ASBR PA 2

Jian Chu (jc86537) & Yang Liu (yl34825)

March 2020

Mathematical Approach

Forward Kinematics

In the $z-axis$, $0.3330 + 0.3160 + 0.3840 = 1.0330$

The end-effector zero position configuration M is the same, which is $M = \begin{bmatrix} 1 & 0 & 0 & 0.0880 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.9260 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

The screw axes S_i in $\{0\}$ are:

i	w_i	q_i
1	(0, 0, 1)	(0, 0, 0.3330)
2	(0, 1, 0)	(0, 0, 0.3330)
3	(0, 0, 1)	(0, 0, 0.6490)
4	(0, -1, 0)	(0.0825, 0, 0.6490)
5	(0, 0, 1)	(0, 0, 1.0330)
6	(0, -1, 0)	(0, 0, 1.0330)
7	(0, 0, -1)	(0.0880, 0, 1.0330)

The screw axes B_i in $\{b\}$ are:

i	w_i	q_i
1	(0, 0, -1)	(-0.0880, 0, 0.5930)
2	(0, -1, 0)	(-0.0880, 0, 0.5930)
3	(0, 0, -1)	(-0.0880, 0, 0.2770)
4	(0, 1, 0)	(-0.0055, 0, 0.2770)
5	(0, 0, -1)	(-0.0880, 0, -0.1070)
6	(0, 1, 0)	(-0.0880, 0, -0.1070)
7	(0, 0, 1)	(0, 0, -0.1070)

Therefore, we can use the $v_i = -w_i \times q_i$ to get the S_i and B_i.
For the Space Frame Forward Kinematics, $T = e^{[S_1]\theta_1} e^{[S_2]\theta_2} ... e^{[S_7]\theta_7} M$.
For the Body Frame Forward Kinematics, $T = M e^{[B_1]\theta_1} e^{[B_2]\theta_2} ... e^{[B_7]\theta_7}$.
Here, $[S_i]$ and $[B_i]$ are 6×1 vectors and equals to $[w, v]^T$.
Thus, $e^{[S]\theta} = \begin{bmatrix} e^{[w]\theta} & * \\ 0 & 1 \end{bmatrix}$, where $* = (I\theta + (1 - \cos\theta)[w] + (\theta - \sin\theta)[w]^2)v$.
And $e^{[w]\theta} = I + \sin\theta[w] + (1 - \cos\theta)[w]^2$.
Thus, we can easily calculate forward kinematics with S_i, B_i and M.

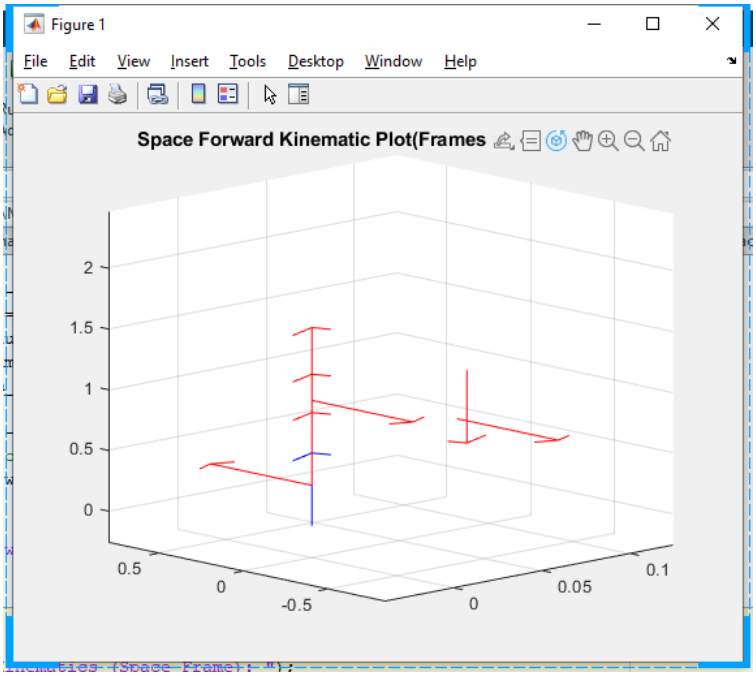


Figure 1
File Edit View Insert Tools Desktop Window Help
Space Forward Kinematic Plot(Frames
2
1.5
1
0.5
0
0.5
0
-0.5
0
0.05
0.1

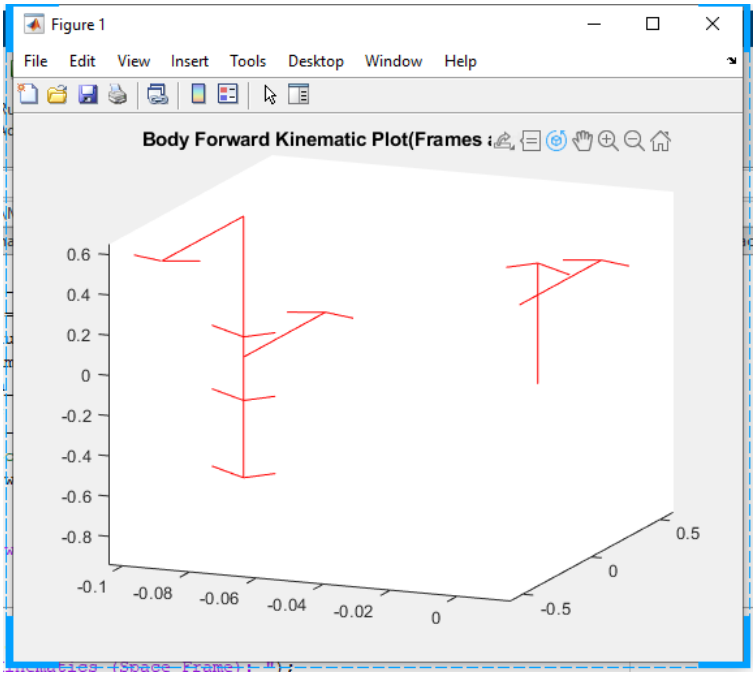


Figure 1
File Edit View Insert Tools Desktop Window Help
Body Forward Kinematic Plot(Frames
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.1
-0.08
-0.06
-0.04
-0.02
0
0.5
0
-0.5

Space Jacobian

From the notes, we know that $\dot{V_s} = S_1\dot{\theta_1} + Ad_{e^{[S_1]\theta_1}}(S_2)\dot{\theta_2} + ...$, and Space Jacobian matrix is given by $J_s = [S_1, Ad_{e^{[S_1]\theta_1}}(S_2), ...]$. Thus, we can first calculate the exponential term, where $T = e^{[S]\theta} = \begin{bmatrix} e^{[w]\theta} & * \\ 0 & 1 \end{bmatrix}$ (the same procedure for forward kinematics). Then, $Ad_T = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \in R^{6\times 6}$.
Therefore, we can get the Jacobian by given S_i and θ_i

Body Jacobian

Similarly, $\dot{V_b} = Ad_{e^{-[B_n]\theta_n}e^{-[B_{n-1}]\theta_{n-1}}...e^{-[B_2]\theta_2}}(B_1)\dot{\theta_1} + Ad_{e^{-[B_n]\theta_n}e^{-[B_{n-1}]\theta_{n-1}}...e^{-[B_3]\theta_3}}(B_2)\dot{\theta_2}... + B_n\dot{\theta_n}$, and the Body Jacobian matrix is given by
$J_b = [Ad_{e^{-[B_n]\theta_n}e^{-[B_{n-1}]\theta_{n-1}}...e^{-[B_2]\theta_2}}(B_1), Ad_{e^{-[B_n]\theta_n}e^{-[B_{n-1}]\theta_{n-1}}...e^{-[B_3]\theta_3}}(B_2), ..., B_n]$. Use the same formula to calculate $T = e^{[B]\theta}$ and Ad_T, we can get the Jacobian by given B_i and θ_i

(g)

Assuming J is invertible, the unit joint-velocity condition can be written:

$$1 = \dot{\theta}^T\dot{\theta} = (J^{-1}\dot{q})^T(J^{-1}\dot{q}) = \dot{q}^TJ^{-T}J^{-1}\dot{q} = \dot{q}^T(JJ^T)^{-1}\dot{q} = \dot{q}^TA^{-1}\dot{q}$$

Letting vi and λi be the the eigenvectors and eigenvalues (all positive) of $A = JJ^T > 0$, the directions of the principle axes of the ellipsoid are vi and the lengths of the principle semi-axes are $\sqrt{\lambda i}$.
The volume V of the ellipsoid is proportional to the product of the semi-axis lengths:

$$\sqrt{\lambda_1\lambda_2...\lambda_m} = \sqrt{det(A)} = \sqrt{det(JJ^T)}$$

For the geometric Jacobian $J \in R^{6\times n}$, we have: $J(\theta) = \begin{bmatrix} J_\omega(\theta) \\ J_v(\theta) \end{bmatrix}$ One measure is the ratio of the longest and shortest semi-axes of the manipulability ellipsoid:

$$\mu_1(A) = \frac{\sqrt{\lambda_{max}(A)}}{\sqrt{\lambda_{min}(A)}} = \sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$$

Isotropy means that it is equally easy to move in any direction and is generally desirable. As the robot approaches a singularity, however $\mu_1(A)$ goes to infinity.
Condition number measure $\mu_2(A)$ is just the sequence of the isotropy measure:

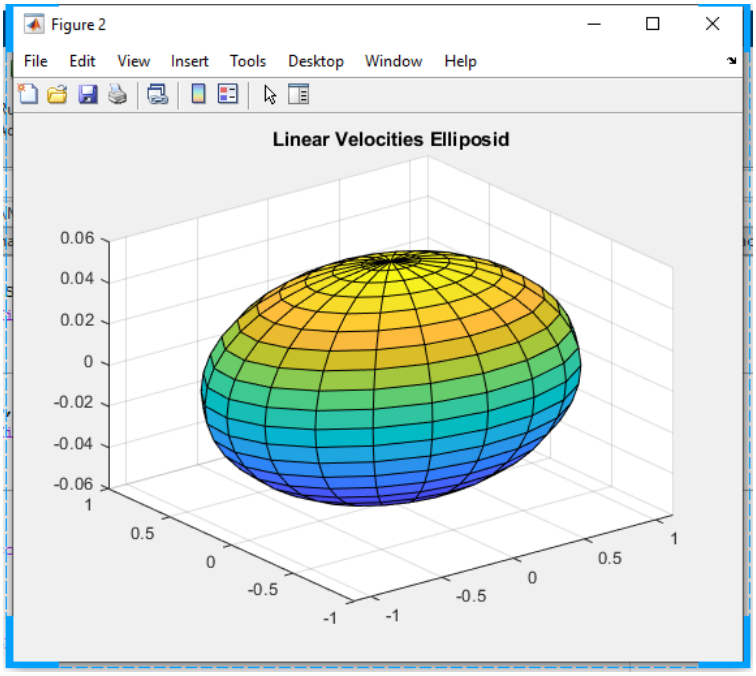
$$\mu_2(A) = \frac{\lambda_{max}}{\lambda_{min}} \geq 1$$

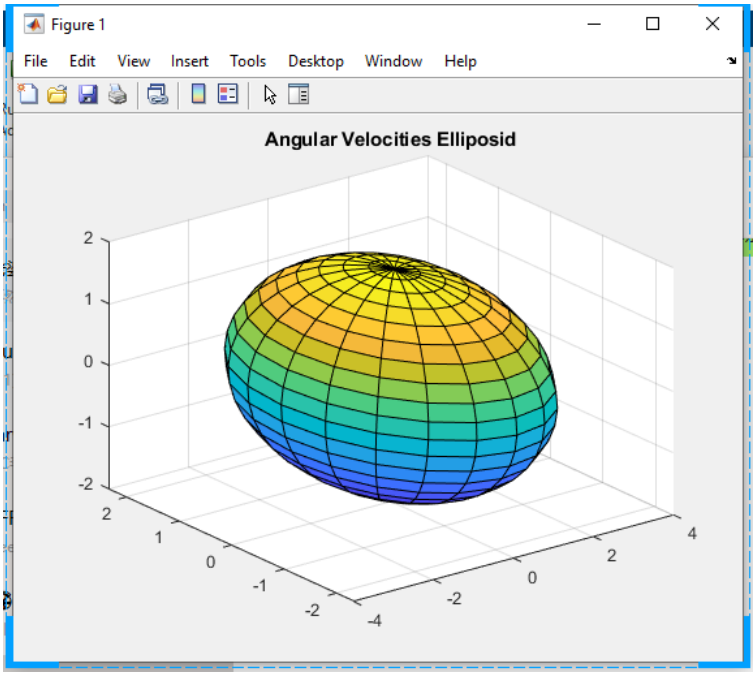
A measure is the volume of the manipulability ellipsoid:

$$\mu_3(A) = \sqrt{\lambda_1\lambda_2...} = \sqrt{det(A)}$$

In this problem, we need to split $6 \times n$ Jacobian matrix to $J_w, J_v \in 3 \times n$, then get two matrices

$A_w = J_w * J_w^T, A_v = J_v * J_v^T$, and do the same procedure as mentioned before.
One thing we need to notice here, the ellipsoid plot function can only draw a standard ellipsoid, we need to use *rotate* function to make the ellipsoid in the right configuration. Fortunately, we use *svd* function to get the singular value decomposition for the A matrix, and the eigenvectors we get formulate a rotation matrix. Use the function in $PA1$, we can easily get the rotation axis and rotate angle. Then we can use the *rotate* function to draw the ellipsoid.





(h)

Suppose we express the end-effector frame using a coordinate vector x governed by the forward kinematics x = $f(\theta)$, a nonlinear vector equation mapping the n joint coordinates to the m end-effector coordinates. Assume that $f : R^n \longrightarrow R^m$ is differentiable, and let x_d be the desired end-effector coordinates. Then $g(\theta)$ for the Newton-Raphson method is defined as , and the goal is to find joint coordinates θ_d such that

$$g(\theta) = x_d - f(\theta_d) = 0$$

Assuming that $J(\theta_0)$ is square (m = n) and invertible, we can solve for $\Delta\theta$ as

$$\Delta\theta = J^{-1}(\theta^0)(x_d - f(\theta^0))$$

$$\theta_1 = \theta_0 + \Delta\theta$$

In the case where J is full rank (rank m for $n > m$ or rank n for $n < m$), i.e., the robot is not at a singularity, the pseudoinverse can be calculated as

$$J^\dagger = J^T(JJ^T)^{-1}$$

$$J^\dagger = (J^TJ)^{-1}J^T$$

Replacing the Jacobian inverse with the pseudoinverse, Equation becomes

$$\Delta\theta = J^\dagger(\theta^0)(x_d - f(\theta^0))$$

To find this Vb, we first calculate the desired configuration in the body frame

$$T_{bd}(\theta^i) = T_{sb}^{-1}(\theta^i)T_{sd} = T_{bs}(\theta^i)T_{sd}$$

. Then Vb is determined using the matrix logarithm

$$[V_b] = logT_{bd}(\theta^i)$$

This leads to the following inverse kinematics algorithm, which is analogous to the above coordinate-vector algorithm:

(a) Initialization: Given T_{sd} and an initial guess $\theta^0 \longrightarrow R^n$, set $i = 0$

(b)Set

$$[V_b] = log(T_{sb}^{-1}(\theta^i)T_{sd})$$

While $||\omega_b|| > \epsilon_\omega$ or$||\omega_b|| > \epsilon_\omega$: set

$$\theta^{i+1} = \theta^i + J_b^\dagger(\theta^i)V_b$$

Increment i

(i)

A computationally simpler algorithm can be derived by finding a relationship between $\dot{q}$ and e that ensures error convergence to zero, without requiring linearization. As a consequence, the error dynamics is governed by a nonlinear differential equation. The Lyapunov direct method can be utilized to determine a dependence $q(\dot{e})$ that ensures asymptotic stability of the error system. Choose as Lyapunov function candidate the positive definite quadratic form

$$V(e) = \frac{1}{2}e^T Ke$$

where K is a symmetric positive definite matrix. This function is so that

$$V(e) > 0 \forall e \neq 0 V(0) = 0$$

Differentiating with respect to time and accounting gives

$$\dot{V} = e^T K\dot{x_d} - e^T K\dot{x_e}$$

$$\dot{V} = e^T K\dot{x_d} - e^T KJ_A(q)\dot{q}$$

$$\dot{q} = J_A^T(q)Ke$$

leads to

$$\dot{V} = e^T K\dot{x_d} - e^T KJ_A(q)J_A^T(q)Ke$$

(j)

This problem can be solved with the method of Lagrange multipliers. Consider the modified cost functional

$$g(\dot{q}, \lambda) = \frac{1}{2}\dot{q}^T W \dot{q} + \lambda^T (v_e - J\dot{q})$$

The requested solution has to satisfy the necessary conditions

$$(\frac{\partial g}{\partial \dot{q}})^T = 0$$

$$(\frac{\partial g}{\partial \dot{\lambda}})^T = 0$$

the sought optimal solution

$$\dot{q} = W^{-1} J^T (J W^{-1} J^T)^{-1} v_e$$

Proceeding in a way similar to the above yields

$$g(\dot{q}, \lambda) = \frac{1}{2}(\dot{q} - \dot{q_0})^T (\dot{q} - \dot{q_0}) + \lambda^T (v_e - J\dot{q})$$

then the solution is

$$\dot{q} = J^\dagger v_e + (I_n - J^\dagger J)\dot{q_0}$$

As can be easily recognized, the obtained solution is composed of two terms. The first is relative to minimum norm joint velocities. The second, termed homogeneous solution, attempts to satisfy the additional constraint to specify via $\dot{q_0}$ the matrix $I - J^\dagger J$ is one of those matrices P, which allows the projection of the vector $\dot{q_0}$ in the null space of J so as not to violate the constraint.It is possible to generate internal motions described by $(I_n - J^\dagger J)\dot{q_0}$that reconfigure the manipulator structure without changing the end-effector position and orientation

(k)

The derived equations are only valid when the jacobian has full rank. Hence, they become meaningless when the manipulator is at singular configuration. The damped least-squares (DLS) inverse:

$$J^\star = J^T (J J^T + k^2 I)^{-1}$$

where k is a damping factor. The problem can be reformed in terms of the minimization of the cost function

$$g^{''}(\dot{q}) = \frac{1}{2}(v_e - J\dot{q})^T (v_e - J\dot{q}) + \frac{1}{2}k^2 \dot{q}^T \dot{q}$$

The introduction of the first term allows a finite inversion error to be tolerated, with the advantage of norm-bounded velocities.

(m)

We use two Matlab method to do the graphical simulation.
Method 1:
This method need the newest version of Matlab, which is Matlab 2020a. In this version, $RoboticsSystemToolbox$ supply the model of this Panda robot, we can just use *loadrobot* function to use this robot. You can check the code in $Simulation.m$.
In our code, we split it into three parts.
First Part: Load the robot file, assign the initial configuration to it. This toolbox use a structure to contain the configuration information.
Second Part: Define the robot parameters, use the code we write to get the forward kinematics and Jacobians.
Third Part: Give the Desired Configuration and Initial guess, use the algorithm we write to get the trajectory, plot the robot.
If you cannot run this code, please see the video we took. $Simulation.mp4$
Method 2:
Please Run $PA2\_Plot.m$(Not $PA2\_Plot.fig$!). We use $PeterCroke$'s toolbox. And make a GUI in Matlab. You can check the code in $PA2_Plot.m$. If you haven't install the toolbox, please see the vedio we took. $Matlab\_Plot.mp4$

(l)

Discuss the results: The calculation results of the final configuration of the end effector by using the method (h)(i)(j)(k) are the same. But the iteration process is different. The convergence speed is different. Doe example, the method from (h) converges faster than the method from (i).
For all the work, we do it together.