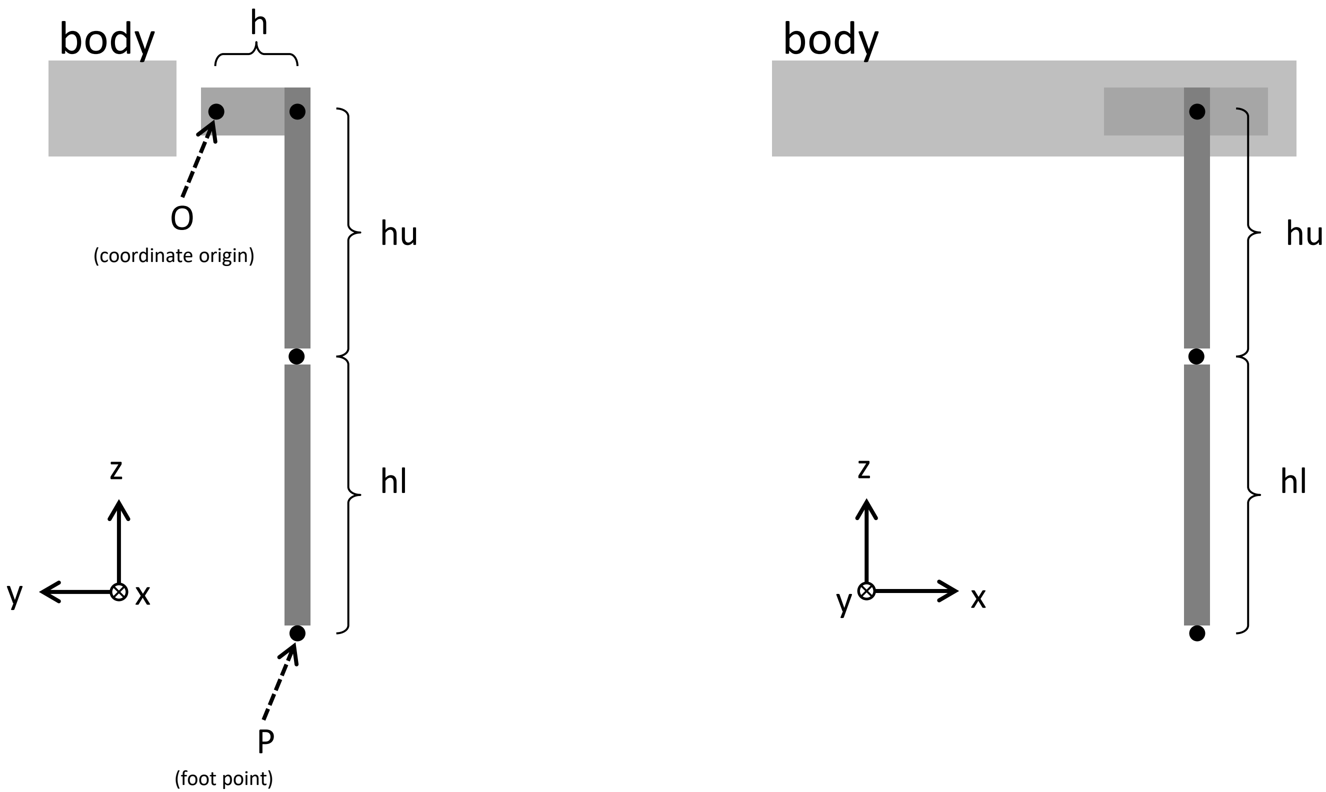
Dog-leg inverse kinematics

RB

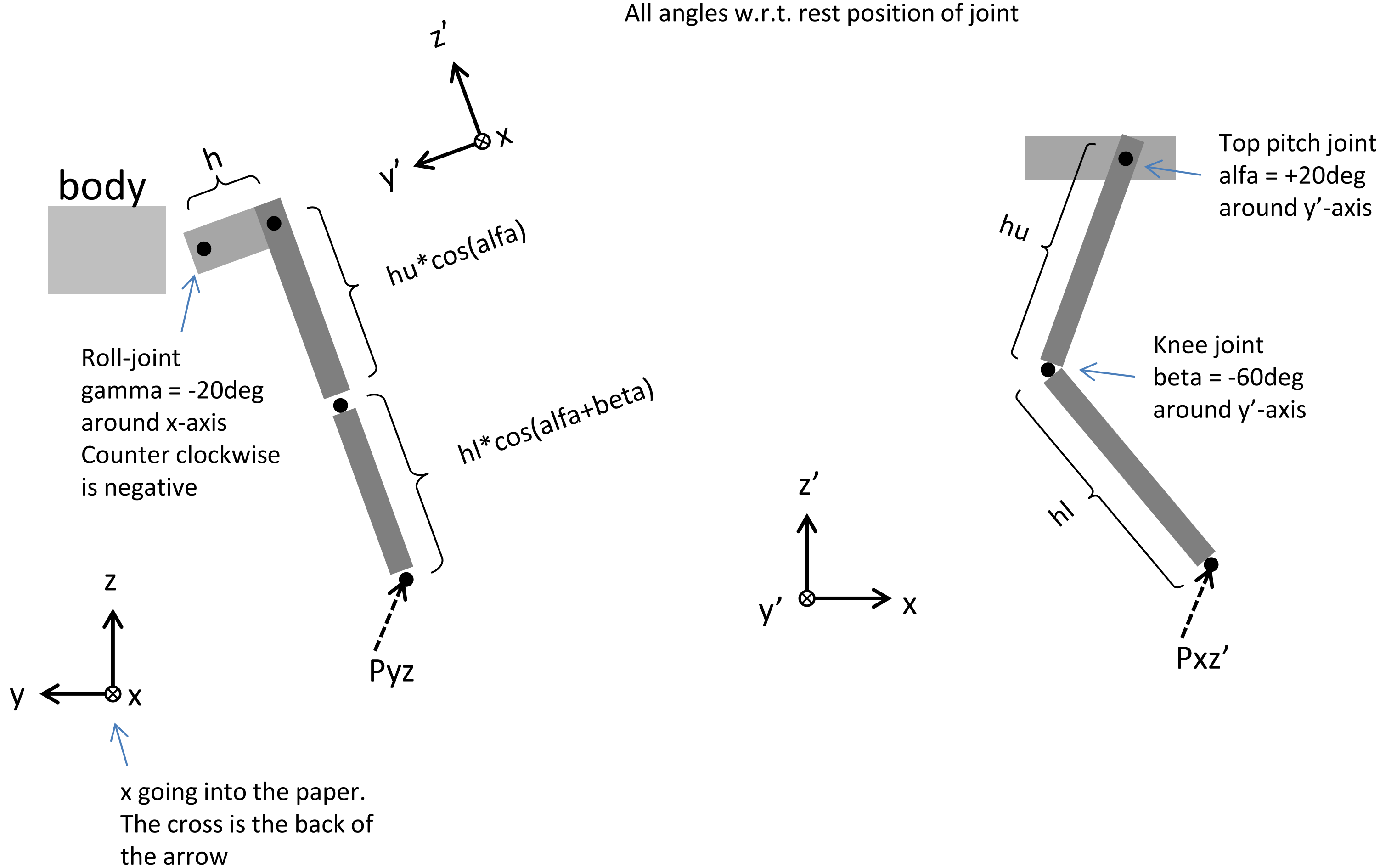
Pivot locations, rest position

(different definitions possible of course)



Pivot locations

View of inclined plane (x-z’ –plane)
All angles w.r.t. rest position of joint



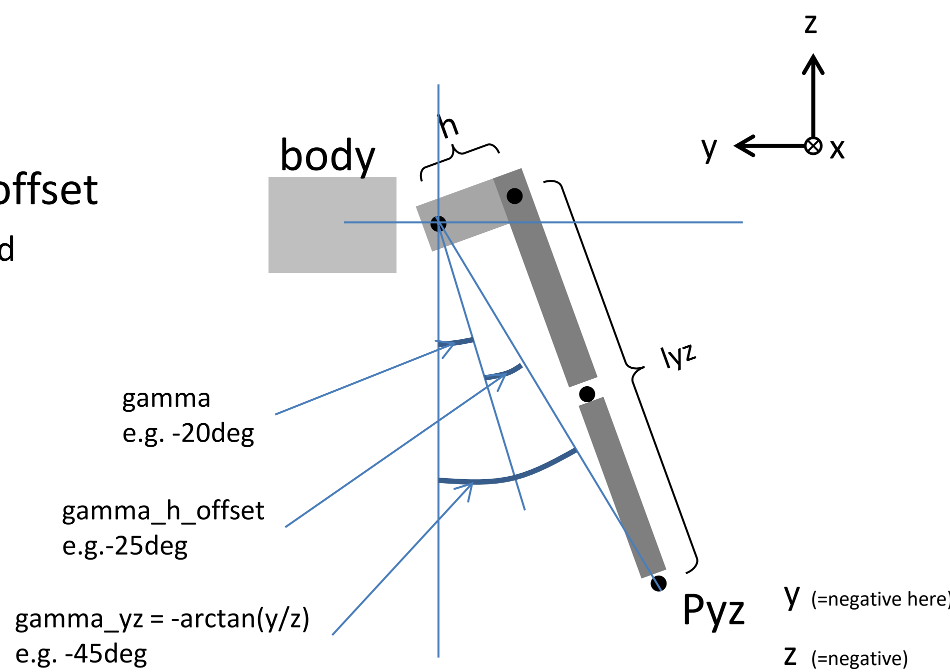
Objective

- Given a desired P in coordinates x, y, z where x, y, z are measured starting from the origin O.
- Given the link lengths h, hl, hu
- Use right handed coordinate system x,y,z
- Use angles to be positive using right-hand rule
- Find the required alfa, beta, gamma

Find gamma first

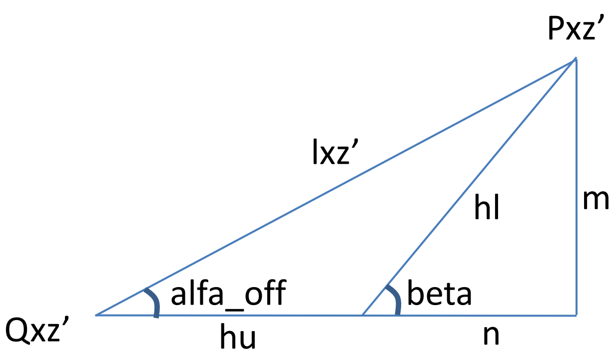
- dyz is the length from O to the projection of P on the y,z plane (=Pyz)
- dyz = sqrt(y^2+z^2)   (Pythagoras)
- lyz is the length from top-pitch-joint to the projection of P on the y,z plane (=Pyz)
- lyz = sqrt(dyz^2-h^2)   (Pythagoras)
- gamma_yz = -arctan(y/z)
- gamma_h_offset = -arctan(h/lyz)
- gamma = gamma_yz - gamma_h_offset

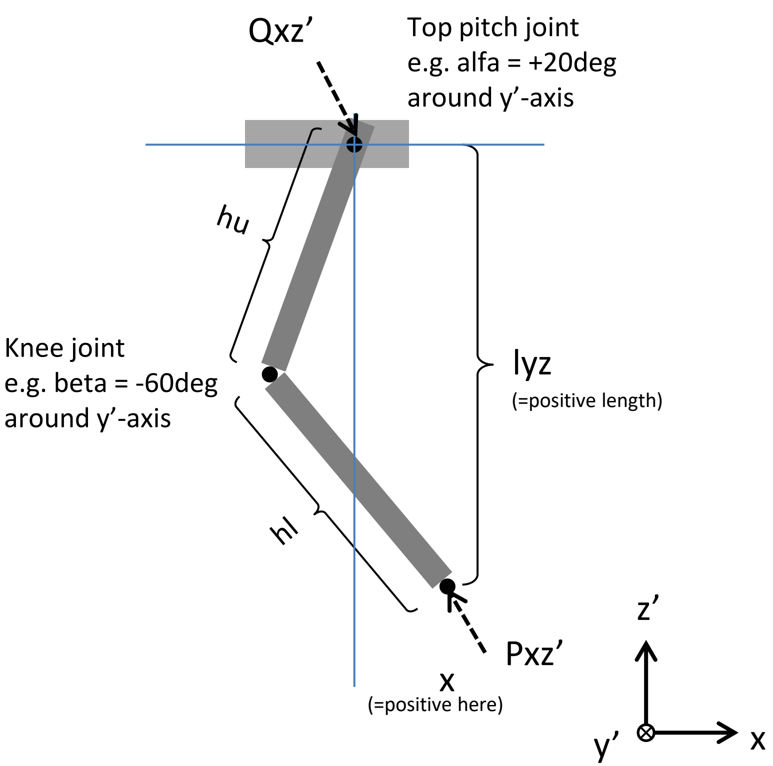
  e.g. (-45deg) – (-25deg) = -20deg = -0.35rad



Now focus on the 2-part leg

- Focus on the inclined x,z'-plane. The leg lies in this plane.
- The leg joints need to be set such that the leg height is lyz in the z' direction
- find length from Qxz' to Pxz'
- $lxz' = \mathrm{sqrt}(lyz^2+x^2)$
- Now we resolve beta
  - with triangle below



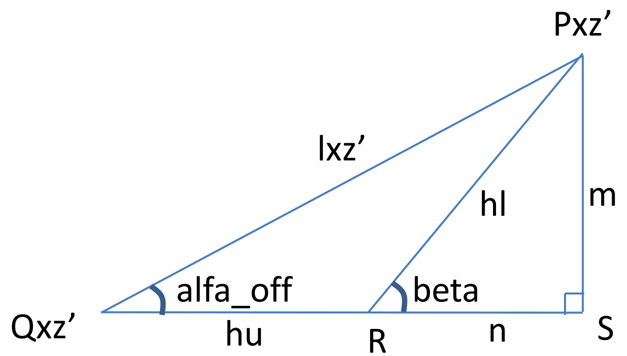


Find beta

- Note that for lxz’ and hu and hl given the triangle Pxz’ Qxz’ R is fixed and beta can be obtained
- Define the right-angled triangle Pxz’ Qxz’ S where S is the right angle
- Introduce lengths n and m
- Then triangle Pxz’ R S is also right-angled
- Pythagoras and a bit of algebra:
  - $(hu+n)^2 + m^2 = lxz'^2$
  - $n^2 + m^2 = hl^2$
  - subtract the second equation from the first
  - $(hu+n)^2 + m^2 - n^2 - m^2 = lxz'^2 - hl^2$
  - $2 * hu * n + hu^2 = lxz'^2 - hl^2$
  - $n = (lxz'^2 - hl^2 - hu^2) / (2 * hu)$
  - **$beta = -\arccos(n / hl)$**

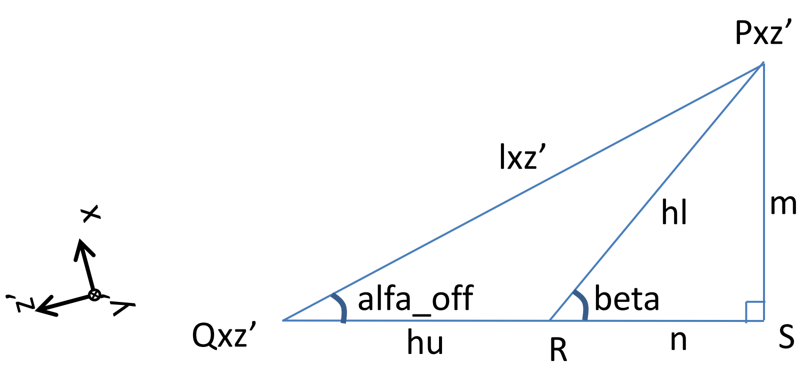
    (because cos(-beta) = n / hl)

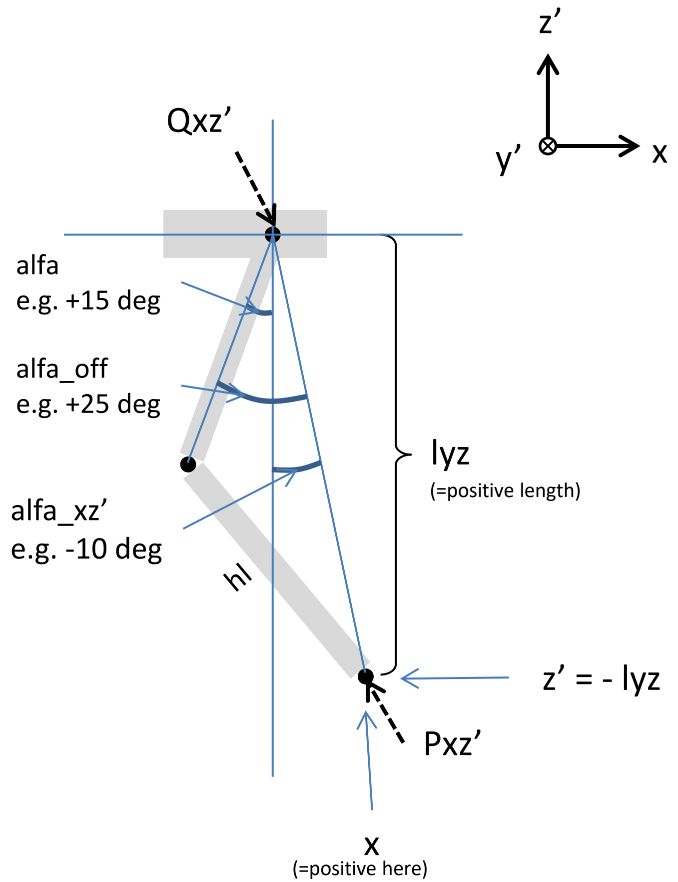
    minus because counter clockwise



Find alfa

- Find alfa_xz' from x and lyz
  - alfa_xz' = -arctan(x / lyz)
- Find alfa_off from the triangle Pxz' Qxz' S
  - alfa_off = arccos((hu+n) / lxz')
- **alfa = alfa_xz' + alfa_off**





Python function

```
def xyztoang(x,y,z,h,hu,hl):
   #Some sqrt’s can be optimized out
   dyz=np.sqrt(y**2+z**2)
   lyz=np.sqrt(dyz**2-h**2)
   gamma_yz=-np.arctan(y/z)
   gamma_h_offset=-np.arctan(h/lyz)
   gamma=gamma_yz-gamma_h_offset
   #
   lxzp=np.sqrt(lyz**2+x**2)
   n=(lxzp**2-hl**2-hu**2)/(2*hu)
   beta=-np.arccos(n/hl)
   #
   alfa_xzp=-np.arctan(x/lyz)
   alfa_off=np.arccos((hu+n)/lxzp)
   alfa=alfa_xzp+alfa_off
   #
   return [alfa,beta,gamma]

#Test
In[1]: xyztoang(0,-0.1,-2,0.1,1,1)
Out[1]: [0.0, -0.0, 0.0]
```

All angles are in radians here.
Multiply them by 180/Pi to get degrees.

Conclusions, observations

- Given x, y, z, h, hu and hl, we found formulas for **alfa, beta and gamma**: Inverse Kinematics with one Python function.
- For simple math it is important that the alfa and beta joint axes have the same orientation (parallel in 3D space).
- For simple math it is important that the gamma joint axis is orthogonal to the alfa joint axis orientation (gamma joint is perpendicular to alfa and beta joints).
- Additional translational or rotational offsets can be added quite easily. This allow more design flexibility and optimization.
- If the foot is a ball, it will just be a small offset in z.
- It seems good to define a consistent x, y, z reference frame and define orientations to not get lost.

Pybullet model