Національний технічний університет України
"Київський політехнічний інститут"

Кафедра приладів і систем орієнтації і навігації

Ю. Ф. Лазарєв

Прикладні методи дослідження алгоритмів систем орієнтації

методичний посібник

Київ – НТУУ "КПІ" – 2015

Зміст

Вступ

Кредитний модуль "Прикладні методи теоретичного аналізу систем орієнтації" призначений для навчання магістрантів і є частиною навчальної дисципліни "Основи теорії чутливих елементів систем орієнтації".

Метою цього посібника є надання викладачеві та студенту теоретичний і практичний матеріал, який забезпечив би поглиблену прикладну підготовку майбутніх магістрів у галузі проектування приладів і систем автономної орієнтації.

1. Визначення кутового положення тіла у просторі

Існують кілька можливостей математичного подання кутового положення тіла у просторі [1, с 16-41]:

1) за допомогою кутів Ейлера-Крилова, тобто трійки значень кутів повороту навколо трьох координатних осей;
2) матрицею напрямних косинусів між осями вихідної системи координат і системи координат, пов'язаної з тілом;
3) параметрами вектора Ейлера єдиного повороту;
4) кватерніоном повороту від вихідної системи координат до кінцевої.

Розглянемо взаємозв'язок між цими видами визначення орієнтації тіла і побудуємо комп'ютерні засоби, що забезпечують перехід від одного виду до інших

1.1. Кути Ейлера-Крилова і матриця напрямних косинусів

За основу приймемо послідовність кутів Ейлера-Крилова, наведену на рис. 1.1

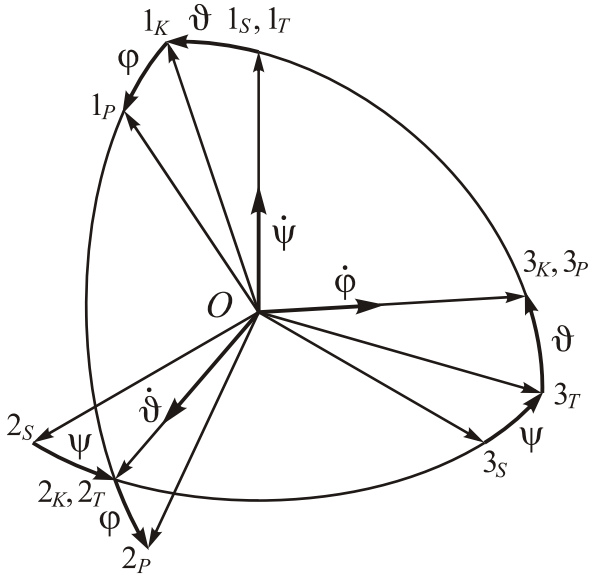


Рис. 1.1. Послідовність поворотів на кути Ейлера-Крилова

Відповідна матриця напрямних косинусів переходу від вихідної системи координат S до кінцевої P має такий вигляд:

C^{SP}	1_P	2_P	3_P
1_S	$\cos\vartheta\cos\varphi$	$-\cos\vartheta\sin\varphi$	$\sin\vartheta$
2_S	$\cos\psi\sin\varphi+\sin\psi\cos\varphi\sin\vartheta$	$\cos\psi\cos\varphi-\sin\psi\sin\varphi\sin\vartheta$	$-\sin\psi\cos\vartheta$
3_S	$\sin\psi\sin\varphi-\cos\psi\cos\varphi\sin\vartheta$	$\sin\psi\cos\varphi+\cos\psi\sin\varphi\sin\vartheta$	$\cos\psi\cos\vartheta$

Складемо універсальну програму, яка обчислює елементи матриці напрямних косинусів за відомими значеннями кутів поворотів навколо координатних осей, номера яких вказано у векторі k. Для цього спочатку створимо процедуру mnke обчислення елементарної матриці напрямних косинусів одного повороту навколо вісі, номер якої задано:

```
function y=mnke(x,k)
% Функція формування Матриці Напрямних Косинусів (y)
% за заданим значення куту (x) та номера повороту (к)
% осі, навколо якої здійснюється поворот

% Лазарєв Ю.Ф.      18-07-1998р.    19-09-1998р.
y = zeros(3,3);  y(k,k) = 1;
if k==1
   y(2,2) = cos(x);    y(3,3) = y(2,2);
   y(3,2) = sin(x);    y(2,3) = - y(3,2);
end
if k==2
   y(1,1) = cos(x);    y(3,3) = y(1,1);
   y(1,3) = sin(x);    y(3,1) = - y(1,3);
end
if k==3
   y(1,1) = cos(x);    y(2,2) = y(1,1);
   y(2,1) = sin(x);    y(1,2) = - y(2,1);
end
```

Тепер універсальна програма обчислення МНК виглядатиме просто:

```
function y = ug2mnk(x,k)
% Процедура знаходження Матриці Напрямних Косинусів за кутами Ейлера
%       y = ug2mnk(x,k)
% обчислює матрицю напрямних косинусів  'y'
% за заданими значеннями вектора 'x' трьох кутів x(1), x(2), x(3)
% послідовних  поворотів та вектора 'к' номерів осей к(1), к(2), к(3),
% навколо яких ці повороти здійснюються
%         Використовується процедура-функція  'mnke'

% (c) Лазарєв Ю.Ф.         18-07-1998р.     09-06-2014
y = mnke(x(1),k(1))*mnke(x(2),k(2))*mnke(x(3),k(3));
```

Наведемо приклад застосування цієї процедури. Нехай послідовність поворотів така, яка вказана на рис. 1.1. Тоді, очевидно, вектор номерів осей послідовних поворотів буде таким k=[1 2 3]. Задамо значення кутів послідовності $\psi = 0{,}4$; $\vartheta = -0{,}3$; $\varphi = 1$. При цьому вектор значень кутів повороту набуде вигляду x = [0.4 -0.3 1].

Складемо керувальну програму:

```
x = [0.4 -0.3 1];  k=[1 2 3];
C = ug2mnk(x,k)
```

В результаті отримаємо матрицю напрямних косинусів у вигляді

```
C =
  5.1617e-001 -8.0389e-001 -2.9552e-001
  7.1287e-001  5.9449e-001 -3.7203e-001
  4.7475e-001 -1.8638e-002  8.7992e-001
```

Задля перевірки правильності результату потрібно утворити процедуру зворотного переходу від заданої МНК до значень кутів послідовних поворотів навколо заданих (номерами) осей. Текст її наведено нижче

```
function y= mnk2ug(x,kv)
%  MNK2UG перетворює матрицю напрямних косинусів у кути Ейлера
% Звернення:
%   y = mnk2ug(x,kv)
%  дозволяє обчислити вектор "y", що складається зі
```

```
%  значень y(1),y(2),y(3) кутів послідовних поворотів
%  навколо координатних осей відповідно з номерами kv(1),
%  kv(2) і kv(3) за заданою матрицею напрямних косинусів "x"
%  розміром (3*3) між вісями початкової та кінцевої
%  систем координат
%       x(i,j) = cos(Iп,Jк),
%   де Iп - позначення осі з номером "i" початкової,
%   а Jк - осі з номером "j" кінцевої систем координат.
% Примітка:
%   kv(1) не повинно дорівнювати kv(2), а kv(2) не повинно
%       дорівнювати kv(3)

%   Ю.Ф.Лазарєв,        Останні змінювання 5-12-2001
k1=kv(1);   k2=kv(2);   k3=kv(3);
k = k1*10^2+k2*10+k3;
if k==123 | k==312 | k==231,
  y(1) = atan2(-x(k2,k3),x(k3,k3));          y(2) = asin(x(k1,k3));
  y(3) = atan2(-x(k1,k2),x(k1,k1));
  sp= x(k1,k3);    cp=sqrt(1-x(k1,k3)^2);
  st=sin(y(3));     ct=cos(y(3));
  if (x(k1,k1)*ct)<0
       y(1) = atan2(x(k2,k3),-x(k3,k3));      y(2) = atan2(sp,-cp);
       y(3) = atan2(x(k1,k2),-x(k1,k1));
  end
elseif  k==132 | k==321 | k==213,
   y(2) = asin(-x(k1,k3));                y(1) = atan2(x(k2,k3),x(k3,k3));
   y(3) = atan2(x(k1,k2),x(k1,k1));
else y(2) = acos(x(k1,k3));
   if  k==121,
      y(1) = atan2(x(2,1),-x(3,1));              y(3) = atan2(x(1,2),x(1,3));
   elseif  k==313,
      y(1) = atan2(x(1,3),-x(1,2));              y(3) = atan2(x(3,1),x(3,2));
   elseif  k==232,
      y(1) = atan2(x(3,2),-x(1,2));              y(3) = atan2(x(2,3),x(2,1));
   elseif  k==212,
      y(1) = atan2(x(1,2),x(3,2));       y(3) = atan2(x(2,1),-x(2,3));
   elseif  k==323,
      y(1) = atan2(x(2,3),x(1,3));       y(3) = atan2(x(3,2),-x(3,1));
   elseif  k==131,
      y(1) = atan2(x(3,1),x(2,1));       y(3) = atan2(x(1,3),-x(1,2));
   end
end
```

Якщо звернутися до цієї процедури

Ug=mnk2ug(C,k),

то отримаємо

Ug = 0.4 -0.3 1

Як бачимо, виходить той самий вектор кутів повороту, що й вихідний. Отже, створені програмні засоби працюють добре.

Якщо система координат S спочатку за допомогою поворотів переходить у систему координат T, а потім шляхом інших поворотів – в систему координат P, то операцію знаходження параметрів повороту, які забезпечують поворот від системи S безпосередньо до системи P, прийнято називати операцією *складання поворотів*. Таку операцію неможливо виконати за допомогою використання лише послідовностей кутів повороту.

Переваги застосування послідовності трьох кутів – *наочність подання поворотів*

Недоліки застосування послідовності трьох кутів *– непристосованість до безпосереднього проеціювання векторів і не існує операції складання поворотів.*

***Переваги застосування матриці напрямних поворотів** – добра пристосованість до проеціювання векторів та існування простого правила складання поворотів*

$$C^{SP} = C^{ST} \cdot C^{TP}, \tag{1.1}$$

МНК результуючого повороту дорівнює добутку МНК складових поворотів, записаних зліва направо у порядку здійснення самих поворотів.

Недоліки застосування матриці напрямних поворотів – не наочність подання поворотів івелика кількість параметрів, що описують кожний поворот.

1.2. Вектор Ейлера єдиного повороту

Відповідно до теореми Ейлера довільне кутове положення однієї системи координат P відносно іншої системи координат S можна подати як єдиний поворот на кут δ (*кут Ейлера*) навколо певно орієнтованої осі (*осі Ейлера*). При цьому (див. [1, с. 28]) кут Ейлера доволі просто визначається з матриці $\mathbf{C}^{SP}$ напрямних косинусів за формулою

$$\cos\delta = \frac{1}{2}[Tr(\mathbf{C}^{SP}) - 1], \tag{1.2}$$

де $Tr(\mathbf{C}^{SP})$ – позначення сліду (trace) матриці напрямних косинусів, тобто суми її діагональних елементів. З самого визначення єдиного повороту випливає, що напрямні косинуси e_{1S}, e_{2S} і e_{3S} осі Ейлера відносно осей 1_S, 2_S, 3_S вихідної системи координат S дорівнюють напрямним косинусам e_{1P}, e_{2P} і e_{3P} цієї осі відносно кінцевої системи координат P, тому можна зняти позначення індексу системи координат і користуватися позначеннями

$$e_1 = e_{1S} = e_{1P}; \qquad e_2 = e_{2S} = e_{2P}; \qquad e_3 = e_{3S} = e_{3P}.$$

Ці напрямні косинуси визначаються співвідношеннями

$$e_1 = \frac{c_{32}^{SP} - c_{23}^{SP}}{2\sin\delta}; \qquad e_2 = \frac{c_{13}^{SP} - c_{31}^{SP}}{2\sin\delta}; \qquad e_3 = \frac{c_{21}^{SP} - c_{12}^{SP}}{2\sin\delta}. \tag{1.3}$$

Зворотний перехід від параметрів єдиного повороту Ейлера до матриці напрямних косинусів можна подати у вигляді

$$\mathbf{C}^{SP} = \mathbf{E} + \sin\delta \cdot (\mathbf{e}\times) + (1 - \cos\delta) \cdot (\mathbf{e}\times)^2, \tag{1.4}$$

де $(\mathbf{e}\times)$ є позначенням кососиметричної матриці

$$(\mathbf{e}\times) = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}. \tag{1.5}$$

Користуючись цими відомостями, створимо кілька комп'ютерних процедур, що здійснюють ці перетворення:

– перетворення матриці напрямних косинусів у параметри єдиного повороту:

```
function [d,e]=MNK2VEul(C)
% Процедура преобразования матрицы направляющих косинусов
% в вектор Эйлерового поворота
%         [d,e]=MNK2VEul(C)
% где d - величина угла поворота вокруг Эйлеровой оси,
% а e - вектор направляющих косинусов этой оси

% Лазарев Ю. Ф.       19-11-2014
cd=(trace(C)-1)/2;  d=acos(cd);
if d==0
   e=[1 0 0];
else
   zn=2*sin(d);
   e(1)=(C(3,2)-C(2,3))/zn;
```

```
    e(2)=(C(1,3)-C(3,1))/zn;
    e(3)=(C(2,1)-C(1,2))/zn;
end
```

– перетворення вектора Ейлера у матрицю напрямних косинусів

```
function C=VEul2MNK(d,e)
% Перетворення вектору Ейлера єдиного повороту у  матрицю напрямних
% косинусів

% Лазарев Ю.Ф.      19-11-2014      02-09-2015
if d==0
   e=[1 0 0];
end
E=eye(3,3);
cd=cos(d);  sd=sin(d);
ex=[0 -e(3) e(2); e(3) 0 -e(1);-e(2) e(1) 0];
C = E + sd*ex + (1-cd)*ex*ex;
```

Перевіримо ці процедури у дії за допомогою програми

```
x = [0.4 -0.3 1];  k=[1 2 3];
C = ug2mnk(x,k)
[d,e]=MNK2VEul(C)
C1=VEul2MNK(d,e)
dC= C1-C
```

В результаті її виконання отримаємо

```
C =
    0.51617    -0.80389    -0.29552
    0.71287     0.59449    -0.37203
    0.47475    -0.018638    0.87992
d =
     1.0526
e =
    0.20339    -0.44333     0.87298
C1 =
    0.51617    -0.80389    -0.29552
    0.71287     0.59449    -0.37203
    0.47475    -0.018638    0.87992
dC =
  1.1102e-016   -1.1102e-016   5.5511e-017
 -1.1102e-016   -1.1102e-016         0
        0              0        1.1102e-016
```

Похибка проведення операцій, як можна впевнитися, не перевищує $2 \cdot 10^{-16}$, що є вельми високою точністю.

Переваги застосування вектора Ейлера – *наочність подання повороту і кількість параметрів, близька до мінімальної.*

Недоліки застосування вектора Ейлера – *не встановлено операції складання поворотів, не пристосованість до проеціювання векторів.*

1.3. Кватерніон повороту

Кватерніоном називають (див. [1, с. 29-36]) гіперкомплексне число вигляду

$$\mathbf{Q} = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3,$$

де q_0, q_1, q_2, q_3 – дійсні числа, які у сукупності називають параметрами Родріга-Гамільтона; $\mathbf{i}, \mathbf{j}, \mathbf{k}$ – уявні одиниці, які мають такі властивості

$\mathbf{i}\circ\mathbf{i}=\mathbf{j}\circ\mathbf{j}=\mathbf{k}\circ\mathbf{k}=-1$; $\mathbf{i}\circ\mathbf{j}=\mathbf{k}=-\mathbf{j}\circ\mathbf{i}$; $\mathbf{j}\circ\mathbf{k}=\mathbf{i}=-\mathbf{k}\circ\mathbf{j}$; $\mathbf{k}\circ\mathbf{i}=\mathbf{j}=-\mathbf{i}\circ\mathbf{k}$;

$\circ$ –позначка кватерніонного добутку.

Величина q_0 складає скалярну частину кватерніону, а величини q_1, q_2, q_3 можна розглядати як складові деякого вектора – матриці-рядка

$$\mathbf{q}=[q_1\ q_2\ q_3]. \tag{1.6}$$

Кватерніон єдиного повороту на деякий кут δ має вигляд

$$\mathbf{Q}^{SP}=\cos\frac{\delta}{2}+\mathbf{e}^{SP}\sin\frac{\delta}{2}, \tag{1.7}$$

де $\mathbf{e}^{SP}\{e_1, e_2, e_3\}$ – одиничний вектор осі повороту.

Переходячи до подання кватерніону в комп'ютерній системі Matlab, подаватимемо кватерніон у вигляді вектора довжиною у чотири елементи

$$\mathbf{Q}=[q_0, q_1, q_2, q_3]=[q_0, \mathbf{q}]. \tag{1.8}$$

Отже, вираз (1.8) дає можливість визначити кватерніон повороту через параметри вектора Ейлера у такий спосіб

$$\mathbf{Q}^{SP}=\left[\cos\frac{\delta}{2}, \sin\frac{\delta}{2}\mathbf{e}\right], \tag{1.9}$$

де $\mathbf{e}\{e_1, e_2, e_3\}$ –одиничний вектор осі повороту.

Зворотне перетворення кватерніона повороту у матрицю напрямних косинусів дається формулою [1, (3.30), с. 35] :

$$\mathbf{C}^{SP}=\mathbf{E}+2[q_0\mathbf{E}+(\mathbf{q}\times)](\mathbf{q}\times), \tag{1.10}$$

де $(\mathbf{q}\times)$ є позначенням кососиметричної матриці

$$(\mathbf{q}\times)=\begin{bmatrix}0 & -q_3 & q_2\\ q_3 & 0 & -q_1\\ -q_2 & q_1 & 0\end{bmatrix}. \tag{1.11}$$

Перейдемо до комп'ютерного програмування.

Перетворення кватерніона у матрицю напрямних косинусів може бути здійснено процедурою

```
function C= kwat2mnk(kw)
% Перетворює кватерніон kw у матрицю напрямних косинусів C
% Звернення:   y = quat2ug(x)
% дозволяє обчислити матрицю напрямних косинусів "y"
% за заданим кватерніоном "x"

% Лазарєв Ю.Ф.      03-09-2015 р.
q0=kw(1);  qx=[0 -kw(4) kw(3); kw(4) 0 -kw(2); -kw(3) kw(2) 0];
E=eye(3,3);
C = E + 2*[q0*E + qx]*qx;
```

Зворотне перетворення можна здійснити функцією

```
function kw = mnk2kwat(C)
% Перетворює матрицю напрямних косинусів C у кватерніон kw
% Звернення:      y = mnk2kwat(x)
% дозволяє обчислити кватерніон "y"
% за заданою матрицею напрямних косинусів  "x"

% Лазарєв Ю.Ф.      03-09-2015 р.
[d,e]=MNK2VEul(C);
cd2= cos(d/2);  sd2= sin(d/2);
q0=cd2; q=sd2*e;
kw=[q0 q];
```

Створимо програму перевірки правильності роботи цих функцій

```
k=[1 2 3];
```

```
Ug= [0.5 0.3 -1];
C= ug2mnk(Ug,k);
kw = mnk2kwat(C);
C1= kwat2mnk(kw);
Ug1=mnk2ug(C1,k)
dUg=Ug1-Ug
```

В результаті її роботи виходить

```
Ug1 =        0.5                0.3                -1
dUg =  1.1102e-016       5.5511e-017      -2.2204e-016
```

що впевнює у адекватності створених програм-функцій і високій точності здійснення перетворень і обчислень.

Переваги застосування кватерніонів –

1) кількість параметрів, близька до мінімальної;

2) існує достатньо просте правило складання поворотів:

$$\mathbf{Q}^{SP} = \mathbf{Q}^{ST} \circ \mathbf{Q}^{TP},$$

а саме, – ***кватерніон результуючого повороту дорівнює кватерніонному добутку кватерніонів складових поворотів, записаних зліва направо у порядку самих поворотів****.*

Недоліки застосування кватерніонів – *відсутність наочності в поданні повороту .*

1.4. Вектор повороту Гіббса

Вектор Гіббса тісно пов'язаний з вектором Ейлера і визначається у такий спосіб

$$\mathbf{g}^{SP} = tg\frac{\delta}{2}\cdot\mathbf{e}, \qquad (1.12)$$

де δ і $\mathbf{e}$ – параметри вектора Ейлера.

Переваги застосування вектора Гіббса –

1) мінімальна кількість параметрів;

2) існує правило складання поворотів:

$$\mathbf{g}^{SP} = \frac{\mathbf{g}^{ST} + \mathbf{g}^{TP} + \mathbf{g}^{ST} \times \mathbf{g}^{TP}}{1 - \mathbf{g}^{ST} \cdot \mathbf{g}^{TP}}, \qquad (1.13)$$

Недоліки застосування вектора Гіббса – *складне правило складання поворотів.*

1.5. Вектор скінченого повороту Родріга

Найбільш розповсюдженим є подання повороту у вигляді так званого вектору скінченого повороту (вектора Родріга):

$$\mathbf{p}^{SP} = 2tg\frac{\delta}{2}\cdot\mathbf{e}, \qquad (1.12)$$

де δ і $\mathbf{e}$ – параметри вектора Ейлера.

Переваги і недоліки застосування вектора Родріга такі самі, що й вектора Гіббса.

2. Структура комп'ютерної моделі БІСО

Безплатформова інерціальна система орієнтації складається з двох головних блоків:

1) блоку вимірювачів, призначеного вимірювати за допомогою гіротахометрів (або інтегрувальних гіротахометрів) значення проекцій абсолютної кутової швидкості основи в окремі моменти часу, розділені так званим кроком опитування вимірювачів (або інтегралів за часом віл цих проекцій (так званих квазікоординат) за один крок опитування);
2) бортового обчислювального пристрою, який за даними вимірювання вектора кутової швидкості основи обчислює (шляхом чисельного інтегрування за тим чи іншим алгоритмом) параметри поточної орієнтації основи.

Розглянемо задачу дослідження похибок БІСО, обумовлених виключно похибками алгоритму чисельного інтегрування, закладеного у обчислювачі БІСО.

Цю задачу розв'язуватимемо шляхом комп'ютерного моделювання.

Відповідна комп'ютерна модель, очевидно, має складатися з таких програмних блоків:

1) блоку імітатора поточного кутового руху основи;
2) блоку імітатора обчислювального пристрою БІСО з вбудованим в нього досліджуваним алгоритмом;
3) блоку порівнювання параметрів орієнтації, отриманих блоком імітатора обчислювача БІСО, з "дійсними" значеннями цих параметрів, отриманих у блоці імітатора руху основи; отримання масиву значень похибок алгоритму як функцій часу.

У якості наочних параметрів поточної орієнтації основи зручніше за все обрати послідовність кутів Ейлера-Крилова.

Наведемо текст UGDvObj_Laz прикладу процедури, яка реалізує імітатор кутового руху основи:

```
function [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t)
% Процедура вычисления текущих параметров движения основания
% для стандартного расположения осей координат и углов основания

% Лазарев Ю. Ф.        01-04-2014        05-09-2015
global psit psim tetm gam omp omt omg ep et eg
% Углы
psi = psit*t + psim*sin(omp*t+ep);      tet = tetm*sin(omt*t+et);
ga=gam*sin(omg*t+eg);
% Скорости углов
pst=psim*omp*cos(omp*t+ep); tett=tetm*omt*cos(omt*t+et);
gat=gam*omg*cos(omg*t+eg);
% Синусы и косинусы
cg=cos(ga); ct=cos(tet);    st=sin(tet);    sg=sin(ga);
% Проекции угловой скорости на связанные оси
omX = gat+pst*st;    omY = pst*ct*cg + tett*sg;
omZ = - pst.*ct.*sg + tett.*cg;
```

Перевіримо правильність роботи цієї функції за допомогою такої допоміжної керувальної програми:

```
% Upr_Obj
% Програма перевірки роботи функції UgDvObj_laz

clear all, clc
global psit psim tetm gam omp omt omg ep et eg
   % Установка параметров движения основания
psit = 0.01;              Tt = 0;            Gt = 0;
psim = 0.08;           tetm = 0.1;        gam = 0.15;
omp = 3;               omt = 1;           omg =  2;
ep = 0*pi/180;         et = 90*pi/180;    eg = -90*pi/180;
h=0.1; t=0:h:30; n=length(t);
```

```
for k=1:n
   T=t(k);
   [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(T);
   Psi(k)=psi;   Tet(k)=tet;   Gm(k)=ga;
end
plot(t,Psi,'o-',t,Tet,'*-',t,Gm,'.-'), grid
set(gca,'FontSize',14)
title('Кутовий рух основи (точний)')
xlabel('Час, с'), ylabel('Кути, радіани')
legend('\psi','\theta','\phi',2)
```

Виклик цієї програми приводить до появи у графічному вікні монітора картини, наведеної на рис. 2.1.

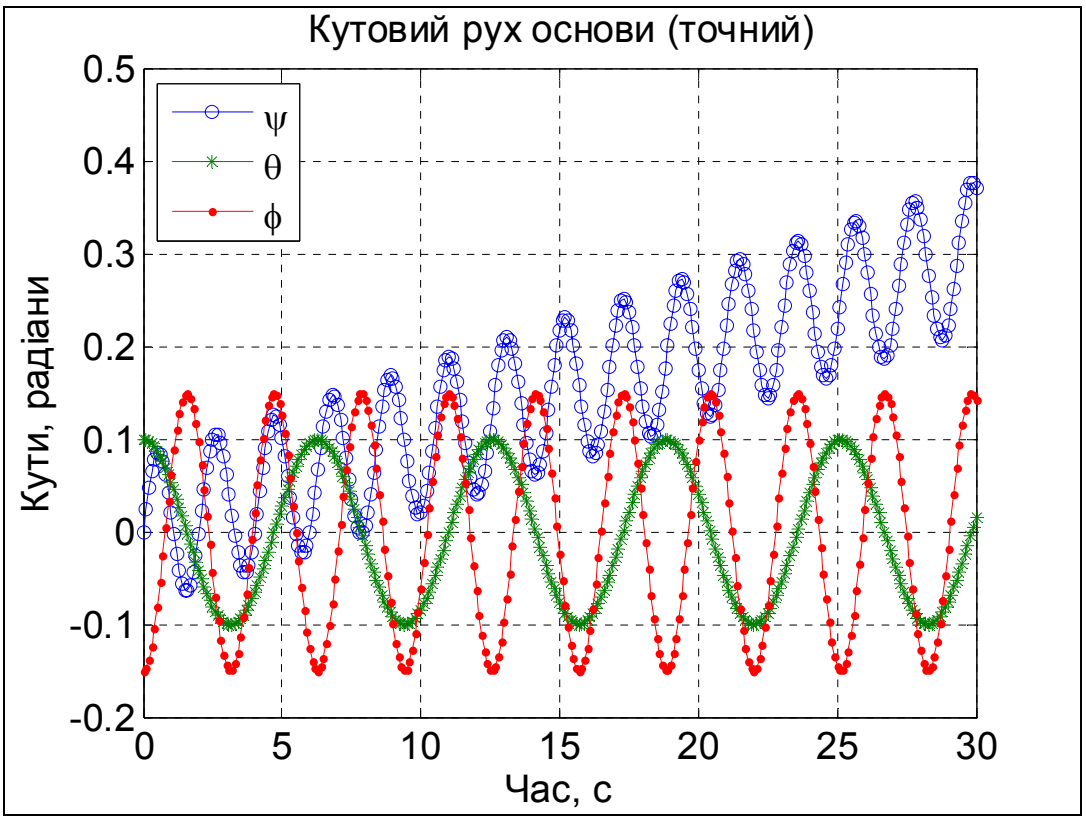


Рис. 2.1. Приклад роботи блоку імітатора руху основи

3. Алгоритми на основі методів Рунге-Кутти

Методи Рунге-Кутти можна пристосувати і для побудови алгоритмів БІСО, дещо їх трансформувавши.

У прикладній математиці [2, с 166-176] методи Рунге-Кутти вважаються однокроковими, бо розраховують значення вектора фазових змінних на наступному кроці інтегрування. При цьому вони використовують обчислення правих частин диференціальних рівнянь у формі Коши у кількох моментах часу всередині кроку інтегрування. Останнє неможливо в алгоритмах БІСО через те, що значення вектору кутової швидкості досяжні лише в окремі моменти часу, розділені інтервалом опитування вимірювачів.

Тому алгоритми методів Рунге-Кутти слід переформатувати у багатокрокові, тобто такі, в яких крок інтегрування складається з кількох кроків опитування в залежності від кількості використовуваних проміжних моментів часу.

Іншою особливістю алгоритмів БІСО є те, що обчислювання вектора фазових змінних наприкінці кроку інтегрування здійснюється тоді, коли крок інтегрування вже завершився, тобто вже відоме нове значення вектора кутової швидкості основи. Тому вже однокроковий алгоритм БІСО може спиратися на знання значення правих частин диференціальних рівнянь у два моменти часу – на початку і наприкінці кроку опитування. У цьому випадку однокроковий алгоритм БІСО може бути побудований на основі модифікованого метода Ейлера.

Для багатокрокових алгоритмів БІСО введемо окрім кроку опитування h ще поняття кроку інтегрування H. Корк інтегрування може складатися з одного кроку опитування $H = h$. Це характерно для так званих розгінних багатокрокових методів, алгоритми яких спираються на дані про кутову швидкість у кількох попередніх моментах часу, тобто потребують попереднього "розгону".

Алгоритми ж, основані на методах Рунге-Кутти, є безрозгінними (самостартувальними). Вони використовують лише ті значення вимірюваної кутової швидкості, які виходять на наступних кількох кроках опитування. В них здійснюється визначення фазових змінних через декілька кроків опитування, тобто $H = m \cdot h$, де m – кількість кроків опитування у одному кроці інтегрування.

Загальна формула чисельного інтегрування на основі методів Рунге-Кутти така

$$\mathbf{y}_{n+m} = \mathbf{y}_n + H \cdot \mathbf{F}(t_n; \mathbf{y}_n), \tag{3.1}$$

де $\mathbf{y}_n$ – вектор значень фазових змінних у момент часу t_n; n – номер початкового кроку опитування; $\mathbf{F}(t_n; y_n)$ – вектор, що є певною функцією значень правої частини $\mathbf{Z}(t_j; y_j)$; $(j = n, n+1, \ldots, n+m)$ диференціальних рівнянь у формі Коши:

$$\frac{d\mathbf{y}}{dt} = \mathbf{Z}(t; \mathbf{y}). \tag{3.2}$$

Значення функції $\mathbf{F}(t_n; y_n)$ визначаються формулами, наведеними у таблиці 3.1.

Таблиця 3.1. Алгоритми БІСО на основі методів Рунге-Кутти

$\mathbf{y}_{n+m} = \mathbf{y}_n + H \cdot \mathbf{F}(t_n; \mathbf{y}_n)$; $\mathbf{k}_1 = \mathbf{Z}(t_n; \mathbf{y}_n)$

Назва	Формула	Допоміжні величини
RK-21 ($H = h$)	$\mathbf{F} = (\mathbf{k}_1 + \mathbf{k}_2)/2$	$\mathbf{k}_2 = \mathbf{Z}(t_n + H; \mathbf{y}_n + H\mathbf{k}_1)$
RK-22 ($H = 2h$)	$\mathbf{F} = \mathbf{k}_2$	$\mathbf{k}_2 = \mathbf{Z}(t_n + H/2; \mathbf{y}_n + H\mathbf{k}_1/2)$
RK-32 ($H = 2h$)	$\mathbf{F} = (\mathbf{k}_1 + 4\mathbf{k}_2 + \mathbf{k}_3)/2$	$\mathbf{k}_2 = \mathbf{Z}(t_n + H/2; \mathbf{y}_n + H\mathbf{k}_1/2)$; $\mathbf{k}_3 = \mathbf{Z}(t_n + H; \mathbf{y}_n + H(2\mathbf{k}_2 - \mathbf{k}_1))$
RK-33	$\mathbf{F} = (\mathbf{k}_1 + 3\mathbf{k}_3)/4$	$\mathbf{k}_2 = \mathbf{Z}(t_n + H/3; \mathbf{y}_n + H\mathbf{k}_1/3)$;

($H = 3h$)		$\mathbf{k}_3 = \mathbf{Z}(t_n + 2H/3;\ \mathbf{y}_n + 2H\mathbf{k}_2/3)$
RK-42 ($H = 2h$)	$\mathbf{F} = (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)/6$	$\mathbf{k}_2 = \mathbf{Z}(t_n + H/2;\ \mathbf{y}_n + H\mathbf{k}_1/2)$; $\mathbf{k}_3 = \mathbf{Z}(t_n + H/2;\ \mathbf{y}_n + H\mathbf{k}_2/2)$; $\mathbf{k}_4 = \mathbf{Z}(t_n + H;\ \mathbf{y}_n + H\mathbf{k}_3)$
RK-43 ($H = 3h$)	$\mathbf{F} = (\mathbf{k}_1 + 3\mathbf{k}_2 + 3\mathbf{k}_3 + \mathbf{k}_4)/8$	$\mathbf{k}_2 = \mathbf{Z}(t_n + H/3;\ \mathbf{y}_n + H\mathbf{k}_1/3)$; $\mathbf{k}_3 = \mathbf{Z}(t_n + 2H/3;\ \mathbf{y}_n + H(\mathbf{k}_2 - \mathbf{k}_1/3))$; $\mathbf{k}_4 = \mathbf{Z}(t_n + H;\ \mathbf{y}_n + H(\mathbf{k}_1 + \mathbf{k}_3 - \mathbf{k}_2))$

Формули у такому поданні не відрізняються від звичайних. Відмінність лише у тому, що крок інтегрування має складатися з кількох кроків опитування. Алгоритм *RK-21* є однокроковим другого порядку. Двокрокові алгоритми *RK-22, RK-32* и *RK-42* мають відповідно другий, третій і четвертий порядок точності. Трикрокові алгоритми *RK-33* и *RK-43* відносяться до третього і четвертого порядків відповідно.

Приведемо текст можливого варіанту процедури інтегрування трикроковим алгоритмом четвертого порядку:

```
function [tout,yout]=RK_43(Zpfun,h,t,y)
% Процедура трехшагового (четырехточечного) алгоритма
% метода Рунге-Кутты четвертого порядка
% Zpfun - символьная переменная, содержащая имя используемой процедуры
%         вычисления значения правой части интегрируемой системы
%         дифференциальных уравнений, записанной в форме Коши
%    h  -  шаг интегрирования
%    t  -  время начала очередного шага интегрирования
%    y  - значение вектора фазовых переменных в момент времени t
%   tout - момент конца очередного шага интегрирования
%   yout - вектор фазовых переменных в конце шага интегрирования

% Лазарев Ю. Ф.          06-09-2015
k1=feval(Zpfun,t,y);
k2=feval(Zpfun,t+h/3,y+h*k1/3);
k3=feval(Zpfun,t+2*h/3,y+h*(k2-k1/3));
k4=feval(Zpfun,t+h,y+h*(k1-k2+k3));
F=(k1+3*k2+3*k3+k4)/8;
yout=y+h*F;    tout=t+h;
```

. При використанні цього алгоритму потрібно попередньо сформувати значення кроку інтегрування за відомим значенням кроку h опитування

H=3*h;

і звертатися до процедури метода у такий спосіб

[tout,yout]=RK_43('ім'я процедури правих частин',H,t,y);

4. Кінематичні рівняння поворотів

В теоретичній механіці під кінематичними рівняннями поворотів розуміють диференціальні рівняння, що пов'язують проекції кутової швидкості тіла, рух якого досліджується, на осі той чи іншої системи координат з похідними від параметрів повороту.

У цьому посібнику мова буде йти про так звані безплатформові інерціальні системи орієнтації (БІСО), які призначені визначати поточну орієнтацію основи, на якій вони встановлені, використовуючи вимірювання проекцій кутової швидкості основи на осі системи координат, пов'язаної з основою. Тому під кінематичними рівняннями розумітимемо такі, в яких похідні від параметрів орієнтації (поворотів) виражаються через проекції ω_{1P}^{PS}, ω_{2P}^{PS} і ω_{3P}^{PS} вектора $\boldsymbol{\omega}^{PS}$ кутової швидкості основи (система координат P) відносно вихідної системи координат S.

Наведемо [1, с. 42-50] кінематичні рівняння, виражені через подані раніше види параметрів орієнтації. Задля однаковості їх зручно подати у стандартній формі Коши, тобто у вигляді

$$\frac{d\mathbf{y}}{dt} = \mathbf{Z}(t;\, \mathbf{y}), \tag{4.1}$$

де $\mathbf{y}$ – вектор фазових змінних $\mathbf{Z}(t;\, y)$ – вектор-функція часу і фазових змінних.

4.1. Кінематичні рівняння Ейлера

Кінематичні рівняння Ейлера, відомі в теоретичній механіці, можна в наших цілях подати у вигляді

$$\begin{cases} \dfrac{d\psi}{dt} = (\omega_{1P}^{PS} \cdot \cos\varphi - \omega_{2P}^{PS} \cdot \sin\varphi)/\cos\vartheta; \\ \dfrac{d\vartheta}{dt} = \omega_{2P}^{PS} \cdot \cos\varphi + \omega_{1P}^{PS} \cdot \sin\varphi; \\ \dfrac{d\varphi}{dt} = \omega_{3P}^{PS} - tg\vartheta \cdot (\omega_{1P}^{PS} \cdot \cos\varphi - \omega_{2P}^{PS} \cdot \sin\varphi). \end{cases} \tag{4.2}$$

Якщо ввести вектор фазових змінних у вигляді

$$\mathbf{y} = [y_1,\, y_2,\, y_3] = [\psi,\, \vartheta,\, \varphi], \tag{4.3}$$

а під вектор-функцією розуміти вектор

$$\mathbf{Z}(t;\, \mathbf{y}) = \left[Z_1,\, Z_2,\, Z_3\right], \tag{4.4}$$

де

$$\begin{aligned} Z_1 &= [\omega_1(t)\cos y_1 - \omega_2(t)\sin y_1]/\cos y_2; \\ Z_2 &= \omega_2(t) \cdot \cos y_1 + \omega_1(t) \cdot \sin y_1; \\ Z_3 &= \omega_3(t) - [\omega_1(t)\cos y_1 - \omega_2(t)\sin y_1]tgy_2; \end{aligned} \tag{4.5}$$

то кінематичні рівняння Ейлера набудуть стандартної форми Коши (4.1).

Наведемо приклад процедури обчислення правих частин кінематичного рівняння Ейлера:

```
function z=UgEul(t,y)
% Процедура обчислення правих частин кінематичних рівнянь Ейлера
%       t – значення поточного моменту часу
%       y – вектор значень кутів повороту
%       z – вектор значень похідних за часом від кутів повороту

% Лазарев Ю. Ф.       15-09-2015
[psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
cf=cos(y(3));   sf=sin(y(3));  ct=cos(y(2));   st=sin(y(2));
```

```
z(1)=(omX*cf-omY*sf)/ct;
z(2)=omY*cf+omX*sf;
z(3)=omZ-st*z(1);
```

4.2. Матричне рівняння Пуассона

Матричне кінематичне рівняння, що використовує матрицю напрямних косинусів, має форму

$$\frac{d\mathbf{C}^{SP}}{dt} = \mathbf{C}^{SP} \cdot (\boldsymbol{\omega}_P^{PS} \times). \tag{4.6}$$

Тут використано позначення кососиметричної матриці

$$(\boldsymbol{\omega}_P^{PS} \times) = \begin{bmatrix} 0 & -\omega_{3P}^{PS} & \omega_{2P}^{PS} \\ \omega_{3P}^{PS} & 0 & -\omega_{1P}^{PS} \\ -\omega_{2P}^{PS} & \omega_{1P}^{PS} & 0 \end{bmatrix}. \tag{4.7}$$

Рівнянню (4.6) можна надати форми (4.1), якщо вважати

$$\mathbf{y} = \mathbf{C}^{SP}; \qquad \mathbf{Z} = \mathbf{y} \cdot (\boldsymbol{\omega}(t) \times), \tag{4.8}$$

де

$$(\boldsymbol{\omega}(t) \times) = \begin{bmatrix} 0 & -\omega_3(t) & \omega_2(t) \\ \omega_3(t) & 0 & -\omega_1(t) \\ -\omega_2(t) & \omega_1(t) & 0 \end{bmatrix}. \tag{4.9}$$

Далее наведено текст відповідної процедури:

```
function z=Poiss(t,y)
% Процедура обчислення правих частин матричного рівняння Пуассона
%   t – значення поточного моменту часу
%   y – матриця напрямних косинусів
%   z - матриця-похідна за часом від матриці напрямних косинусів

% Лазарев Ю. Ф.        15-09-2015
[psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
OMx =[0 -omZ omY; omZ 0 -omX; -omY omX 0];
z = y*OMx;
```

4.3. Рівняння через вектор Ейлера

Позначимо вектор Ейлера

$$\boldsymbol{\varphi} = \delta \cdot \mathbf{e}, \tag{4.10}$$

де $\mathbf{e}$ – позначення вектора

$$\mathbf{e} = [e_1 \; e_2 \; e_3].$$

Кінематичне рівняння через вектор Ейлера у матричній формі має вигляд

$$\frac{d\boldsymbol{\varphi}}{dt} = \boldsymbol{\omega}_P^{PS} + \frac{1}{2}(\boldsymbol{\varphi} \times)\boldsymbol{\omega}_P^{PS} + \frac{1 - \frac{\delta}{2} ctg \frac{\delta}{2}}{\delta^2}(\boldsymbol{\varphi} \times)(\boldsymbol{\varphi} \times)\boldsymbol{\omega}_P^{PS}, \tag{4.11}$$

де позначено матриці

$$\boldsymbol{\omega}_P^{PS} = [\omega_{1P}^{PS} \; \omega_{2P}^{PS} \; \omega_{3P}^{PS}]; \qquad (\boldsymbol{\varphi} \times) = \begin{bmatrix} 0 & -\varphi_3 & \varphi_2 \\ \varphi_3 & 0 & -\varphi_1 \\ -\varphi_2 & \varphi_1 & 0 \end{bmatrix}. \tag{4.12}$$

Позначимо

$$\mathbf{y} = [y_1, y_2, y_3] = [\varphi_1, \varphi_2, \varphi_3], \tag{4.13}$$

$$\mathbf{Z} = \boldsymbol{\omega}(t) + \frac{1}{2}(\mathbf{y}\times)\boldsymbol{\omega}(t) + \frac{1-\frac{y}{2}ctg\frac{y}{2}}{y^2}(\mathbf{y}\times)(\mathbf{y}\times)\boldsymbol{\omega}(t), \tag{4.14}$$

де

$$(\mathbf{y}\times) = \begin{bmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{bmatrix}; \qquad y = \sqrt{y_1^2 + y_2^2 + y_3^2};$$

$$\boldsymbol{\omega}(t) = [\omega_1(t), \omega_2(t), \omega_3(t)]. \tag{4.15}$$

Тоді рівняння (4.11) набудуть форми Коши (4.1).

Недоліком рівняння (4.11) є невизначеність значення коефіцієнта при його третьої складової за умови $\delta = 0$. Враховуючи, що за малих значень δ цей коефіцієнт може подати у вигляді ряду

$$\frac{1-\frac{\delta}{2}ctg\frac{\delta}{2}}{\delta^2} = \frac{1}{12}\left(1 + \frac{\delta^2}{60} + \frac{\delta^4}{2520} + \frac{\delta^6}{10800} + \ldots\right),$$

краще користуватися рівнянням виду

$$\frac{d\boldsymbol{\varphi}}{dt} = \boldsymbol{\omega}_P^{PS} + \frac{1}{2}(\boldsymbol{\varphi}\times)\boldsymbol{\omega}_P^{PS} + \frac{1}{12}\left(1 + \frac{\delta^2}{60} + \frac{\delta^4}{2520} + \frac{\delta^6}{10800} + \ldots\right)(\boldsymbol{\varphi}\times)(\boldsymbol{\varphi}\times)\boldsymbol{\omega}_P^{PS}. \tag{4.16}$$

С точністю до членів шостого порядку мализни можна використовувати таку процедуру правих частин векторного рівняння Ейлера:

```
function z=V_Eul(t,y)
% Процедура обчислення правих частин векторного рівняння Ейлера
%   t – значення поточного моменту часу
%   y – значення поточного вектора Ейлера
%   z - значення поточного вектора похідної за часом від вектора Ейлера

% Лазарев Ю. Ф.       15-09-2015
[psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
OM =[omX omY omZ]';
d = norm(y);  e=y/mVE;  yX=[0 -y(3) y(2); y(3) 0 y(1); -y(2) y(1) 0];
z = OM + yX*OM/2 + yX*yX*OM/12;
```

4.4. Кватерніонне кінематичне рівняння

Якщо використовувати матричну форму (1.8) позначення кватерніону повороту то кватерніонне рівняння орієнтації складатиметься з двох рівнянь

$$\begin{cases} \dfrac{dq_0}{dt} = -\dfrac{1}{2}\mathbf{q}^T \cdot \boldsymbol{\omega}_P^{PS} \\ \dfrac{d\mathbf{q}}{dt} = \dfrac{1}{2}[q_0 \cdot \boldsymbol{\omega}_P^{PS} + (\mathbf{q}\times)\cdot\boldsymbol{\omega}_P^{PS}] \end{cases}, \tag{4.16}$$

де використані позначення матриць (1.6) і (1.11).

Нехай

$$\mathbf{y} = [y_1, \mathbf{y}_2] = [q_0, q_1, q_2, q_3], \tag{4.17}$$

$$\mathbf{Z} = [Z_1, \mathbf{Z}_2] = \left[-\frac{1}{2}\mathbf{y}_2^T \cdot \boldsymbol{\omega}(t); \quad \frac{1}{2}\left[y_1 \cdot \boldsymbol{\omega}(t) + (\mathbf{y}_2\times)\cdot\boldsymbol{\omega}(t)\right]\right]. \tag{4.18}$$

Тоді система рівнянь (4.16) перетворюється на рівняння вигляду (4.1).

Відповідна процедура обчислення правих частин рівнянь може виглядати так:

```
function z=Kwatern(t,y)
% Процедура обчислення правих частин кватерніонного рівняння орієнтації
%   t – значення поточного моменту часу
%   y – значення поточного кватерниону поворота
%   z - значення поточної похідної за часом від кватерніону поворота

% Лазарев Ю. Ф.        15-09-2015
[psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
OM =[omX omY omZ];
q0=y(1);    q=y(2:4);
z(1) = -q*OM'/2;
z(2:4) = (q0*OM + cross(q,OM))/2;
```

4.5. Рівняння через вектор Гіббса

Рівняння орієнтації, подане через вектор Гіббса і виражене у матричній формі, можна подати так

$$\frac{d\mathbf{g}^{SP}}{dt}=\frac{1}{2}\left[\boldsymbol{\omega}_P^{PS}+\left(\mathbf{g}^{SP}\times\right)\boldsymbol{\omega}_P^{PS}+\left(\left(\mathbf{g}^{SP}\right)^T\cdot\boldsymbol{\omega}_P^{PS}\right)\mathbf{g}^{SP}\right], \tag{4.19}$$

де

$$\mathbf{g}^{SP}=\left[g_{1p}^{SP}\ g_{2p}^{SP}\ g_{3p}^{SP}\ \right]; \tag{4.20}$$

$$\left(\mathbf{g}^{SP}\times\right)=\begin{bmatrix}0 & -g_{3p}^{SP} & g_{2p}^{SP}\\ g_{3p}^{SP} & 0 & -g_{1p}^{SP}\\ -g_{2p}^{SP} & g_{1p}^{SP} & 0\end{bmatrix};\quad \left(\mathbf{g}^{SP}\right)^T=\left[g_{1p}^{SP}\ g_{2p}^{SP}\ g_{3p}^{SP}\ \right]^T=\begin{bmatrix}g_{1p}^{SP}\\ g_{2p}^{SP}\\ g_{3p}^{SP}\end{bmatrix}.$$

У позначеннях (див. (4.20), (4.21))

$$\mathbf{y}=\mathbf{g}^{SP}, \tag{4.21}$$

та

$$\mathbf{Z}=\frac{1}{2}\left[\boldsymbol{\omega}(t)+\left(\mathbf{y}\times\right)\boldsymbol{\omega}(t)+\left(\mathbf{y}^T\cdot\boldsymbol{\omega}(t)\right)\mathbf{y}\right]. \tag{4.22}$$

рівняння (4.19) прийме форму (4.1).

```
function z=Gibbs(t,y)
% Процедура обчислення правих частин векторного рівняння Гіббса
%   t – значення поточного моменту часу
%   y – поточне значення  вектора Гіббса
%   z - поточне значення вектора похідної за часом від вектора Гіббса

% Лазарев Ю. Ф.        15-09-2015
[psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
OM =[omX omY omZ]';
yX=[0 -y(3) y(2); y(3) 0 y(1); -y(2) y(1) 0];
z = (OM + yX*OM + (y'*OM)*y)/2;
```

4.6. Рівняння через вектор Родріга

Рівняння орієнтації через вектор Родріга у матричній формі багато в чому аналогічні рівнянню через вектор Гіббса

$$\frac{d\mathbf{p}^{SP}}{dt}=\boldsymbol{\omega}_P^{PS}+\frac{1}{2}\left(\mathbf{p}^{SP}\times\right)\boldsymbol{\omega}_P^{PS}+\frac{1}{4}\left(\left(\mathbf{p}^{SP}\right)^T\cdot\boldsymbol{\omega}_P^{PS}\right)\mathbf{p}^{SP}, \tag{4.23}$$

де використані позначення

$$\mathbf{p}^{SP}=\left[p_{1p}^{SP}\ p_{2p}^{SP}\ p_{3p}^{SP}\ \right]; \tag{4.24}$$

$$\left(\mathbf{p}^{SP}\times\right)=\begin{bmatrix} 0 & -p_{3p}^{SP} & p_{2p}^{SP} \\ p_{3p}^{SP} & 0 & -p_{1p}^{SP} \\ -p_{2p}^{SP} & p_{1p}^{SP} & 0 \end{bmatrix}; \quad \left(\mathbf{p}^{SP}\right)^{T}=\begin{bmatrix} p_{1p}^{SP} & p_{2p}^{SP} & p_{3p}^{SP} \end{bmatrix}^{T}=\begin{bmatrix} p_{1p}^{SP} \\ p_{2p}^{SP} \\ p_{3p}^{SP} \end{bmatrix}.$$

Форму (4.1) це рівняння набуває, якщо використати позначення

$$\mathbf{y}=\mathbf{p}\,; \qquad \mathbf{Z}=\boldsymbol{\omega}(t)+\frac{1}{2}\left(\mathbf{y}\times\right)\boldsymbol{\omega}(t)+\frac{1}{4}\left(\mathbf{p}^{T}\cdot\boldsymbol{\omega}(t)\right)\mathbf{y}\,. \tag{4.25}$$

Текст відповідної процедури може бути таким:

```
function z=Rodriges(t,y)
% Процедура обчислення правих частин векторного рівняння Родріга
%   t – значення поточного моменту часу
%   y – поточне значення  вектора Родріга
%   z - поточне значення вектора похідної за часом від вектора Родріга

% Лазарев Ю. Ф.         15-09-2015
[psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
OM =[omX omY omZ]';
yX=[0 -y(3) y(2); y(3) 0 y(1); -y(2) y(1) 0];
z = OM + yX*OM/2 + (y'*OM)*y/4;
```

5. Розробка керувальної програми

Мета керувальної програми – на основі попередніх створених процедур організувати виконання наступної послідовності дій:

1) задати значення постійних параметрів, що визначають кутовий рух основи;
2) задати значення кроку інтегрування, кроку опитування і тривалості модельного часу, протягом якого виявлятимуться алгоритмічні похибки заданого алгоритму БІСО;
3) обчислити значення параметрів кутового положення у початковий момент часу;
4) організувати циклічне звернення до процедури алгоритму чисельного інтегрування заданого диференціального рівняння орієнтації і формування масиву похибок визначення кутів Ейлера на кожному циклі (кроці) інтегрування;
5) створити процедуру обчислення середньої швидкості змінювання похибок визначення кутів орієнтації (дрейфу похибок) за відомими (обчисленими) масивами алгоритмічних похибок;
6) оформити графічне вікно монітора комп'ютеру, де були б наведені: а) графіки залежності алгоритмічних похибок від часу; б) текстова інформація про усі використані задані параметри руху основи і параметри інтегрування; в) обчислені дрейфи похибок з усіх кутів.

Розглянемо приклад такої програми.

Дослідимо похибки чисельного інтегрування матричного рівняння Пуассона шістьма методами Рунге-Кутти, алгоритми яких були наведені раніше.

Задля виявлення дрейфів похибок задамо наступний рух основи:

$$\psi(t) \equiv 0; \qquad \vartheta(t) = a_m \sin(\omega t + \varepsilon); \qquad \varphi(t) = a_m \sin(\omega t),$$

за якого основа здійснює синхронні коливання з однаковою амплітудою навколо двох ортогональних осей (з кутів ϑ та φ) і не обертається навколо третьої осі.

Нижче приведено текст відповідної керувальної програми за ім'ям Pois_RuKu_UgSk_upr.

```
% Pois_RuKu_UgSk_upr
% Керувальна програма дослідження алгоритмичних похибок чисельного
% інтегрування матричного рівняння Пуассона методами Рунге-Кутти
%      Використовувані процедури
%   UgDvObj_laz - обчислення поточних точних значень кутів повороту основи
%      і проекцій його кутової швидкості на осі звязаноїсистемы координат
%   Ru_Ku_21 - перший метод Рунге-Кутти 2-го порядка (однокроковий)
%   Ru_Ku_22 - другий метод Рунге-Кутти 2-го порядка (двокроковий)
%   Ru_Ku_31 - перший метод Рунге-Кутти 3-го порядка (двокроковий)
%   Ru_Ku_32 - другий метод Рунге-Кутти 3-го порядка (трикроковий)
%   Ru_Ku_41 - перший метод Рунге-Кутти 4-го порядка (двокроковий)
%   Ru_Ku_42 - другй метод Рунге-Кутти 4-го порядка (трикроковий)
%   Ur_Poisson - обчислення правої частини рівняння Пуассона
%   vect2ksm   - перетворення вектора в кососиметричну матрицю
%   mnke       - побудова матриці напрямних косинусів,
%                відповідну одному повороту навколо заданої осі
%   Ug2Mnk     - перетворення послідовності в матрицю напрямних косинусів
%   Mnk2Ug1 - перетворення матриці напрямних косинусів  у кути Ейлера
%   Drift_L        - обчислення величини дрейфа за масивом даних
%   GRAF_Ru_Ku_Laz1 - побудова графиків 18 масивів даних від часу
% Лазарев Ю. Ф.              14-11-2014       15-09-2015
clc, clear all
global psim tetm gam omp omt omg ep et eg
global h sname sprogram DR
```

```
sprogram='Pois-RuKu-UgSk-upr';    sname='Лазарев Ю. Ф.';
   % Задання значень постійних параметрів кутового руху основи
psim=0.;       tetm = 0.1;     gam=0.1;
omp=0;          omt=1;          omg=1;
ep=0*pi/180;    et=0*pi/180;    eg=0*pi/180;
       % Завдання параметрів інтегрування
 t=0;           % Початковий момент часу
 h=0.01;        % Крок опитування вимірювачів
 tfinal=100;    % Кінцевий час інтегрування
% Розрахунок матриці напрямних косинусів у початковий момент часу
 [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t); %Розрахунок кутів орієнтації
 ugV=[psi,tet,ga];  kV=[2,3,1];  C0=ug2mnk(ugV,kV);  % МНК
%%  Метод Рунге-Кутти 21 другого порядку (однокроковий)
H=h;    C2=C0;
   % Початкові значення обчислювальних масивів похибок
  t=0;     imas = 1;          tt2(imas)=t;
   dpsi2(imas)=0;   dtet2(imas)=0;   dga2(imas)=0;
while (imas-1)*H<tfinal
        t=(imas-1)*H;   % формування значення поточного часу
        [tout,yout]=Ru_Ku_21('Ur_Poisson',H,t,C2);% Інтегрування
        C2=yout; t=tout;   % Формування МНК і моменту часу через крок
        imas=imas+1;    % Змінювання значення лічильника
        tt2(imas)=tout; % Формування масиву моментів часу
        [ps2,te2,ga2]=Mnk2Ug1(C2);  % Визначення кутів за новою МНК
                % Розрахунок точних значень кутів
        [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
                % Розрахунок похибок визначення кутів
        dpsi2(imas)=psi-ps2;  dtet2(imas)=tet-te2;   dga2(imas)=ga-ga2;
end
%%  Метод Рунге-Кутти 22 другого порядку (двокроковий)
H=2*h;  C2=C0;    t=0;     imas=1;          tt22(imas)=t;
   dpsi22(imas)=0;   dtet22(imas)=0;   dga22(imas)=0;
while (imas-1)*H<tfinal
        t=(imas-1)*H;
        [tout,yout]=Ru_Ku_22('Ur_Poisson',H,t,C2);
        C2=yout; t=tout;
        imas=imas+1;
        tt22(imas)=tout;    % Масив моментів часу
        [ps2,te2,ga2]=Mnk2Ug1(C2);
        [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
        dpsi22(imas)=psi-ps2;  dtet22(imas)=tet-te2;   dga22(imas)=ga-ga2;
end
%%% Метод Рунге-Кутти 31 третього порядку (двокроковий)
H=2*h;  C3=C0;  t=0;    imas=1;  tt31(imas)=t;
dpsi31(imas)=0;   dtet31(imas)=0;   dga31(imas)=0;
while (imas-1)*H<tfinal
        t=(imas-1)*H;
        [tout,yout]=Ru_Ku_31('Ur_Poisson',H,t,C3);
        C3=yout; t=tout;   imas=imas+1;
        tt31(imas)=tout;                % Масив моментів часу
        [ps3,te3,ga3]=Mnk2Ug1(C3);
        [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
        dpsi31(imas)=psi-ps3;  dtet31(imas)=tet-te3;   dga31(imas)=ga-ga3;
end
%%% Метод Рунге-Кутти 32 третього порядку (трикроковий)
H=3*h;  C3=C0;  t=0;    imas=1;  tt3(imas)=t;
dpsi3(imas)=0;   dtet3(imas)=0;   dga3(imas)=0;
while (imas-1)*H<tfinal
        t=(imas-1)*H;
        [tout,yout]=Ru_Ku_32('Ur_Poisson',H,t,C3);
        C3=yout; t=tout;            imas=imas+1;
        tt3(imas)=tout;             % Масив моментів часу
        [ps3,te3,ga3]=Mnk2Ug1(C3);
```

```
        [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
        dpsi3(imas)=psi-ps3;  dtet3(imas)=tet-te3;   dga3(imas)=ga-ga3;
end
%%% Метод Рунге-Кутти 41 четвертого порядку (двокроковий)
H=2*h;  C4=C0;  t=0;    imas=1;  tt4(imas)=t;
dpsi4(imas)=0;   dtet4(imas)=0;   dga4(imas)=0;
while (imas-1)*H<tfinal
        t=(imas-1)*H;
        [tout,yout]=Ru_Ku_41('Ur_Poisson',H,t,C4);
        C4=yout; t=tout;              imas=imas+1;
        tt4(imas)=tout;               % Масив моментів часу
        [ps4,te4,ga4]=Mnk2Ug1(C4);
        [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
        dpsi4(imas)=psi-ps4;   dtet4(imas)=tet-te4;  dga4(imas)=ga-ga4;
end
%%% Метод Рунге-Кутти 42 четвертого порядку (трикроковий)
H=3*h;  C4=C0;  t=0;    imas=1;  tt42(imas)=t;
dpsi42(imas)=0;   dtet42(imas)=0;   dga42(imas)=0;
while (imas-1)*H<tfinal
        t=(imas-1)*H;
        [tout,yout]=Ru_Ku_42('Ur_Poisson',H,t,C4);
        C4=yout; t=tout;              imas=imas+1;
        tt42(imas)=tout;                % Масив моментів часу
        [ps4,te4,ga4]=Mnk2Ug1(C4);
        [psi,tet,ga,omX,omY,omZ]=UgDvObj_laz(t);
        dpsi42(imas)=psi-ps4; dtet42(imas)=tet-te4; dga42(imas)=ga-ga4;
end
           % Обчислення дрейфів
[ConsT,Dr12]=drift_L(tt2,dpsi2);    [ConsT,Dr122]=drift_L(tt22,dpsi22);
[ConsT,Dr131]=drift_L(tt31,dpsi31); [ConsT,Dr13]=drift_L(tt3,dpsi3) ;
[ConsT,Dr14]=drift_L(tt4,dpsi4);    [ConsT,Dr142]=drift_L(tt42,dpsi42);
[ConsT,Dr222]=drift_L(tt22,dtet22); [ConsT,Dr322]=drift_L(tt22,dga22);
[ConsT,Dr231]=drift_L(tt31,dtet31); [ConsT,Dr331]=drift_L(tt31,dga31);
[ConsT,Dr22]=drift_L(tt2,dtet2);    [ConsT,Dr23]=drift_L(tt3,dtet3);
[ConsT,Dr24]=drift_L(tt4,dtet4);    [ConsT,Dr32]=drift_L(tt2,dga2);
[ConsT,Dr33]=drift_L(tt3,dga3);     [ConsT,Dr34]=drift_L(tt4,dga4);
[ConsT,Dr242]=drift_L(tt42,dtet42); [ConsT,Dr342]=drift_L(tt42,dga42);
DR=[Dr12,Dr13,Dr14,Dr22,Dr23,Dr24,Dr32,Dr33,Dr34,...
      Dr122,Dr222,Dr322,Dr131,Dr231,Dr331,Dr142,Dr242,Dr342];
  % Виведення графічної і текстової інформації у графічне вікно монітору
GRAF_Ru_Ku_Laz1(tt2(1:end-1),dpsi2(1:end-1),dtet2(1:end-1),dga2(1:end-1),...
   tt22(1:end-1),dpsi22(1:end-1),dtet22(1:end-1),dga22(1:end-1),...
   tt31(1:end-1),dpsi31(1:end-1), dtet31(1:end-1),dga31(1:end-1),...
   tt3(1:end-1),dpsi3(1:end-1), dtet3(1:end-1),dga3(1:end-1),...
   tt4(1:end-1),dpsi4(1:end-1),dtet4(1:end-1),dga4(1:end-1),...
   tt42(1:end-1),dpsi42(1:end-1),dtet42(1:end-1),dga42(1:end-1))
```

Спеціальна функція drift_L, текст якої наведено нижче, виконує операцію апроксимування масиву даних X при заданому масиві t відповідних аргументів поліномом першого ступеня (прямою на графіку X(t)):

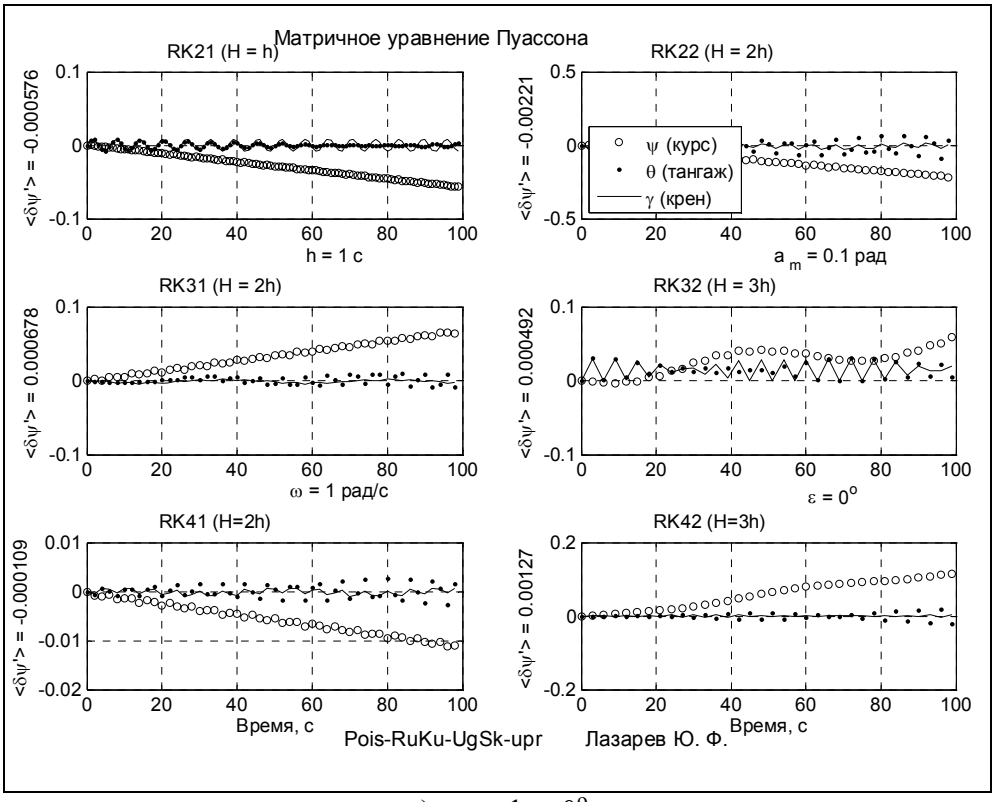
```
function [ConsT,Dr]=drift_L(t,X)
% Процедура обчислення дрейфу функції X часу
% студент Цибульник     04-10-2012
P=polyfit(t,X,1); Dr=P(1);          ConsT=P(2);
```

При цьому виходять значення двох величин: Dr – кутового коефіцієнту прямої і ConsT – ординати точки перетинання прямої з віссю ординат.

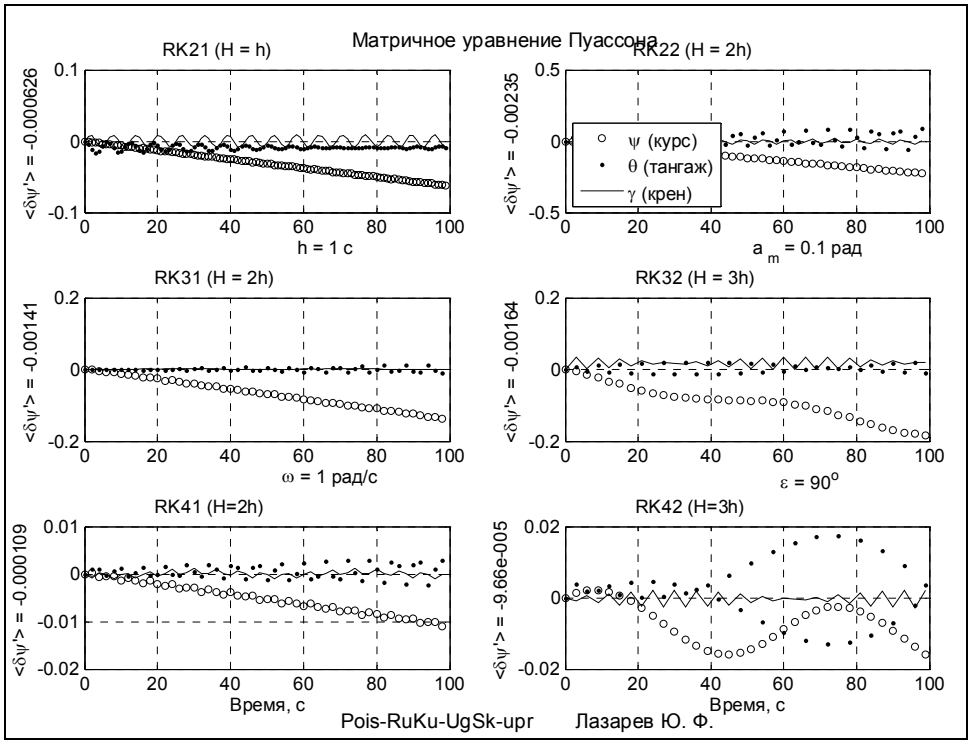
На рисунках 5.1…5.3 показані результати роботи цієї програми за наступних значень параметрів руху

$a_m = 0{,}1$ радіана; $\omega = 0{,}1$ радіан / с.

Рисунок 5.1 подає результати за кроку опитування вимірювачів $h = 1$ с і при двох значеннях зсуву фаз між коливаннями основи $\varepsilon = 0^o$ и $\varepsilon = 90^o$. Аналогічно, рисунки 5.2 и 5.3 демонструють результати при $h = 0{,}1$ та $h = 0{,}01$ с.

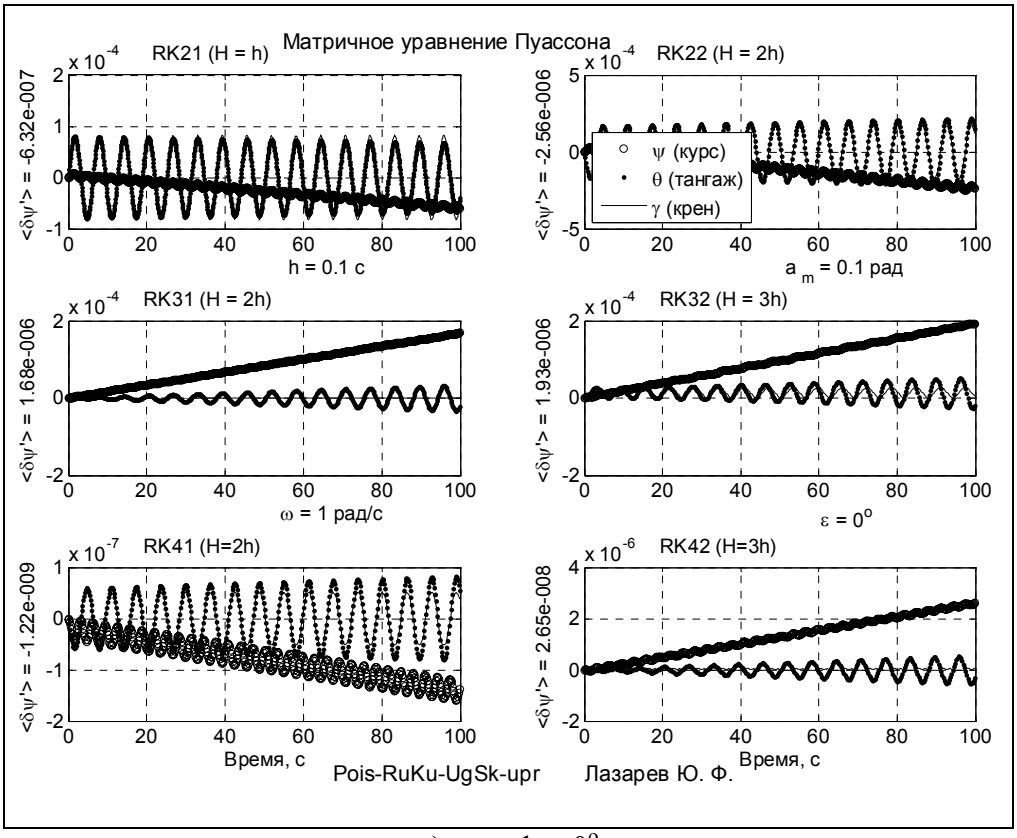


а) зсув фаз 0^o

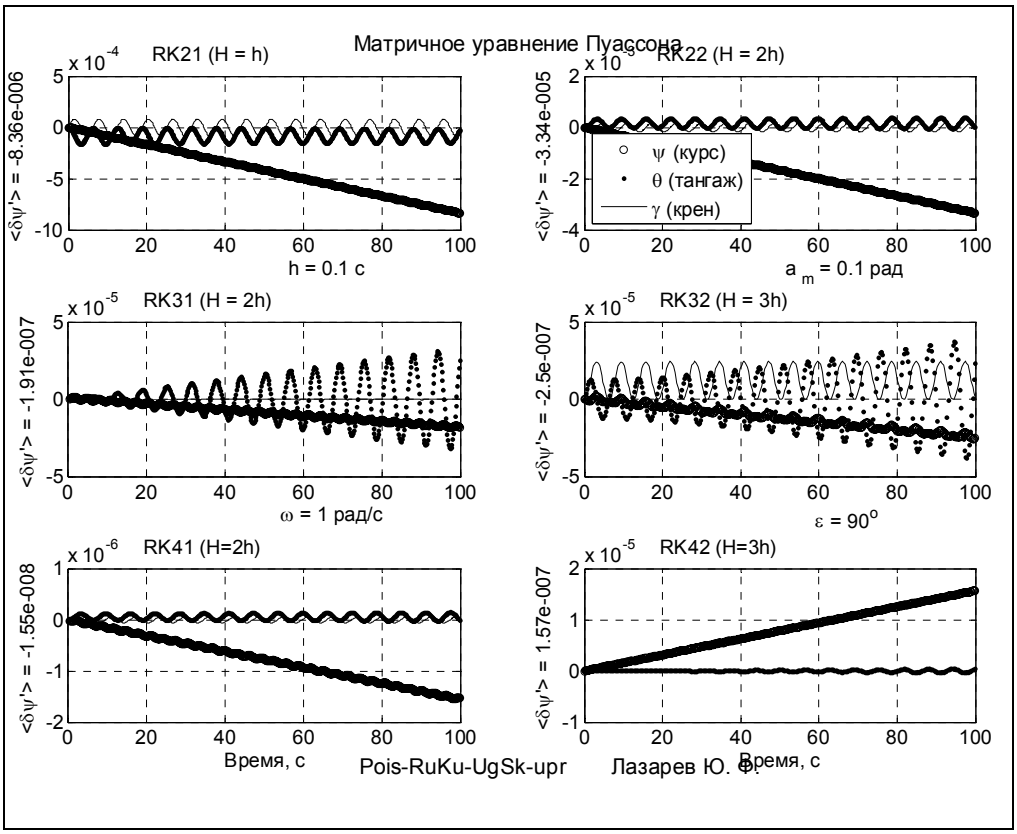


а) зсув фаз 90^o

Рис. 1. Крок опитування 1 с

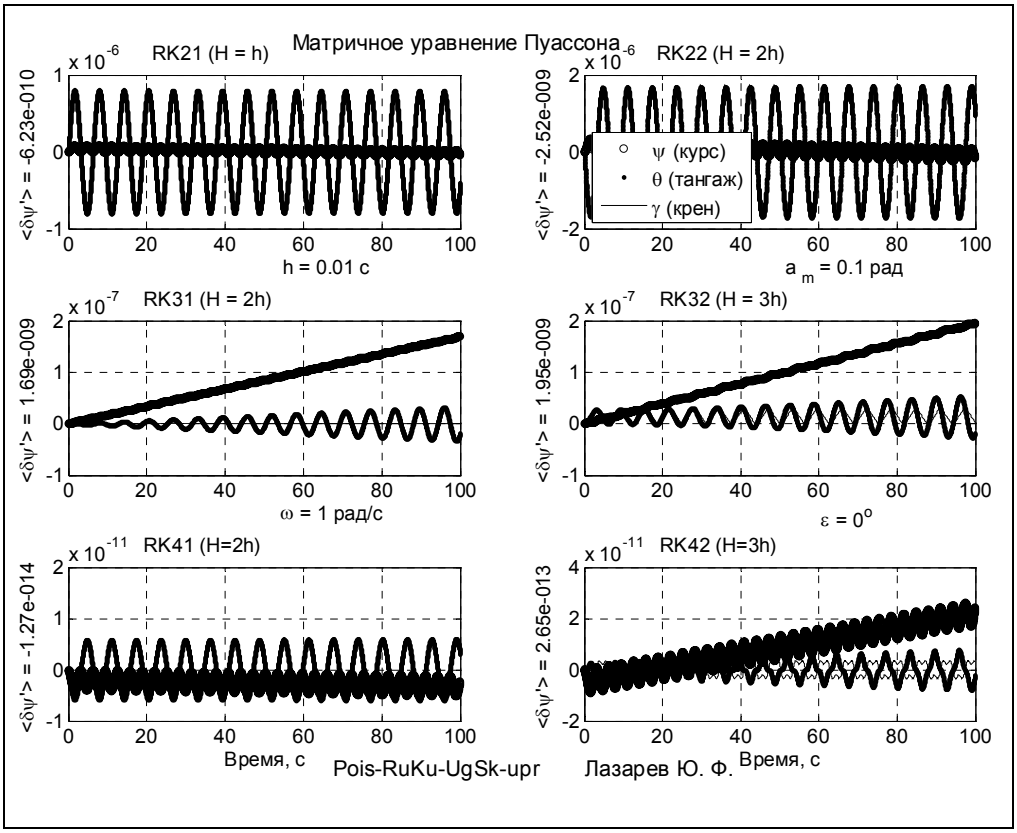


а) зсув фаз 0^o

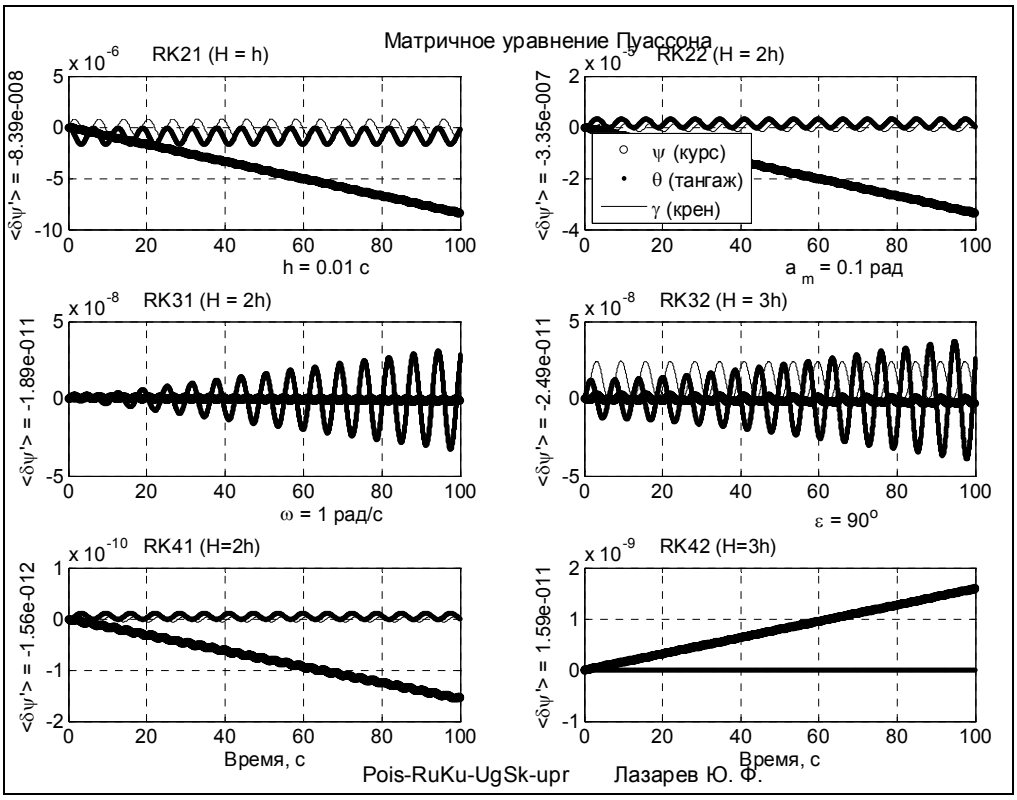


а) зсув фаз 90^o

Рис. 5.2. Крок опитування 0,1 с



а) зсув фаз 0°



а) зсув фаз 90°

Рис. 5.3. Крок опитування 0,01 с

6. Завдання для досліджень

Кожному студенту дається окреме завдання (див. таблицю) на дослідження, в якому вказується:

1) вид рівняння орієнтації, яке інтегрується в обчислювальному пристрої БІСО;
2) три види алгоритмів, що досліджуються на точність.

Таблиця. Варіанти завдань

Варіант	Вид рівняння орієнтації	Види алгоритмів Рунге-Кутти
1	Кінематичні Ейлера	21; 31; 41
2	Пуассона	21; 31; 41
3	Кватерніонне	21; 31; 41
4	Векторне Ейлера	21; 31; 41
5	Векторне Гіббса	21; 31; 41
6	Векторне Родріга	21; 31; 41
7	Кінематичні Ейлера	22; 32; 42
8	Пуассона	22; 32; 42
9	Кватерніонне	22; 32; 42
10	Векторне Ейлера	22; 32; 42
11	Векторне Гіббса	22; 32; 42
12	Векторне Родріга	22; 32; 42

Завдання полягає в наступному:

1) встановити залежність дрейфу похибки визначення куту ψ від зсуву фаз для кожного з заданих алгоритмів при трьох значеннях кроку опитування вимірювачів $h = 1;\ \ 0{,}1;\ \ 0{,}01$ с; побудувати графіки цих залежностей в діапазоні зсуву фаз від -180^o до +180^o через 30^o;

2) встановити залежність максимального (амплітудного при зсуві фаз) значення дрейфу від амплітуди коливань основи;

3) встановити залежність максимального (амплітудного при зсуві фаз) значення дрейфу від частоти коливань основи;

4) встановити залежність максимального (амплітудного при зсуві фаз) значення дрейфу від кроку опитування вимірювачів для кожного з заданих алгоритмів; побудувати графіки цих залежностей в логарифмічному масштабі при $h = 1;\ \ 0{,}4;\ \ 0{,}1;\ \ 0{,}04;\ \ 0{,}01;\ \ 0{,}004;\ \ 0{,}001$ с.

Зробити висновки щодо встановлених залежностей і властивостей дрейфів алгоритмів.

Література

1. Лазарєв Ю. Ф. Кінематика твердого тіла: навчальний посібник. – К.: НТУУ "КПІ", 2014. – 61 с. [електронний ресурс: http://kafpson.kpi.ua/arch.htm]
2. Лазарєв Ю. Ф. Моделювання на ЕОМ: Навчальний посібник. – К.: Корнійчук, 2007. – 290 с.