Kinematic Solvers

Kinematic

Study of displacement and it's first and second derivate, without considering Force (Mass and Inertia)

Kinematics of Robot

Kinematics study of joint value, joint velocity and joint acceleration with respective to other links.

Forward Kinematics

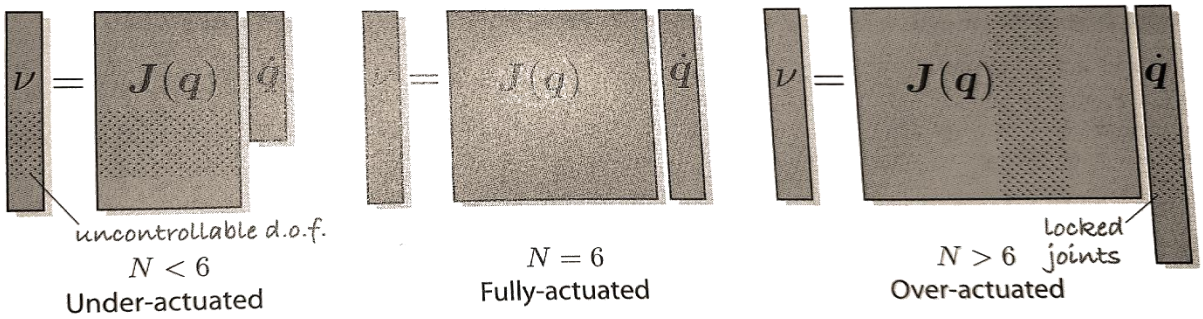
- Given Joint value
- To find End Effector Position
- Solution Multiply Joint value by Jacobian.

Inverse Kinematics

- Given End Effector Position
- To find Joint value
- Solution Multiply End Effector Position by Inverse Jacobian.

Jacobian is Matrix of order M*N where

- M is DOF in cartesian space
- N is No of Joints of Manipulator



Kinematic Solvers

- An algorithmic which does the solving part, i.e., last step mentioned above.
- Especially during IK, solver gets any one of the following situation
  - Existence of multiple solutions.
  - Possible non-existence of a solution.
  - Singularities

What is singularity?

- Singularity makes axis parallel and thus losing a DOF
- Mathematically they are points at which determinant of Jacobin is zero.
- In terms of intuition we have infinite ways to achieve singularity configuration.
- The speed of some joints becomes suddenly very large
- The cartesian velocity of the end-effector is significantly reduced.
- It gets complicated with increasing DOF

How to avoid singularities?

Avoid them by avoiding the arm to move into such configuration. (cartesian admittance control)

Under-actuated

Not Possible without sacrificing any one value of dimension.

Fully-actuated

Replace Jacobian with damp-Inverse Jacobian however it will introduce some error in q

$$\dot{q} = (J(q) + pI)^{-1}\nu$$

Over-actuated

$$\dot{q} = \underbrace{J(q)^{+}\nu}_{\text{end-effector motion}} + \underbrace{NN^{+}\dot{q}_{ns}}_{\text{null-space motion}}$$

Here we project the desired joint space into a null space such that, it won't affect the end effector Cartesian motion.

Major Kinematic Solver

- KDL
- IK-Fast
- Track-IK

Selection of Kinematic Solver

1. **IK Fast** is faster than **KDL** due to analytical method but it can solve **only up to 6 DOF**.
2. Our robotic arm has 7 DOF plus an End effector, hence **IK Fast is rule out**.
3. Now when we compare **Track-IK with KDL**

Humanoid Kinematics Chain			IK Error	IK Technique													
Robot	Chain Description	DOFs	Position/ Rotation Error	Plain KDL		KDL-RR		SQP		SQP-DQ		SQP-SS		SQP-L2		TRAC-IK	
				Solve Rate (%)	Avg Time (ms)	Solve Rate (%)	Avg Time (ms)	Solve Rate (%)	Avg Time (ms)	Solve Rate (%)	Avg Time (ms)	Solve Rate (%)	Avg Time (ms)	Solve Rate (%)	Avg Time (ms)	Solve Rate (%)	Avg Time (ms)
Robonaut2	Arm + Waist	8	1E-6 / 1E-6	78.18	0.61	85.36	0.75	17.93	3.43	96.46	1.06	97.76	0.91	97.12	1.81	98.72	0.68
	Arm	7	1E-6 / 1E-6	85.82	0.61	91.21	0.50	27.66	2.95	97.76	0.88	98.85	0.64	98.29	1.31	99.54	0.40
	Leg + Waist	8	1E-6 / 1E-6	76.05	0.71	81.21	0.83	17.41	3.52	96.93	1.34	97.99	1.16	97.08	2.17	99.13	0.71
	Leg	7	1E-6 / 1E-6	60.82	0.58	77.37	0.87	22.31	3.42	97.59	1.32	98.36	1.13	97.81	1.88	99.26	0.73
TRACBot	Long Arm	7	1E-6 / 1E-6	78.88	0.39	90.13	0.59	26.85	3.24	99.85	0.93	99.88	0.75	99.83	1.73	99.95	0.44
	Short Arm	6	1E-6 / 1E-6	46.09	0.17	71.86	0.63	26.91	2.92	98.55	1.09	97.45	1.01	97.08	1.56	98.81	0.68
Atlas 2013	Arm + Back (r,p,y)	9	1E-6 / 1E-6	89.59	0.79	91.95	0.84	31.00	3.04	99.57	0.65	99.88	0.47	99.72	1.37	99.89	0.39
	Arm + Back (p,y)	8	1E-6 / 1E-6	87.76	0.78	91.21	0.85	28.69	2.75	99.82	0.60	99.91	0.41	99.88	1.14	99.90	0.38
	Arm + Back yaw	7	1E-6 / 1E-6	85.36	0.59	90.84	0.69	30.53	2.63	99.84	0.53	99.92	0.36	99.89	0.95	99.95	0.34
	Arm	6	1E-6 / 1E-6	75.53	0.15	90.66	0.25	27.04	2.25	99.71	0.60	99.11	0.43	99.23	0.83	99.85	0.26

Atlas 2015	Arm + Back (r,p,y)	10	1E-6 / 1E-6	93.93	0.75	94.18	0.76	19.93	3.35	98.90	0.94	99.44	0.81	99.37	1.73	99.65	0.56
	Arm + Back (p,y)	9	1E-6 / 1E-6	91.22	0.83	91.99	0.85	22.00	3.45	99.28	0.86	99.52	0.78	99.54	1.47	99.72	0.53
	Arm + Back yaw	8	1E-6 / 1E-6	83.26	0.66	86.60	0.72	27.65	3.13	99.20	0.79	99.53	0.68	99.57	1.25	99.50	0.51
	Arm	7	1E-6 / 1E-6	75.39	0.39	85.50	0.55	28.38	3.02	98.98	0.78	99.32	0.65	99.28	1.09	99.45	0.42
Valkyrie	Arm + Back (r,p,y)	10	1E-6 / 1E-6	84.32	1.08	85.43	1.10	24.97	3.26	99.63	0.85	99.82	0.63	99.72	1.69	99.90	0.57
	Arm + Back (p,y)	9	1E-6 / 1E-6	77.62	1.25	80.40	1.32	24.91	3.36	99.38	0.82	99.65	0.60	99.44	1.45	99.85	0.58
	Arm + Back yaw	8	1E-6 / 1E-6	66.73	1.08	77.49	1.28	27.90	3.06	99.17	0.82	99.69	0.58	99.52	1.26	99.67	0.59
	Arm	7	1E-6 / 1E-6	44.83	0.60	82.10	1.16	31.67	2.96	99.05	0.90	99.61	0.62	99.40	1.16	99.83	0.51

The above table is taken from, “TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics” by **Patrick Beeson** and **Barrett Ames**

4. We can clearly see that, **Track-IK is faster in computational time and better in solve rate than KDL**
5. Reason?

5a) Track IK uses KDL's enhancement solver (which detects and mitigates local minima that can occur when joint limits are encountered during gradient descent)

5b) A **Sequential Quadratic Programming** IK formulation that uses quasi-Newton methods that are known to better handle non-smooth search spaces

KDL disadvantage

1) frequent convergence failures for robots with joint limits,

2) no actions taken when the search becomes “stuck” in local minima,

3) inadequate support for Cartesian pose tolerances,

4) no utilization of tolerances in the IK solver itself.

Issues 2–4 above can be mitigated by simple implementation enhancements to KDL’s inverse Jacobian solver.

Issue 1, however, requires consideration of alternative IK formulations. i.e., **sequential quadratic programming**.

Sequential Quadratic Programming

- Sequential quadratic programming (SQP) is an iterative method for nonlinear optimization with unequal and equality constraints
- SQP methods are used on objective function and the constraints which are twice continuously differentiable.
    - objective function is the function that it is desired to maximize or minimize.
- If the problem is **unconstrained**, then the method **reduces to Newton's method** for finding a point where the gradient of the objective vanishes.
- If the problem has only **equality constraints**, then the method is equivalent to applying Newton's method to the first-order optimality conditions, **or Karush–Kuhn–Tucker** conditions, of the problem.
- **Critical points of the objective function will also be critical points of the Lagrangian function** and vice versa because the Lagrangian function is equal to the objective function at a KKT point.

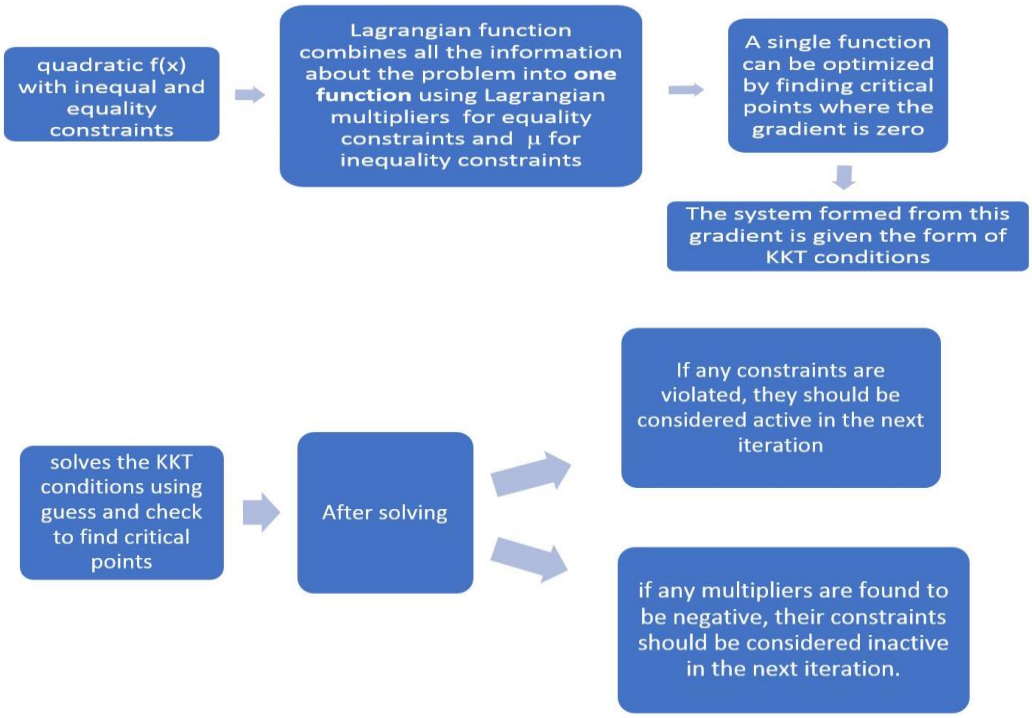
Quadratic subproblem with linear constraints can be solved efficiently:

- General case: $\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$
- KKT condition: $\begin{bmatrix} \mathbf{Q} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} -\mathbf{c} \\ \mathbf{b} \end{Bmatrix}$
- Solution: $\begin{Bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{bmatrix} \mathbf{Q} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{Bmatrix} -\mathbf{c} \\ \mathbf{b} \end{Bmatrix}$

Disadvantage

It incorporates several derivatives, which likely need to be worked analytically in advance of iterating to a solution, so SQP becomes **quite cumbersome** for large problems with many variables or constraints.

Process



It iterates till we have solution coverages.

Newton's Method

- It is to improve a guess in proportion to how quickly the function is changing at the guess and inversely proportional to how the function is accelerating at the guess.
    - a long, steep incline in a function will not be close to a critical point, so the improvement should be large

    - a shallow incline that is rapidly expiring is likely to be near a critical point, so the improvement should be small.

- Near minimums, a positive gradient should decrease the guess and vice versa, and the divergence is positive
- Near maximums, a positive gradient should increase the guess and vice versa, but the divergence is negative.
- This sign convention also prevents the algorithm from escaping a single convex or concave region; the improvement will reverse direction if it overshoots.
- Newton's method will find the critical point closest to the original guess.
- Incorporating Newton's Method into the active set method will transform the iteration above into a matrix equation.