Stability Analysis Software Platform Dedicated for a Hexapod Robot

Sorin Mănoiu-Olaru and Mircea Nițulescu

Department of Mechatronics, Blvd. Decebal. 107, 200440, Craiova, Romania
manoiusorin2006@yahoo.com, 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 using Matlab software package. The simulation software was created using geometrical modelling based on Denvait-Hartenberg algorithm and analyses the static stability of the robot in different stages of locomotion on horizontal surface for different leg configuration. The paper includes some experimental results related to the static gravitational stability depending on the support polygon formed by the legs on the ground.

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

1 Introduction

Walking machines allow locomotion on rough and irregular surfaces with a high degree of softness [1]. This is why legged machines received increasing attention by the scientific community [2]. Current vehicles we are used to have wheels for locomotion. Wheeled vehicle can achieve high speed with a relative low control complexity but only on structured terrain. Since most of the earth's land surface is inaccessible to regular vehicles there is a need for mobile robots that can handle difficult terrain.

The conventional walking machine with three degrees of freedom for each leg has great flexibility during terrain motion [3]. An important drawback of legged machines is the complexity of the control required to achieve walking even in completely flat and horizontal surface in which much simpler wheeled machines work perfectly well [4]. This means that the use of legged machines is only justified if they can walk on irregular terrain with certain degree of confidence.

Most popular hexapods can be grouped into two categories: rectangular (with two groups of three legs distributed symmetrically on the two sides) and hexagonal (round or hexagonal body with evenly distributed legs).

The motion of legged robots can be divided into statically and dynamically stable. Static stability 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.

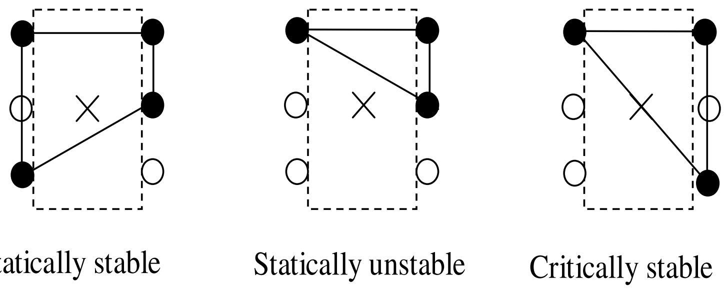

Fig. 1. Stability cases for a hexapod robot: statically stable, unstable cases and critically stability

Some of the most important advantages of legged locomotion are [5]:

- accommodation to uneven terrain
- use of isolated footholds
- providing active suspension
- environmental effects of legged vehicles are less than wheeled or tracked vehicles

Among disadvantages of legged locomotion we can enumerate:

- artificial walking mechanisms are so far heavy due to large number of actuators
- control of walking is very complex and so far walking vehicles are rather slow
- bad payload-weight-to-mechanism-weight ratio compared to wheeled or tracked vehicles
- appearance of an impact force with each step made.

2 Hexapod Robot Model

The legged locomotion on natural terrain presents a set of complex problems (foot placement, obstacle avoidance, load distribution, general stability) [6] 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 modelling becomes a useful tool in understanding systems complexity and in testing and simulating different control approaches [7], [8].

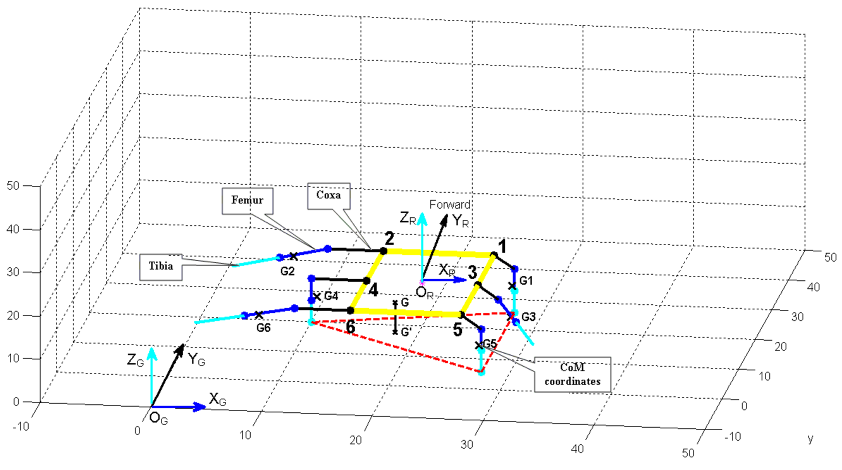

Fig. 2. Hexapod robot structure

The robot structure considered has 6 identical legs and each leg has 3 degree of freedom (RRR) (Fig. 2). All the relevant points have been put on the model as can be seen from figure 2: 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 $O_G(X_G,Y_G,Z_G)$, and direction of motion.

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.

2.1 Robot Leg

The successful design of a legged robot depends to a large extent on the leg design chosen. 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.

A three-revolute kinematical chain has been chosen for each leg mechanism in order to mimic the leg structure (Fig. 3). 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…6, and the fixed frame $O_R(X_R,Y_R,Z_R)$.

The coordinate frames for the robot legs are assigned as in fig. 3. The assignment of link frames follows the Denavit- Hartenberg direct geometrical modelling algorithm. The robot leg is made of links and joints as noted in figure 3. The different links of the robot legs are called *coxa*, *femur* and *tibia*.

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).

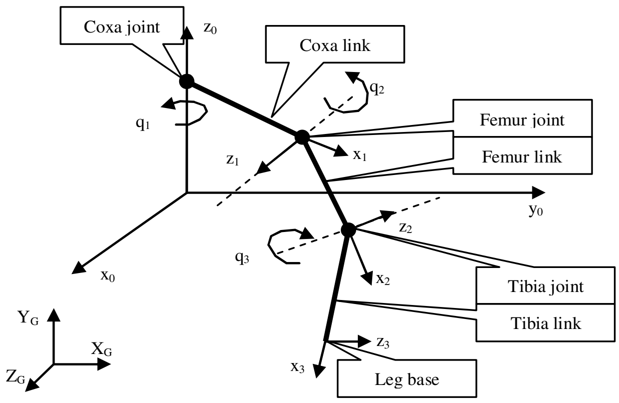

Fig. 3. Model and coordinate frame for leg kinematics

The general form for the transformation matrix from link i to link i-1 using Denavit Hartengberg parameters is given in equation 1:

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

The transformation matrix is a series of transformations:

- translate d_i along z_{i-1} axis,
- rotate θ_i about z_{i-1} axis,
- translate a_i along x_{i-1} axis,
- rotate α_i about x_{i-1} axis.

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

$$T_{O_G}^{base} = T_{O_G}^{O_R} T_{O_R}^{coxa} T_{coxa}^{femur} T_{femur}^{tibia} T_{tibia}^{base} \quad (2)$$

The product of the last three matrixes determines the geometrical model of the leg:

$$T_0^3 = T_0^1 T_1^2 T_2^3 \quad (3)$$

The centre of mass of each link is positioned relative to the link frame by a position vector p_i= [x_i, yi, zi, 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:

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

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

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

where:

m_{li} – mass of link i

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

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

where:

$$m_{Li} = \sum_{i=1}^{3} m_{li} \quad (7)$$

m_{Li} - mass of leg i

Limitations for each joint are:

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

3 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 [9] (**G**raphic **U**ser **I**nterface **D**esign **E**nvironment).

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

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

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.

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.

3.1 Program Interface

When the interface is launched the robot is first drawn in a stable configuration as shown in fig. 4. 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 2 parts: joints control, position 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.

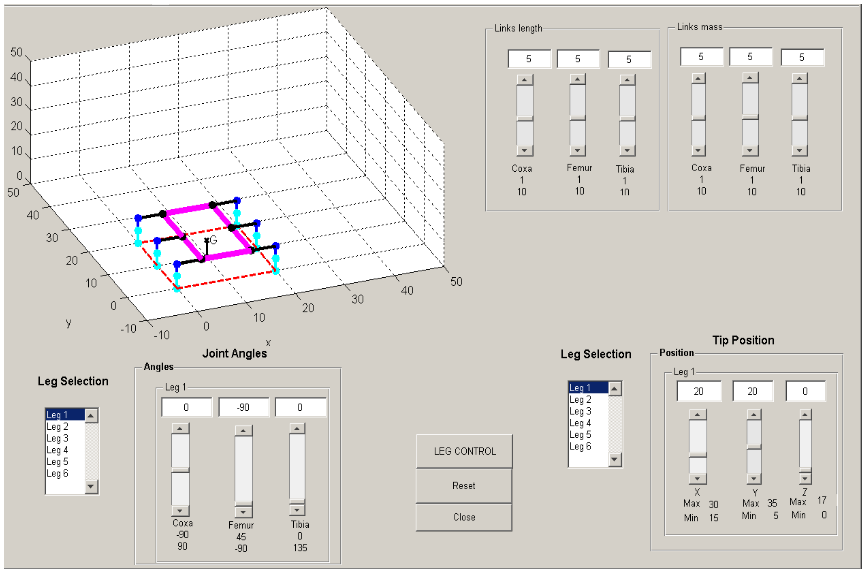

Fig. 4. Matlab robot platform interface

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. 4 launches a second interface like in fig. 5. The second interface allows control of a single leg. This interface can be used both in offline mode or online mode.

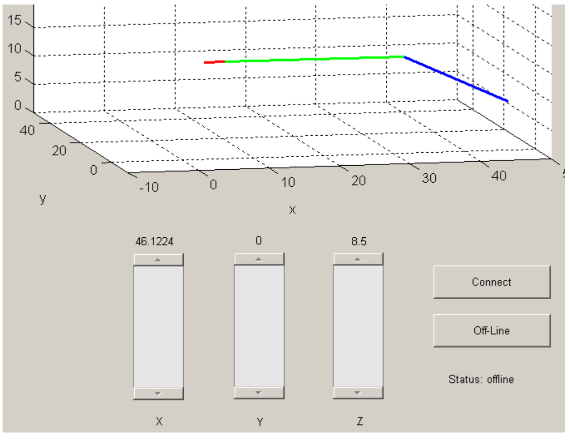

Fig. 5. The communication interface used to control the real leg using Matlab

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.

3.2 Leg Control

The system proposed by the authors (Fig. 6) is similar with the system xPC-Target component of Matlab. Of course our system does not have its performance but it is a lot cheaper. The software that make possible the communication between Arduino board and Matlab has been released with the last version of Matlab. Mathworks [10] 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.

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.

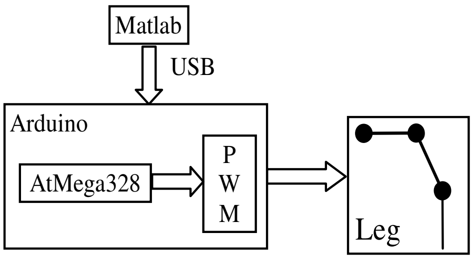

Fig. 6. Leg control diagram

3.3 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.

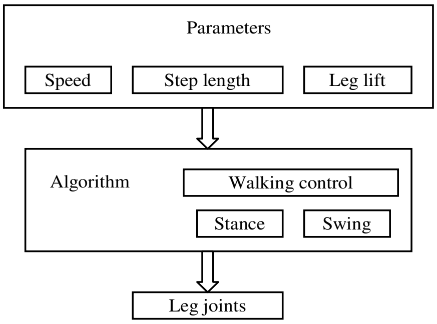

Fig. 7. Walking Algorithm

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. Parameters introduce in the algorithm (Fig. 7) 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:

$$\begin{aligned} &\mathrm{x}=leglength \\ &\mathrm{y}=\frac{steplength}{2}-t*speed \\ &z=-legheight \end{aligned} \tag{8}$$

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:

$$dist = \frac{(2 * leglift - steplength) * t}{swing\_cycle} \tag{9}$$

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.

3.4 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:

- support phase or stance when the robot uses the leg to support and propel.
- 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 is true
  - o else condition is false

3.5 Simulation Algorithm for Stability Analysis

The authors 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:

- support phase or stance when the robot uses the leg to support and propel.
- 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)
  - o determine the rotation line using the minimum distance from G' to support polygon's sides
  - o rotate the robot about the line found
  - o determine which legs are on the ground
  - o evaluation of the static stability condition

4 Experimental Results

4.1 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. 8.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' for now on) must be inside the support polygon. To solve this problem the above algorithm is applied.

As it can be seen in figure 8.a, when the robot falls the first legs that reach the ground are those nearest the ground. Also G' is not inside the support polygon and the robot continuing its falling and rotates about the line determined by the leg 2 and leg 3. The rotation line is determined by calculating the minimum distance from G' to polygon's sides. In this case the first leg closest to ground that will provide support is leg 5.

In figure 8.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 4.

In figure 8.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.

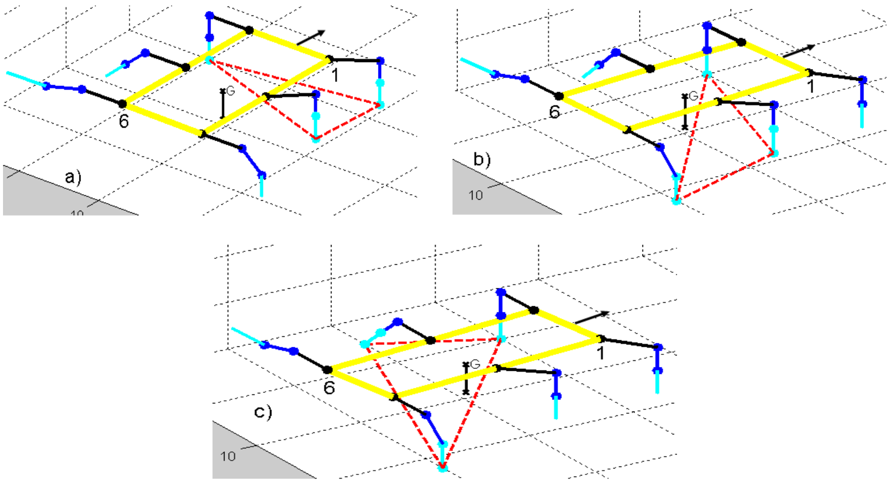

Fig. 8. Free fall analysis a. Phase one of falling, **b.** Phase two of falling, **c.**Phase three – statically stable

4.2 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. This analysis can also be interpreted as a continuation of *free fall analysis* case if the condition of stability has been met.

At legs considered on the ground the tip will become rounded as can be seen in all figures.

In figure 9.a the robot has static stability, the support polygon described by the legs on the ground is a quadrilateral and the projection of the centre of mass is inside the support polygon.

Next, lifting leg 5 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. 9.b).

Following the algorithm the next leg closest to the ground is leg number 4. In figure 9.c the projection of G is inside the newly support polygon, the robot stops falling and becomes statically stable.

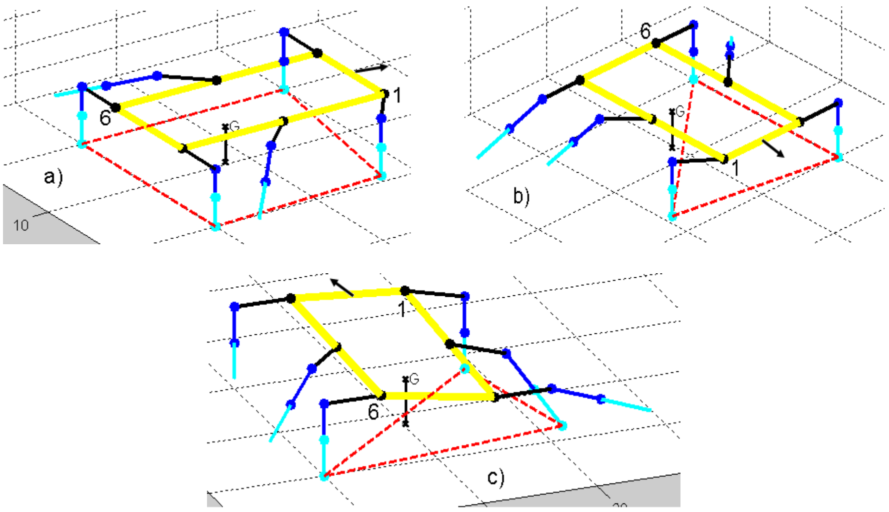

Fig. 9. Transitory analysis. a. Phase one: statically stable, **b.** Phase two of falling **c.** Phase three: statically stable

4.3 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 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.

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

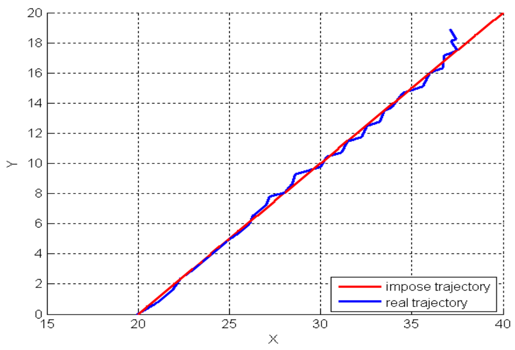

Fig. 10. Hardware leg control results

5 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 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).

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

The program can also be used for legged mobile robots with 4 or 8 legs using minor code modifications.

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.

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

Hardware leg control will be imbuing to better precision.

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

1. Giorgio, F., Pierluigi, R.: Mechanics and Simulation of Six-Legged Walking Robots. In: Climbing & Walking Robots Towards New Applications, p. 546. Itech Education and Publishing, Vienna (2007) ISBN 978-3-902613-16-5
2. Nelson, G.M., Quinn, R.D., Bachmann, R.J., Flannigan, W.C.: Design and Simulation of a Cockroach-like Hexapod Robot. In: Proceedings of the 1997 IEEE International Conference on Robotics and Automation Albuquerque, New Mexico (April 1997)
3. Maki, K.H.: Bioinspiration and Robotics:Walking and Climbing Robots. I-Tech Education and Publishing, Croatia (2007) ISBN 978-3-902613-15-8
4. Carbone, G., Ceccarelli, M.: Legged Robotic Systems. In: Cutting Edge Robotics, pp. 553–576. ARS Scientific Book, Wien (2005)
5. Nitulescu, M.: Robotic Systems with Navigation Capabilities. Universitaria Craiova, Romania (2002) ISBN: 973-8043-143-3
6. Krzysztof, W., Dominik, B., Andrzej, K.: Control and environment sensing system for a six-legged robot. Journal of Automation, Mobile Robotics & Intelligent Systems 2(3) (2008)
7. Lungu, V.: Artificial Emotion Simulation Model and Agent Arhitecture. In: Proc. of the 18th Int. Conf. on Control Systems and Computer Science, Romania, pp. 716–721 (May 2011)
8. Robotin, R., Lupea, D., Lazea, G., Dobra, P.: Mobile Robots Path Planning with Heuristic Search. In: Proc. of the 18th Int. Conf. on Control Systems and Computer Science, Romania, pp. 391–396 (May 2011)
9. Brian, R., Hunt, R., Lipsman, L., Rosenberg, J.M.: A Guide to MATLAB for Beginners and Experienced Users. Cambridge University Press (2001) ISBN-I3: 978-0-521-00859-4
10. Matlab Guide Help, `http://www.mathworks.com/help/techdoc/ref/guide.html`
11. Barreto, J., Trigo, A., Menezes, P., Dias, J.: Kinematic and dynamic modeling of a six legged robot
12. Bensalem, S., Gallien, M., Ingrand, F., Kahloul, I.: Nguyen Thanh-Hung, Designing autonomous robots. IEEE Robotics & Automation Magazine 16 (March 2009)
13. Fahimi, F.: Autonomous Robots: Modeling, Path Planning, and Control. Springer (2008)
14. Knight, A.: Basic of Matlab and Beyond. CRC Press, LLC (2000) ISBN: 8493-2039-9
15. Marchand, P., Thomas, H.O.: Graphics and GUIs with MATLAB, 3rd edn. Chapman & Hall/CRC (2003) ISBN 1-58488-320-0
16. Sarjoun, S., Kantor, G., Maiwand, G., Alfred, A.: Inertial Navigation and Visual Line Following for a Dynamical Hexapod Robot. In: Proceedings of the 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems Las Vegas, Nevada (2003)