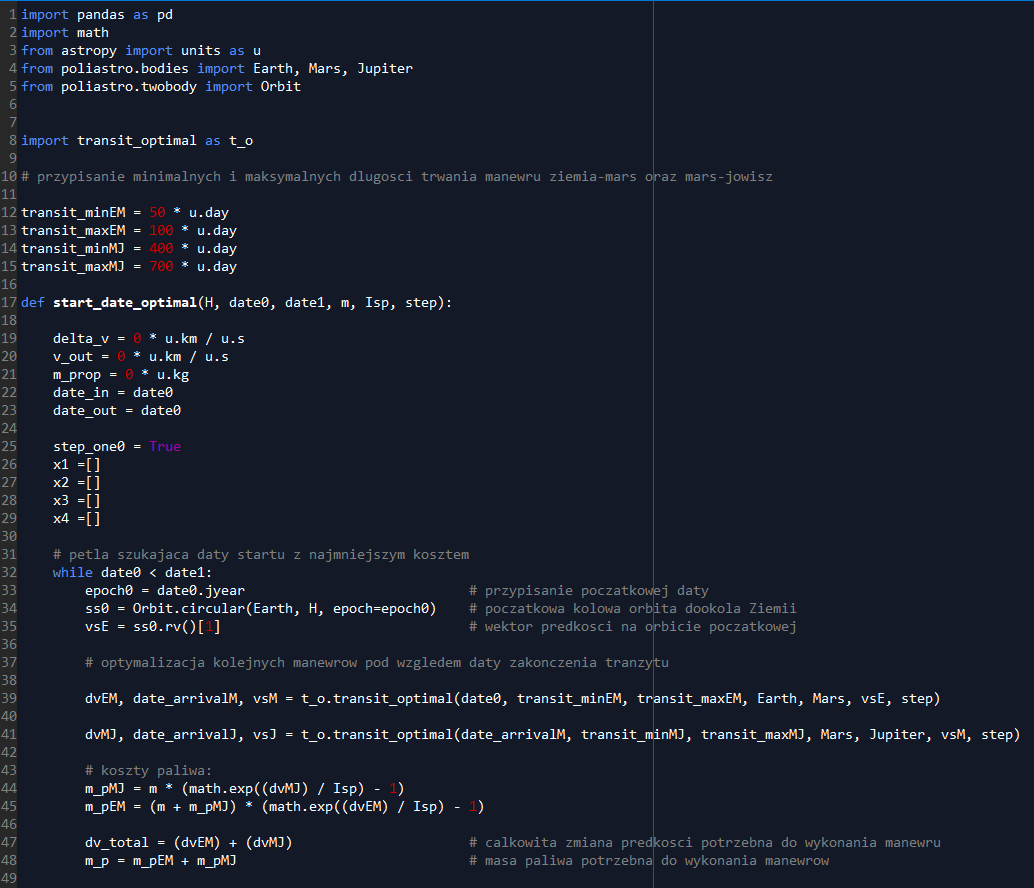
## Funkcja optymalizująca transfer pomiędzy planetami

Główną częścią funkcji *transit\_optimal* jest pętla *while*. Odbywa się ona od najwcześniejszej zadanej daty do daty zamykającej analizowane okno startowe. Krok tej pętli jest zadany w głównej części programu. W pętli zadawane są data startu i data przylotu. Następnie dla tych dat obliczane są pozycja i prędkość planety 1 dla daty startu i analogicznie pozycja i prędkość planety 2 dla daty przylotu. Dla tych wartości rozwiązywane jest zagadnienie Lambert'a i uzyskiwany jest niezbędny do wykonania manewru przyrost prędkości satelity. następnie za pomocą instrukcji warunkowej *if else* następuje sprawdzenie, czy przyrost prędkości jest mniejszy w kolejnym kroku czasowym. Funkcja zwraca wektor przyrostów prędkości i datę przylotu.

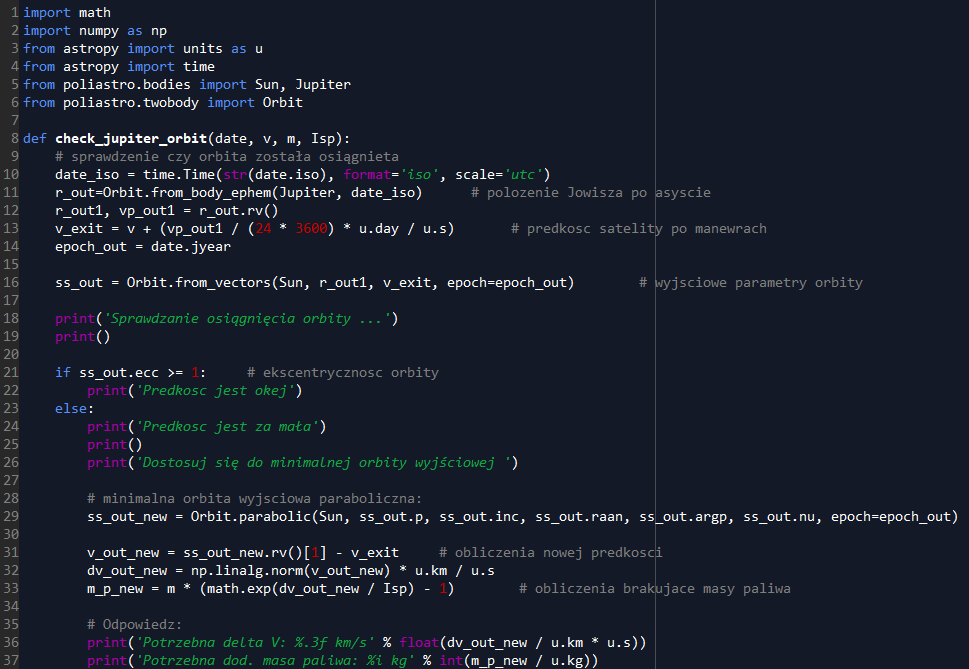


## Funkcja optymalizująca datę startu

Funkcja *start\_date\_optimal* ma za zadanie wyznaczyć optymalną datę startu ze względu na koszt transferów. Początkowo zadaje się maksymalne i minimalne czasy trwania poszczególnych transferów. Po inicjalizacji zmiennych rozpoczyna się pętla *while*, która trwa dopóki *date\_0* nie zrówna się z *date\_1* . Po zakończeniu każdego obiegu pętli czas jest 'przesuwany' do przodu o zadany krok *step*. Na jej początku ustalana jest pozycja i prędkość satelity na początkowej orbicie kołowej wokół Ziemi dla danego czasu. Następnie używając wcześniej zdefiniowanej funkcji *transit\_optimal* optymalizuje się manewry, dzięki czemu otrzymywane są całkowity koszty paliwa i zmiana prędkości. Kolejne dane istotne z punktu widzenia wyników są dodawane do zdefiniowanych wcześniej zmiennych. Na końcu pętli *while* występuje instrukcja sterująca *if else* mająca za zadanie optymalizację daty startu w każdym kolejnym kroku.

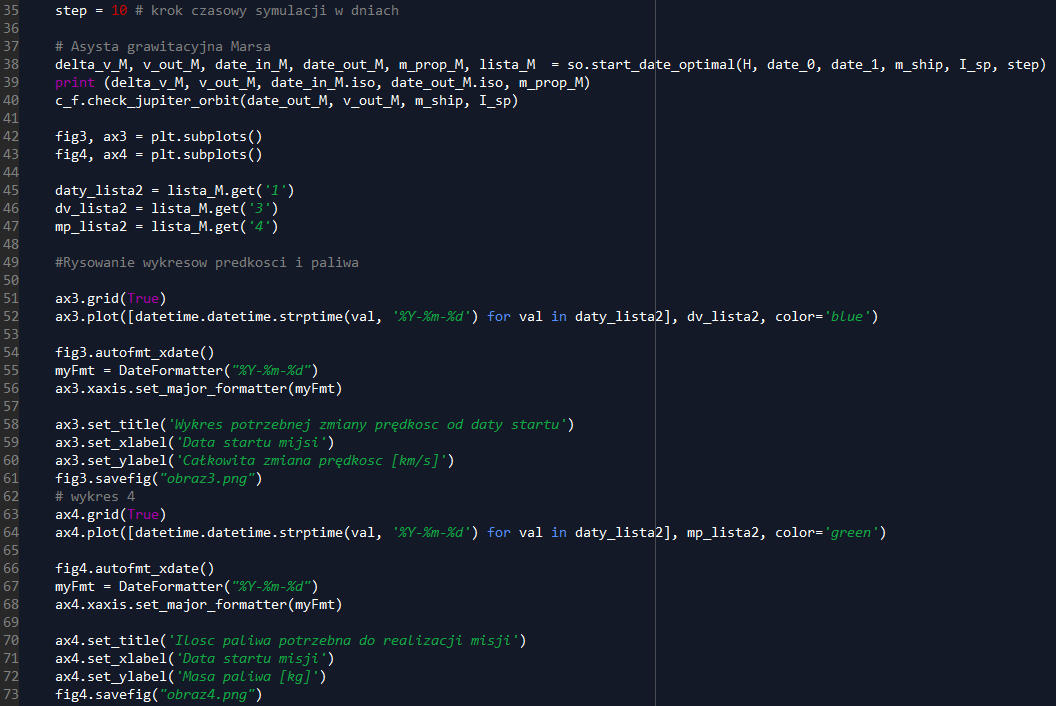
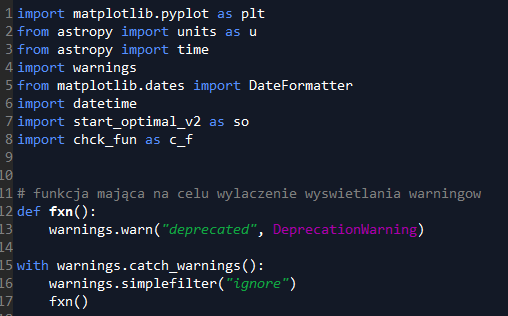


## Funkcja sprawdzająca osiągniętą orbitę



## Główna część programu

Głowna program używa funkcji *start\_date\_optimal* do obliczenia przyrostów prędkości, mas paliwa, dat asysty grawitacyjnej Marsa. Następnie używając funkcji sprawdzającej sprawdzany jest warunek osiągnięcia orbity Marsa. Dalsza część głównego programu to wizualizacja wykresów paliwa i przyrostów prędkości.

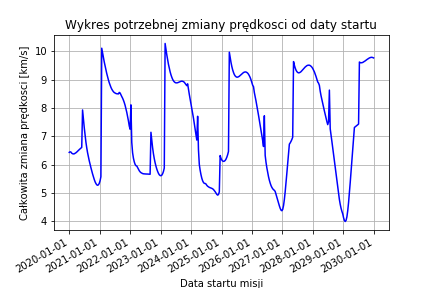


# Obliczenia

Obliczenia przeprowadzono dla kilku zestawów danych obrazujących wpływ parametrów początkowych symulacji na wyniki. Analizowany przedział czasowy: od 01-01-2020r. g.12 do 01-01-2030r. g.12 .

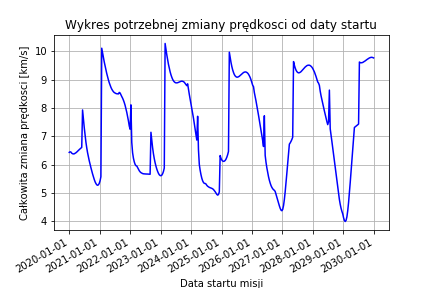
|  |  |  |
| --- | --- | --- |
| zestaw #1 (step = 10 dni) | | |
| masa | I\_sp | H |
| 500 | 1000 | 300 |
| 1000 | 1000 | 300 |

**1) m = 500 kg**





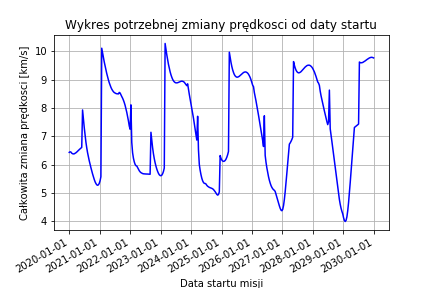
**2) m = 1000 kg**





|  |  |  |
| --- | --- | --- |
| zestaw #2 (step = 10 dni) | | |
| masa | I\_sp | H |
| 1000 | 450 | 300 |
| 1000 | 1000 | 300 |

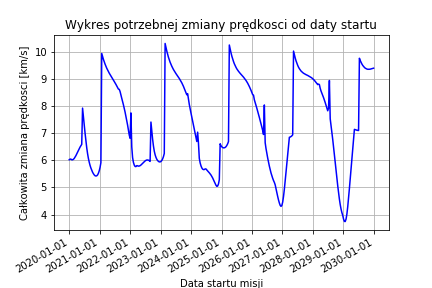
**1) I\_sp = 450**

****

****

|  |  |  |
| --- | --- | --- |
| zestaw #3 (step = 10 dni) | | |
| masa | I\_sp | H |
| 1000 | 1000 | 300 |
| 1000 | 1000 | 30000 |

**1) H = 30000**

****

****

# Wnioski

# Bibliografia