# A SIMPLIFIED KNEE MODEL

**Ellen Klonowski**
Department of Biomedical Engineering
Lerner Research Institute, Cleveland Clinic
Cleveland, Ohio, USA

## Introduction

The following knee model is adapted from Three-Dimensional Modelling of the Human Knee Joint [1], a simplified representation of the human body's most complex joint. It presents a three dimensional static left knee model to analyze the structure-function relationship provided certain loading conditions and movement. The model consists of a distal femur and proximal tibia surface along with 16 ligament components to represent the four ligaments of the knee. Two coordinate systems, femur and tibia, describe its position. Millimeters (mm) are used for position measurements and force is measured in Newtons (N). This mathematical model is executed using a single Python script with Anaconda3 Distribution [https://www.anaconda.com/distribution/] and Pycharm [https://www.jetbrains.com/pycharm/] as its environment to run the code. It provides figures of the anatomical surfaces of the knee and ligaments. The code is useful to calculate a database of force and position provided a certain femur origin point (x, y, z) or angle of the knee ($\alpha, \beta, \theta$). It also has the capability to find the femur origin point and angles of the knee required to keep the knee in static equilibrium when prescribed certain external forces and moments. The knee is represented as a deformable contact model. Ultimately, the goal is to solve static equilibrium equations as a function of external, contact, ligament, and residual forces and moments using Equations 1 and 2.

$$\overline{Fe} + \overline{Fc} + \overline{Fl} + \overline{Fr} = \overline{0} \quad (1)$$

$$\overline{Me} + \overline{Mc} + \overline{Ml} + \overline{Mr} = \overline{0} \quad (2)$$

## Articulating Surfaces Definition

Two coordinate systems used in the model are the femur (u, v, w) and tibia (x, y, z) coordinate systems. The x-axis of the tibia and u-axis of the femur provide anterior (positive x) and posterior directions of knee movement while the y-axis of the tibia and v-axis of the femur provide lateral (positive y) and medial directions the knee. The z-axis of the tibia and w-axis of the femur direct the knee proximally (positive z) and distally. Rotation angles of the knee includes flexion ($+\theta$), internal rotation ($\alpha$) and varus ( $+\beta$). Provided the transformation matrix (T), femur origin point ($R_o$) and a point on the femur ($R_i'$), the corresponding point on the tibia ($R_i$) can be defined using the following equation:

$$R_i' = R_o + T(^{t,f}) * R_i \quad (3)$$

The rotational transformation matrix (T) is used to describe points on the femur coordinate system in the tibia coordinate system in Figure 1 below.

**Figure 1. Rotational Transformation matrix.**

$$T= \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

Articulating surfaces of the knee in this model are the femur and tibia, modeled using two sets of parametric equations of an ellipsoid and two horizontal planes to define the tibia surfaces for lateral and medial surfaces. Uppercase phi ($\Phi$) and lowercase phi ($\phi$) represent the ellipsoid's angle of projection in the x-y plane and the angle form the polar axis, respectively. $\Phi$ consists of several linearly spaced angles that range from 0° to 360° and $\phi$ is defined as linearly spaced angles that range from -90° to 0° for plotting purposes. The range of points plotted in the lateral and medial directions is limited reflect the approximate contact areas of the articulating surfaces. Each of the surfaces utilizes its own Python function and accepts input values x, y, and z. The tibia surface equations return a z-value after x and y are input, and is used as the third (z) coordinate. Each set of surface coordinates are appended to their own list. Figure 2 illustrates the equations and plots of articulating surfaces.

**Figure 2. Articulating surfaces and equations.**

**Lateral Femur Surface**

$x = 30.961418 * \cos(\varphi) * \cos(\Phi) + 6.220455$
$y = 28.315550 * \cos(\varphi) * \sin(\Phi) + 27.834953$
$z = 19.509878 * \sin(\varphi) + 0.024089$

**Lateral Tibia Surface**

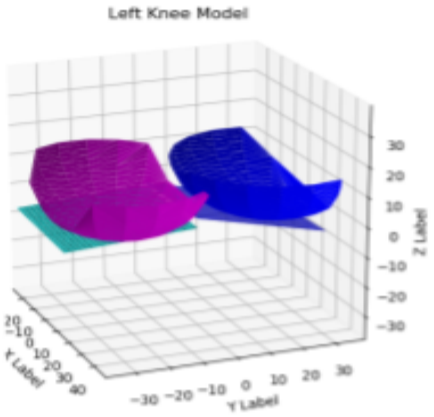
$f_L(x,y,z) = .022731 * x - .247293 * y - 1.75356 = 0$

**Medial Femur Surface**

$x1 = 33.662991 * \cos(\varphi) * \cos(\Phi) + 10.961876$
$y1 = 26.383162 * \cos(\varphi) * \sin(\Phi) - 21.706642$
$z1 = 20.116209 * \sin(\varphi) + 2.454443$

**Lateral Tibia Surface**

$f_M(x,y,z) = .022731 * x - .247293 * y - 1.75356 = 0$




Geometric surfaces are necessary to obtain a triangulation of the surface. The code does not require a visualization of the plot, but is helpful to see the change in position with different rotation angles and femur origin points. The Delaunay Triangulation [2] function accepts the lateral and medial surface coordinates of the femur

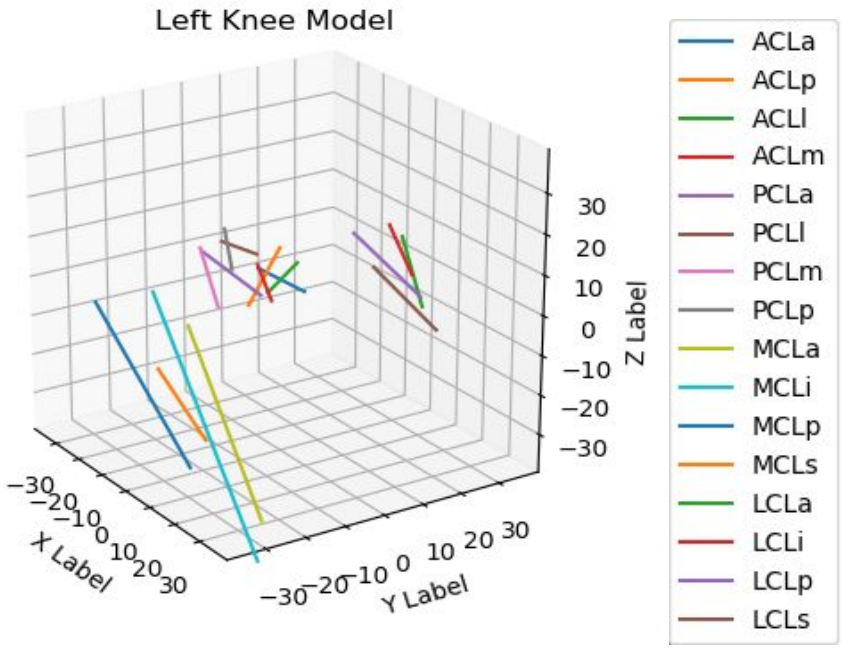
and returns 'verts', a three dimensional array that consists of *n* number of triangles and its vertices. These triangles are necessary to compute the area of the surface and contact force.

**Ligament Definition**

Ligaments are modeled as nonlinear tension spring elements. They act as connective tissue for the femur and tibia to allow movement of the knee. Anterior and posterior cruciate ligaments, medial and lateral collateral ligaments each consist of four elements for a total of 16. Insertion areas reflect the most anterior, posterior, medial, and lateral points of the ligament on the bone. The text details the insertion sites on both the femur and the tibia, provided by Martelli (1997) [3]. It is important to note that the femur insertion sites must be transformed into the tibia coordinate system using Equation 3.

One stiffness value exists for each ligament, in this case provided by literature [4]. Slack length is defined as the length of the ligament before tension is applied. Data from previous research is used to optimize these values based on displacement as a result of external forces applied on the tibia. These values are stored in a class inside of the calculate residual function that contain each ligament component's name, femur insertion point, tibia insertion point, stiffness, and slack length. Figure 3 is a plot of the left knee ligaments with no rotation of the knee and the femur position at its origin.

**Figure 3. Left knee ligaments.**



**Contact Forces and Moments**

Contact forces and moments are important when creating a knee model because they describe the force on the bone as a function of load applied and position. Contact force and moment is calculated for the medial and lateral sides of the knee separately.

Deformable contact represents the contact force in this model as opposed to rigid because it can calculate any type of contact without increasing computation time. Contact force in this case can also be described as pressure distribution of the femur on the tibia. The position of the femur relative to the tibia provides the necessary contact areas to find the penetration distance of the two surfaces. In this model, the contact area of the femur surfaces are represented by the area variable. Area of each triangle is calculated by finding the distance between each vertex to find the perimeter (4) and then Heron's formula (5) to calculate area to append each value to a list. The list of areas is used for the midpoint integration over the contact area to find force and moment. Next, normal vectors from each triangle are found using the cross product of sides A and C, eventually dividing by the magnitude for the unit normal vector. It is important to note that the normal vectors on the femur should point outward in the tibia's direction. The compression modulus, S, from Equation 6 is a constant in this model where $\nu$ is Poisson's ratio (0.45), E is the elastic modulus (5 MPa), and b is the surface thickness (2 mm).

$$P = \frac{(A+B+C)}{2} \tag{4}$$

$$Area = \sqrt{P(S-A)(P-B)(P-C)} \tag{5}$$

$$S = \frac{(1-\nu)E}{(1+\nu)(1-2\nu)} \tag{6}$$

The centroid of each triangle is used to find the displacement from the center of each triangle by subtracting the z component of each triangle's centroid from the corresponding z value on the tibia with the same x and y value. Un, Equation 7, is the surface displacement of the tibial and femoral surface, stored in a list. 'Idx_penetrating' is the variable that determines whether two surfaces are in contact. If the difference in the z value on the femur from the tibia is greater than zero, the surfaces are penetrating and considered in contact. The built in Numpy function, Numpy.where, checks if the difference in z values (zd) is greater than zero and stores those indices. The enumerate Python function loops over areas, normal vectors, centroids and the z values to store only values that satisfy the 'Idx_penetrating' condition. Once only the values of interest are obtained, the normal surface stress ($\sigma_n$) is found using Equations 8 and 9 to find the contact force ($\overline{Fc}$) on the knee by summing forces on each element of the knee.

$$Un = z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^{T} T^{(t,f)} \tag{7}$$

$$\sigma_n = S \frac{Un}{b} \quad (8)$$

$$\overline{Fc} = -T^{(t,f)} \left( \sum_{i=1}^{mL} \sum_{j=1}^{nL} \sigma_n \overline{n'} r_{lj}^2 \cos\alpha_{lj} \Delta\alpha \Delta\beta + \sum_{i=1}^{mM} \sum_{j=1}^{nM} \sigma_n \overline{n'} r_{lj}^2 \cos\alpha_{lj} \Delta\alpha \Delta\beta \right) \quad (9)$$

Contact moment, provided by Equation 10, is defined as the moment caused by the rotation of the femur on the tibia. The function to find contact moment is similar to the contact force, except for the finding vector from the distance from the centroids on the femur and the normal vector on that surface.

$$\overline{Mc} = -T^{(t,f)} \left( \sum_{i=1}^{mL} \sum_{j=1}^{nL} \sigma_n \overline{R'} \times \overline{n'} r_{lj}^2 \cos\alpha_{lj} \Delta\alpha \Delta\beta + \sum_{i=1}^{mM} \sum_{j=1}^{nM} \sigma_n \overline{R'} \times \overline{n'} r_{lj}^2 \cos\alpha_{lj} \Delta\alpha \Delta\beta \right) \quad (10)$$

**Ligament Forces and Moments**

It is important to analyze on ligaments provided certain movements to understand how they move relative to the knee. Force ($\overline{Fr}$) and moment ($\overline{Ml}$) on the ligament is determined with respect to position on the femur given in Equations 11 and 12. The ligaments are defined by their boundaries on the femur and the tibia. After points on the femur are transformed according to rotation angles and femur origin, the magnitude of the ligament force ($\overline{Fj}$) is calculated based on strain ($\varepsilon_j$) given in Equation 13. The force elongation-relationship is not linear and is dependent upon the strain on the ligament. $\lambda_j$, Equation 14 is a unit vector in the direction of the ligament provided by the vector given by the two boundary conditions, divided by the magnitude. The ligament force function will find $\overline{Fj}$ based on the strain and stiffness ($k_j$) derived from Wismans (1980) and Blankevoort et al. (1991). $\overline{Fl}$ is the sum of all ligament forces. Ligament moment represents the moment from the transformed femur point to the femur origin.

$$\overline{Fl} = \sum_{j=1}^{16} F_j \lambda_j \quad (11)$$

$$\overline{Ml} = \sum_{j=1}^{16} T^{(t,f)} \overline{R'}_j \times Fj \, \overline{\lambda}_j \quad (12)$$

$$F_j = \begin{cases} 0.0 & \varepsilon_j \le 0.0 \\ \kappa_j \varepsilon_j^2 / 4\varepsilon_{lj} & 0.0 < \varepsilon \le 2\varepsilon_{lj} \\ \kappa_j (\varepsilon_j - \varepsilon_l) & 2\varepsilon_{lj} \le \varepsilon_j \end{cases}$$

$$\varepsilon_j = (L_j - L_{0j}) / L_{0j} \quad (13)$$

$$\bar{\lambda}_j = \frac{\overline{Rj} - \overline{Ro} - T^{(t,f)}\,\overline{R'j}}{(\overline{Rj} - \overline{Ro} - T^{(t,f)}\,\overline{R'j})^T (\overline{Rj} - \overline{Ro} - T^{(t,f)}\,\overline{R'j})} \quad (14)$$

**External Forces and Moments**

External forces ($\overline{Fe}$) and moments ($\overline{Me}$) include muscle and gravity forces applied to the femur in any direction.

**Residual Forces and Moments**

Residual forces ($\overline{Fr}$) and moments ($\overline{Mr}$) are introduced to the model. It is the force or moment required to hold the knee joint in equilibrium provided the other three constraints (contact, ligament, external). These forces are solved for in the model in two different methods. First, the residual moment and force functions are separate Python functions that each return a value, residual force or residual moment vector are given as:

$$\overline{Fe} + \overline{Fc} + \overline{Fl} = \overline{Fr} \quad (14)$$

$$\overline{Me} + \overline{Mc} + \overline{Ml} = \overline{Mr} \quad (15)$$

This method is used if the external load is known and prescribed to the model at one specified femur origin and set of rotation angles.

The second method uses a nonlinear solver to find the femur origin and rotation angles that result in zero residual force and moment vectors. The code outputs a femur origin point and a set of rotation angles that keep the system in equilibrium without residual force, provided an external load. Scipy optimize's *optimize.root* function accepts the previously calculated contact and ligament moments and forces to serve as input to solve for the residuals. The solver requires an initial guess for the femur origin and rotation angles. Output includes whether the solution converges, number of iterations, and solutions.

**Model Structure**

The model is designed to write values to a CSV file based on several knee positions, or just a single position. Results from the solver method are displayed in the Pycharm window. The code creates a database of resulting forces from different femur origin and rotation angles to numerically display the structure-function relationship of the knee. It includes all femur origin points, all rotation angles, all ligament forces and moments, contact forces and moments, and residual forces and moments in a range specified by the user. This is information is helpful to analyze changes in force on the knee as the position changes. Since the bones are modeled as rigid surfaces, the increase in force can be dramatic from a small change in position.

Improvements in the code and its development include the ability to perform calculations using indexing rather than looping on every displacement iteration. Initially, the code was designed using a Python for loop to solve for the equilibrium equations at one displacement condition. The next iteration included a for loop to solve for several positions of the knee and write to a CSV file, but was very time consuming. This version was helpful in determining the effect on the knee as a result of load because it provided several sets of results. Next, the code was modified to utilize indexing so that it could perform multiple calculations at the same time instead of one by one. A large grid using the Numpy meshgrid function creates the combinations of femur origin points and rotation angles as inputs to solve the equilibrium equations.

The code can be further improved by validating results for comparison with the text. Patterns in the database results seem to follow the plots in the text, but are not exact. Also, the code can be improved by allowing a larger range displacement inputs because it currently crashes Python if run with too large of a range.

**References**

[1] Erdemir, A. (September 1998). *Three-Dimensional Modelling of the Human Knee Joint*. The Middle East Technical University.

[2] Bird, B. "Python Matplotlib plot_trisurf Polygon Data." *Stack Overflow*, stackoverflow.com/questions/41233786/python-matplotlib-plot-trisurf-polygon-data.

[3] Blankevoort LL, Huiskes RR. Ligament-Bone Interaction in a Three-Dimensional Model of the Knee. ASME. *J Biomech Eng.* 1991;113(3):263-269. doi:10.1115/1.2894883.

[4] Martelli, S. (1997) Istituti Ortopedici Rizzoli, Laboratorio di Biomeccanica, Bologna, Italy.