Estymacja parametrów modelu

Autor: Wiktor Murawski

Przedmiot: Modelowanie matematyczne
Prowadzący: dr inż. Jakub Wagner

Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych

Oświadczam, że niniejsza praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu Modelowanie matematyczne, została wykonana przeze mnie samodzielnie.

Warszawa
27 grudnia 2024

Spis treści

1 Lista Symboli i Akronimów

URRZ	układ równań różniczkowych zwyczajnych
t	czas
$x_k(t)$ i $y_k(t)$	współrzędne położenia k-tego obiektu w chwili t
m	masa obiektu
G	stała grawitacyjna
$r_{jk}(t)$	odległość pomiędzy obiektami j i k dla $j, k = 1, 2, 3$ w chwili t.

2 Wprowadzenie

Dane zawarte w pliku `data_30.csv` reprezentują wyniki pomiaru położenia trzech obiektów o identycznych masach m, przyciągających się grawitacyjnie. Trajektorie ruchu tych obiektów opisane są następującym układem nieliniowych równań różniczkowych zwyczajnych drugiego rzędu:

$$\begin{cases} \frac{d^2x_1(t)}{dt^2} = Gm\Big(\frac{x_2(t)-x_1(t)}{r_{12}^3(t)} + \frac{x_3(t)-x_1(t)}{r_{13}^3(t)}\Big) \\ \frac{d^2y_1(t)}{dt^2} = Gm\Big(\frac{y_2(t)-y_1(t)}{r_{12}^3(t)} + \frac{y_3(t)-y_1(t)}{r_{13}^3(t)}\Big) \\ \frac{d^2x_2(t)}{dt^2} = Gm\Big(\frac{x_3(t)-x_2(t)}{r_{23}^3(t)} + \frac{x_1(t)-x_2(t)}{r_{12}^3(t)}\Big) \\ \frac{d^2y_2(t)}{dt^2} = Gm\Big(\frac{y_3(t)-y_2(t)}{r_{23}^3(t)} + \frac{y_1(t)-y_2(t)}{r_{12}^3(t)}\Big) \\ \frac{d^2x_3(t)}{dt^2} = Gm\Big(\frac{x_1(t)-x_3(t)}{r_{13}^3(t)} + \frac{x_2(t)-x_3(t)}{r_{23}^3(t)}\Big) \\ \frac{d^2y_3(t)}{dt^2} = Gm\Big(\frac{y_1(t)-y_3(t)}{r_{13}^3(t)} + \frac{y_2(t)-y_3(t)}{r_{23}^3(t)}\Big) \end{cases} \tag{1}$$

gdzie:

- t oznacza czas,
- $x_k(t)$ i $y_k(t)$ to współrzędne położenia k-tego obiektu dla $k = 1, 2, 3$,
- m to masa obiektu,
- G to stała grawitacyjna,
- $r_{jk}(t) \equiv \sqrt{[x_k(t) - x_j(t)]^2 + [y_k(t) - y_j(t)]^2}$ dla $j, k = 1, 2, 3$.

Wyniki pomiaru położenia są zaburzone oraz nieznana jest masa m obiektów. Wyznaczono przybliżone współrzędne położeń początkowych obiektów oraz przybliżenie masy m, a następnie rozwiązano układ (1) w celu wyznaczenia współrzędnych położenia obiektów w chwilach t zapisanych w pliku `query_30.csv`.

3 Metodyka i Wyniki Doświadczeń

3.1 Przekształcenie URRZ

W celu rozwiązania URRZ (1) drugiego rzędu za pomocą procedury `ode45`, przekształcono układ do URRZ rzędu pierwszego:

$$\begin{cases} \dfrac{dx_1(t)}{dt} = v_{x1} \\ \dfrac{dy_1(t)}{dt} = v_{y1} \\ \dfrac{dx_2(t)}{dt} = v_{x2} \\ \dfrac{dy_2(t)}{dt} = v_{y2} \\ \dfrac{dx_3(t)}{dt} = v_{x3} \\ \dfrac{dy_3(t)}{dt} = v_{y3} \\ \dfrac{dv_{x1}(t)}{dt} = Gm\Big(\dfrac{x_2(t) - x_1(t)}{r_{12}^3(t)} + \dfrac{x_3(t) - x_1(t)}{r_{13}^3(t)}\Big) \\ \dfrac{dv_{y1}(t)}{dt} = Gm\Big(\dfrac{y_2(t) - y_1(t)}{r_{12}^3(t)} + \dfrac{y_3(t) - y_1(t)}{r_{13}^3(t)}\Big) \\ \dfrac{dv_{x2}(t)}{dt} = Gm\Big(\dfrac{x_3(t) - x_2(t)}{r_{23}^3(t)} + \dfrac{x_1(t) - x_2(t)}{r_{12}^3(t)}\Big) \\ \dfrac{dv_{y2}(t)}{dt} = Gm\Big(\dfrac{y_3(t) - y_2(t)}{r_{23}^3(t)} + \dfrac{y_1(t) - y_2(t)}{r_{12}^3(t)}\Big) \\ \dfrac{dv_{x3}(t)}{dt} = Gm\Big(\dfrac{x_1(t) - x_3(t)}{r_{13}^3(t)} + \dfrac{x_2(t) - x_3(t)}{r_{23}^3(t)}\Big) \\ \dfrac{dv_{y3}(t)}{dt} = Gm\Big(\dfrac{y_1(t) - y_3(t)}{r_{13}^3(t)} + \dfrac{y_2(t) - y_3(t)}{r_{23}^3(t)}\Big) \end{cases} \tag{2}$$

3.2 Wyznaczenie warunków początkowych położenia i ich pochodnych

Z pliku `data_30.csv` odczytano wyniki pomiaru położeń trzech obiektów. Wyznaczono przybliżone wartości pochodnych $x'_k(t_0)$ i $y'_k(t_0)$ dla $k = 1, 2, 3$ korzystając ze wzoru różnicy w przód:

$$x'_0 = \frac{x_1 - x_0}{t_1 - t_0}$$

Uzyskano następujące warunki początkowe położeń i ich pochodnych:

x_1	y_1	x_2	y_2	x_3	y_3
-0.554795098	0.311684711	0.231422269	-0.376185606	0.322675278	0.0579379401
v_{x1}	v_{y1}	v_{x2}	v_{y2}	v_{x3}	v_{y3}
-0.255664946	-0.557361492	0.778963645	0.481299004	-0.399831836	0.250095091

Tabela 1: Wartości x_k i y_k oraz ich pochodne v_{xk} i v_{yk} dla $t = t_0$ zaokrąglone do 9 cyfr znaczących

3.3 Aproksymacja masy

W celu znalezienia wartości masy m odpowiednio bliskiej faktycznej masie obiektów przekształcono (1) następująco:

$$Gm = \frac{d^2x_1(t)}{dt^2} \frac{1}{\dfrac{x_2(t) - x_1(t)}{r_{12}^3(t)} + \dfrac{x_3(t) - x_1(t)}{r_{13}^3(t)}}$$

Analogicznie dla y_1, x_2, y_2, x_3, y_3. Wartość drugiej pochodnej wyznaczono korzystając ze wzoru różnicy centralnej:

$$x'_n = \frac{x_{n+1} - x_{n-1}}{t_{n+1} - t_{n-1}}$$
$$x''_n = \frac{x'_{n+1} - x'_{n-1}}{t_{n+1} - t_{n-1}}$$

W ten sposób, dla różnych n, wyznaczono różne wartości mające być przybliżeniem iloczynu Gm. Przez względnie dużą niedokładność danych pomiarowych przybliżenia były błędne (na przykład ujemne bądź bardzo odchylone od reszty), w szczególności te na końcach przedziału. W celu zminimalizowania wpływu błędów pomiarów dane wygładzono korzystając ze średniej ruchomej. Kilka skrajnych wartości z danych zignorowano. Następnie sprawdzono, dla której współrzędnej wyznaczone iloczyny Gm są najmniej oddalone od średniej.
Okazało się, że najlepsze przybliżenie początkowe daje wzór na Gm wykorzystujący drugą pochodną x_1. Otrzymano $Gm = 0.361115456$ (dokładność do 9 cyfr znaczących).

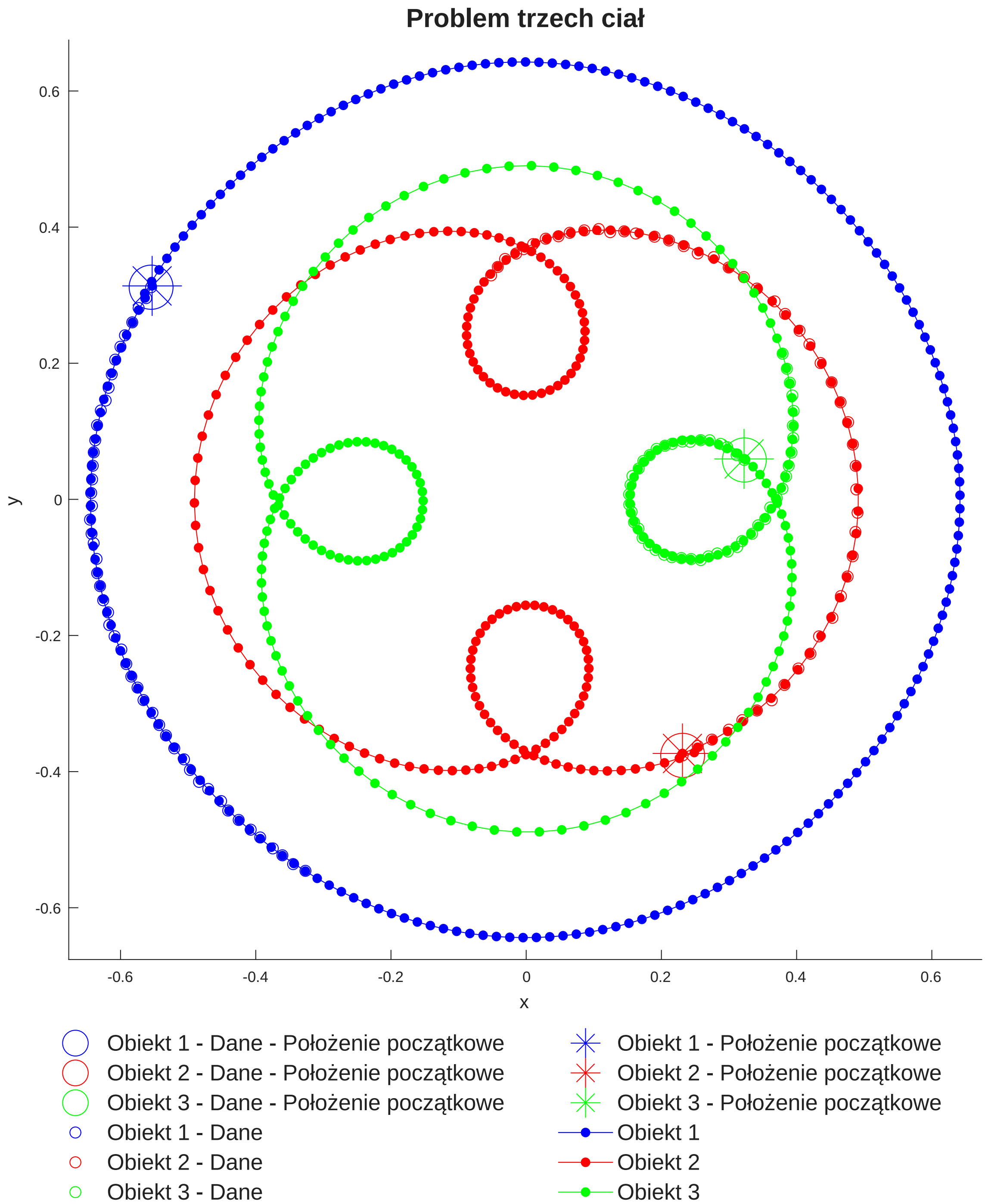
3.4 Dopasowanie parametrów poprzez minimalizację funkcji błędu

Funkcję błędu zdefiniowano jako funkcję przekształcającą 13-elementowy wektor $\mathbf{p} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & x_3 & y_3 & v_{x1} & v_{y1} & v_{x2} & v_{y2} & v_{x3} & v_{y3} & Gm \end{bmatrix}^T$ zawierający warunki początkowe, na pojedynczą wartość błędu średniokwadratowego, obliczanego na podstawie różnic między danymi pomiarowymi a wyznaczonymi estymatami rozwiązania na podstawie parametrów początkowych $\mathbf{p}$.
Do znalezienia minimum funkcji błędu wykorzystano wbudowaną w środowisko `MATLAB` funkcję `fminsearch` warunkami początkowymi p_0. Funkcja zwróciła wektor warunków początkowych $\mathbf{p}_{\mathbf{min}}$, dla których wartość błędu powinna być minimalna.

	$\mathbf{p}_0$	$\mathbf{p}_{\min}$
x_1	-0.554795097997622	-0.553263048869668
y_1	0.311684711285866	0.313624244608031
x_2	0.231422269108973	0.231127385195129
y_2	-0.376185606022192	-0.373336032975038
x_3	0.322675278263600	0.322262184381840
y_3	0.057937940057362	0.059426057383732
v_{x1}	-0.255664945942007	-0.369138275745365
v_{y1}	-0.557361491961784	-0.610833886439908
v_{x2}	0.778963645043103	0.778373447699517
v_{y2}	0.481289003900639	0.287622684238942
v_{x3}	-0.399831835911587	-0.409745525949235
v_{y3}	0.250095091068599	0.322932407732306
Gm	0.361115455784322	0.361230297281287

3.5 Rozwiązanie URRZ z wyznaczonymi parametrami

Do rozwiązania URRZ (2) wykorzystano wbudowaną w środowisko `MATLAB` funkcję `ode45` z warunkami początkowymi $\mathbf{p_{min}}$. Wyznaczono położenia wszystkich trzech ciał w momentach danych w pliku `query_30.csv`. Trajektorie trzech obiektów przedstawiono na rysunku 1.



Rysunek 1: Wyznaczone estymaty orbit obiektów

Dla obliczonych estymatów położeń, zawartą w pliku `test_solution_30.p` funkcją `test_solution_30`, obliczono wskaźnik dokładności rozwiązania $\Delta = 0.0169 < 0.02$.

4 Dyskusja Wyników i Wnioski

Problem trzech ciał, będący jednym z klasycznych zagadnień mechaniki nieba oraz teorii chaosu, nie ma ogólnego rozwiązania analitycznego. Z tego względu, rozwiązania tego problemu są często badane numerycznie, z czego wynikają liczne problemy, na przykład dotyczące dokładności danych wejściowych. W problemie trzech ciał, jakość danych pomiarowych ma kluczowe znaczenie dla dokładności uzyskiwanych wyników. Nawet niewielkie błędy w danych pomiarowych mogą prowadzić do znaczących odchyleń w trajektoriach obiektów. W związku z tym, proces optymalizacji parametrów początkowych miał kluczowe znaczenie dla uzyskania precyzyjnych wyników.

Podczas dopasowywania parametrów początkowych, kluczowym było wyznaczenie wartości iloczynu Gm wystarczająco bliskiej wartości faktycznej. Drobne zmiany w wartości Gm, na przykład zmiana wartości $Gm \approx 0.361$ na $Gm = 0.360$ lub $Gm = 0.362$ (różnica mniejsza niż 0.3%) sprawia, że rozwiązanie przestaje być zadowalające. Choć dane pomiarowe położeń zawierały błędy, funkcja `fminsearch`, używająca algorytmu korzystającego z metody numerycznej Neldera-Meada do wyznaczania ekstremum, pozwoliła na uzyskanie zadowalających warunków początkowych.

Zastosowanie funkcji `ode45`, korzystającej z jawnej metody Rungego-Kutty 4-go rzędu z poprawką (metoda Dormanda-Prince'a) z adaptacyjnym krokiem, do rozwiązania układu równań różniczkowych okazało się stabilne i skuteczne, nawet w kontekście chaotycznego charakteru problemu trzech ciał. Dzięki temu możliwe było uzyskanie zadowalająco dokładnych trajektorii obiektów w przyszłych momentach czasowych, wykraczających poza zakres danych pomiarowych.

Uzyskane rozwiązanie jest jednym z periodycznych rozwiązań problemu trzech ciał. W układzie trzech ciał, periodyczność oznacza, że po pewnym czasie obiekty wracają do swoich początkowych pozycji i prędkości, co prowadzi do cyklicznego powtarzania się trajektorii. Chociaż układ trzech ciał jest w ogólności chaotyczny i wrażliwy na początkowe warunki, istnieją szczególne przypadki, w których rozwiązanie jest periodyczne. Takie trajektorie są stabilne w sensie, że ich kształt i zachowanie powtarzają się w równych odstępach czasu.

Bibliografia

1. Dokumentacja MATLAB: `https://www.mathworks.com/help/matlab`.
2. `https://en.wikipedia.org/wiki/Three-body_problem`

Listing Programów

plik: Projekt2.m

```
function result = Projekt2()
  % Wczytanie danych pomiarowych
  data = readtable("data_30.csv");
  t_data = data.t;
  x1 = data.x1; y1 = data.y1;
  x2 = data.x2; y2 = data.y2;
  x3 = data.x3; y3 = data.y3;
  data = [x1, y1, x2, y2, x3, y3];

  % Wczytanie wartości czasu z zapytania
  data2 = readtable("query_30.csv");
  t_query = data2.t;

  % Przybliżenie początkowych wartości pochodnych
  dx1 = ApproximateDerivative(t_query(1:2), x1(1:2));
  dy1 = ApproximateDerivative(t_query(1:2), y1(1:2));
  dx2 = ApproximateDerivative(t_query(1:2), x2(1:2));
  dy2 = ApproximateDerivative(t_query(1:2), y2(1:2));
  dx3 = ApproximateDerivative(t_query(1:2), x3(1:2));
  dy3 = ApproximateDerivative(t_query(1:2), y3(1:2));

  % Przybliżenie początkowe masy; Gm = 0.361115455784322 -> delta = 0.0169
  Gm = ApproximateMass(t_data, x1, x2, x3, y1, y2, y3);

  % Warunki początkowe
  p0 = [ x1(1);  y1(1);  x2(1);  y2(1);  x3(1);  y3(1);
        dx1(1); dy1(1); dx2(1); dy2(1); dx3(1); dy3(1); Gm];

  % Minimalizacja
  opts = optimset('TolX', 1e-6, 'TolFun', 1e-6, ...
                  'MaxIter', 1e4, 'MaxFunEvals', 1e4);
  pmin = fminsearch(@(p) Criterium(p, t_data, data), p0, opts);

  % Zdefiniowanie rozwiązywanego URRZ
  odefun = @(t, z) ODEFunction(t, z, Gm);

  % Rozwiązanie URRZ za pomocą ode45
  opts = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);
  [~, z] = ode45(odefun, t_query, pmin(1:12), opts);

  % Przypisanie estymatów wartości x1, y1, x2, y2, x3, y3 w danych chwilach
  x1 = z(:, 1); y1 = z(:, 2);
  x2 = z(:, 3); y2 = z(:, 4);
  x3 = z(:, 5); y3 = z(:, 6);
  result = [x1, y1, x2, y2, x3, y3];

  % Wizualizacja orbit
  Visualize(data, result);

  % Wyświetlenie wyniku testu dokładności
  test_solution_30(x1, y1, x2, y2, x3, y3);

end % function
```

plik: `ApproximateDerivative.m`

```
function dy = ApproximateDerivative(t, y)
  N = length(y);
  dy = zeros(N, 1);

  dy(1) = (y(2) - y(1)) / (t(2) - t(1)); % Różnica wprzód
  for n = 2:N-1
    dy(n) = (y(n+1) - y(n-1)) / (t(n+1) - t(n-1)); % Różnica środkowa
  end
  dy(N) = (y(N) - y(N-1)) / (t(N) - t(N-1)); % Różnica wstecz

end % function
```

plik: `ODEFunction.m`

```
function dzdt = ODEFunction(~, z, Gm)
    x1 = z(1); y1 = z(2);
    x2 = z(3); y2 = z(4);
    x3 = z(5); y3 = z(6);
    dx1 = z(7);  dy1 = z(8);
    dx2 = z(9);  dy2 = z(10);
    dx3 = z(11); dy3 = z(12);

    r12 = sqrt((x1 - x2)^2 + (y1 - y2)^2);
    r13 = sqrt((x1 - x3)^2 + (y1 - y3)^2);
    r23 = sqrt((x2 - x3)^2 + (y2 - y3)^2);

    ddx1 = Gm * ((x2 - x1)/r12^3 + (x3 - x1)/r13^3);
    ddy1 = Gm * ((y2 - y1)/r12^3 + (y3 - y1)/r13^3);
    ddx2 = Gm * ((x3 - x2)/r23^3 + (x1 - x2)/r12^3);
    ddy2 = Gm * ((y3 - y2)/r23^3 + (y1 - y2)/r12^3);
    ddx3 = Gm * ((x1 - x3)/r13^3 + (x2 - x3)/r23^3);
    ddy3 = Gm * ((y1 - y3)/r13^3 + (y2 - y3)/r23^3);

    dzdt = [ dx1;  dy1;  dx2;  dy2;  dx3;  dy3;
            ddx1; ddy1; ddx2; ddy2; ddx3; ddy3];

end % function
```

plik: `Criterium.m`

```
function J = Criterium(p, t, data)
  Gm = p(13);

  % Zdefiniowanie rozwiązywanego URRZ
  odefun = @(t, z) ODEFunction(t, z, Gm);

  % Rozwiązanie URRZ za pomocą ode45
  opts = odeset('RelTol', 1e-9, 'AbsTol', 1e-9);
  [~, z] = ode45(odefun, t, p(1:12), opts);

  % Obliczenie błędu średniokwadratowego
  J = sum((z(:, 1:6) - data(:, 1:6)).^2, "all");

end % function
```

plik: ApproximateMass.m

```
function m = ApproximateMass(t, x1, x2, x3, y1, y2, y3)
  N = length(t);

  % Wygładzanie zaburzonych danych
  window_size = 3;
  for i = 1:5
    x1 = movmean(x1, window_size);
    x2 = movmean(x2, window_size);
    x3 = movmean(x3, window_size);
    y1 = movmean(y1, window_size);
    y2 = movmean(y2, window_size);
    y3 = movmean(y3, window_size);
  end

  % Przybliżanie wartości pierwszej pochodnej
  dx1 = ApproximateDerivative(t, x1);
  dy1 = ApproximateDerivative(t, y1);
  dx2 = ApproximateDerivative(t, x2);
  dy2 = ApproximateDerivative(t, y2);
  dx3 = ApproximateDerivative(t, x3);
  dy3 = ApproximateDerivative(t, y3);


  % Przybliżanie wartości drugiej pochodnej
  ddx1 = ApproximateDerivative(t, dx1);
  ddy1 = ApproximateDerivative(t, dy1);
  ddx2 = ApproximateDerivative(t, dx2);
  ddy2 = ApproximateDerivative(t, dy2);
  ddx3 = ApproximateDerivative(t, dx3);
  ddy3 = ApproximateDerivative(t, dy3);

  % Wyznaczenie macierzy przybliżonych wartości masy
  m_mat = zeros(N, 6);
  for i = 1:N
    r12 = sqrt((x1(i)-x2(i))^2 + (y1(i)-y2(i))^2);
    r13 = sqrt((x1(i)-x3(i))^2 + (y1(i)-y3(i))^2);
    r23 = sqrt((x2(i)-x3(i))^2 + (y2(i)-y3(i))^2);
    m_mat(i, 1) = ddx1(i)/((x2(i) - x1(i))/r12^3 + (x3(i) - x1(i))/r13^3);
    m_mat(i, 2) = ddy1(i)/((y2(i) - y1(i))/r12^3 + (y3(i) - y1(i))/r13^3);
    m_mat(i, 3) = ddx2(i)/((x3(i) - x2(i))/r23^3 + (x1(i) - x2(i))/r12^3);
    m_mat(i, 4) = ddy2(i)/((y3(i) - y2(i))/r23^3 + (y1(i) - y2(i))/r12^3);
    m_mat(i, 5) = ddx3(i)/((x1(i) - x3(i))/r13^3 + (x2(i) - x3(i))/r23^3);
    m_mat(i, 6) = ddy3(i)/((y1(i) - y3(i))/r13^3 + (y2(i) - y3(i))/r23^3);
  end

  % Zignorowanie wartości ze skrajów przedziału dla każdej zmiennej
  m_mat = abs(m_mat(10:N-9, 1:6));

  % Obliczenie wskaźnika odchylenia od średniej wartości wartości masy
  % w kolumnie dla każdej kolumny
  mean_vals = zeros(1, 6);
  deviation = zeros(1, 6);
  for i = 1:6
    mean_vals(i) = median(m_mat(:, i));
    for j = 1:length(m_mat(:, i))
      deviation(i) = abs(sum(m_mat(:, i) - mean_vals(i)));
    end
```

```
  end

  % Wybranie kolumny z wartościami o najmniejszym odchyleniu
  [~, best_idx] = min(deviation);

  % Przypisanie do wartości zwracanej średniej arytm. z najlepszej kolumny
  m = mean(m_mat(:, best_idx));

end % function
```

plik: `Visualize.m`

```
function [] = Visualize(data, calc)
  x1_data = data(:, 1); y1_data = data(:, 2);
  x2_data = data(:, 3); y2_data = data(:, 4);
  x3_data = data(:, 5); y3_data = data(:, 6);
  x1_calc = calc(:, 1); y1_calc = calc(:, 2);
  x2_calc = calc(:, 3); y2_calc = calc(:, 4);
  x3_calc = calc(:, 5); y3_calc = calc(:, 6);

  c1 = "#0000FF";
  c2 = "#FF0000";
  c3 = "#00FF00";
  figure(Name='Projekt2_30'); clf; hold on;
  title('Problem trzech ciał', FontSize=16);
  xlabel('x'); ylabel('y');
  axis equal;
  maxy = max(max([y1_calc, y2_calc, y3_calc]));
  miny = min(min([y1_calc, y2_calc, y3_calc]));
  maxx = max(max([x1_calc, x2_calc, x3_calc]));
  minx = min(min([x1_calc, x2_calc, x3_calc]));
  a = 1.05;
  ylim([a*miny, a*maxy]);  xlim([a*minx, a*maxx]);
  %set(gca, "Color", 'k')

  % Wyświetl położenia początkowe z danych
  opts = {'LineStyle', 'none', 'Marker', 'o', 'MarkerSize', 24};
  plot(x1_data(1), y1_data(1), opts{:}, Color=c1, ...
    DisplayName='Obiekt 1 - Dane - Położenie początkowe');
  plot(x2_data(1), y2_data(1), opts{:}, Color=c2, ...
    DisplayName='Obiekt 2 - Dane - Położenie początkowe');
  plot(x3_data(1), y3_data(1), opts{:}, Color=c3, ...
    DisplayName='Obiekt 3 - Dane - Położenie początkowe');

  % Wyświetl położenia z danych
  opts = {'LineStyle', 'none', 'Marker', 'o', 'MarkerSize', 6};
  plot(x1_data, y1_data, opts{:}, Color=c1, ...
    DisplayName='Obiekt 1 - Dane');
  plot(x2_data, y2_data, opts{:}, Color=c2, ...
    DisplayName='Obiekt 2 - Dane');
  plot(x3_data, y3_data, opts{:}, Color=c3, ...
    DisplayName='Obiekt 3 - Dane');

  % Wyświetl wyznaczone położenia początkowe
  opts = {'LineStyle', 'none', 'Marker', '*', 'MarkerSize', 30};
  plot(x1_calc(1), y1_calc(1), opts{:}, Color=c1, ...
    DisplayName='Obiekt 1 - Położenie początkowe');
```

```
  plot(x2_calc(1), y2_calc(1), opts{:}, Color=c2, ...
    DisplayName='Obiekt 2 - Położenie początkowe');
  plot(x3_calc(1), y3_calc(1), opts{:}, Color=c3, ...
    DisplayName='Obiekt 3 - Położenie początkowe');

  % Wyświetl wyznaczone położenia
  opts = {'LineStyle', '-', 'Marker', '.', 'MarkerSize', 16};
  plot(x1_calc, y1_calc, opts{:}, Color=c1, ...
    DisplayName='Obiekt 1');
  plot(x2_calc, y2_calc, opts{:}, Color=c2, ...
    DisplayName='Obiekt 2');
  plot(x3_calc, y3_calc, opts{:}, Color=c3, ...
    DisplayName='Obiekt 3');

  legend('show', Color='#FFFFFF', Location='southoutside', ...
    NumColumns=2, fontsize=14, box='off');

end % function
```