Dynamic Equations

1. 27 simultaneous equations of motion based on FBD
2. Torque summation about arbitrary ground point to remove virtual inertial forces
3. Solved using matrices to represent torque (8) and limb force (16) equations
4. Assumes knowledge of forces acting on foot
5. Assumes pure torque applied at joints
6. Verified using static poses
7. Foot forces approximated for foot swing, drag, and impulse
8. –FBDs and diagrams-
9. Results verified with ‘common sense’ and literature checks
10. **Results and conclusions**

Motion Study

1. Determine maximum mounting position based on geometry
2. Calculating minimum and maximum leg extension
3. Determine maximum stroke length from difference in extension
4. Determine minimum piston compression from specifications
5. Adjust mounting and stroke length to optimize gait
6. Ensure pistons are usable for future iterations
7. **Results and conclusions**

Pneumatic/Compressor Specifications

1. Maximum and average torque converted to pressures and flow rates
2. Moment arm and desired torque calculates maximum required piston force
3. Piston bore diameter and stroke selected based on cost and geometry
4. Maximum air pressure calculated from moment arm and piston area
5. Average flow rate calculated from stroke length
6. Compressor and air storage selected based on average flow rate and maximum pressure
7. Piston selected based on bore size, stroke length, and maximum pressure/force output
8. **Results and conclusions**

FEA of Legs

1. Finite element analysis based on worst case scenario of leg joints seizing up during maximum forces applied
2. 2D analysis used due to simple 2D planar geometry of joint and fewer vertices
3. Automatically generated ANSYS mesh using primarily quad elements due to increased accuracy over CST
4. Manual mesh refinement in areas with structural error – generally region where forces were applied
5. **Results and conclusions**

Dynamics

To determine the internal forces felt in the joints and the required torques for locomotion a dynamic mathematical model of the robot was constructed. From the specifications the robot was known to have four legs with two links each all attached to a main chassis. Summing the force and torque around each link of the robot resulted in 27 simultaneous equations used to calculate the state of the robot. To simplify the calculations it was assumed that the robot exhibited purely planar motion and all leg torque is applied purely at the hip and knee. A diagram of the mathematical notation used in the model is shown below:

X

Y

Θthigh

Θbody

Θshank

Leg 1

Leg 2

Leg 3

Leg 4

Thigh

Shank

Figure 1: Mathematical notation for the robot model. The body angle is measured relative to the horizontal, the thigh angle is measured relative to the body, and the shank angle is measured relative to the thigh.

To determine the equations of motion for the overall system free body diagrams were developed for the shanks, thighs, and bodies. This resulted in a system of 27 equations with 32 unknowns. The free body diagrams are given below for a single leg and the chassis:

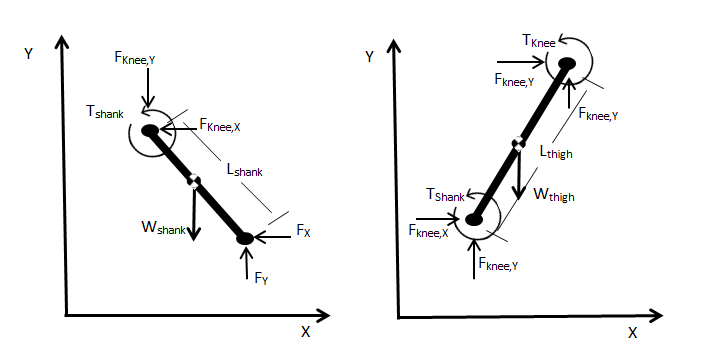


Figure 2: Free body diagrams for the shank (left) and thigh (right). Summing forces and masses on the legs generates 24 equations and 32 unknowns.

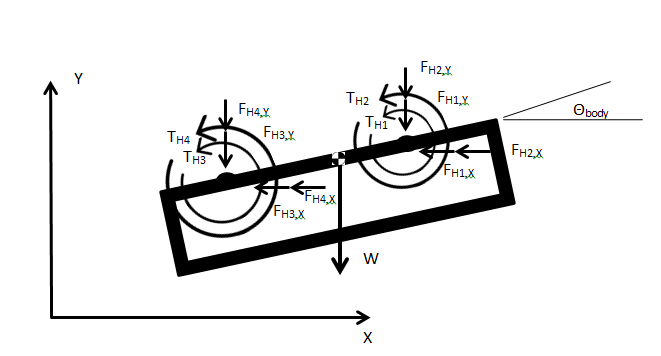


Figure 3: Free body diagram of the chassis at an arbitrary angle. The body brings the system to 27 equations and 32 unknowns.

To simplify the model into a solvable form it was assumed that the reactionary forces of the feet were known, and equations of motion for the body were discarded. This reduced the simulation to a system of 24 equations and unknowns, which could be solved to find the torque applied and internal force felt at each joint.

The final step in developing a solvable model for the robot is to deal with the non-inertial reference frame of the robot. Due to the robot’s acceleration during motion, taking a force and moment sum of each joint will neglect inertial forces acting on the robot. The coriolis and centrifugal forces are two examples of inertial forces that need to be accounted for when doing a force sum in a non-inertial reference frame. **Reference here**.

There are three main ways to deal with dynamics in non-inertial reference frames. The most common way is to simply add in the inertial forces to make newton’s laws of motion valid. Adding in the inertial forces is the most conceptually straight forward and the simplest to solve mathematically. However without a physical system to test it would be very intensive to determine the extent that non inertial forces are affecting a system as complicated as ours.

The second method is to use lagrangian mechanics to solve the system in an arbitrary coordinate system. Due to the general energy based nature of lagrangian mechanics inertial forces are accounted for during equation derivation. Unfortunately the core curriculum at MSOE does not mention lagrangian mechanics and it would be too time intensive to learn an additional dynamic system on top of constructing the senior design project. **Refrences here**.

The final method is to have an arbitrary ground point as the reference for the system. This eliminates the inertial forces acting on the robot while also allowing newton’s laws of motion to be used for the dynamical analysis. After giving the robot an arbitrary ground location the sum of the torque equation needed to be modified using the **parallel axis theorem?** A subset of the developed equations is given below:

Where *r* is a distance vector to the arbitrary ground point, *Fhip* and *Fknee* are the internal forces on the hip and knee joints, *Thip* and *Tknee* are the torques applied to the hip and knee joints, and *Tequivalent* is calculated as follows:

Where *Izz* is the moment of inertia of the link about the z axis. **SITE SOURCE PLZ**.

The torque and force equations can then be put into the following matrix forms:

Where T is an 8x1 matrix of torques, A is an 8x8 matrix of 1’s and 0’s, and B is an 8x1matrix containing the other torque equation elements such as distances and equivalent torques. F is a 16x1 matrix of internal joint forces, C is a 16x16 matrix of 1’s and 0’s, and D is a 16x1 matrix containing masses, accelerations, and weights.

Equations ### can then be solved by inverting the A and C matrices to get the following solution form:

This creates a system of equations where the torques and forces are dependent on the robot link’s mass and inertia, the angular position, velocity, and acceleration of each joint, and the Cartesian acceleration of each joint. The mass and inertial values for the robot were taken from the SolidWorks model developed during the previous design phase and a kinematic model was developed to calculate the required position, velocity, and acceleration values during motion.

To determine the position of the hip, knee, foot, and thigh CGs a kinematic model for the robot was developed. The kinematic model assumed each leg is a serial manipulator grounded at the CG of the robot at a position (x, y) relative to the arbitrary ground. A diagram for the front right leg is given below:

LH1

LK1

LF1

LT1

LS1

Figure 4: Length definitions for the front right robot leg. All positions are measured with polar notation using a series of angles and distances from the point (x,y) at the CG of the robot chassis. For the kinematic equations the leg is considered a serial manipulator consisting of revolute joints connected to a ground reference at point (x, y). All angles use the earlier notation.

From the above diagram it is easy to perform a serial manipulator analysis to determine the position of any point on the robot leg. The position of the foot in terms of the body, hip, and knee angles is given below:

Where F is the Cartesian foot position. All angles follow the convention given earlier.

The equations were then symbolically stored in a MATLAB script and derived with the following generic function **reference**:

Which is equivalent to:

Where *J(f)* is the Jacobian matrix of function *f*. This approach allows rapid iteration of design versions without changing the differentiation of the joint velocity and accelerations. The partial derivative of the position is multiplied by the position Jacobian to acquire the velocity function, and the velocity function is multiplied by the velocity Jacobian to achieve the acceleration. A simplified form of the kinematics which assumed *LH1 =* θ*body = 0* was hand derived and matched exactly with the MATLAB differentiation.

In addition a static case close form solution was created for the torques and forces at each joint in a single leg. After modifying the simulation parameters to fit a single static leg the identical equations were output by the simulation. Although having a correct static case doesn’t verify the entire dynamical derivation, it is useful to spot any fundamental errors in the simulation.

After deriving the dynamic and kinematic equations the required torques and resultant forces at each joint are functions of the following 11 x 1 state vector q and its two time derivatives:

To calculate q and its time derivatives for the robot a second kinematic simulation was constructed with some simplifications. The simulation was simplified to a single leg with the foot following an arbitrary path at a constant speed. This allows the angles and their derivatives to be calculated for a single leg. The values of the other legs could then be approximated either as constant values or values based on from the simulation.

A semi-ellipse was selected as the initial foot path. The path was selected due to its mathematical simplicity and the ease of changing the step length and height parameters. It is also a similar shape to the more complicated pear-shaped quartic, which is the path many organic creature’s feet follow during motion. A constant speed was selected to simplify the step analysis. A plot of the simulation in action is given below:



Figure 5: Kinematic step simulation for a single robot leg. The leg follows the elliptical path with a constant velocity.

To get a close approximation of the maximum stresses and torques felt by the robot a 0.5 m long step was selected to be completed in 1 second which satisfies the specification of moving a maximum of 0.5 m/s.

After a close inspection of the step it should be clear that there is a corner at 0.25 m, which causes an unrealistically large spike in acceleration and therefore force and torques felt. To correct this error the step was split into three phases. The first two phases, swinging and dragging, were calculated as described above. The third phase, impulse, used the velocity of the leg to calculate the foot force which was then used to calculate internal forces and torques in the joints. The impulse method was derived as follows from the conversion of linear momentum **reference**:

Where *m* and *v* are the mass and velocity of their respective bodies, and *f(t)* is the impulse force applied to the foot. Assuming *f(t)* is constant and *vfloor* = *vtotal* = *0* the following equation results:

Which calculates the force applied to the foot. This force can then be used in conjunction with the other simulations to determine the impulse force felt by the robot joints.

After running the simulation for the swing and drag phase the vector *q* is also known for a single leg. In order to find the torques required and internal forces at each joint some simplifying assumptions were made about the robot’s gait to determine the remaining values in *q* and the foot forces.

The first simplification made was that the robot performed a drag gait to move. This means the robot locks three of its legs and uses the fourth to drag itself along the ground. It was also assumed that the robot chassis is moving at a constant velocity of 0.5 m/s forward. These simplifications don’t capture the entirety of the dynamical motion, but it is accurate enough to get a rough idea of the expected dynamic forces without spending months doing only dynamical analyses of the system.

To finish defining the state vector *q* a value of 90 degrees was selected for the remaining hip angles, and a value of zero degrees was selected for the body and knee angles. The derivatives of these values were also set to zero due to the static nature of this gait. Finally the foot force values were calculated for the swing and drag phases of the elliptical step. For the drag portion of the step the following free body diagram was used to calculate foot force values:

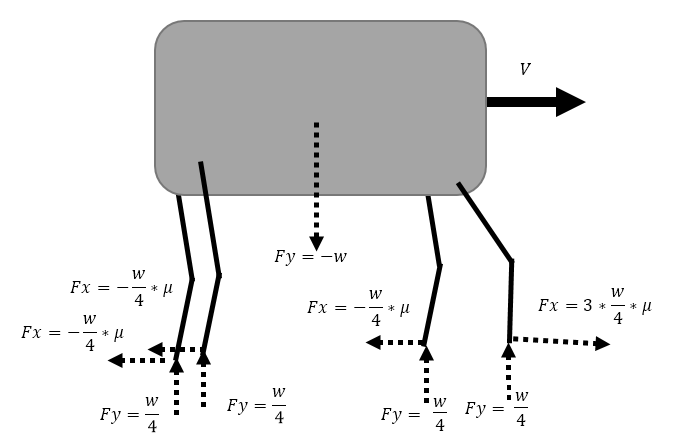


Figure 6: Free body diagram for the drag portion of the gait. All four legs are assumed to be on the ground of the robot while travelling at a constant velocity V.

Where μ is the dynamic friction coefficient of the feet. This resulted in the following equations:

Which, when simplified, leads to the following:

Equation **##** simplifies to weight divided by four because the rotational acceleration of the body was assumed to be zero. Thus to prevent an unbalanced tipping moment the force of each foot in the y direction must be equal. For the simulation a value of 0.5 was used for the coefficient of friction. This is a common value for rubber sliding across a hard surface. **Src?**

Finally the foot forces during the swing phase were calculated. To simplify the calculations it was assumed that the robot body was completely static as the foot swung out. There are a few inaccuracies in this assumption, but the amount of time it would take to determine a foot reaction force model for the robot would exceed the time allotted for the project. With this simplification and leg geometry setup it is trivial to calculate that the x and y components of force acting on the swinging leg is zero. Additionally because the legs are completely vertical it is trivial to conclude the x component of the other foot forces are zero as well, while the y direction is one third of the robot weight.

After performing all of the derivations a MATLAB script was developed to run the simulations and output the state of the robot at the maximum torque and maximum internal forces. The script can be seen in **Appendix Q**. The maximum values output from the script are tabulated in the figure below:

Table 1: Simulation results for the Swing, Drag, and Impulse phases of the simulation. Values are labeled as [Type][Location][Leg], so TK1 is the Torque applied at the Knee on Leg 1, and FH2x is the Force felt by at the Hip on Leg 2 in the X direction.

|  |  |  |  |
| --- | --- | --- | --- |
| *Value* | *Swing* | *Drag* | *Impulse* |
| TH1 [N-m] | 383.2008 | 67.082 | 0 |
| TH2 [N-m] | 2.9008 | -3.1367 | 0 |
| TH3 [N-m] | -0.7823 | -1.3573 | 0 |
| TH4 [N-m] | 2.9008 | -3.1367 | 0 |
| TK1 [N-m] | 1.3135 | 99.3253 | 0 |
| TK2 [N-m] | 17.5289 | 6.8914 | 0 |
| TK3 [N-m] | -13.2918 | -9.8418 | 0 |
| TK4 [N-m] | 17.5289 | 6.8914 | 0 |
| FK1x [N] | 557.6 | 33.7092 | 12.3869 |
| FH1x [N] | 911 | 91.9408 | 0 |
| FK2x [N] | 0 | 17.25 | 0 |
| FH2x [N] | 0 | 17.25 | 0 |
| FK3x [N] | 0 | 17.25 | 0 |
| FH3x [N] | 0 | 17.25 | 0 |
| FK4x [N] | 0 | 17.25 | 0 |
| FH4x [N] | 0 | 17.25 | 0 |
| FK1y [N] | 789.9 | 145.8055 | 50.3962 |
| FH1y [N] | 1321.2 | 199.8142 | 34.5 |
| FK2y [N] | -42.4 | 30.8703 | 34.5 |
| FH2y [N] | -38.2 | 26.652 | 34.5 |
| FK3y [N] | -42.4 | 30.8703 | 34.5 |
| FH3y [N] | -38.2 | 26.652 | 34.5 |
| FK4y [N] | -42.4 | 30.8703 | 34.5 |
| FH4y [N] | -38.2 | 26.652 | 34.5 |

It can be seen from the above table that the forces and torques are an order of magnitude higher during the swing phase compared to the other two. At first this may seem unintuitive, but it does make sense. The drag and impulse columns represent forces that are being used to either stop or maintain a velocity, whereas the swing column includes forces used to achieve high accelerations due to the motion of the foot.

With these simulation results the pneumatic cylinders and legs can be sized to fit the walking specifications. The results were used with a motion study and cost analysis to determine the maximum force output and therefore pressure required by the pneumatic system. This allowed the other components of the pneumatic circuit to be sized. Additionally the worst case scenario forces for the legs were determined allowing their design to be iterated and ensured they did not fail.