Combinatorics-based method for derivation of solution sets for Inverse Kinematics
June 18, 2024

1

One of our main learnings in this project is that robot kinematic equation solutions in general are NOT described by a tree structure! Instead they are a more general graph. For example, two variable solutions might be independent of each other but contribute to the other variables' solutions (multiple 'roots' to the graph). Our previous solution was overly complex due to lingering assumptions from the tree structure idea. We have done a complete re-write of the solution set generator process with seemingly good results. The key insight was to develop the table of solutions, adding rows as each solved variable creates combinations.

In the new approach, for each solved variable, we distinguish between the number of "solutions" and the number of "versions" it has. The number of solutions is determined by the mathematical form of the solution for example:

$$x = \sin(\theta) \tag{1}$$

has two solutions (and also two versions).

$$\theta = [\arcsin(x), \pi - \arcsin(x)] \tag{2}$$

However a variable whose solution is single valued, could have multiple versions of it depends on a variable with multiple versions. For example,

$$y = \theta + \pi \tag{3}$$

y has one solution, but two versions

$$y = [\pi + \arcsin(x), 2\pi - \arcsin(x)] \tag{4}$$

depending on which version of θ is used. We identify previously solved unknowns (such as x) appearing in the right hand side of solution equations as "dependencies".

In general, an unknown may have multiple solutions to its equation, and may have more than one dependency, and all the combinations of versions of the dependencies generate versions of the current variable. To summarize, if a variable has n_s solutions and n_d combinations of its dependencies, Then it has $n_s \times n_d$ versions. If a variable x_i has m different variables as dependencies, then it as

$$n_{vi} = n_{sj} \Pi_{j=1}^{m} n_{vj} \tag{5}$$

Note that we have studied this problem solely for the case of $n_{sj} \in \{1, 2\}$.

We illustrate this with a small system of equations which is easy to solve but has these versioning characteristics of IK problems. Let the problem be to find all sets of the unknowns $[x_1 \dots x_5]$ which satisfy the following five equations.

$$x_1 - x_4 - x_2 = 0 \tag{6}$$

$$x_2 - x_4/2 = 0 \tag{7}$$

$$9 - x_3 = 0 \tag{8}$$

$$x_4 - \sqrt{x_3} = 0 \tag{9}$$

$$\sqrt{4} - x_5 = 0 \tag{10}$$

where each of the x_i are unknowns and the other terms (here they are numbers) are known. By inspection, we can solve these in the order $[(x_3, x_5)(tie), x_4, x_2, x_1]$.

1. x_3 is a trivial solution: $x_3 = 9$ and there is only one solution.

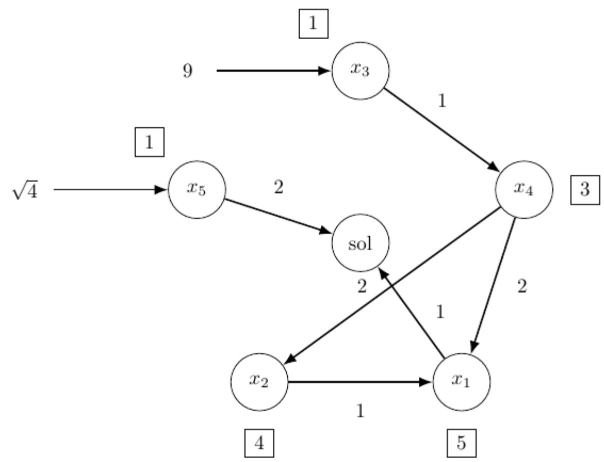

Figure 1: Dependencies to solution of Equations 6 to 10. Edge weights are n_{si} of the origin of the edge. The square annotation gives the solution order of each variable. "sol" node is the full solution set.

2. x_5 is simple, but there are two solutions: $[\sqrt{4}, -\sqrt{4}]$.
3. x_4 similarly has two solutions: $[\sqrt{9}, -\sqrt{9}]$.
4. x_2 has only one solution, $x_4/2$, but we have two *versions* due to the two solutions of x_4: $[1.5, -1.5]$
5. x_1 depends on both x_4 and x_2 so though it has one solution, it has four versions (in general distinct): $[4.5, -4.5, -4.5, 4.5]$ due to the combinations of versions of its two dependencies.

var.,	multiplicity	Solutions or Versions
x_3	1	$[9]$
x_5	2	$[2, -2]^S$
x_4	2	$[3, -3]^S$
x_2	1	$[1.5, -1.5]^V$
x_1	1	$[4.5, -4.5, -4.5, 4.5]^V$

The solutions dependencies can be represented as a graph (Figure 1), but unlike a tree, the graph is not especially helpful in finding the solution sets.

Next, we assemble solution vectors by following the solution order as follows:

1. x_3 has one solution, $x_{3s1} = 9$ giving one version

$$x_{3v1} = x_{3s1} = 9 \tag{11}$$

We will continue by collecting these versions into a set of vectors:

$$[x_{3v1}] \tag{12}$$

2. x_5 has two solution but depends on no previous unknowns. Its versions are thus $x_{5v1} = x_{5s1} = 2, x_{5v2} = x_{5s2} = -2$. We thus double the number of rows giving:

$$\begin{bmatrix} x_{3v1} & x_{5v1} \\ x_{3v1} & x_{5v2} \end{bmatrix} \tag{13}$$

note that the order of these two variable columns does't matter since they are independent of each other.

3. x_4 has two solutions based on the multiple solutions of Eqn 9 which depend on x_3 (having only one version) but these solutions are independent of x_4 and x_5. We thus have to double the rows again and we have to flip the order of the newly added rows to generate the full set of combinations:

$$\begin{bmatrix} x_{3v1} & x_{5v1} & x_{4v1} \\ x_{3v1} & x_{5v2} & x_{4v2} \\ x_{3v1} & x_{5v2} & x_{4v1} \\ x_{3v1} & x_{5v1} & x_{4v2} \end{bmatrix} \tag{14}$$

4. x_2, similarly has one solution (so we do not double the rows) but depends on the versions of x_4 so it has 4 versions.

$$\begin{bmatrix} x_{3v1} & x_{5v1} & x_{4v1} & x_{2v1} \\ x_{3v1} & x_{5v2} & x_{4v2} & x_{2v2} \\ x_{3v1} & x_{5v2} & x_{4v1} & x_{2v3} \\ x_{3v1} & x_{5v1} & x_{4v2} & x_{2v4} \end{bmatrix} \tag{15}$$

5. x_1 has only one solution, but it depends on two variables: x_2 which has 2 versions, and x_4 which has two versions. Applying Eqn 5. we have 4 versions:

$$\begin{aligned} x_{1v1} &= x_{4v1} + x_{2v1} \\ x_{1v2} &= x_{4v2} + x_{2v2} \\ x_{1v3} &= x_{4v1} + x_{2v3} \\ x_{1v4} &= x_{4v2} + x_{2v4} \end{aligned} \tag{16}$$

but we do not increase the number of rows because of the single solution to Eqn 6.

$$\begin{bmatrix} x_{3v1} & x_{5v1} & x_{4v1} & x_{2v1} & x_{1v1} \\ x_{3v1} & x_{5v2} & x_{4v2} & x_{2v2} & x_{1v2} \\ x_{3v1} & x_{5v2} & x_{4v1} & x_{2v3} & x_{1v3} \\ x_{3v1} & x_{5v2} & x_{4v2} & x_{2v4} & x_{1v4} \end{bmatrix} \tag{17}$$

The above process can be summarized as:

Once an algorithm such as IKBT has found solutions to all n_u unknowns we can then collect valid solution vectors into an $n_u \times n_v$ matrix as follows:
For each variable x_i in solution order:

1. Determine its number of solutions from its solution method. Example: $\sqrt{x}$ has 2 solutions)
2. Determine its number of dependencies on previously solved unknowns and determine the number of versions of each dependency: n_{vj}. Example: $y_5(y_2, y_4)$ where y_2 has 4 versions and y_4 has 2 versions.
3. Compute the number of versions for x_i, n_{vi}, using Eqn 5.
4. If the new number of versions is a multiple of the n_d per Eqn 5, then repeat the previous rows n_{si} times, but flip their order[1].
5. Number the versions x_{ivj} where i selects the variable and j selects the version number.
6. Enumerate and save the expression for each version. For example: if the solutions are
   - $x_{4s1} = -\sqrt{9}$
   - $x_{4s2} = \sqrt{9}$

   Then renumber them by row number x_{4vi} and add them as a new column to the solution vector matrix. If n_{si} is less than the number of versions, n_{vi}, copy the solutions n_{di} times and append them to to get n_{vj} rows.

[1]Open Issue: Understood so far only for $n_{sj} \in \{1, 2\}$. If a variable had e.g. 4 solutions, how to flip the quadrupled rows??