**Metody Numeryczne [MNUM]  
Projekt 3.40**

Tomasz Pawlak, Informatyka, 304104  
Prowadzący: dr hab. inż. Andrzej Karbowski

**Zad 1.**

Ruch punktu na płaszczyźnie jest opisany równaniami:

Należy obliczyć przebieg trajektorii ruchu tego punktu w przedziale dla warunków początkowych:

Rozwiązanie znaleźć korzystając z zaimplementowanego solwera **metody Rungego-Kutty czwartego rzędu (RK4) przy zmiennym kroku z szacowaniem błędu metodą zdwajania kroku.**

**Opisy algorytmów**

Metoda Rungego-Kutty RK4 ze zmiennym krokiem omija problem wykonywania małych kroków w miejscach, gdzie nie jest to konieczne, przez co zmniejsza błąd spowodowany dokładnością reprezentacji liczb.

Samą metodę możemy formułować dla dowolnej rzędu, wybieramy w zadaniu czwarty, gdyż jest to kompromis pomiędzy najwyższą dokładnością – wyrażaną jako rząd metody *p* oraz nakład obliczeń na jedną iterację i powiązany z tym błąd zaokrągleń. Główny wzór definicji metody prezentuje się następująco:

, gdzie

Przekładając wzory z rozdziału 7.1.1 podręcznika na nazwy parametrów użytych przeze mnie w zadaniu otrzymujemy wzór iteracyjnego postępowania znajdywania kolejnych iteracji dla metody 4-rzędu oraz układu dwóch funkcji równań różniczkowych z zadania:

gdzie

Wzory te mają zastosowanie nie tylko do obliczenia kroku całkowitego, ale również są wykorzystane w metodzie zdwajania półkroków.

Metoda zdwajania kroków wymaga również policzenia błędu pomiędzy przejściem między półkrokowym algorytmem oraz pełnym równaniem kroku. Aby tego dokonać należy oszacować błąd dwóch półkroków o długości . Ogólny wzór na część główną błędu metody rzędu 4 dla kroku h jest postaci:

Założenie dokładności obliczeń zadane przez użytkownika zakłada że dla kroku *h* błąd jest w proporcji:

Stąd możemy policzyć współczynnik jako:

Parametry dokładności dane przez użytkownika określamy natomiast jako estymatę błędu:

A w algorytmie wprowadzone zostały stałe określające dane dokładności jako:

- dokładność względna  
 - dokładność bezwzględna

Prezentowany w poniższych punktach wykresy estymaty zakładają wybór mniejszego modułu błędu, który analogicznie został wypracowany dla parametru alpha.

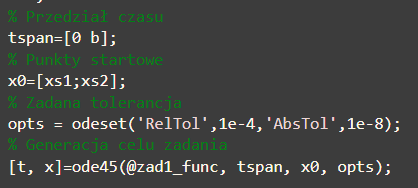
Dla układu 2 równań przyjmuje się współczynnik wyliczony dla najbardziej krytycznego równania, zatem wybieramy nowy współczynnik po każdej iteracji jako:

Realizacja algorytmu zakłada również wykorzystanie algorytmu z Rysunku 7.6/174 podręcznika do przedmiotu przebudowanego, lecz spełniającego warunki schematu blokowego.

**Implementacja**

W pliku ode.m zawarta została implementacja algorytmu ode45 celem rozwiązania równania. Metoda, aby obsłużyć układ równań różnicowych wymaga przekazania zewnętrznego uchwytu do metody, zatem byłem zmuszony uprościć solwer do przyjmowania jedynie parametrów startowych oraz oraz długości obserwowanego przedziału *b*.

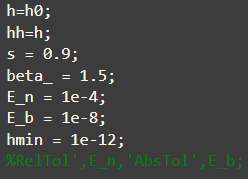
Reszta znaczących parametrów została wprowadzona jako stałe lokalne metody (Tolerancja Relatywna RelTol i Absolutna AbsTol są znane jest z poprzednich wzorów jako kolejno tolerancja względna i bezwzględna i domyślnie ustawiona na 1e-4 oraz 1e-8. Wpływ tolerancji powoduje poprawianie wyników i wpływa niekorzystnie na czas operacji algorytmu. Samo wywołanie funkcji jest intuicyjne i kod nie wymaga dalszych objaśnień:



Metoda zad1\_func zawiera generację naszego układu równań.

Plik rk4z.m zawiera implementacje algorytmu RK4 „Z” zmiennym krokiem z szacowaniem błędu metodą zdwajania kroku. Parametrami wejścia do algorytmu są dwa równania różniczkowe dane jako funkcje, współrzędne startowe układu oraz , a także parametr końca przedziału obserwacji i pierwotny krok . W solwerze zakładam, że solwer zawsze startuje z chwilą t=0.

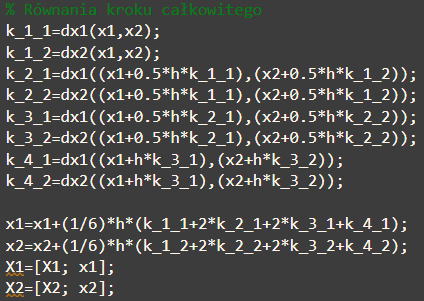
Lokalne parametry solwera wylistowane są poniżej:



*h –* krok algorytmu zmieniany zgodnie ze schematem  
*hh –* wektor kroków algorytmu  
s – współczynnik bezpieczeństwa używany przy obliczaniu *h\_new*, czyli proponowanych wartości kroku na kolejnych iteracjach zgodnie ze wzorem:

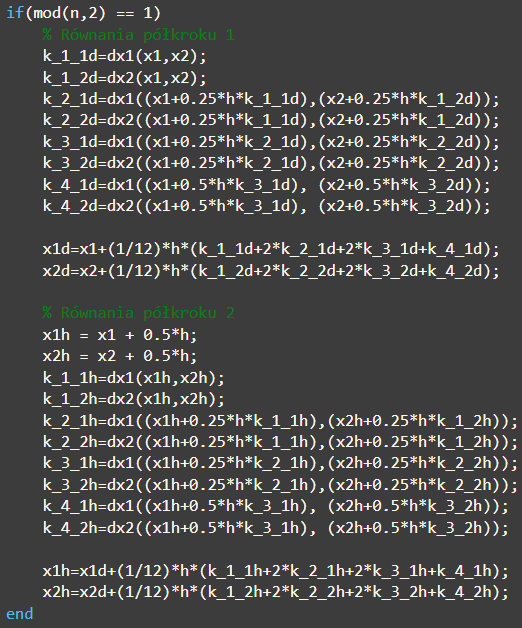
*beta\_* - heurystyczne ograniczenie maksymalnego wzrostu długości kroku w jednej iteracji  
 – minimalny krok algorytmu, którego wpływ zakłada poprawianie wyników operacji kalkulacji następnych punktów.

Równanie kroku całkowitego zakłada operacje podane w algorytmie z 1p. niniejszego sprawozdania:

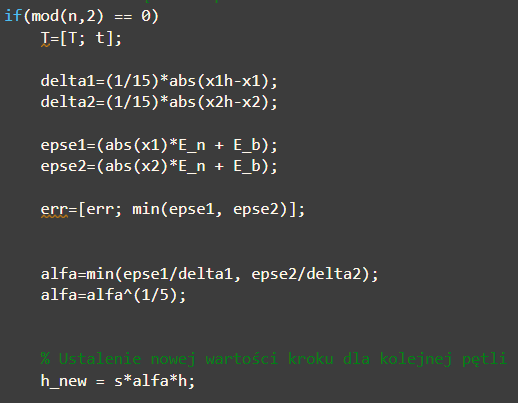


Nowe rezultaty zostają również dopisane do wektorów X1 oraz X2, celem prezentacji danych.

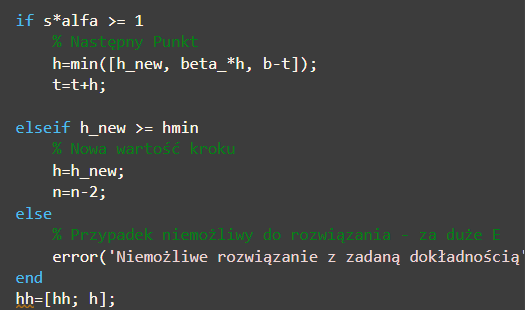
Równania półkroku zakładają wykonanie dwóch półkroków o długości po sobie, zgodnie z algorytmem:



Na podstawie porównania obu wyznaczonych punktów zostają wyznaczane błędy oraz estymowany zostaje następny punkt jako h\_new, w przypadku, gdy sa>=1 możliwe jest że punkt nie jest najmniejszy z zakresu h\_new, beta\_\*h, b-t. Ograniczenia wynikają z tego, aby różnice nie narastały gwałtownie lecz stopniowo w przypadku prób poprawiania wyniku:

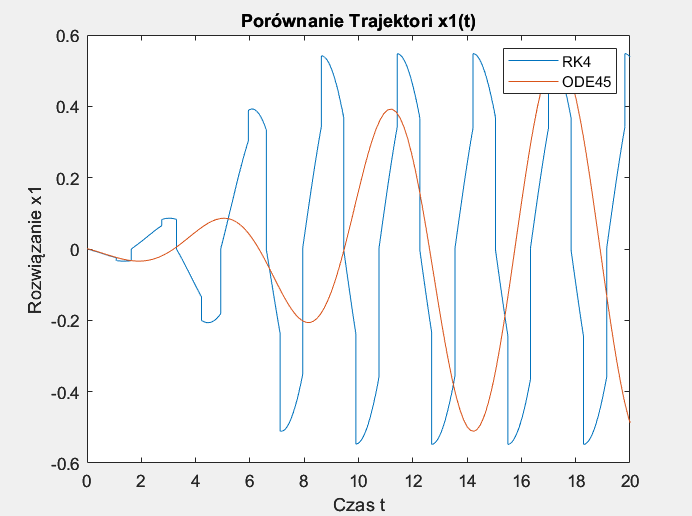


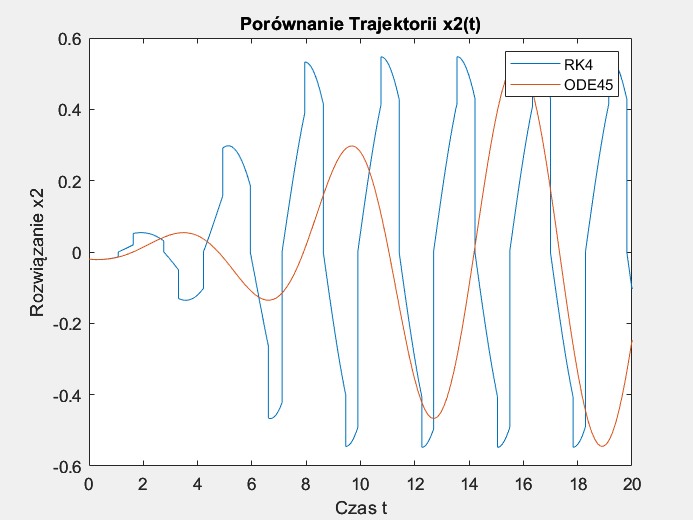
Poniżej znajduje się rozwiązanie schematu z Rys. 7.6 podręcznika.



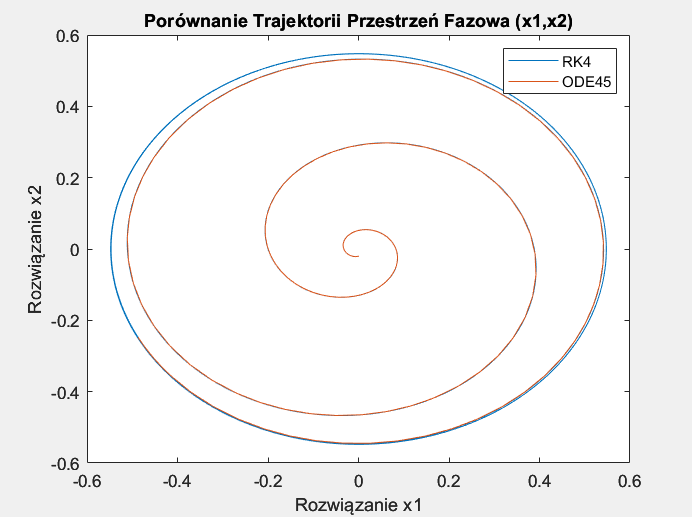
**Wykresy porównawcze**

W pliku comparison.m zawarte są wywołania obu algorytmów, porównanie czasów wywołania algorytmów oraz wykresy porównawcze trajektorii punktów:

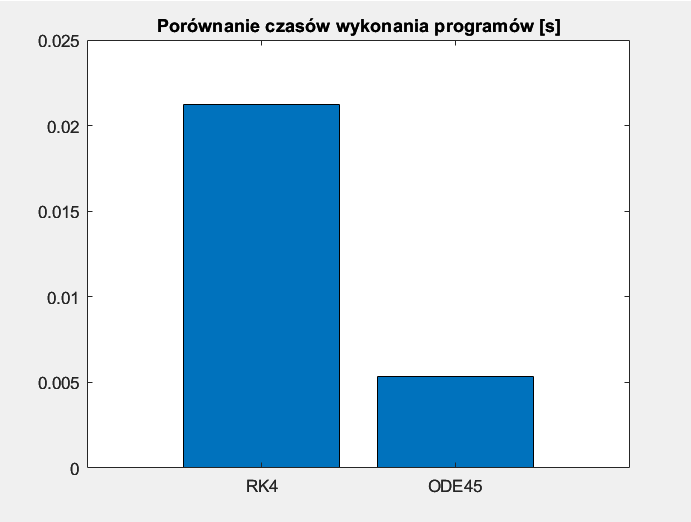




Powyższe wykresy niestety rozjeżdżają się względem siebie, nie jest to sytuacja pożądana. Zakładając, że popełniłem błędy w implementacji można zauważyć, że okres ruchu ciała w RK4 jest mniejszy, lecz Amplitudy z grubsza pokrywają się, lecz RK4 nie jest tak płynny – ma mniejszą dokładność.

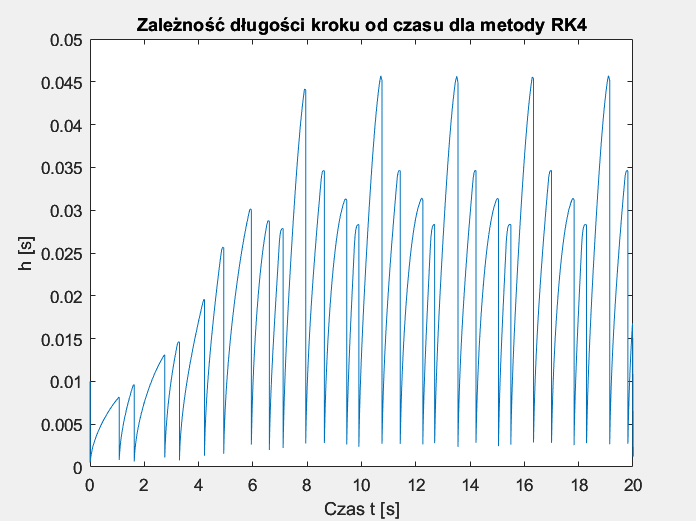


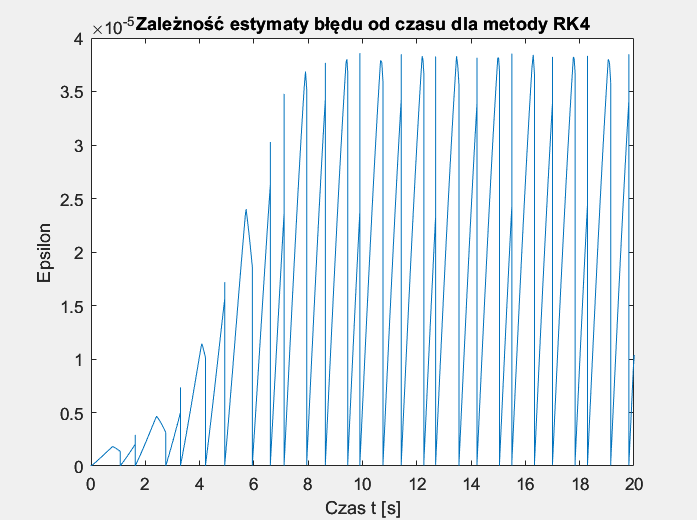
Mimo poprzednich błędów osiągamy zbliżone wykresy, lecz z naciskiem metody RK4 na krążenie po orbicie, więc błąd „przyspieszył” ruch punktu w danym obszarze.



Algorytm RK4 jest również wolniejszy około 4-krotnie od implementacji ODE45. Jednym z przyczyn tego zjawiska jest na pewno addytywne rozszerzanie listy w moim algorytmie zamiast stałej alokacji pamięci.

**Wykresy zależności dla metody RK4**





Jak widać metoda zakłada schodkowe zwiększanie długości kroków, aż do momentu przekroczenia błędu. Wynika to z wyborów następnego kroku regularnie zmienianego w algorytmie. Tendencja wykresu zgadza się z założeniami: im większy „skręt” punktu na wykresie fazowym, tym większy krok można przyjąć zachowując proporcje.

Zależność estymaty błędu również zgadza się z moimi założeniami – błąd narasta ze wzrostem iteracji. Od czasu w przedziale 8-20 mamy już sytuacje „krążenia po orbicie” znanego z wykresów – wykres jest wyprowadzony poprawnie lecz implementacja powoduje zbyt szybkie ustabilizowanie błędu na „stałopodobnym” poziomie.

**Krok początkowy, minimalny , Wartość dokładności względnej i bezwzględnej.**

Krok początkowy definiuje stopień przeskoku między punktami – chcąc uzyskać płynniejszy wykres w przestrzeni fazowej musimy zastosować mniejszy krok początkowy zwiększając przy tym znacznie nakład mocy obliczeniowej procesora na generacje licznych iteracji.

Krok minimalny definiuje natomiast dokładność wyników – program może zawiesić się, gdy przy próbie zwiększania dokładności zmniejszamy nazbyt znacznie krok. Krok posiada korelacje ze współczynnikami dokładności względnej i bezwzględnej.