Приложение

Приложение 1.

Алгоритм вычисления углов поворота сервоприводов исходя из решенной обратной кинематической задачи для всей ноги

```
%перевод из радианов в градусы
rad2deg = pi/180;

%параметры матриц и векторов
AXIS = 3;                    %количество осей декартового
пространства
LEGS = 6;                    %количество ног гексапода
JOINTS = 3;                  %количество шарниров на 1й ноге

%индексы
X = 1;
Y = 2;
Z = 3;
Q1 = 1;
Q2 = 2;
Q3 = 3;
LEG_1 = 1;
LEG_2 = 2;
LEG_3 = 3;
LEG_4 = 4;
LEG_5 = 5;
LEG_6 = 6;
x_axis = [1 0 0];
y_axis = [0 1 0];
z_axis = [0 0 1];

%допустимая погрешность вычислений(+определение нуля)
EPS = 1E-06;

%геометриччческие параметры
RADIUS = 100;                %расстояние отн. BASE до СК0
(правильный шестиугольник)
COXA_LENGHT = 26.0;          %длина таза
FEMUR_LENGHT = 50.0;         %длина бедра
TIBIA_LENGHT = 54.0;         %длина голени
panel_w = [0 0 5];           %толщина основной пластины
head_w = [0 -5 0];           %толщина головы
L1 = COXA_LENGHT  * x_axis;
L2 = FEMUR_LENGHT * x_axis;
L3 = TIBIA_LENGHT * x_axis;

%расположение точек ног на основной панели
CK0_1 = [-RADIUS*cos(50*rad2deg) RADIUS*sin(50*rad2deg) 0];
CK0_2 = [-RADIUS 0 0];
```

```
CK0_3 = [-RADIUS*cos(50*rad2deg) -RADIUS*sin(50*rad2deg) 0];
CK0_4 = [RADIUS*cos(50*rad2deg) RADIUS*sin(50*rad2deg) 0];
CK0_5 = [RADIUS 0 0];
CK0_6 = [RADIUS*cos(50*rad2deg) -RADIUS*sin(50*rad2deg) 0];
CK0_Head =  [0 RADIUS*sin(50*rad2deg) 0];

%расположение точек косых ног
CK1_1 = -1*[-COXA_LENGHT*cos(50*rad2deg)/2
COXA_LENGHT*sin(50*rad2deg)/2 0];
CK2_1 = -1*[-FEMUR_LENGHT*cos(50*rad2deg)/2
FEMUR_LENGHT*sin(50*rad2deg)/2 0];
CK3_1 = -1*[-TIBIA_LENGHT*cos(50*rad2deg)/2
TIBIA_LENGHT*sin(50*rad2deg)/2 0];

CK1_3 = -1*[-COXA_LENGHT*cos(50*rad2deg)/2 -
COXA_LENGHT*sin(50*rad2deg)/2 0];
CK2_3 = -1*[-FEMUR_LENGHT*cos(50*rad2deg)/2 -
FEMUR_LENGHT*sin(50*rad2deg)/2 0];
CK3_3 = -1*[-TIBIA_LENGHT*cos(50*rad2deg)/2 -
TIBIA_LENGHT*sin(50*rad2deg)/2 0];

CK1_4 = [COXA_LENGHT*cos(50*rad2deg)/2
COXA_LENGHT*sin(50*rad2deg)/2 0];
CK2_4 = [FEMUR_LENGHT*cos(50*rad2deg)/2
FEMUR_LENGHT*sin(50*rad2deg)/2 0];
CK3_4 = [TIBIA_LENGHT*cos(50*rad2deg)/2
TIBIA_LENGHT*sin(50*rad2deg)/2 0];

CK1_6 = [COXA_LENGHT*cos(50*rad2deg)/2 -
COXA_LENGHT*sin(50*rad2deg)/2 0];
CK2_6 = [FEMUR_LENGHT*cos(50*rad2deg)/2 -
FEMUR_LENGHT*sin(50*rad2deg)/2 0];
CK3_6 = [TIBIA_LENGHT*cos(50*rad2deg)/2 -
TIBIA_LENGHT*sin(50*rad2deg)/2 0];

%ограничения углов в градусах
Q1_MAX =  30;
Q1_MIN = -30;
Q2_MAX =  80;
Q2_MIN = -80;
Q3_MAX = 90;
Q3_MIN = -90;

%параметры ходьбы
INIT_HEIGHT = -100;     %начальная высота платформы

%координаты векторов управления

C_L_X(1) = 70;
C_L_Y(1) = -85;

C_L_X(2) = 110;
```

```
C_L_Y(2) = 0;

C_L_X(3) = 70;
C_L_Y(3) = 85;

C_L_X(4) = -70;
C_L_Y(4) = -85;

C_L_X(5) = -110;
C_L_Y(5) = 0;

C_L_X(6) = -70;
C_L_Y(6) = 85;

%длины поремещений в декартовых и угловых координатах
DELTA_Y = 15;        %mm
DELTA_Z = 15;        %mm
DELTA_PHI = 9;       %degrees

% задержка на исполнение механической частью системы
DELAY_MS = 250;

%далее сам расчет

%устанавливаем ограничения углов поворота шарниров
maxStateAngle(Q1) = Q1_MAX;
maxStateAngle(Q2) = Q2_MAX;
maxStateAngle(Q3) = Q3_MAX;
minStateAngle(Q1) = Q1_MIN;
minStateAngle(Q2) = Q2_MIN;
minStateAngle(Q3) = Q3_MIN;

%устанавливаем углы поворота СК0 относительно BASE
legBaseAngle(LEG_1) = 130.0;         %угол ню
legBaseAngle(LEG_2) = 180.0;
legBaseAngle(LEG_3) = 230.0;
legBaseAngle(LEG_4) = 50.0;
legBaseAngle(LEG_5) = 0.0;
legBaseAngle(LEG_6) = 310.0;

%массив номеров шарниров
jointNumbers(LEG_1,Q1) = 7;
jointNumbers(LEG_1,Q2) = 6;
jointNumbers(LEG_1,Q3) = 5;
jointNumbers(LEG_2,Q1) = 11;
jointNumbers(LEG_2,Q2) = 10;
jointNumbers(LEG_2,Q3) = 9;
jointNumbers(LEG_3,Q1) = 15;
jointNumbers(LEG_3,Q2) = 14;
jointNumbers(LEG_3,Q3) = 13;
jointNumbers(LEG_4,Q1) = 24;
jointNumbers(LEG_4,Q2) = 25;
```

```
jointNumbers(LEG_4,Q3) = 26;
jointNumbers(LEG_5,Q1) = 20;
jointNumbers(LEG_5,Q2) = 21;
jointNumbers(LEG_5,Q3) = 22;
jointNumbers(LEG_6,Q1) = 16;
jointNumbers(LEG_6,Q2) = 17;
jointNumbers(LEG_6,Q3) = 18;

%обнуление матриц
controlVector = [0 0 0];
rotateBaseToZero = [0 0 0;0 0 0;0 0 0];
controlVectorSC0 = [0 0 0];
currentStateVector = [0 0 0];
currentState = [0 0 0];
calculatedStateVector = [0 0 0];
calculatedState = [0 0 0];
angleRotation = [0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0; 0 0 0];

%решение обратной кинематической задачи для кадой ноги и
нахождение углов
%поворота для каждого двигателя
for i = 1:LEGS,
    %заданный нами контрольный вектор
    controlVector(1) = C_L_X(i);
    controlVector(2) = C_L_Y(i);
    controlVector(3) = INIT_HEIGHT;

    %матрица поворота BASE -> CK0
    rotateBaseToZero(1,1) =  cos((legBaseAngle(i) -
90.0)*rad2deg); %вторая картинка внизу
    rotateBaseToZero(1,2) =  sin((legBaseAngle(i) -
90.0)*rad2deg);
    rotateBaseToZero(2,1) = -sin((legBaseAngle(i) -
90.0)*rad2deg);
    rotateBaseToZero(2,2) =  cos((legBaseAngle(i) -
90.0)*rad2deg);
    rotateBaseToZero(3,3) = 1.0;

    %умножаем матрицу поворота на контрольынй вектор
    controlVectorSC0 = rotateBaseToZero * controlVector';
    controlVectorSC0 = controlVectorSC0';
    %считаем текущий вектор состояния currentstatevector для
нулевых углов
    currentStateVector(X) = sin(currentState(Q1)*rad2deg) *
(COXA_LENGHT - TIBIA_LENGHT * sin(currentState(Q2)*rad2deg) *
sin(currentState(Q3)*rad2deg) + cos(currentState(Q2)*rad2deg) *
(FEMUR_LENGHT + TIBIA_LENGHT * cos(currentState(Q3)*rad2deg)));
    currentStateVector(Y) = cos(currentState(Q1)*rad2deg) *
(COXA_LENGHT - TIBIA_LENGHT * sin(currentState(Q2)*rad2deg) *
sin(currentState(Q3)*rad2deg) + cos(currentState(Q2)*rad2deg) *
(FEMUR_LENGHT + TIBIA_LENGHT * cos(currentState(Q3)*rad2deg)));
```

```
    currentStateVector(Z) = - TIBIA_LENGHT *
cos(currentState(Q2)*rad2deg) * sin(currentState(Q3)*rad2deg) -
sin(currentState(Q2)*rad2deg) * (FEMUR_LENGHT + TIBIA_LENGHT *
cos(currentState(Q3)*rad2deg));

    calculatedStateVector(X) = currentStateVector(X);
    calculatedStateVector(Y) = currentStateVector(Y);
    calculatedStateVector(Z) = currentStateVector(Z);

    %скалдываем две матрицы и получаем итоговые вектор положения
конца всей конечности
    calculatedStateVector = controlVectorSC0 +
currentStateVector;

    %расчет первого угла (таза)
    calculatedState(Q1) =
(atan2(calculatedStateVector(X),calculatedStateVector(Y)))/rad2d
eg;

    %расчет второго и третьего угла (бедро и голень)

    %расчет локальных координат конца ноги(центра второй
окружности)
    localX = sqrt(calculatedStateVector(X) *
calculatedStateVector(X) + calculatedStateVector(Y) *
calculatedStateVector(Y)) - COXA_LENGHT;
    localY = calculatedStateVector(Z);

    %объявляем координаты точки пересечения
    intersectionX = 0.0;
    intersectionY = 0.0;
    %находим коэффициенты уравнения прямой
    A = -2.0 * localX;
    B = -2.0 * localY;
    C = localX * localX + localY * localY + FEMUR_LENGHT *
FEMUR_LENGHT - TIBIA_LENGHT * TIBIA_LENGHT;
    %расчет координат точки пересечения окружностей
    x0 = -A*C/(A * A + B * B);
    y0 = -B*C/(A * A + B * B);
    aaa_bigger = FEMUR_LENGHT * FEMUR_LENGHT * (A * A + B * B) +
EPS;
    aaa_smaller = C* C;
    if (C * C > FEMUR_LENGHT * FEMUR_LENGHT * (A * A + B * B) +
EPS)
        %нет пересечений
        abcd = 0;
        funcRes = false;
    elseif (abs(C * C - FEMUR_LENGHT * FEMUR_LENGHT * (A * A + B
* B)) < EPS)
        %1 пересечение(касательная)
        intersectionX = x0;
        intersectionY = y0;
```

```
    else
        %2 пересечения
        D = FEMUR_LENGHT * FEMUR_LENGHT - C * C / (A * A + B *
B);
        mult = sqrt (D / (A * A + B * B));
        Ax = x0 + B * mult;
        Bx = x0 - B * mult;
        Ay = y0 - A * mult;
        By = y0 + A * mult;
        %выбор точки пересечения
        if (Ay < By)
            intersectionX = Ax;
            intersectionY = Ay;
        else
            intersectionX = Bx;
            intersectionY = By;
        end
    end
    %расчет углов Q2, Q3 исходя из координат точки в локальной
СК
    calculatedState(Q2) = (atan2(-1*intersectionY,
intersectionX))/rad2deg;
    calculatedState(Q3) = (atan2(intersectionY - localY, localX
- intersectionX))/rad2deg - calculatedState(Q2);

    %задаем расчитанные углы
    angleRotation(i,1) = calculatedState(Q1);
    angleRotation(i,2) = calculatedState(Q2);
    angleRotation(i,3) = calculatedState(Q3);

    %корректировка левой стороны HEXY
    if i < 4
        angleRotation(i,1) = -angleRotation(i,1);
        angleRotation(i,2) = -angleRotation(i,2);
        angleRotation(i,3) = -angleRotation(i,3);
    end
end

angleRotation = angleRotation + [-50 0 0; 0 0 0; 50 0 0; 50 0 0;
0 0 0; -50 0 0];
```

Приложение 2.

Код программы поиска кратчайшего маршрута

```
int N[10][10]; //размер нашей карты

	struct trajectory
	{
		int X;
		int Y;
	};
	typedef struct trajectory Path;
	Path root[50]; //назначаем массив с координамтами точек от первой до последней по которым надо идти, чтобы пройти маршрут по траектории

	int map[][10] = {
		{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
		{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },
		{ 0, 1, 1, 0, 0, 0, 1, 1, 1, 0 },
		{ 0, 1, 1, 1, 1, 0, 1, 1, 1, 0 },
		{ 0, 1, 1, 1, 1, 0, 0, 1, 1, 0 },
		{ 0, 1, 1, 1, 1, 1, 0, 1, 1, 0 },
		{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },
		{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },
		{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },
		{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
	};//карта местности, где "0" - пройти нельзя, "1" - пройти можно

	map[7][3] = -1; //начало маршрута
	map[3][6] = -2; //конец маршрута

	for (int i = 0; i < 10; i++)
	{
		for (int j = 0; j < 10; j++)
		{
			N[i][j] = map[i][j];
			if (map[i][j] == 0)
				N[i][j] = 100; //назначаем непроходимым клеткам значение 100
			if (map[i][j] == 1)
				N[i][j] = 150; //назначем проходимым клеткам значение 150
			if (map[i][j] == -2)
				N[i][j] = 150; //назначаем концу пути 150
			if (map[i][j] == -1)
				N[i][j] = 0; //назначаем началу пути 0
		}
	}

	for (int iter = 0; iter < 100; iter++) //строим волну, начиная из начала пути
	{
		for (int i = 0; i < 10; i++)
		{
			for (int j = 0; j < 10; j++)
			{
			  if (N[i][j] == iter)
			  {
			    if ((i + 1)<10 && N[i + 1][j] != 100 && N[i + 1][j] > iter + 1)
			      N[i + 1][j] = iter + 1;
			    if ((i - 1) >= 0 && N[i - 1][j] != 100 && N[i - 1][j] > iter + 1)
			      N[i - 1][j] = iter + 1;
			    if ((j + 1)<10 && N[i][j + 1] != 100 && N[i][j + 1] > iter + 1)
			      N[i][j + 1] = iter + 1;
```

```
                        if ((j - 1) >= 0 && N[i][j - 1] != 100 && N[i][j - 1] > iter + 1)
                           N[i][j - 1] = iter + 1;

                      }
                  }
            }
      }

      if (N[3][6] == 150) //проверка на возможность прохождения пути
      {
            printf("Can't make the route!");
            std::cin.get();
      }

      int znch0 = N[3][6]; //определяем колиество точек в маршруте
      root[N[3][6]].X = 3; //координаты конца маршрута по X
      root[N[3][6]].Y = 6; //координаты конца маршрута по У
      N[3][6] = -1; //задаем концу маршрута значение -1

      for (int znch = (znch0 - 1); znch >= 0;) //ищем путь из конца в начало
      {
            for (int i = 0; i < 10; i++)
            {
                  for (int j = 0; j < 10; j++)
                  {
                        if (N[i][j] == znch) //как только находим значение, на 1
меньшее чем предыдущее, тогда проверяем..
                        {
                              if (N[i + 1][j] == -1 || N[i - 1][j] == -1 || N[i][j +
1] == -1 | N[i][j - 1] == -1)//..чтобы клетка с этим значением была рядом по расположению
с предыдущей клеткой..
                              {
                              root[N[i][j]].X = i; //..если это совпадает, и
следующая клетка рядом с предыдщей и ее значение меньше на единицу, то записываем
координаты этой новой точки
                              root[N[i][j]].Y = j;
                              N[i][j] = -1; //задаем клетке значение -1 и переходим к
следующей точке и так, пока не придем в начало маршрута
                              znch = znch - 1;
                              }
                        }
                  }
            }
      }

      int orient[2];      //массив с ориентацией робота в начальной точке
      int coord[2];       //массив с координатами точки
      int movements[100]; //массив со всеми движениями
      orient[0] = -1;     //ориентация робота..
      orient[1] = 0;      //..на свевер
      int g,i;
      for (i = 0, g = 0; i < znch0;) //определяем, какие команды подаем роботу
(налево(1), прямо(2) или направо(3))
      {
            coord[0] = root[i + 1].X - root[i].X;
            coord[1] = root[i + 1].Y - root[i].Y;
            if (coord[0] == orient[0] && coord[1] == orient[1])
            {
                  movements[g] = 2; //2 - идем прямо
                  i++;
                  g++;
            }
            else if (coord[1] == orient[0] && orient[1] == 0)
            {
```

```
                        movements[g] = 1; //1 - поворачиваем налево при этих условиях и
возвращаемся в начало алгоритма, чтобы пойти прямо
                        orient[0] = 0;
                        orient[1] = coord[1];
                        g++;
                }
                else if (coord[0] == orient[1] && orient[0] == 0)
                {
                        movements[g] = 3; //3 - поворачиваем направо при этих условиях и
возвращаемся в начало алгоритма, чтобы пойти прямо
                        orient[0] = coord[0];
                        orient[1] = 0;
                        g++;
                }
                else if (coord[1] == -orient[0] && orient[1] == 0)
                {
                        movements[g] = 3; //3 - поворачиваем направо при этих условиях и
возвращаемся в начало алгоритма, чтобы пойти прямо
                        orient[0] = 0;
                        orient[1] = coord[1];
                        g++;
                }
                else if (coord[0] == -orient[1] && orient[0] == 0)
                {
                        movements[g] = 1; //1 - поворачиваем налево при этих условиях и
возвращаемся в начало алгоритма, чтобы пойти прямо
                        orient[0] = coord[0];
                        orient[1] = 0;
                        g++;
                }
        }

        for (int i = 0; i < g; i++)
        {
                printf("%2d ", movements[i]);
                std::cout << std::endl;
        }               //выводим попорядку команды для робота, где 1 - поворот налево, 2 -
идем прямо, 3 - поворот направо

        std::cin.get();
```

Приложение 3.

Маршрутно-технологическая карта сборки ноги шестиногого шагающего робота

№ опер.	**Название операции**	**Содержание операции**	**Оборудование**	**Приспособление**	**Инструмент**	**Время на операцию (минуты)**
10	Монтажная	1. Зажать деталь Д5 горизонтально в тисках 2. Вставить в деталь Д5 два сервопривода СП1 3. Вставить в деталь Д5 две детали Д6 4. Надеть на два детали Д6 и два сервопривода СП1 деталь Д7 5. Продеть болт М3х35 через отверстия в деталях Д5 и Д7 и скрепить конструкцию гайкой М3	Монтажный стол	Тиски 7200-0203 ГОСТ 16518-96	Отвертка 7810-0976 ГОСТ 10753-86	2

№ опер.	Название операции	Содержание операции	Оборудование	Приспособление	Инструмент	Время на операцию (минуты)
15	Монтажная	1. На деталь Д2 надеть деталь Д1 2. Прикрутить к детали Д3 с помощью двух винтов 2-2,5х6 деталь Д12 3. Деталь из пункта 2 скрепить с деталью из пункта 1 болтом М3х14 и гайкой М3 4. К одному из приводов СП1 прикрутить деталь из пункта 3 с помощью винта 2-2,5х6	Монтажный стол		Отвертка 7810-0976 ГОСТ 10753-86	1.5
20	Монтажная	1. Продеть в дальнее от центра детали Д4 отверстие болт М3х14 и закрепить его гайкой М3	Монтажный стол	Тиски 7200-0203 ГОСТ 16518-96	Отвертка 7810-0976 ГОСТ 10753-86	1.5

№ опер.	Название операции	Содержание операции	Оборудование	Приспособление	Инструмент	Время на операцию (минуты)
		2. Смазать болт М3х14 смазкой ГОСТ 21150-87 3. Прикрепить деталь Д4 к детали Д2 с помощью болта М3х14 и гайки М3 4. Вытащить деталь из тисков				
25	Монтажная	1. Закрепить деталь Д9 горизонтально вверх выгравированным кругом в тисках 2. Вставить в ближнее к выгравированному кругу отверстие детали Д9 деталь Д10 3. Закрепить конструкцию с помощью болта М3х14 и	Монтажный стол	Тиски 7200-0203 ГОСТ 16518-96	Отвертка 7810-0976 ГОСТ 10753-86	2

№ опер.	Название операции	Содержание операции	Оборудование	Приспособление	Инструмент	Время на операцию (минуты)
		гайки М3 4. Прикрутить деталь Д12 с помощью двух винтов 2-2,5х6				
30	Монтажная	1. В деталь Д11 вставить сервопривод СП1 в положении шестерня ближе к круглому отверстию детали Д11 2. Деталь Д11 вставить в деталь из операции №25 3. Надеть на детали Д11 и Д10 деталь Д8 выгравированным кругом внутрь 4. Закрепить деталь Д8 двумя болтами М3х14 и гайками	Монтажный стол		Отвертка 7810-0976 ГОСТ 10753-86	1

№ опер.	Название операции	Содержание операции	Оборудование	Приспособление	Инструмент	Время на операцию (минуты)
		М3 5. Продеть болт М3х14 в отверстие детали Д8 и закрепить его гайкой М3 6. Смазать болт М3х14 смазкой ГОСТ 21150-87				
35	Монтажная	1. Вытащить деталь из тисков 2. Соединить деталь из операции №30 и деталь из операции №25 через деталь №12 и сервопривод СП1 с помощью винта 2-2,5х6	Монтажный стол	Тиски 7200-0203 ГОСТ 16518-96	Отвертка 7810-0976 ГОСТ 10753-86	1
40	Контрольная	1. Подсоединяем плату управления роботом к ЭВМ 2. Подсоединяем один из приводов СП1 к плате	Монтажный стол ЭВМ			2

№ опер.	Название операции	Содержание операции	Оборудование	Приспособление	Инструмент	Время на операцию (минуты)
		управления. 3. Проверяем привод на работоспособность 4. Повторяем пункты 2 и 3 для остальных двух приводов				

Приложение 4.

Блок-схема волнового алгоритма Ли

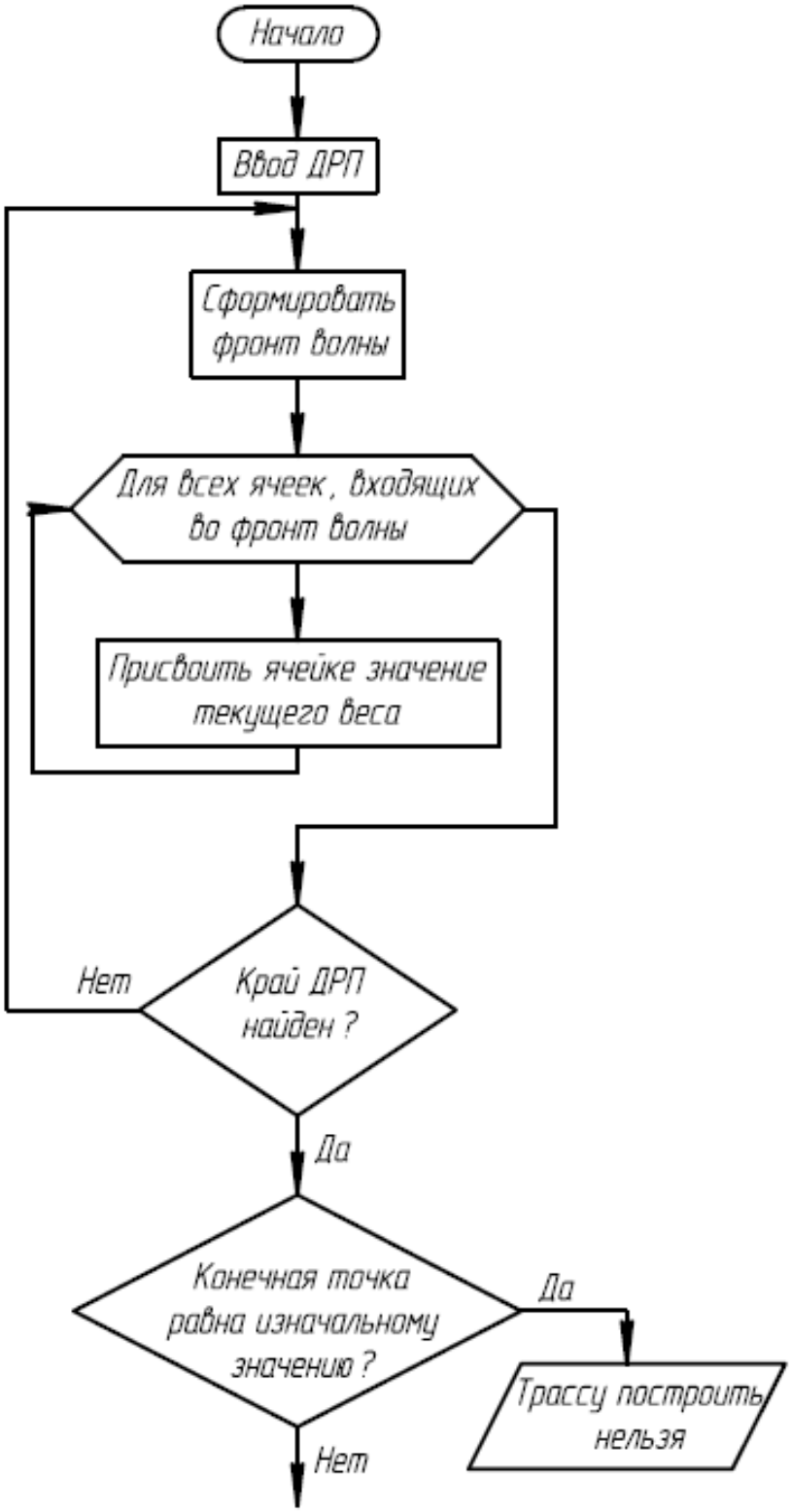

Рис.1. Начало блок-схемы волнового алгоритма

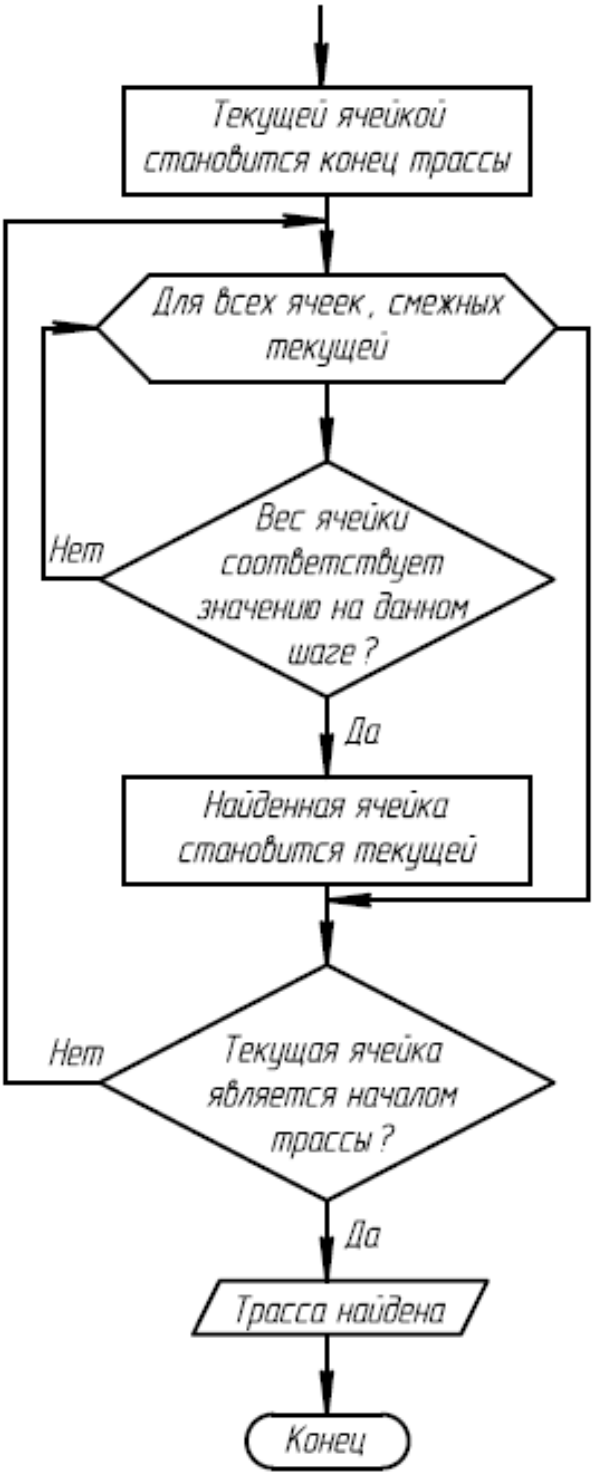

Рис. 2. Конец блок-схемы волнового алгоритма