Basic Walking Simulations and Gravitational Stability Analysis for a Hexapod Robot Using Matlab

Sorin Mănoiu-Olaru*, Mircea Nițulescu **

**Department of Automation, Electronics and Mechatronics, University of Craiova, Romania (e-mail:manoiusorin2006@yahoo.com)*
*** Department of Automation, Electronics and Mechatronics, University of Craiova, Romania (e-mail:nitulescu@robotics.ucv.ro)*

Abstract: In this paper the authors present a software program to simulate hexapod robot stability in gravitational field for a certain configuration of legs and some basic walking simulations using Matlab software package. First the complete kinematical model of the leg is calculated and the direct dynamical model is presented. The kinematical model of the leg was obtained using Denavit – Hartenberg algorithm. A workspace analysis of the leg is made in order to analyze collision during walking and impose the necessary constrains. Then a hexapod robot structure using this kind of leg is presented and simulated using Matlab software package. A virtual simulation platform was created for the robot in order to simulate robot static stability and basic movement algorithm. The analysis of the robot static stability is made for different cases of locomotion on horizontal surface and for different leg configuration. Also the simulation platform allows connection of a physical leg to the computer through Arduino Duemilanove development board in order to simulate different movement algorithms and test the functionality of the structure. This part is an experimental one and will be improve in the future. Moving the leg tip from one point to another is made in two phases: stance phase and swing phase. The paper includes some experimental results related to the static gravitational stability depending on the support polygon, single leg control and some basic walking simulations.

Keywords: Matlab, gravitational stability, Denavit-Hartenberg representation, model, hexapod.

1. INTRODUCTION

The nature invented the leg and humans invented the wheel. In nature, most arthropods have six legs to easily maintain static stability, and it has been observed that a larger number of legs do not increase walking speed. Moreover, hexapod robots show robustness in case of leg faults. For these reasons, hexapod robots have attracted considerable attention in recent decades (Bensalem et. al. 2009).

Most of the earth's surface is inaccessible to regular vehicles. Today's robots are mostly designed for traveling over relatively smooth, leveled or inclined surfaces.

The terrain in question is either outdoor environments, that is generally considered difficult for mobile robots, or indoor environments like staircases, doorsteps or tight corners can cause difficulties.

An important drawback of legged machines is the complexity of the control required to achieve walking even on completely flat and horizontal surface in which much simpler wheeled machines work perfectly well (Carbone 2005). The difficulty is not moving the individual legs, but in coordination of the movement of the legs and the body. The legged robot control system must generate a sequence of leg and body motions, a gait, which will propel it along desired trajectory.

Gait generation is the formulation and selection of a sequence of coordinated leg and body motions that propel the robot along the desired path.

The most studied problem for multi-legged robots concerns how to determine the best sequence for lifting off and placing the feet. Hexapod gaits have been widely studied as a function of robot characteristics.

The large diversity of the existing walking animals offers innumerable examples of the possibilities of this type of locomotion.

The gait analysis and selection requires an appreciable modeling effort for the improvement of mobility with legs in structure/unstructured environments. Nowadays studies are focused primarily on using artificial neural networks, fuzzy logic or central pattern generators for both leg coordination and leg control.

It is well known that to maintain a structure's position in a three dimensional space requires three point of support. Machines with three or more legs continuously in contact with the ground are said to be statically balanced if they maintain their projection of the centre of gravity within the polygon determined by the legs on the support plane. The polygon is known as "the support polygon" (Fig. 1). One of the most studied problem for multilegged robots concerns the stability analyze during lifting off and placing the legs. The motion of legged robots can be divided into statically and dynamically stable. Static sta bility means that the robot is stable at all times during its gait cycle. Dynamic stability means that the robot is only stable when it is moving. For legged robots, static stability demands that the robot has at least three legs on the ground at all times and the robot's centre of mass is inside the support polygon, i.e. the convex polygon formed by the feet supporting the robot (Fig. 1).

On the left side four legs provide support and the centre of mass is located inside the support polygon so the robot is statically stable. On the middle the bottom left leg has been lifted, putting the centre of mass outside the support polygon which made the robot unstable. On the right side three legs provide support and the centre of mass is located on one side of the support polygon. This case is called critical stability.

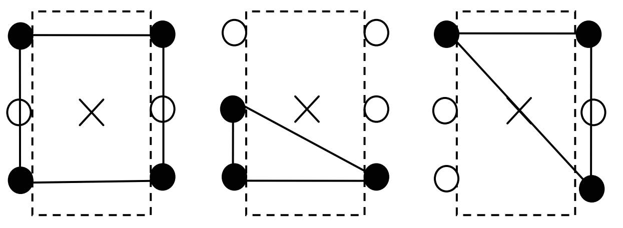


Fig. 1. Stability cases for a hexapod robot: stable, unstable critically.

Arduino Duemilanove is a development board equipped with an AVR ATMEGA328 microcontroller. Today all the microcontrollers are made with CMOS technology because of the large density of integration at a lower cost

AVR uses Harvard architecture, with separate memories and buses for program and data.

Some advantages of using Arduino Duemilanove are:

-easy to program. Programming environment is easy-to-use for both beginners and advanced programmers

-open source. Lots of code libraries are available for free for a wide range of external components (sensors, actuator, LCD).

2. ROBOTIC LEG MODEL

The successful design of a legged robot depends mostly on the design of the chosen leg. Since all aspects of walking are ultimately governed by the physical limitations of the leg, it is important to select a leg that will allow a maximum range of motion and that will not impose unnecessary constraints on the walking.

2.1 Direct Kinematics

A three-revolute kinematical chain has been chosen for each leg mechanism in order to mimic the leg structure (Fig. 2). A direct geometrical model for each leg mechanism is formulated between the moving frame $O_i(x_i,y_i,z_i)$ of the leg base, where i=1…3, and the fixed frame $O_G(X_G,Y_G,Z_G)$.

The coordinate frames for the robot legs are assigned as in fig. 2. The assignment of link frames follows the Denavit-Hartenberg direct geometrical modeling algorithm.

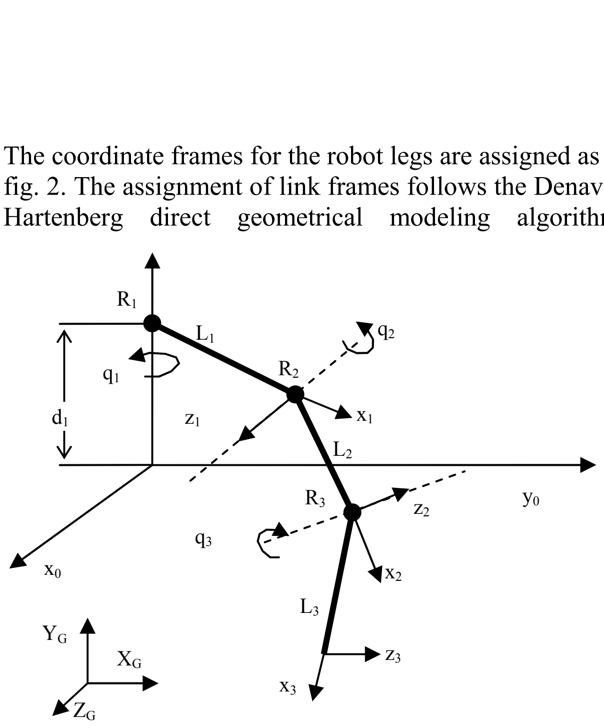


Fig. 2. Model and coordinates frame for leg kinematics.

The robot leg frame starts with link 0 which is the point on the robot where the leg is attached; link 1 is the coxa, link 2 is the femur and link 3 is the tibia. Legs are distributed symmetrically about an axis in the direction of motion (Y in this case). The general form for the transformation matrix from link i to link i-1 using Denavit Hartenberg parameters [Schilling 1990] is given in (1):

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cdot \cos\alpha_i & \sin\theta_i \cdot \sin\alpha_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cdot \cos\alpha_i & -\cos\theta_i \cdot \sin\alpha_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1)$$

The transformation matrix is a series of transformations:

1. Translate d_i along z_{i-1} axis,
2. Rotate θ_i about z_{i-1} axis,
3. Translate a_i along x_{i-1} axis,
4. Rotate α_i about x_{i-1} axis.

The overall transformation is obtained as a product between three transformation matrixes:

$$T_{coxa}^{base} = T_{coxa}^{femur} T_{femur}^{tibia} T_{tibia}^{base} \qquad (2)$$

Considering fig. 2 and using (2) the coordinates of the leg tip are:

$$\begin{aligned} x &= \cos\theta_1 \cdot (L_1 + L_2 \cdot \cos\theta_2 + L_3 \cdot \cos(\theta_2 - \theta_3)) \\ y &= \sin\theta_1 \cdot (L_1 + L_2 \cdot \cos\theta_2 + L_3 \cdot \cos(\theta_2 - \theta_3)) \\ z &= d_1 + L_2 \cdot \sin\theta_2 + L_3 \cdot \sin(\theta_2 - \theta_3) \end{aligned} \qquad (3)$$

where:

d_1 is the distance from the ground to coxa joint.

L_i are the lengths of the leg links.

The centre of mass of each link is positioned relative to the link frame by a position vector $p_i = [x_i, y_i, z_i, 1]^T$. To find the position of the centre of mass of each link relative to leg frame, the coordinates p_i are multiplied with the D-H transformation giving the centre of mass positions:

$$p_{CoM_i} = T_0^i p_i, i = 1...3 \tag{4}$$

The position of centre of mass of the leg is calculated using the following equations:

$$x_g = \frac{\sum_{i=1}^{3} x_i \cdot m_{li}}{\sum_{i=1}^{3} m_{li}}, y_g = \frac{\sum_{i=1}^{3} y_i \cdot m_{li}}{\sum_{i=1}^{3} m_{li}}, z_g = \frac{\sum_{i=1}^{3} z_i \cdot m_{li}}{\sum_{i=1}^{3} m_{li}} \tag{5}$$

where m_{li} is the mass of link i.

2.2 Inverse Kinematics

The geometrical model described above establishes a connection between the joint variable and the position and orientation of the end frame. The inverse kinematics problem consists of determining the joint angles from a given position and orientation of the end frame. The solution of this problem is important in order to transform the motion assigned to the end frame into the joint angle motions corresponding to the desired end frame motion.

The goal is to find the three joint variables θ_1, θ_2, and θ_3 corresponding to the desired end frame position. The end frames orientation is not an issue, since we are only interested in its position.

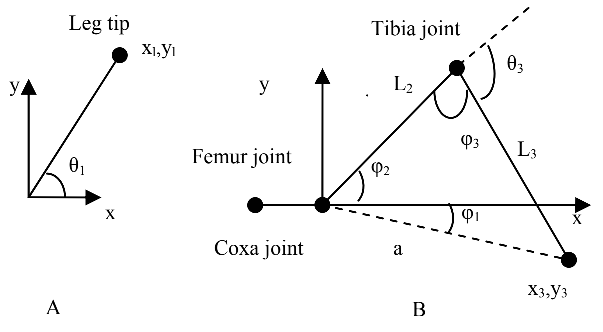


Fig. 3. Illustrations for solving inverse kinematics.

Using (3) and considering the following constraints: all joints allow rotation only about one axis, femur and tibia always rotate on parallel axes, and the physical limitation of each joint we can determine the joint angle.

The coxa joint angle can be found using atan2(y,x) function as can be seen from fig.3.A.

$$\theta_1 = a\tan 2(y_l, x_l) \tag{6}$$

In order to determine the other two angles a geometrical approach was considered.

To further simplify the approach the leg tip coordinates were transformed to coxa frame using the transformation matrix below:

$$T_{coxa}^{femur} = \begin{pmatrix} \left(R_{femur}^{coxa}\right)^T & -\left(R_{femur}^{coxa}\right)^T \bullet d_{femur}^{coxa} \\ 0 & 1 \end{pmatrix} \tag{7}$$

The angle φ_2 which is the angle relating to the femur servo position, can be derived directly from the triangle (fig. 3.B):

$$\theta_2 = \varphi_2 \tag{8}$$

The angle φ_1 is the angle between the x-axis and line *a* and can be calculated with atan2 function:

$$\varphi_1 = a\tan 2(y_3, x_3) \tag{9}$$

where x_3 and y_3 are the leg tip coordinates in coxa frame.

If we consider φ_t to be the entire femur span and apply the law of cosines results:

$$\varphi_t = \text{acos}\left(\frac{L_2^2 + a^2 - L_3^2}{2 \cdot L_2 \cdot a}\right) \tag{10}$$

where:

$$a = \sqrt{x_3^2 + y_3^2} \tag{11}$$

Next the femur angle can be found from:

$$\theta_2 = \text{acos}\left(\frac{L_2^2 + a^2 - L_3^2}{2 \cdot L_2 \cdot a}\right) + a\tan 2(y_3, x_3) \tag{12}$$

Again, applying the law of cosines we find the φ_3 angle:

$$\varphi_3 = \text{acos}\left(\frac{L_2^2 + L_3^2 - a^2}{2 \cdot L_2 \cdot L_3}\right) \tag{13}$$

Considering fig. 3B we see that θ_3 can be found as follows:

$$\theta_3 = \pi - \varphi_3 \tag{14}$$

2.3 Dynamic Model

The purpose of robot dynamics is to determine the generalized forces required to not only overcome the self-inertial loads (due to the weight of the links inside the robot) but also to produce the desired input motions to the robot's mechanism (Fahimi 2008).

In order to obtain a more precise model we divided the mass of each link in two (M_i- servomotor mass, m_i-link mass, $M_i > m_i$) (Fig. 4).

Considering the generalized coordinates vector q= $[q_1, q_2, q_3]^T$ the generalized vector forces can be computed using the below equation:

$$\tau_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \tag{15}$$

where: $L(q,\dot{q}) = E_c(q,\dot{q}) - E_p(q)$.

Considering that all three servomotors have the same mass M_1=M_2=M_3=M and the last two links have the same

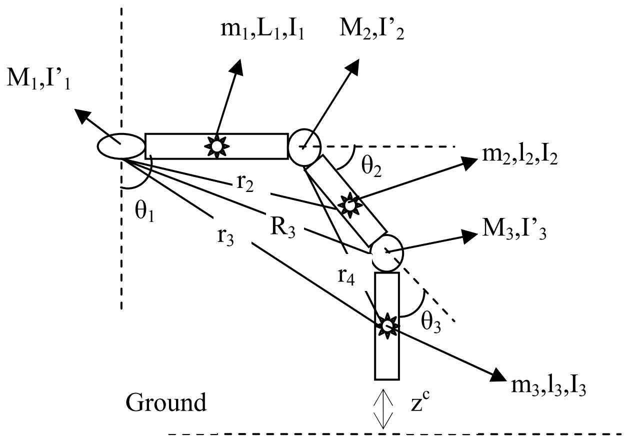


Fig. 4. Illustrations for developing dynamic model of the leg.

mass $m_2=m_3=m_2$ but different lengths the expressions for generalized forces are:

$$\tau_1 = \ddot{\theta}_1 \cdot \left(I'_1 + I''_1 + M \cdot \left(l_1^2 + R_3^2 \right) + m_2 \cdot \left(r_2^2 + r_3^2 \right) \right) + \dot{\theta}_1 \cdot \left(M \cdot \dot{R}_3^2 + m_2 \cdot \left(\dot{r}_2^2 + \dot{r}_3^2 \right) \right) \tag{16}$$

$$\tau_2 = \ddot{\theta}_2 \left(I'_2 + I''_2 + M \cdot l_2^2 + m_3 \cdot r_4^2 \right) + m_3 \cdot \dot{r}_4^2 \cdot \dot{\theta}_2 - \{g \cdot [l_3 \cdot \cos(\theta_2 + \theta_3) \cdot (3 \cdot M + m_1 + \frac{3 \cdot m_2}{2}) + l_2 \cdot \cos(\theta_2) \cdot (2 \cdot M + m_1 + \frac{m_2}{2})]\} \tag{17}$$

$$\tau_3 = \ddot{\theta}_3 \cdot \left(I'_3 + I''_3 \right) - g \cdot l_3 \cdot \cos(\theta_2 + \theta_3) \cdot \left(3 \cdot M + m_1 + \frac{3 \bullet m_2}{2} \right) \tag{18}$$

where:

I'_i -are the moments of inertia associated with the servomotors;

I''_i -are the moments of inertia associated with the links;

r_i -radius of instantaneous circle of rotation of the centre of mass associated with the link i of the leg, i=2...4;

R_3 -radius of instantaneous circle of rotation of the servomotor 3.

3. WORKSPACE ANALYSIS of the LEG

There are several things that are important when defining the leg workspace. A basic one is that the workspace should not be larger than the reach of the leg. Workspace must be define in order to maximize motion but in the same time to minimize singularities or any other pitfalls (Maki 2007).

Workspace is also useful for walking algorithm to prevent collision of adjacent legs by imposing restrictions for direct or inverse kinematics.

The actuators used on the legs are capable of rotating 180 degrees. In order to minimize leg collisions the coxa angle was limited between $-\pi/4$ and $\pi/4$. This restriction was introduced due to mechanical construction of the leg. To get the most out of motion the operation space of variable θ_2 was set from $-\pi/2$ to $\pi/4$ and θ_3 from 0 to $3*\pi/4$. In our case there is a singularity when the x and y coordinates of the leg tip are zero. In this configuration the value for θ_1 is arbitrary. To avoid this scenario we can define the x coordinate greater than zero.

Physical limitations due to mechanical construction for each joint angle are:

$q_{coxa} = [-\pi/4, \pi/4]$,
$q_{femur} = [-\pi/4, \pi/2]$,
$q_{tibia} = [0, 3*\pi/4]$.

The defined workspace with the above limitation is presented in fig. 5, 6 and 7.

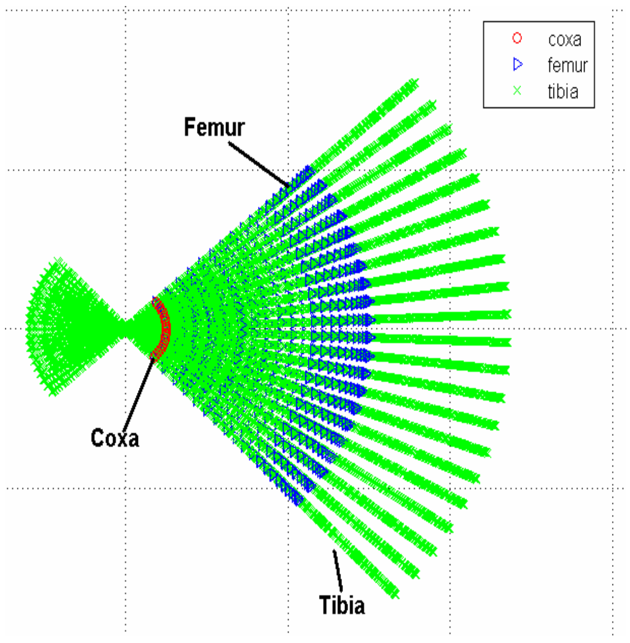


Fig. 5. Leg workspace (XY view).

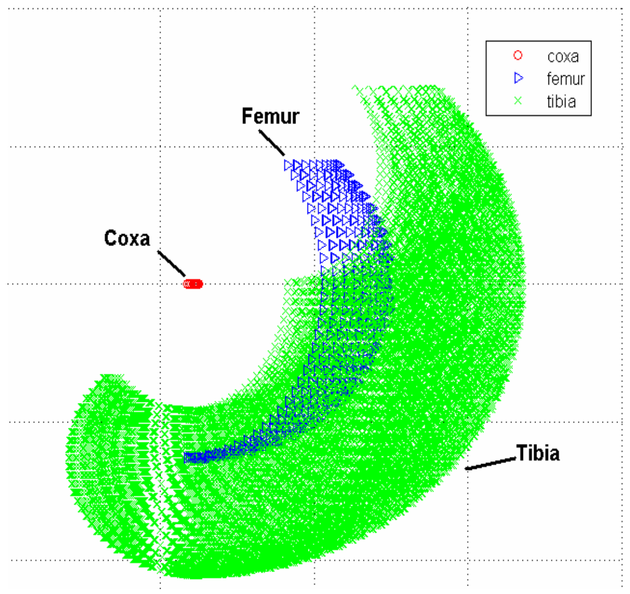


Fig. 6. Leg workspace (XZ view).

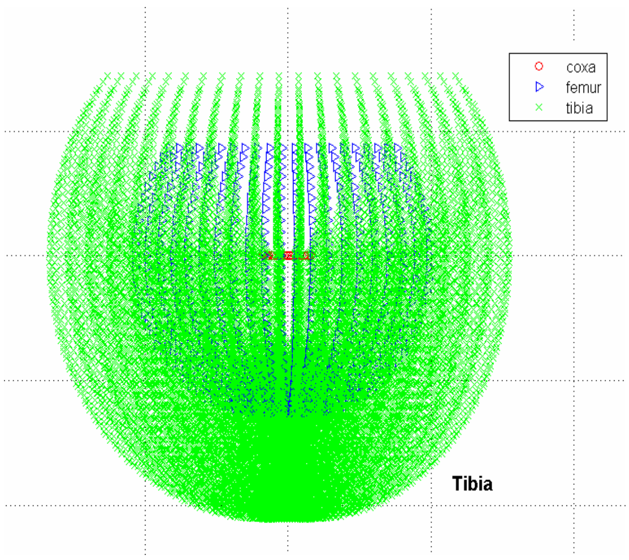


Fig. 7. Leg workspace (YZ view).

The leg workspace with the important point is presented in fig. 8 and synthesized in table 1.

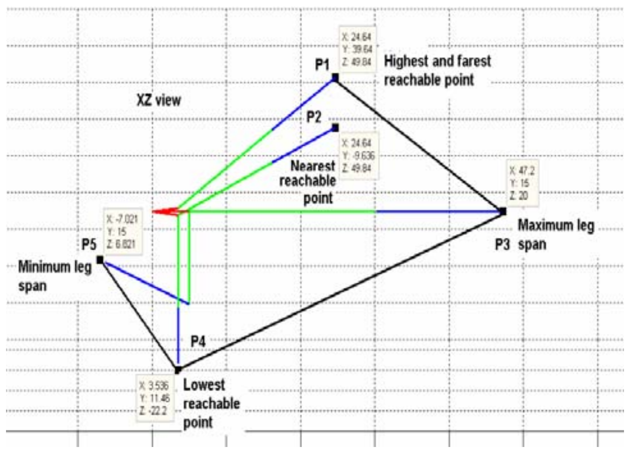


Fig. 8. Leg workspace. Extreme interest points.

Tabel 1. Leg workspace

Coordinates	Min	Max
X	-7	47,2
Y	-9,6	39,6
Z	-22,2	49,8

Valid for l_1=5, l_2=25.2, l_3=17 and for coxa joint coordinates (0, 15, 20)

4. Hexapod Robot Model

The legged locomotion on natural terrain presents a set of complex problems (foot placement, obstacle avoidance, load distribution, general stability) (Krzysztof et al 2008) that must be taken into account both in mechanical construction of vehicles and in development of control strategies. One way to handle these issues is using models that mathematically describe the different situations. Therefore modeling becomes a useful tool in understanding systems complexity and for testing and simulating different control approaches (Barreto et al, Fahimi 2008).

The robot structure considered has 6 identical legs and each leg has 3 degree of freedom (RRR) (Fig. 9). All the relevant points have been put on the model as can be seen from fig. 9: coordinates of the centre of mass of each leg G_i, i=1...6; leg numbering for easy understanding (1 to 6), coordinates of the centre of mass of the robot G, projection of the centre of mass into the support polygon G', robot's centre of symmetry with the attached frame $O_R(X_R,Y_R,Z_R)$, the global frame $O_w(X_w,Y_w,Z_w)$.

The global frame is the frame that all other frames will be defined relative to. The global frame is rigidly attached to the lower left corner of the world so that the z-axis is vertical and the xy-plane is aligned with the floor surface.

The origin of the robot coordinates is attached in the centre of symmetry with the z-axis pointing up, the x-axis pointing left and the y-axis pointing forward.

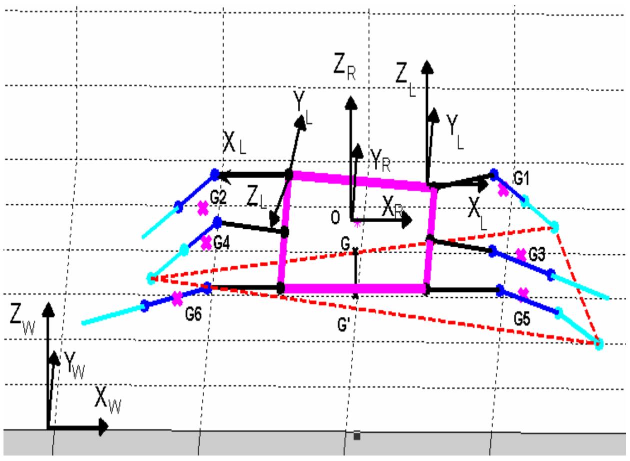


Fig. 9. Hexapod robot structure

The overall transformation from global frame to each leg tip is obtained as follows:

$$T_{tip_i}^{W} = T_{R}^{W} T_{coxa_i}^{R} T_{tip}^{coxa_i} \tag{19}$$

where:

T_R^W is the transformation matrix between the global frame and the robot frame and defines the rotation about y,x,z (roll, pitch, yaw matrix).

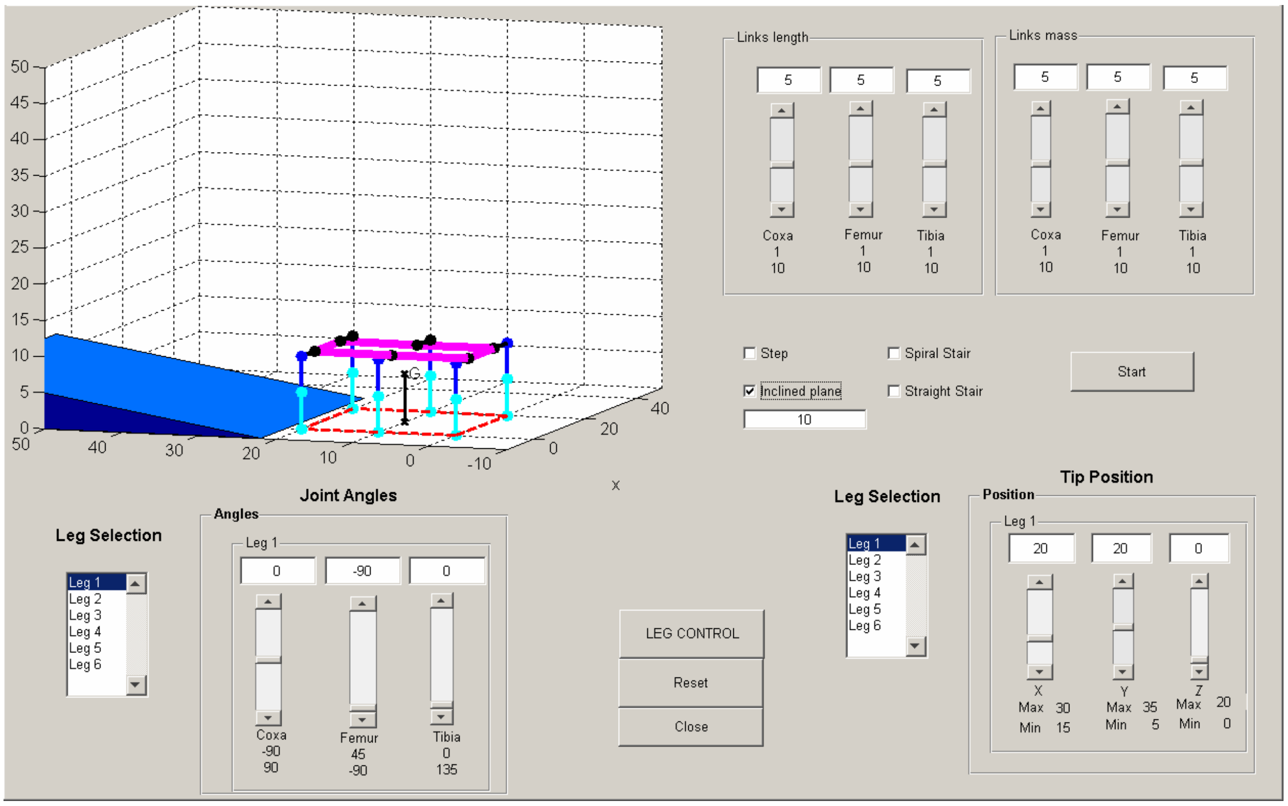
$T_{coxa_i}^R$ is the transformation matrix from the robot centre to each leg.

The position of robot's centre of mass is calculated using the following equations:

$$X_G = \frac{\sum_{i=1}^{6} x_{gi} m_{Li}}{\sum_{i=1}^{6} m_{Li}}, Y_G = \frac{\sum_{i=1}^{6} y_{gi} m_{Li}}{\sum_{i=1}^{6} m_{Li}}, Z_G = \frac{\sum_{i=1}^{6} z_{gi} m_{Li}}{\sum_{i=1}^{6} m_{Li}} \tag{20}$$

where:

$m_{Li} = \sum_{i=1}^{3} m_{li}$, m_{li} - mass of each leg



Links length
	Coxa	Femur	Tibia
Value	5	5	5
Min	1	1	1
Max	10	10	10

Links mass
	Coxa	Femur	Tibia
Value	5	5	5
Min	1	1	1
Max	10	10	10

[ ] Step [ ] Spiral Stair
[✓] Inclined plane 10 [ ] Straight Stair

Joint Angles

Leg Selection: [✓ Leg 1] Leg 2 Leg 3 Leg 4 Leg 5 Leg 6

Angles — Leg 1
	Coxa	Femur	Tibia
Value	0	-90	0
	-90	45	0
	90	-90	135

Tip Position

Leg Selection: [✓ Leg 1] Leg 2 Leg 3 Leg 4 Leg 5 Leg 6

Position — Leg 1
	X	Y	Z
Value	20	20	0
Max	30	35	20
Min	15	5	0

Fig. 10. Hexapod robot control interface.

5. SIMULATION PLATFORM

The main purpose of this simulation platform is to show how the support polygon modifies when legs lose contact with ground and if the projection of the centre of mass is within the support polygon.

The simulation program was made using MatLab GUIDE (Brian et al 2001).

The developed software platform can be used to analyze what happens with the hexapod robot in gravitational field and it also allows communication with the leg in real world.

The analysis of the hexapod robot in gravitational field can be group in two modes:
- free fall mode. In this mode the mechanical configuration of the legs is defined by their joints values, no additional move is allowed.
- transitory analysis. This mode is used to analysis what happens with the robot between two static regimes.

The communication between the physical leg and Matlab environment is made using Arduino Duemilanove development board.

Giving a set of values for legs joints yields a stationary mechanical configuration which generates a certain support polygon in relation with which we analyze the gravitational stability of the hexapod robot.

For a certain configuration of legs, the robot is statically stable if the projection of the centre of mass is inside the support polygon; it is at stability limit if the projection of the centre of mass is on one side of the support polygon and it is statically unstable if the projection of the centre of mass is outside the support polygon. In this last case the robot is shifting gradually its support polygon by lifting/touching the ground with its legs, due to gravity, until the condition for static stability is accomplished.

The shape of the support polygon needed for minimum static stability is the triangle.

5.1 Program Interface

When the interface is launched the robot is first drawn in a stable configuration as shown in fig. 10. The interface is divided basically into 2 areas: in the upper left the robot is displayed (plotted) according to the values set by the user and in the upper right and bottom we can find the controls for the robot. The controls for the robot are structured mainly in 3 parts: joints control, position control, walking control. Joints control and position control consists of a list where the leg number it put on and a panel where the user can set the values for each joint or position (using direct and inverse kinematics). Walking control consist of choosing the terrain on which the robot will move. The terrain can be choosing using the immediate (controls) checkboxes.

The controls for each leg are encapsulated into a panel identified by leg number. Every slider has an editable textbox where the value is displayed and controls a certain link of the robot. If we use a slider the associated editable textbox value is updated and vice versa. Also the robot leg position is updated with the data from the slider or from the textbox. The controls for link length or mass affect all the legs because they are considered identical.

The button label *Leg Control* from fig. 10 launches a second interface like in fig. 11. The second interface allows both hardware and simulated control for a single leg. This interface can be used both in offline mode or online mode.

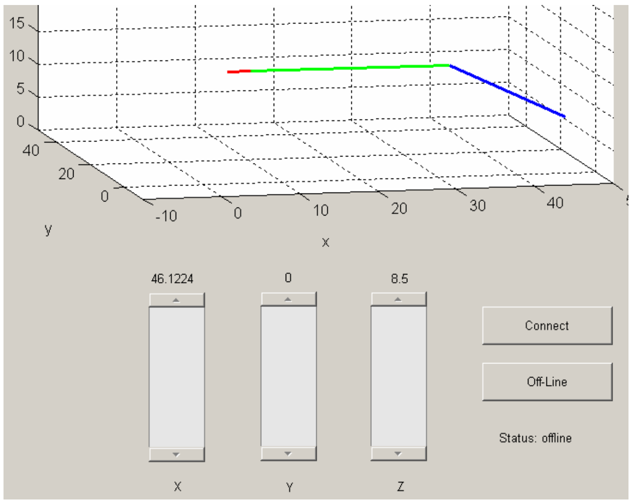


Fig. 11. Leg control using Arduino Duemilanove

The graphical representation of the robot also allows seeing the shape of the support polygon which is updated according to the legs on the ground. The support polygon is drawn only if the distance from the tip of the leg to the ground is smaller than a threshold. This threshold is modifiable (but not present in the interface) and was introduced as a way to compensate certain position errors that may occur due to real servomotors.

The simulation program shows all the stages the robot goes through for a better understanding. For simplicity and better understanding of the robot stages the model is drawn in a simpler way.

5.2. Simulation Algorithm

The authors have elaborated an algorithm in order to achieve the goal of analyzing the static stability of a hexapod robot in gravitational field. The algorithm is structured in 5 steps as following:

```
- setting the joints values
- determine the mechanical configuration
- determine which legs are on the ground
- evaluation of the static stability condition
- while (condition of static stability = false)
        - determine the rotation line using the minimum
          distance from G' to support polygon's sides
        - rotate the robot about the line found
        - determine which legs are on the ground
        - evaluation of the static stability condition
```

The determination of static stability condition is resumed at finding if the projection of robot's centre of mass is inside the support polygon. For this the authors used the following algorithm:

```
- determine the convex polygon
- determine the area of the convex polygon (A)
- form the n triangle using 2 consecutive sides of the support polygon and the projection of centre of mass, G', (e.g. ABG', BCG'… etc.)
- determine the areas of the n triangles formed ($A_i$)
- if ($\Sigma A_i = A$) then condition=true
      - else condition=false
```

5.3 Hardware Leg Control

The system proposed by the authors (Fig. 12) is similar with the system xPC-Target component of Matlab. The software that make possible the communication between Arduino board and Matlab has been released with the last version of Matlab. Mathworks has also developed support for Arduino in Simulink. A part of the communication software is uploaded on the Arduino board and plays the server role.

There are two programs that can be uploaded on the board:
- *adiosrv.pde* with which all the input and output of the board can be manipulated.
- *motorsrv.pde* designed exclusively for motor control via a motor shield.

The major drawback of using the original *motorsrv.pde* was that this file was developed as support for a specific motor shield that can only control 2 servomotors. So we modify the file in a way that now allows using all 6 PWM channels available.

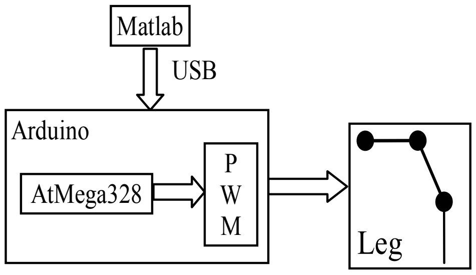


Fig. 12. Leg control system

The other part is a Matlab class (*arduino.m*) and plays the role of the client. Once the class is instantiated it makes possible sending commands over USB port.

The direction of motion of the servo is automatically determined. If we set a rotation with 90^0 and then a 30^0 rotation the servo will rotate 60^0 anticlockwise.

5.4 Walking Algorithm

During walking or running the leg move cyclically and in order to facilitate analysis or control, the motion of the leg is often partitioned in two parts:
- support phase or stance, when the robot uses the leg to support and propel.
- transfer phase or swing, when the leg is moved from one foothold to the next.

The stance part of the walking algorithm is supposed to move the leg in a straight line. The swing part of the algorithm must lift the leg off the ground, move it back to the starting position and lower it down to the ground again. The walking algorithm is based on the kinematical model of the leg.

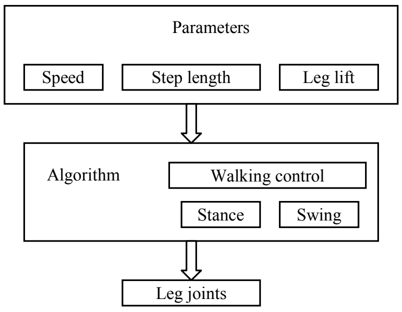


Fig. 13. Walking algorithm diagram

Parameters introduce in the algorithm (Fig. 13) are:
- speed; define the speed of the leg tip,
- step length; define the length of the step,
- leg lift; this defines how high the leg is lifted when it's in the swing-cycle.

For a smooth straight motion, the swing time must be equal to the stance time for each leg.

In the stance part of a step, the leg only moves in a straight line. At time t from the start of the step, the coordinates (x, y, z) for the trajectory will be:

$$\begin{aligned} &x = leglength \\ &y = \frac{steplength}{2} - t * speed \\ &z = -legheight \end{aligned} \qquad (21)$$

In the swing part of the step, the leg first has to be lifted of the ground, and then moved back to where the next step is supposed to start and then lowered to the ground. To find out where in the swing-cycle the leg is at time t from the start of the cycle we first have to calculate the total length the leg has to travel:

$$dist = \frac{(2 * leglift - steplength) * t}{swing\_cycle} \qquad (22)$$

One step consists of one stance cycle and one swing cycle but to maintain a smooth motion of the leg, it's important that the swing cycle continues where the stance cycle ended, and that the swing cycle ends where the new stance cycle starts. This does not only hold for the position, but also for the speed and the direction of the speed.

6. EXPERIMENTAL RESULTS

6.1 Free fall analysis

The free fall analysis represents what happens with the robot left to fall on the ground from a height greater than the extension of the legs. Keeping in mind that the joints are locked by the values prescribed by the user, no extra movements are allowed (no active stability). The only force that acts upon the robot is the gravitational force. For a given set of joint values the robot passes through many transitory stages until it becomes statically stable (Fig. 14. a, b, and c). Legs that have contact with the ground determine the shape of the support polygon (triangle, quadrilateral, pentagon or hexagon). In order to know if the robot achieves static stability the projection of G (G') must be inside the support polygon. To solve this problem the above algorithm is applied.

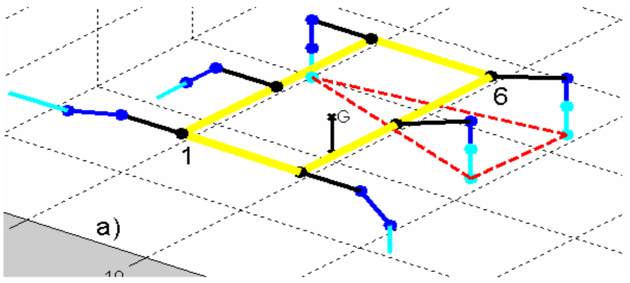


Fig. 14.a: Phase one of falling.

In fig. 14.b it can be seen that even in this configuration G' is not inside the support polygon and the robot continues it's falling and rotates about the line determined by leg 2 and leg 5 until the first leg touches the ground, which in this case, is leg 3.

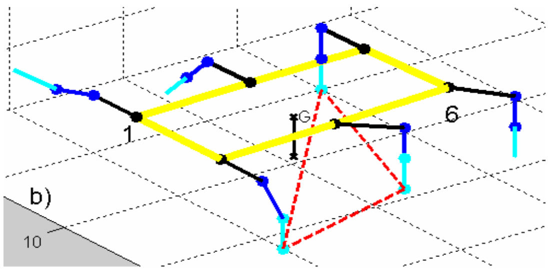


Fig. 14.b: Phase two of falling.

In fig. 14.c a new configuration is formed and if the algorithm described above it is applied, point G' is inside the support polygon and the robot becomes statically stable and the falling stops.

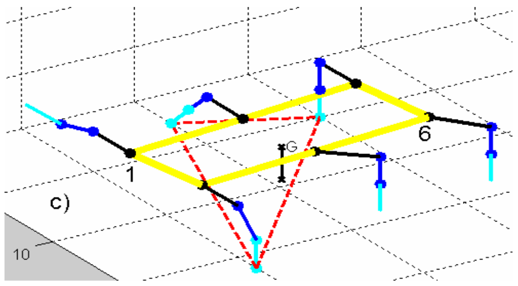


Fig. 14.c: Phase three – statically stable.

6.2 Transitory Analysis

In this mode of analysis the robot passes between two static regimes. A static regime is identified by the condition of stability. The user can alter a static regime using the controls for joint values or position of the leg tip. This analysis can also be interpreted as a continuation of *free fall analysis* case if the condition of stability has been met.

In fig 15.a the robot has static stability, the support polygon described by the legs on the ground is a quadrilateral (formed by legs 1, 2, 5 and 6) and the projection of the centre of mass is inside the support polygon.

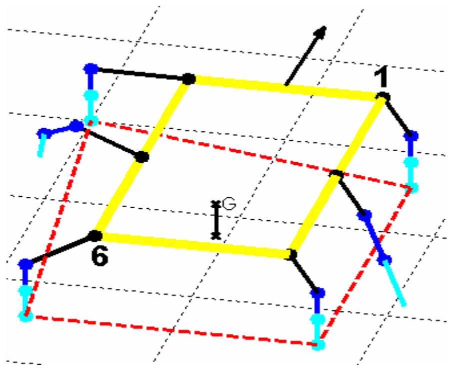


Fig. 15.a: Phase one – statically stable.

Next, lifting leg 2 the support polygon changes its shape becoming a triangle. Using the algorithm described above, the projection of G is not inside the support polygon and the robot becomes statically unstable and starts falling (Fig. 15.b).

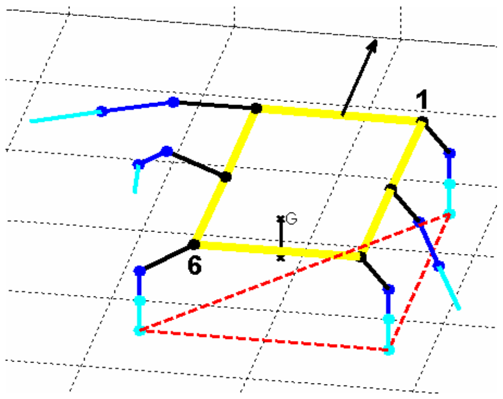


Fig. 15.b: Phase two: falling

Following the algorithm the next leg closest to the ground is leg number 4. In fig. 15.c the projection of G is inside the newly support polygon, the robot stops falling and becomes statically stable.

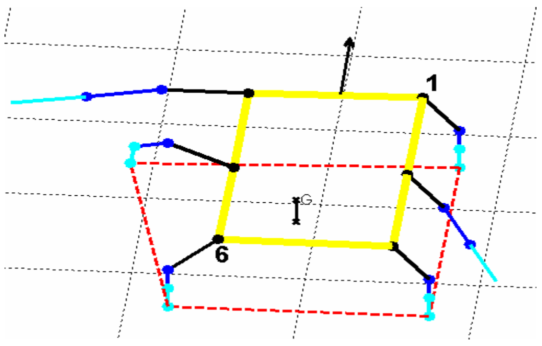


Fig. 15.c: Phase three – statically stable

6.3. Hardware Leg Control

The algorithm for controlling the leg joints, including direct kinematics and inverse kinematics, was implemented in Matlab in order to analyze leg performance.

The results of the tests can be view in the chart below.

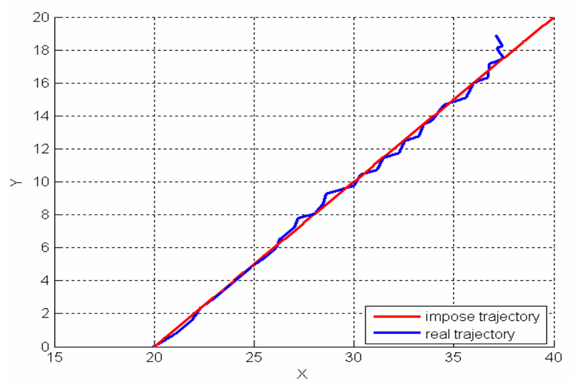


Fig. 16. Hardware leg control results.

The main errors occurred due to the fact that we can only send integer numbers for joint values.

Other errors occurred due to the fact that the joint are assembled using bolts and nuts. Normal usage of the robot causes the nuts and bolts to loosen causing a lot of clearance in the joints.

6.4 Inclined Plane Walking Approach

The movement on inclined plane is realized by moving one leg from one side at the time as can be seen from fig. 17, 18 and 19. As impose restrictions for the robot we can enumerate the following: the robot body must be leveled with respect to the ground, coxa angle is set for π/4. The angle for the plane is 10^0.

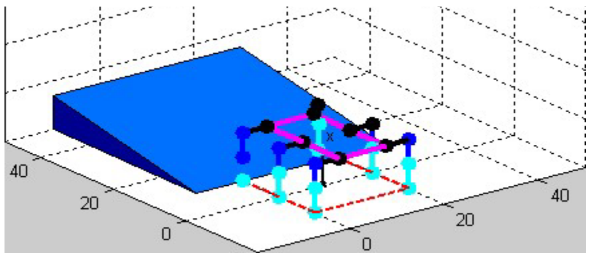


Fig. 17. Moving leg 1

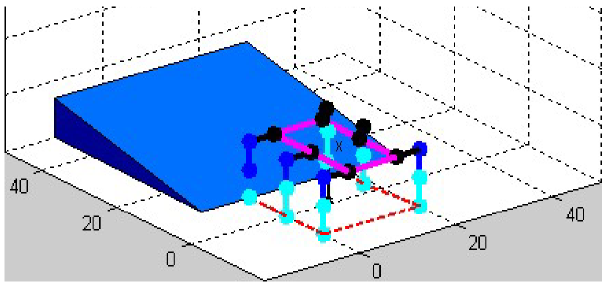


Fig. 18. Moving leg 3.

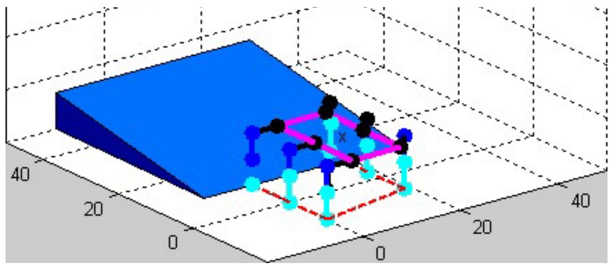


Fig. 19. Moving leg 5.

Once all the legs from one side have been moved to a new position the robot body will be dragged using all the coxa joints so that the robot can move forward (Fig. 20).

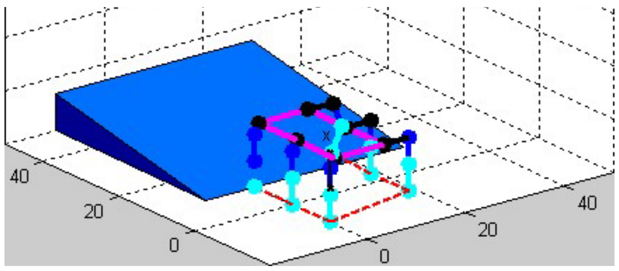


Fig. 20. Robot body drag.

After the body has been dragged to the new position the legs 2, 4 and 6 from the other side must be moved forward (Fig. 21, 22 and 23).

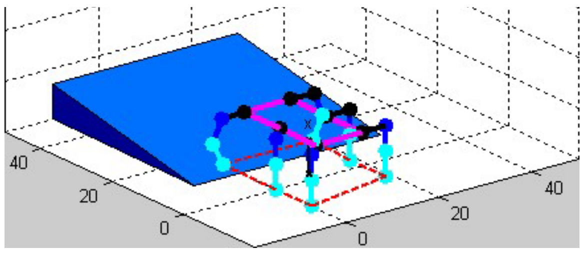


Fig. 21. Moving leg 2.

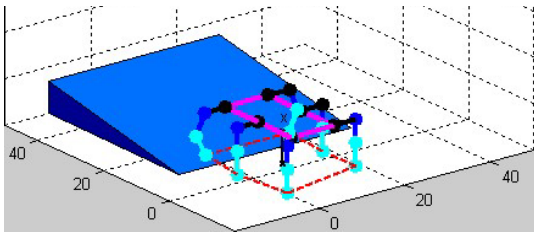


Fig. 22. Moving leg 4.

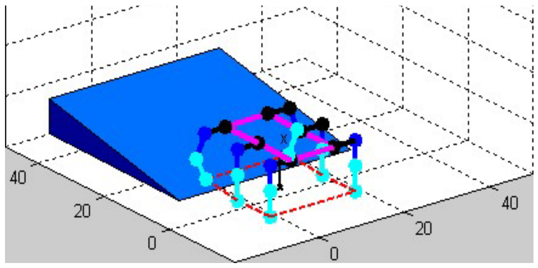


Fig. 23. Moving leg 6.

After this cycle of movement the body is dragged forward again and the algorithm resumes. Finally the robot with all the legs are on the inclined plane with its body leveled as can be seen from fig. 24.

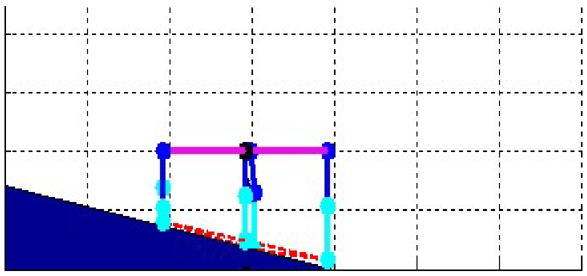


Fig. 24. Final position of the robot on the inclined plane.

Also it can be seen the trajectory of the robot on the inclined plane.

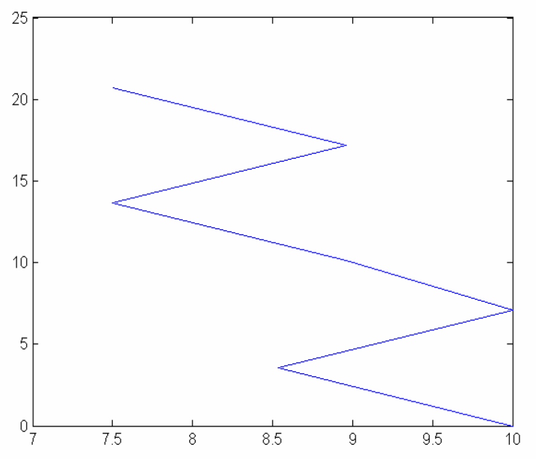


Fig. 25. Trajectory of the robot during walking

6.5 Stair Step Walking Approach

As with the previous case of movement, this case respects the algorithm describe in the previous chapter. The movement on stair step is realized by moving one leg from one side at the time as can be seen from fig. 26, 27 and 28. As impose restrictions for the robot we can enumerate the following: the robot body must be leveled with respect to the ground, coxa angle is set for $\pi/4$. The height of the stair step is 1/3 of the height of the robot.

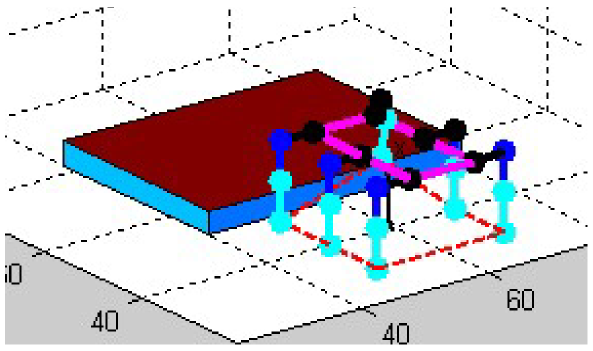


Fig. 26. Moving leg 1.

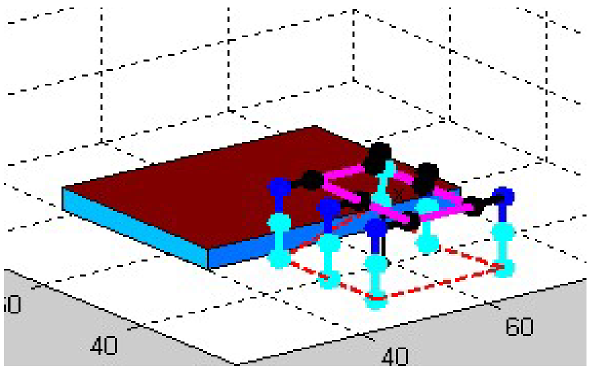


Fig. 27. Moving the leg 3.

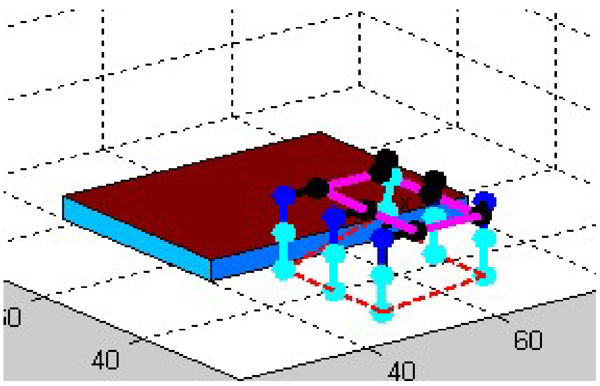


Fig. 28. Moving the leg 5.

Once all the legs from one side have been moved to a new position the robot body will be dragged using all the coxa joints so that the robot can move forward (Fig. 29).

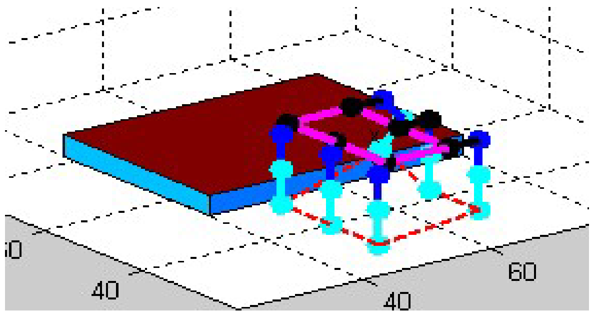


Fig. 29. Robot body drag

After the body has been dragged to the new position the legs 2, 4 and 6 from the other side must be moved forward (Fig. 30, 31 and 32).

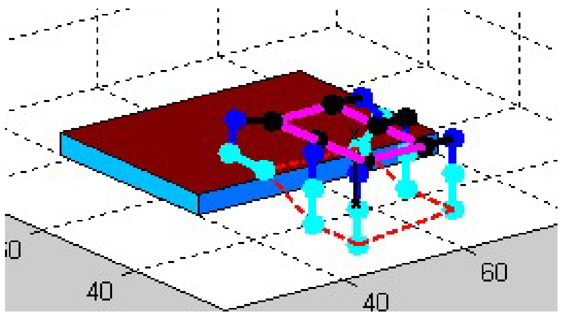


Fig. 30. Moving leg 2.

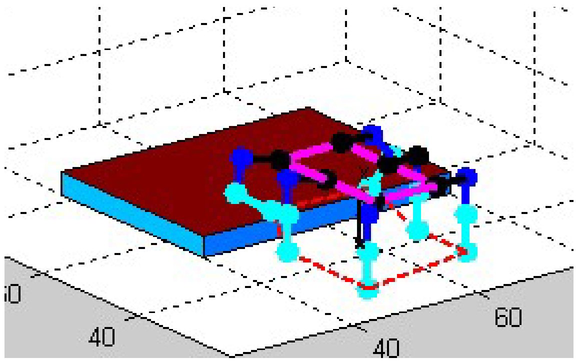


Fig. 31. Moving leg 4.

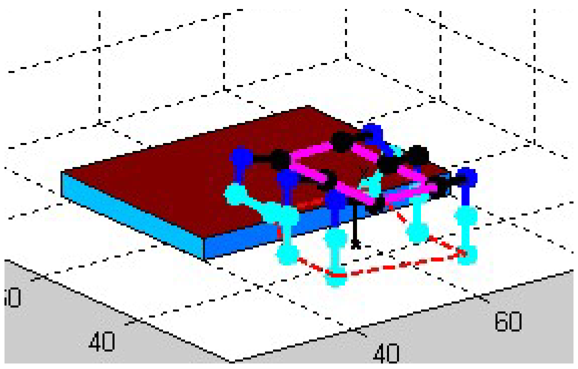


Fig. 32. Moving leg 6.

After this cycle of movement the body is dragged forward again and the algorithm resumes. Finally the robot with all the legs are on the stair step with its body leveled as can be seen from fig 33.

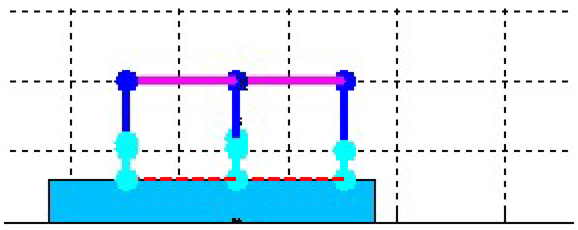


Fig. 33. Final position of the robot.

Also it can be seen the trajectory of the robot on during walking.

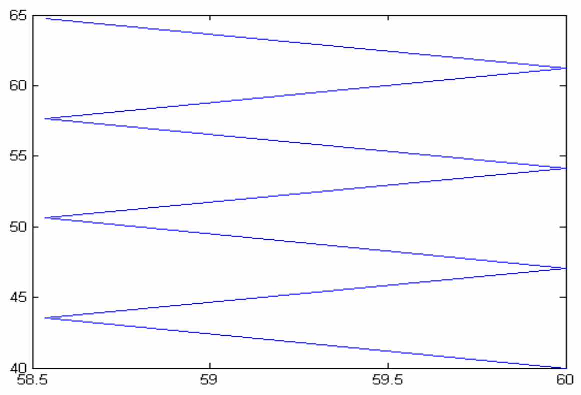


Fig. 34. Trajectory of the robot during walking

7. CONCLUSION

In this paper a simulation platform for legged mobile robots was presented, that allows stability analysis and full control of the robot.

Free fall analysis is useful for investigating what happens with the robot on uneven terrain or for accidents that may happen due to lose of contact, slippery surface, servomotor failure, power supply failure.

Transitory analysis represents a more important case for locomotion, gait generation. When starting to develop a gait cycle we can have a big picture of what happens when the robot starts to move, how the support polygon changes and what actions (what leg should be actuated) must be applied to the robot in order to meet condition of stability.

The interface will be imbuing with additional walking algorithms and controls.

The interface was designed to be simple and intuitive and to offer the user a simple and efficient way to control every aspects of the robot (angles, masses, lengths, leg control).

The two analyses made in this article represent the bases for next activities on static stability.

Hardware leg control will be imbuing to better precision.

In the future the program will be upgraded permitting additional controls and functions for stability analysis (including dynamic stability) on uneven ground and implementing collision detection algorithms. Also the results of these studies represent the bases for different strategies of locomotion on different terrains. The experimental results will become a standard for a real hexapod robot.

ACKNOWLEDGMENT

This work was partially supported by strategic grant POSDRU/88/1.5/S/50783, Project ID 50783 (2009), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resource Development 2007 – 2013.

REFERENCES

Barreto J., Trigo A. ,Menezes P., Dias J., *Kinematic and dynamic modeling of a six legged robot*.

Bensalem, S.,Gallien, M., Ingrand, F., Kahloul, I., Nguyen Thanh-Hung (2009), Designing autonomous robots, *IEEE Robotics & Automation Magazine*, Vol. 16, March

Brian R., Hunt R., Lipsman L., Rosenberg J. M.,(2001), *A guide to MATLAB for beginners and experienced users,* ISBN-I3:978-0-521-00859-4, Cambridge University Press

Carbone G., Ceccarelli M., (2005), Legged robotic systems, *Cutting Edge Robotics,* ARS Scientific Book, pp. 553-576, Wien.

Fahimi, F. (2008), Autonomous robots: modeling, path planning, and control, Springer.

Krzysztof W., Dominik B., Andrzej K.,(2008) Control and environment sensing system for a six-legged robot, *Journal of Automation, Mobile Robotics & Intelligent Systems*, Volume 2 N^{o}3.

Maki K. H.(2007) *Bioinspiration and Robotics:Walking and Climbing Robots*, ISBN 978-3-902613-15-8, I-Tech Education and Publishing, Croatia

Schilling R. J. (1990) , *"Fundamentals of robotics: analysis and control"*, ISBN: 0-13-344433-3, Prentice Hall, New Jersey, USA.

http://www.atmel.com/dyn/resources/prod_documents/8271S.pdf

www.mathworks.com