# 1 Pybullet Documentation Extended

**softBodyAnchor** (int body1, int link1, int body2, int link2, vec3 offset)
Pins vertex of a soft body to the vertex of a rigid/multi body or world. Returns constraint unique ID. See removeConstraint API to remove constraint

@param

- body1 - a non-negative integer denoting the body unique id of a deformable
- link1 - an integer denoting the index of the link. Default is -1 for the base
- body2 - a non-negative integer denoting the body unique id of a rigid/multibody with vertices defined (e.g. obj). Default is -1 for the world
- link2 - an integer denoting the index of the link. Default is -1 for the base
- offset (optional) - unclear

@return - an integer denoting the constraint unique ID

**getMeshData** (int bodyID)
Experimental API used to mesh information (indices, vertices)

@param

- bodyID - a non-negative integer denoting the body unique id of a mesh
- note: other params outlined in pybullet doc do not appear to work

@return - a tuple (numVertices, (vertices)) where each vertex in 'vertices' is a tuple (x,y,z) and their index in 'vertices' is their corresponding link index

# 2 URDF Tags

- The 'neohookean' tag of a 'deformable' takes it's 'mu' and 'lambda' parameters (the lame parameters) in kPa (This is suspected given 60 mu and 200 lambda work and 60 Kpa =0.06 MPa which is close to the approximated 0.1 MPa for hyperelastic material elastin).
- The 'virtual' tag of a 'deformable' is constrained to only a 'filename' attribute that loads a vtk/obj file. The deformable can only be relocated upon loading (loadURDF) and therefore cannot have a changed center of mass given you cannot specify an inertial reference frame relative to a nonexistent link reference frame
- loadSoftBody can support a spring model and neohookean model while URDF only supports neohookean (URDF also allows setting inertial properties)
- The only known supported URDF element tags for deformables: 'collision_margin', 'repulsion_stiffness', 'friction', 'neohookean'.

# 3 Lame Parameter Notes

- The General Rule: $\mu \leq \lambda$
- The difference between $\mu$ and $\lambda$ cannot be too large (e.g. $< 400$) but to be honest you should use your reference of phsyical properties to guide yourself. For example, elastic moduli ($\mu$) should be at least be 60 MPa because anything lower becomes too hyperelastic for the simulation to handle.
- My general rule is to start out at a decent Young's Modulus, increase $\lambda$ from the $\mu$ value until the simulation breaks and then everything is fair game. In other words, from the value of a decent Young's Modulus until the breaking lambda, any combination of two values in that range as long as the general rule is true should not break the simulation
- too many hyperelastic pieces will cause the sim to crash so as the number of nehookean pieces grow, the stiffer each one should become