Manfred

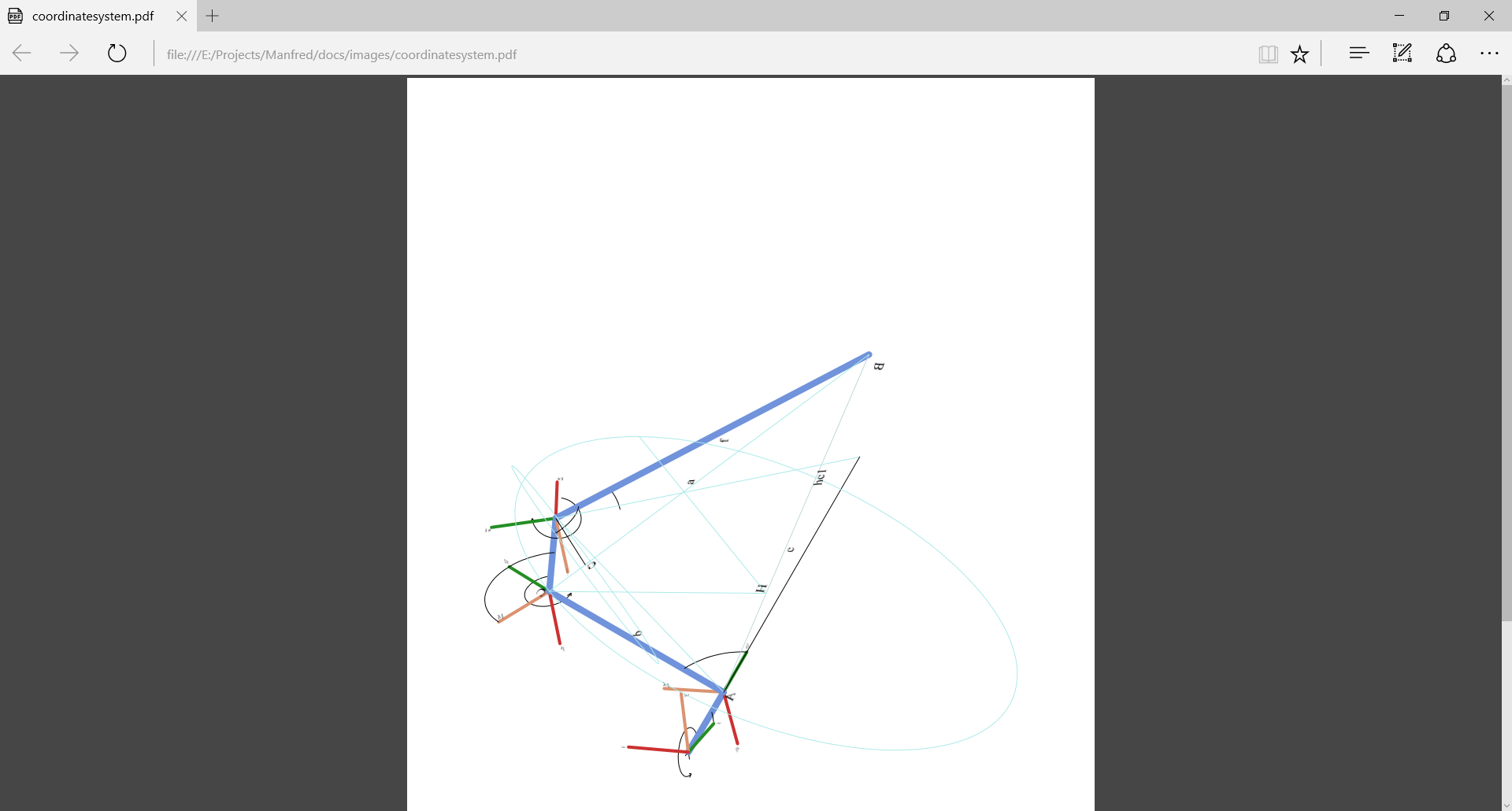
Jochen Alt



Leg Design

Typical Hexapods have 3-5 DOF per leg, most of them have only one DOF that move the from left to right and all other DOFs to move it up and down. I never go the reason behind, since when looking at most of the hexapods you get the impression of a “brutal” gait since the entire leg including the thigh (or so-called “femur”) is moving while \*not\* trying to minimize the mass that is to be accelerated. There’s just one hexapod I am aware of (“Weaver”) that spends one DOF for turning a leg such that the foot can move forward and backward without moving the femur. Unfortunately, this DOF is not used during walking but only to compensate standing on a ramp and levelling the body’s orientation.

In order to minimize the moved mass it seems that turning the knee is helpful, what led to a 4 DOF leg design with a turning possibility at the knee:



The turning knee allows to move the foot point point forward and backward without moving the tibia. In the end, this should allow for more efficient movements since the tibia moves just half the way compared to a classical design without a turning knee.

Kinematics

Kinematics is about computation of the leg’s touch point out of the joint angles and vice versa. First is simple, latter is tricky. The coordinate systems are illustrated above, such that we can derive the Denavit Hardenberg, which is:

The transformation from anglei to anglei+1 is given via

1. rotation around the z-axis by joint angle
2. translation along the z-axis by *d*, and
3. translation along the x-axis by *a*
4. rotating around the x-axis by 

So, the Denavit Hardenberg parameters are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Joint | a[°] | a[mm] | d[mm] |  |
| hip | 90° | 0 | d0 |  |
| thigh | 0 | d1 | 0 |  |
| knee | 90° | d2 | 0 |  |
| [ alignment | 90 | 0 | 0 | 90 ] |
| lower leg | 90° | 0 | d3 |  |

The [alignment] line is necessary to let the lower leg turn according to the convention that turns around the z-axis.

The general definition of a Denavit-Hardenberg (*DH*) transformation is

|  |  |
| --- | --- |
|  | (6‑1) |

which is a homogeneous matrix with two rotations (x,z) and two translations (x,z).

Combined with the DH parameters, the following DH matrixes define the transformation from one joint to its successor:

|  |  |
| --- | --- |
|  | (6‑2) |
|  | (6‑3) |
|  |  |
|  | (6‑4) |
|  |  |
|  | (6‑5) |
|  |  |
|  | (6‑6) |
|  |  |

Forward Kinematics

With the DH transformation matrixes at hand, computation of the leg’s pose out of the joint angles is straight forward. The matrix representing the gripper’s pose is

|  |  |
| --- | --- |
|  | (6‑8 |

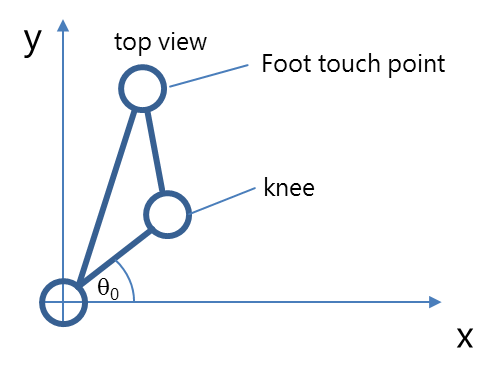
By multiplying the transformation matrix with the origin (as homogeneous vector), we get the absolute coordinates of the foot touch point (*FTP*) centre point in world coordinate system (i.e. relative to the legs’s base).

|  |  |
| --- | --- |
|  | (6‑9) |

That was easy. The tricky part comes now.

Inverse Kinematics

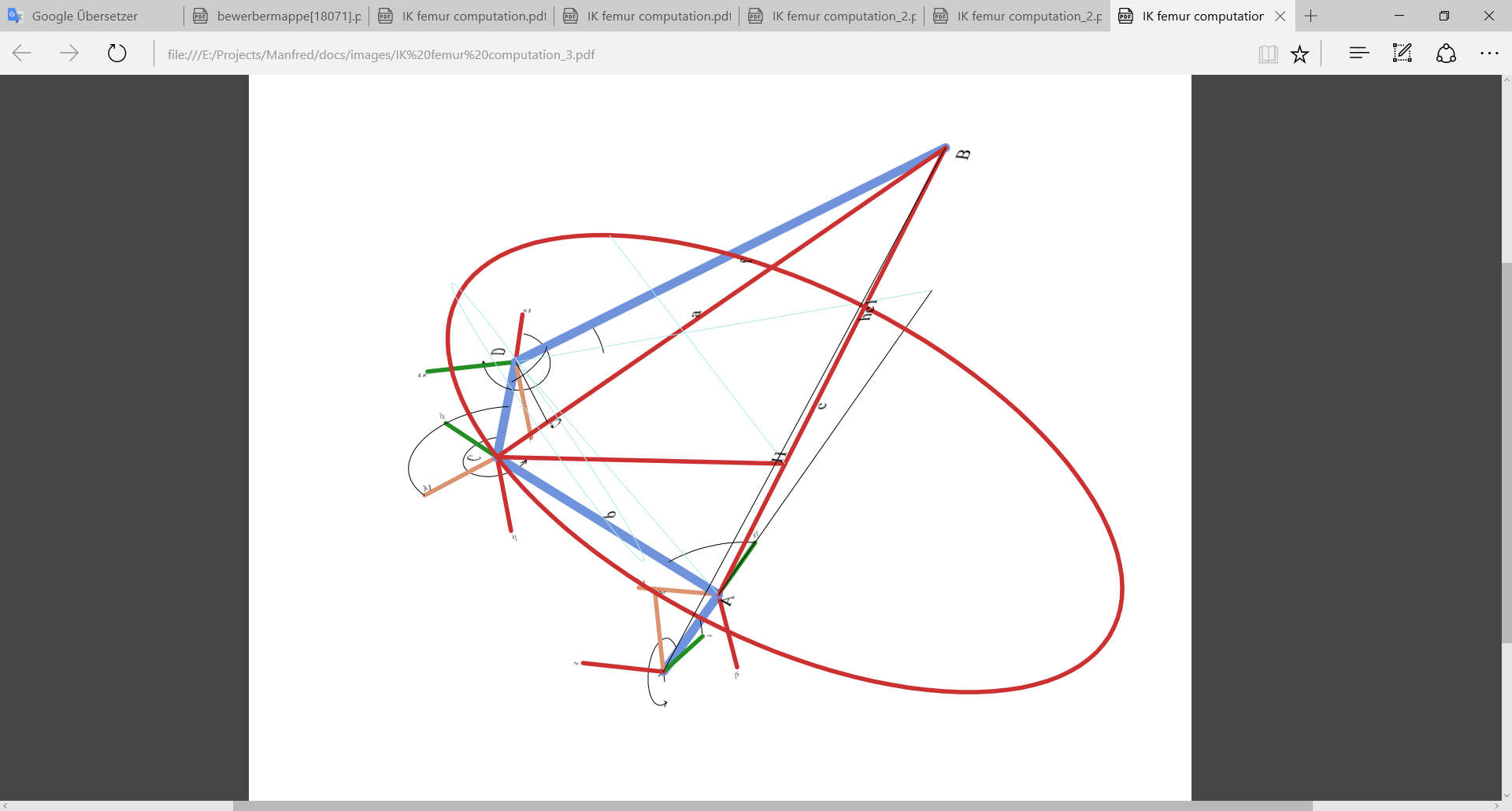
Inverse kinematics denotes the computation of all joint angles out of the foot touch point’s position (*FTP*). Since the leg has four joints, it is becomes clear that there is an infinite number of solutions for that, so we need to predefine one angle with an arbitrary definition. Having the objective in mind of moving the higher limbs of the leg as little as possible, I arbitrarily chose ** and set it as angle bisector of the hidden of the foot touch point to the hip (bird’s perspective):



We get

Later on, we will need the coordinates of end of the first limb (A) which is

Computation of the second angle ** at point A requires a geometric analysis. The leg is denoted in blue, all construction lines are red.

We consider the triangle from FTP, A, and B. Since the two lines and are of fixed length, has a static length as well. So, the point C is upon the circle with the centre H and the radius of the triangle’s height. Additionally, C is defined as function of ** and **, so we should be able to derive ** by intersecting the circle with *C( ,)*.

The only thing we need to do is to express that in terms of coordinates. First, we compute the length of a, b and c:

Now that the triangle is defined, we can compute the height by Herons formula:

The base of the height H is defined by

Now we need to define the circle *K* with radius *h* and centre *H*. This is done by

with S orthogonal to beginning from *H* and T orthogonal to S and :

So, with the arbitrary assumption and the length

we get

(This equation could be simplified, but this way programming is easier by computing the y coordinate and deriving the x coordinate)

There are two possibilities for S, representing two configuration with knee up and knee down. We always take the healthy one where the knee is above the foot touch point. Finally, T is defined by its orthogonality to S and its length :

Having the circle defined we need to intersect it with the possible positions of C:

Hereby denotes . We consider only the equations of x and y coordinates and solve these for . Equating gives

This needs to be solved by in order to get point C. Unfortunately we have sin and cos in the equation, but luckily with the same parameter. So Wikipedia helps with sinusoids:

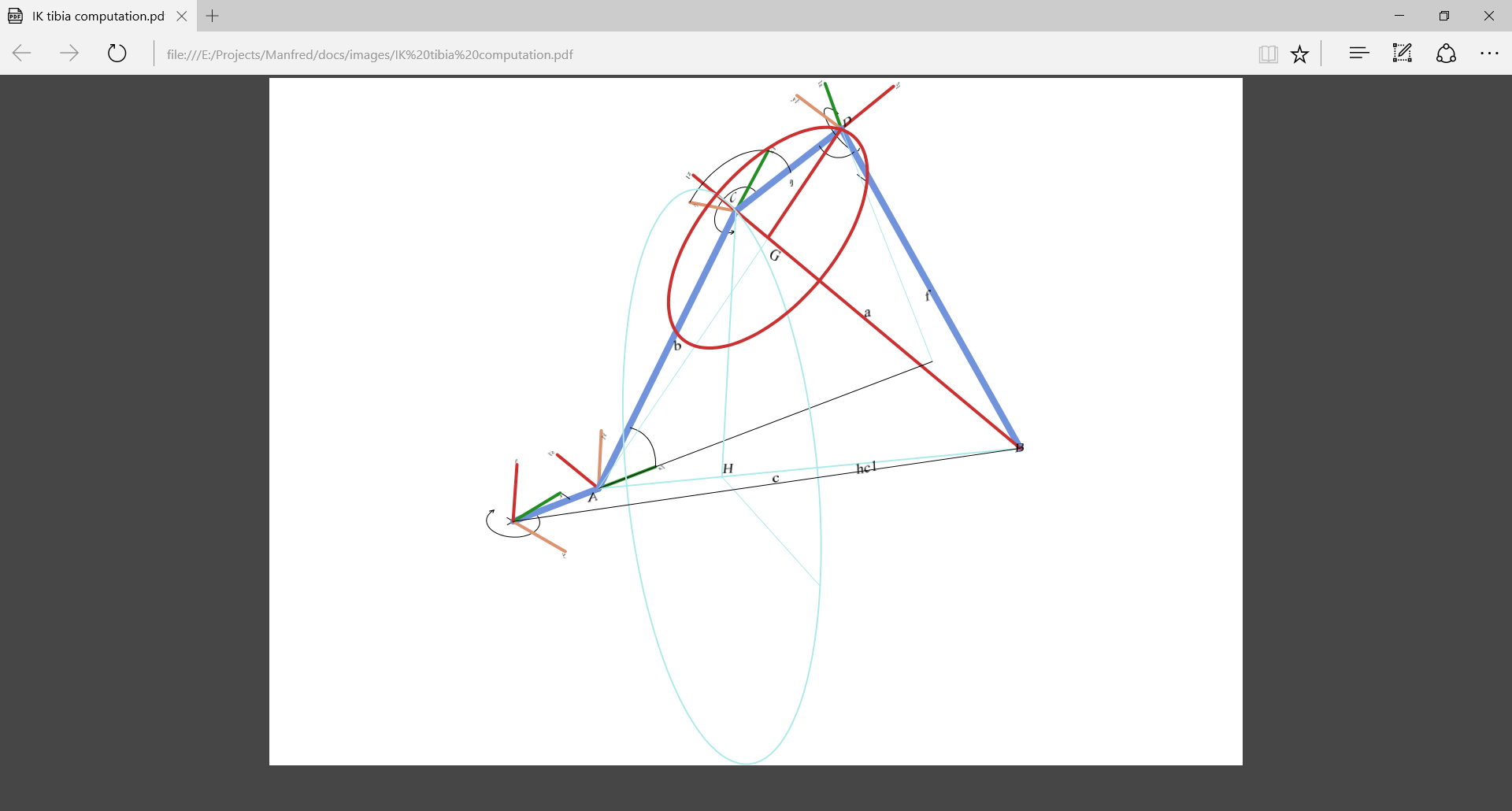
that can be used to solve the equation above for 

a =

Out of  we get C by , out of C we compute **by considering the z-coordinate of C:

which results in

The first angle is always the hardest, time for a beer.

The next angle **is done by the same approach but the red triangle BCD. First we compute point G where the height meets BC, then we compute the circle with centre G and radius ha. We call this circle K:

with

This time, we intersect the circle with the pane P of the triangle ACG, which is

Intersection happens by equating the circle with the pane P:

We consider the x and y coordinates only and by solving for  and equating we get

This can be solved for  by applying the sinusoid above giving two solutions for D.

Additionally, the transformation matrix for D gives

Therefore we can solve for ** by use of sinusoids again. Considering

with and results in two solutions of **:

Since we have two solutions for D, we need to check by another coordinate of D, we take the y coordinate:

and compare, which solution D gives the same value like the forward computation with the transformation matrizes (Geometrically, the second solution represents an unhealthy position with knee down and foot up).

Finally, the last angle ** does not need a special trick, but can be derived out of the z coordinate of transformation matrix, since there is only one *s3*:

We solve directly for **:

and that’s it. Surprisingly complex for a leg with only 4 degrees of freedom.