# 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;

%геометричческие 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)))/rad2deg;

%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;

}

}

}

}

int znch0 = N[3][6]; //определяем колиество точек в маршруте

root[N[3][6]].X = 3; //координаты конца маршрута по Х

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();