Serial-Links Dynamics and Numerical Methods

Yu Lin

September 4, 2020

Introduction

Original problem: forward dynamic simulation of a fourbar.

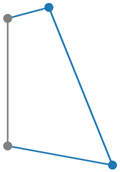

Figure 1: Fourbar

- Governing equations – Modeling.
- How to solve it.

System Governing Equations

Lagrangian equations with holonomic constraints.

- Generalized coordinates and velocity: $\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)$.
- Compute kinetic energy T, potential energy V input power Π and energy loss Δ.
- Constrains $C(\boldsymbol{q}) = \mathbf{0}$

System governing equations

$$L = T - V + \boldsymbol{\lambda} \cdot C(\boldsymbol{q}) \tag{1}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = \frac{\partial \Pi}{\partial \dot{q}_j} - \frac{\partial \Delta}{\partial \dot{q}_j} \tag{2}$$

where L is Lagrangian, $\boldsymbol{\lambda}$ is vector of Lagrangian multipliers.

System governing equations

- The choice of generalized coordinates $\boldsymbol{q}(t)$ is arbitrary, but will give different constraints and governing equations. In this case, the choice of $\boldsymbol{q}(t)$ are $\{x(t), y(t), \theta(t)\}$ of CG of each link, i.e.

$$\boldsymbol{q}(t) = \{x_i(t), y_i(t), \theta_i(t)\}_{i=1,2\ldots n_{links}}.$$

- For fourbar, degree of freedom is 1, $\boldsymbol{q}(t)$ is 9×1 vector, thus there are 8 constraints equations, and $\boldsymbol{\lambda}$ is 1×8 vector, $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1,2,\ldots 8}$.

System governing equations

from equation (2), the DAE is:

$$\boldsymbol{M}\dot{\boldsymbol{q}}(t) = f(\boldsymbol{q}(t), \boldsymbol{\lambda}, \boldsymbol{u}(t)) \tag{3}$$

where $\boldsymbol{M}$ is 9×9 diagonal mass matrix.
The DAE can be converted to ODE by eliminating $\boldsymbol{\lambda}$, equation (3) can be rewritten as:

$$\begin{bmatrix} \ddot{\boldsymbol{q}}(t) \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M} & A(\boldsymbol{q}(t))^T \\ A(\boldsymbol{q}(t)) & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{u}(t) \\ -\dot{A}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ f_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \tag{4}$$

where $A(\boldsymbol{q}(t))$ is the Jacobian of $C(\boldsymbol{q}(t))$.

In state space form

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ f_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \tag{5}$$

Approximate Model

The second method to model the system is to approximate the constraints as "springs". Approximate Lagrangian multipliers $\lambda_i = k_i C_i(\boldsymbol{q}(t))$. The violation of constraints is converted to restoring force.

The state space ODE can be written as:

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}\{\boldsymbol{u}(t) - A(\boldsymbol{q}(t))^T[\boldsymbol{K}C(\boldsymbol{q}(t))]\} \end{bmatrix} \tag{6}$$

where $\boldsymbol{K} = diag[k_i]$ is the stiffness matrix.

Null Space

The third method for modeling is using null space. Take the derivative of constraints,

$$\dot{C(\boldsymbol{q}(t))} = A(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) = 0 \tag{7}$$

The general solution is:

$$\dot{\boldsymbol{q}}(t) = S(\boldsymbol{q}(t))v \tag{8}$$

where $S(\boldsymbol{q}(t))$ is a vector in null space of $A(\boldsymbol{q}(t))$, i.e. $A(\boldsymbol{q}(t))S(\boldsymbol{q}(t)) = 0$, v is 1 dimension independent velocity.
After some manipulation, the state space ode for this method is:

Null Space ODE

$$\begin{bmatrix} \dot{\boldsymbol{q}}(t) \\ \dot{v} \end{bmatrix} = \begin{bmatrix} S(\boldsymbol{q}(t))v \\ (S^T M S)^{-1}(S^T \boldsymbol{u} - S^T M \dot{S} v) \end{bmatrix} \tag{9}$$

Solving ODEs

Solve initial conditions.

- Root finding. May have non-unique solutions

Which solver to use?

- RK23, RK45, etc.

Performance compare between solvers and models.

- Computation cost and accuracy.

Currently based on Scipy.

Explicit Model Result

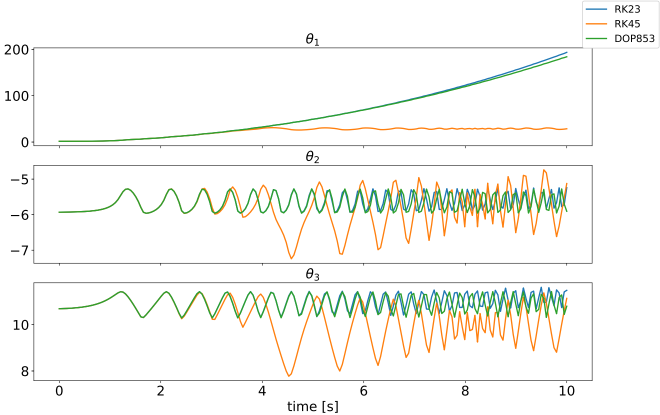

Figure 2: Results of Explicit Model with different solvers

Solvers	RK23	RK45	DOP853
Time(s)	1.26	0.60	1.15

Approximate Model Result

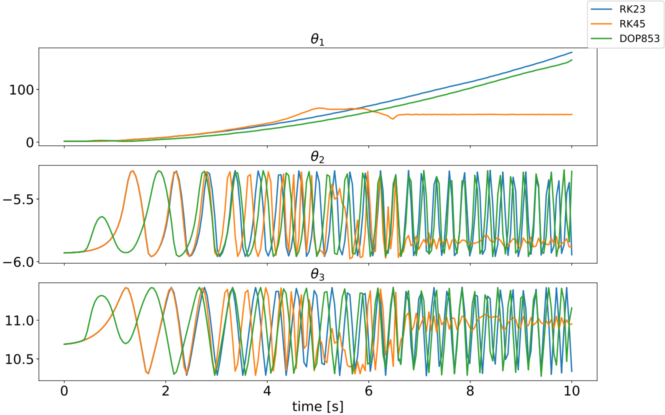

Figure 3: Results of Approximate Model with different solvers, $k_i = 10^6 N/m$

Solvers	RK23	RK45	DOP853
Time(s)	31.27	83.60	43.93

Null Space Model Result

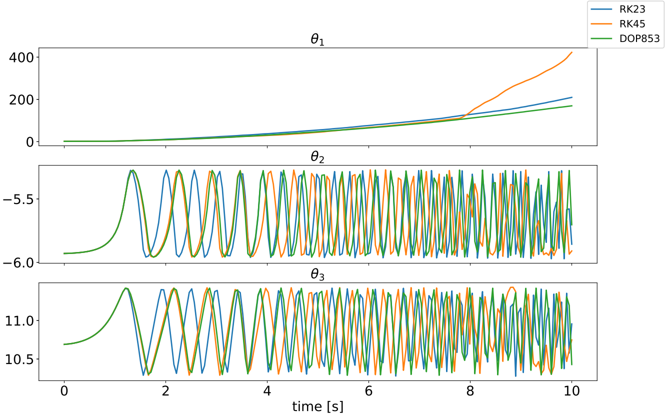

Figure 4: Results of Approximate Model with different solvers

Solvers	RK23	RK45	DOP853
Time(s)	4.85	9.13	5.40

Compare Between Models

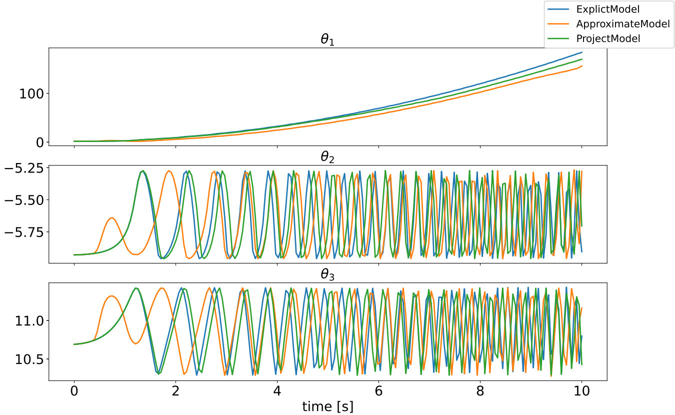

Figure 5: Compare model results with DOP853

Models	Explicit	Approximate	Null Space
Time(s)	1.24	43.29	5.53

Current and Future Work

Current work

- A python class for open and close serial-links modeling.
- Force and torque input on link CGs.
- Solution demos.

Future work

- Add Springs and dampers.
- More to study on numerical methods.
- Different platforms.
- Expand to 3-D.