Coordinate Reference systems

geograbra program

Cartesian coordinate spaces

Model [GCode] centered at (0,0) for Wally preprocessor
gX gY gZ [mm]

Extruder Plane XY … [mm]
eX = …
eY = …
eZ defined by bed

Bed moves in YZ plane
bX = eX
bY = ƒ(eY, bZ) can vary by up to 20+ mm
bZ (0..150) [mm]

Should be encapsulated in an object

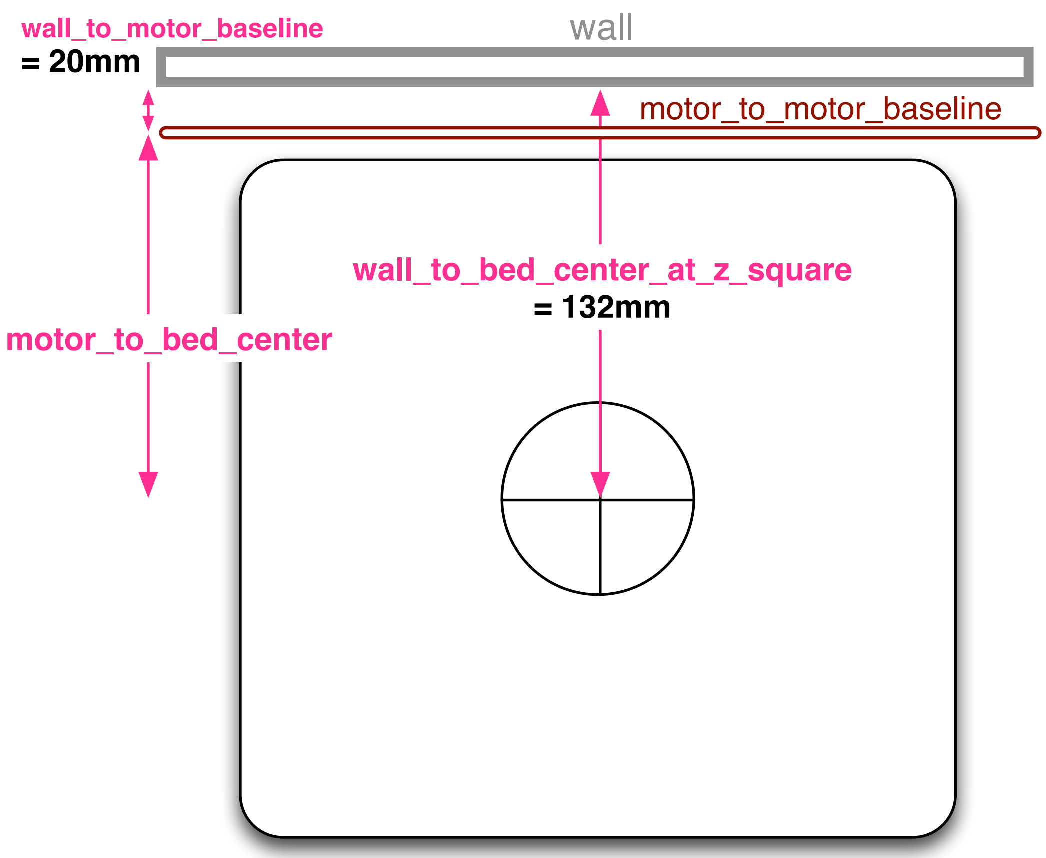
Bed tilt compensated Z
btX =
btY =
btZ = ƒ(btX, btZ)

Polar coordinates

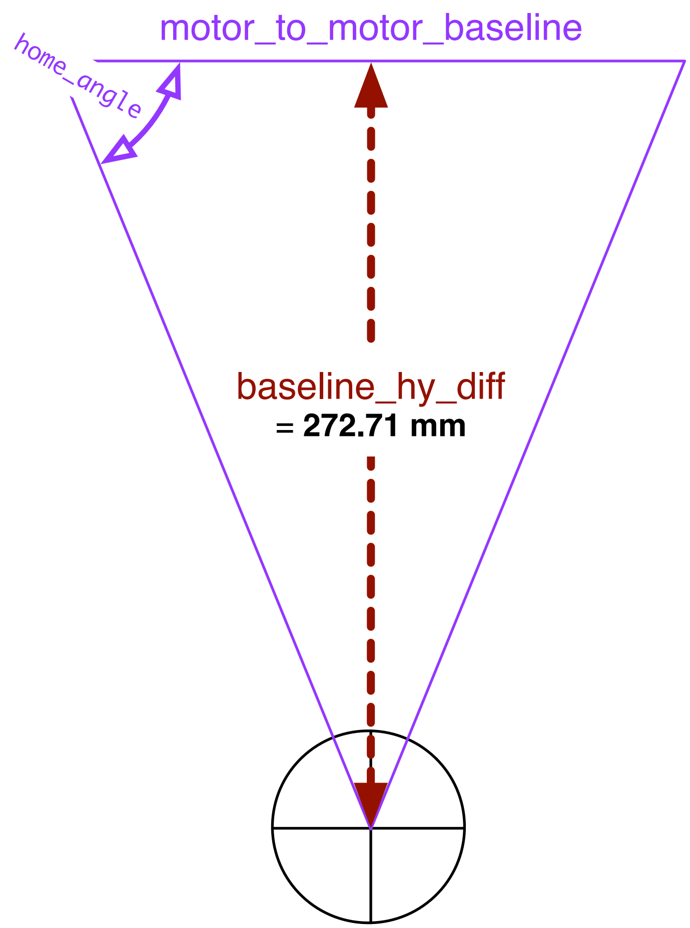
LR big pulley angle
straight arms is 180° (π)

Stepper scaling

```
degrees_per_motor_step = 1.8 #UI depends on your stepper motor
motor_steps_per_rev = 360.0 / degrees_per_motor_step
motor_micro_steps = 16 #UI set on motor drive board
micro_steps_per_rev = motor_micro_steps * motor_steps_per_rev
steps_per_gcode_unit = 8 #UI set in repetier
gunits_per_rotation = micro_steps_per_rev / steps_per_gcode_unit
rads_per_step = 2*math.pi / gunits_per_rotation
```

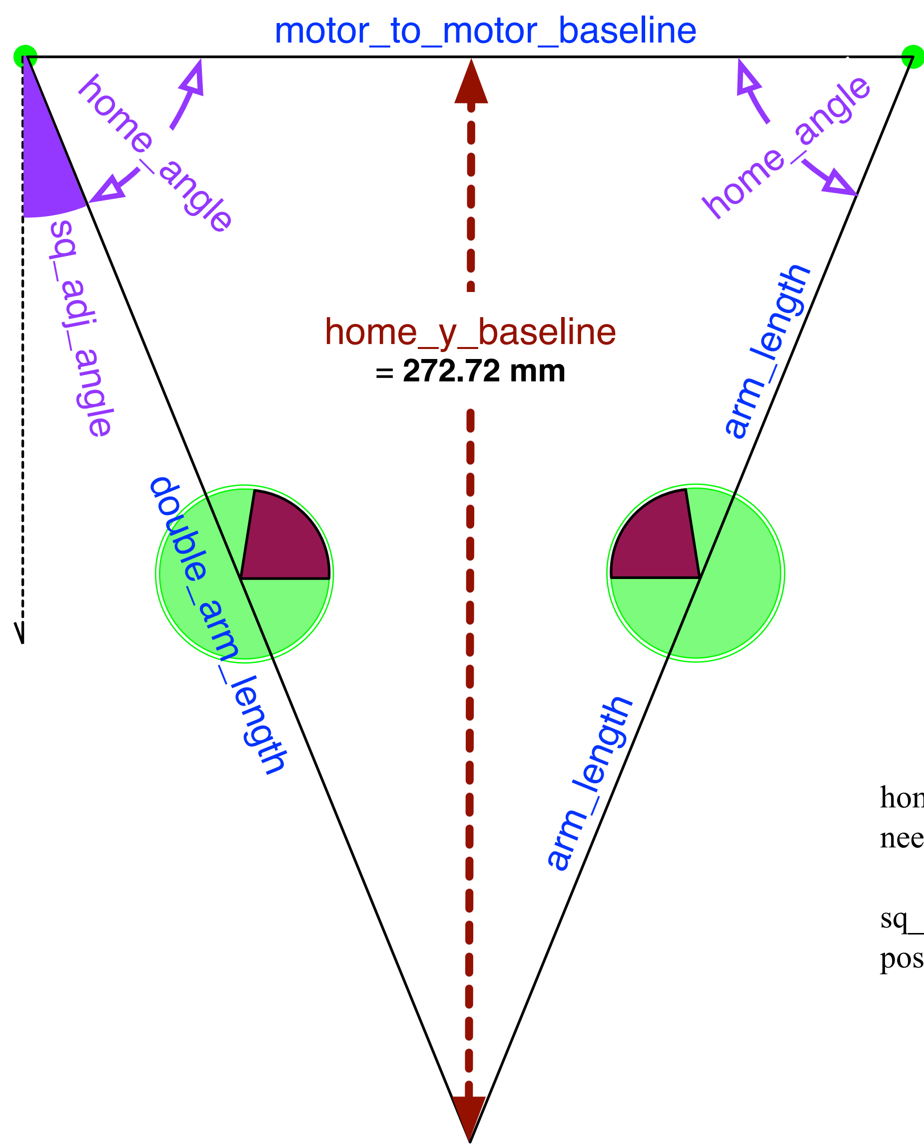



- Bed moves in Y as function of Z position




**Model Origin**
- **GCode origin at center of bed at most raised**
- **GCode (x,y,z) variables (gX, gY, gZ)**

```
baseline_hy_diff = sqrt(double_arm_length**2 -
    ( motor_to_motor_baseline/2.0 )**2)
# 272.71 mm
```




# Extruder Plane
# Home Geometry

***defines small pulleys' zero radian angle***

Constants

```
home_y_baseline =
      sqrt( double_arm_length**2 -
      ( motor_to_motor_baseline / 2.0 )**2 )
# 272.71mm

home_angle = asin( home_y_baseline /
      double_arm_length ) # 65.37°

sq_adj_angle = math.pi / 2 - home_angle
# 1280
```

home_angle is needed for calculating xy_arm's change from home is needed to adjust the small pulley angles effect on the big pulley angles.

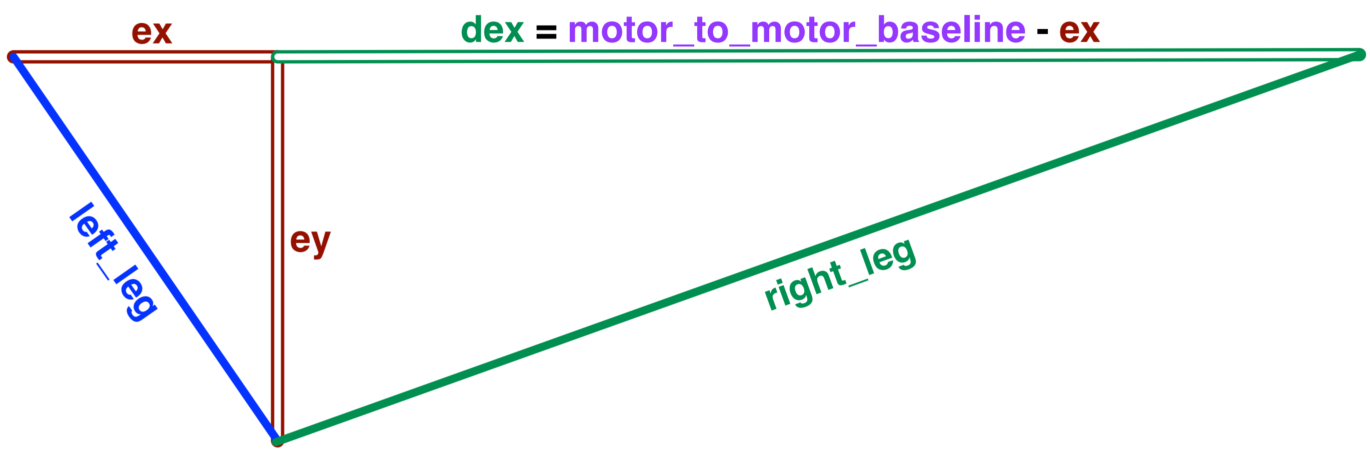
sq_adjust_angle is needed for adjusting pulley calculations at arms squared position

# Extruder Plane Geometry

Pythagorean Theorem a**2 + b**2 = c**2

**ex = gX + (motor_to_motor_baseline / 2)**

**ey = gY - mb_dfz**



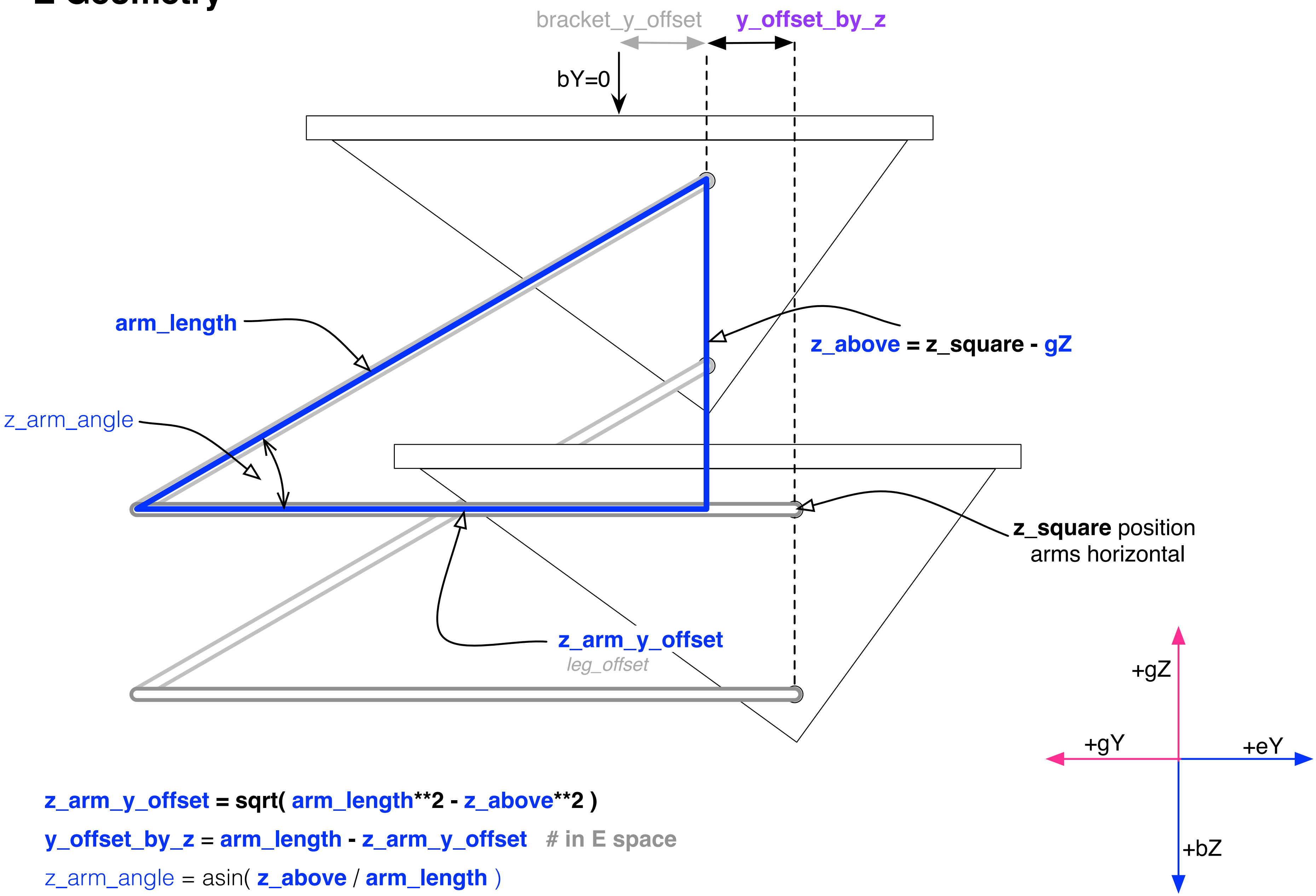

left_leg is c in Pythagorean Theorem

left_leg = sqrt(ex**2 + ey**2)

right_leg is c in Pythagorean Theorem

right_leg = sqrt( **dex****2 + **ey****2)

# Z Geometry




**z_arm_y_offset = sqrt( arm_length**2 - z_above**2 )**

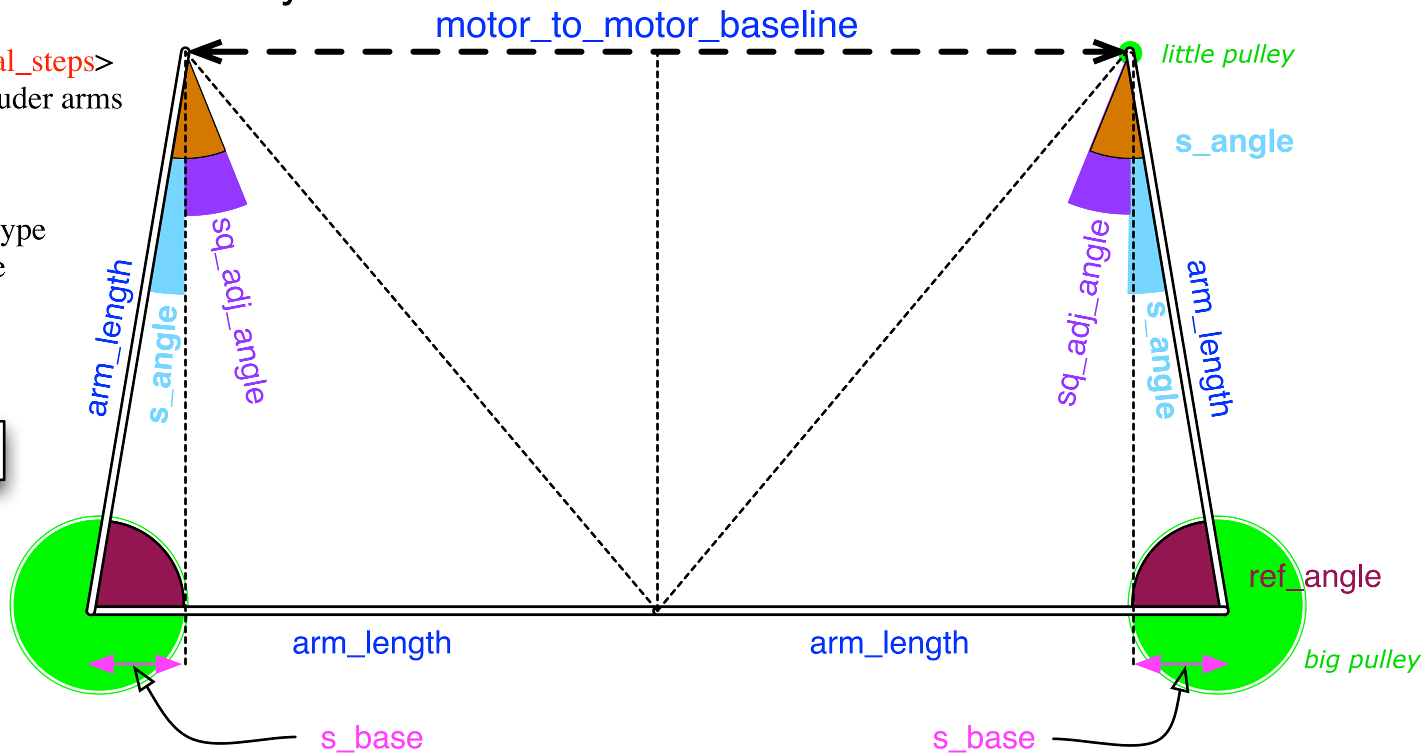
**y_offset_by_z** = **arm_length - z_arm_y_offset** **# in E space**

z_arm_angle = asin( **z_above** / **arm_length** )

# Pulley Calibration Geometry

G1 X<cal_steps> Y<cal_steps> is the value where extruder arms are collinear.

Start from home then increment X and Y or type G1 commands until the arms are in a line.




## Example

cal_steps = 1022.0

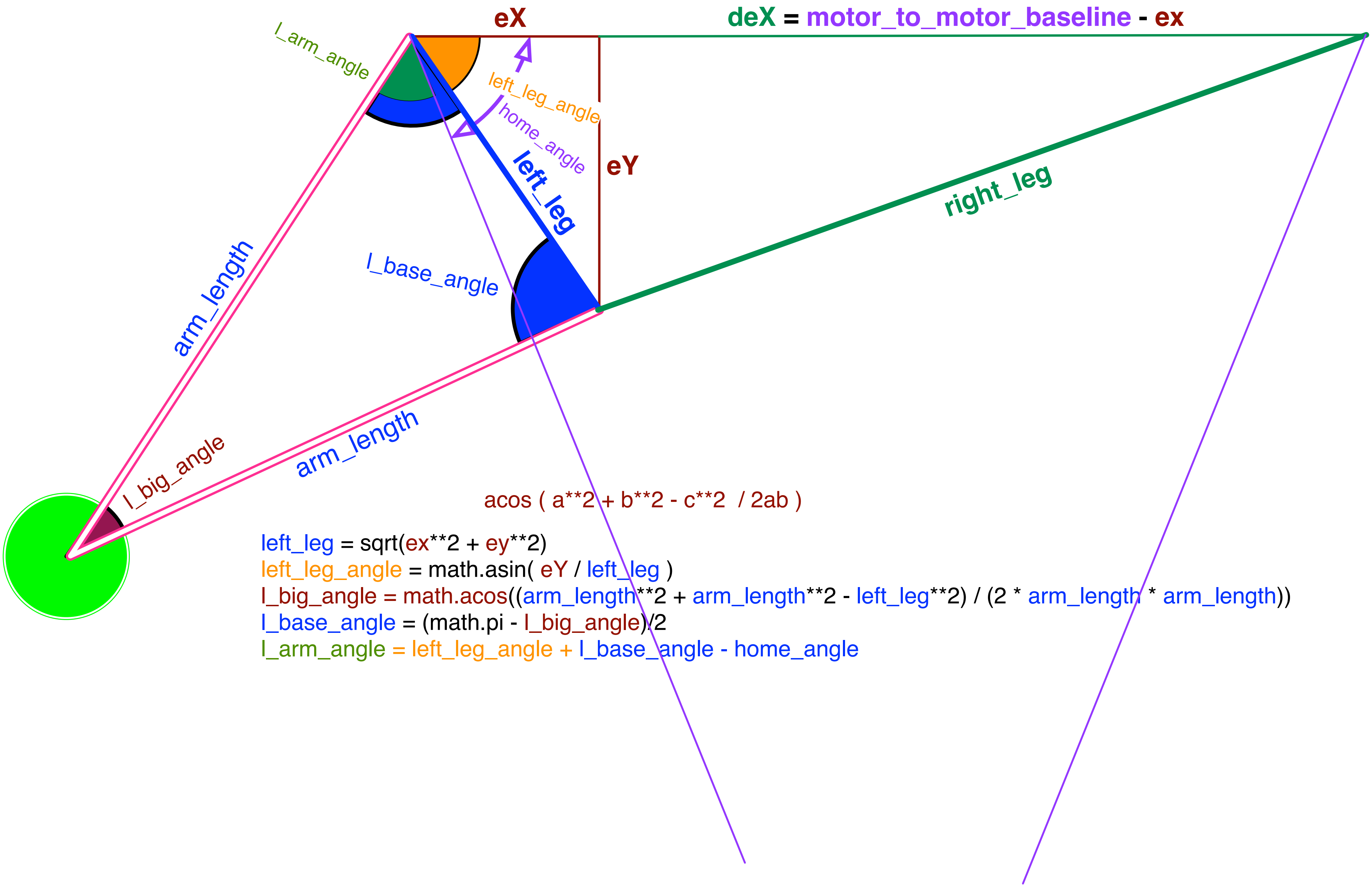
These constants **may** help map LR motor angles to big pulley angles. Little to big pulley mechanical advantage should be 10:100 (10) with possible deviations from theory due to imperfections.

## Constants

```
s_base = ( 2 * arm_length ) - motor_to_motor_baseline
s_angle = math.asin( s_base / arm_length ) # about 19.5°
ref_angle = math.pi / 2 - s_angle  # about 70.52°
little_pulley_rad_offset = sq_adj_angle + s_angle
```

At pulley calibration we've moved big pulley's angle from **π** (180°) to ref_angle (70.52°) using cal_steps but the arm above the motor moved from home position thru little_pulley_rad_offset. So R little pulley turns are not linearly related to the big pulley turns but depend on the arm angle above the motors **and L pulley turns.** The right pulleys moved CW while the arm moved CCW subtracting little_pulley_rad_offset to little pulley's work.

```
big_cal_rads = ref_angle
little_cal_rads = ( cal_steps * rads_per_step ) - little_pulley_rad_offset
little_to_big_cal_rads_ratio = little_cal_rads / big_cal_rads
```




```
left_leg = sqrt(ex**2 + ey**2)
left_leg_angle = math.asin( eY / left_leg )
l_big_angle = math.acos((arm_length**2 + arm_length**2 - left_leg**2) / (2 * arm_length * arm_length))
l_base_angle = (math.pi - l_big_angle)/2
l_arm_angle = left_leg_angle + l_base_angle - home_angle
```