Lesson 4-A

Procedures in Computational Kinematics

• For computational kinematics, we consider two solution methods:
  — Coordinate Partitioning method
  — Appended Constraint method
• These methods are introduced through a simple example

Solution Methods

• A set of algebraic equations, linear or nonlinear, can be arranged for numerical solutions in two different forms

  Example: Solve the set of equations for \( z = 3 \):
  
  \[
  x + 2y - z = 2 \\
  2x - y + z = 3
  \]

• Form I -- For \( z = 3 \) solve 2 equations in 2 unknowns
  
  \[
  x + 2y = 5 \\
  2x - y = 0
  \]
  
  — This is the Coordinate Partitioning method!
• Form II -- Solve 3 equations in 3 unknowns
  
  \[
  x + 2y - z = 2 \\
  2x - y + z = 3 \\
  z = 3
  \]
  
  — This is the Appended Constraint method!
• The two forms of presenting the kinematic equations, Coordinate Partitioning method and Appended Constraint method are further explained through a simple mechanism. The example is a four-bar system formulated for kinematic analysis in the classical form and not in the body-coordinate formulation. This is done since the classical form presents fewer kinematic constraints, compared to the body-coordinate method. So, having fewer equations allows us to concentrate on the two methods and their differences without getting lost in too many equations and too many unknowns.

2D Example: Four-bar Mechanism

• The crank of this four-bar mechanism, link 1, rotates according to the following driver equation:
  
  \[
  (d)\Phi = \theta_1 - \frac{\pi}{2} - 2\pi t = 0
  \]

• Determine the coordinates, velocities, and accelerations at \( t = 0 \)
• We first state all of the necessary equations, then we formulate and solve the problem with (a) the coordinate partitioning method and (b) the appended constraint method
  
  \( \ell_1 = 1.0, \ell_2 = 3.0, \ell_3 = 2.2, a = 2.0, b = 0.5 \)
• The necessary constraints for this mechanism are:
Position constraints
\[1.0 \cos \theta_1 + 3.0 \cos \theta_2 - 2.2 \cos \theta_3 - 2.0 = 0\]
\[1.0 \sin \theta_1 + 3.0 \sin \theta_2 - 2.2 \sin \theta_3 - 0.5 = 0\]

Velocity constraints
\[-\sin \theta_1 \dot{\theta}_1 - 3.0 \sin \theta_2 \dot{\theta}_2 + 2.2 \sin \theta_3 \dot{\theta}_3 = 0\]
\[\cos \theta_1 \dot{\theta}_1 + 3.0 \cos \theta_2 \dot{\theta}_2 - 2.2 \cos \theta_3 \dot{\theta}_3 = 0\]

Acceleration constraints
\[-\sin \theta_1 \ddot{\theta}_1 - 3.0 \sin \theta_2 \ddot{\theta}_2 + 2.2 \sin \theta_3 \ddot{\theta}_3 = 0\]
\[\cos \theta_1 \ddot{\theta}_1 + 3.0 \cos \theta_2 \ddot{\theta}_2 - 2.2 \cos \theta_3 \ddot{\theta}_3 = 0\]

Driver constraints (position, velocity, and acceleration)
\[\dot{\Phi}_1 = \theta_1 - \frac{\pi}{2} - 2\pi t = 0\]
\[\dot{\Phi}_2 = \dot{\theta}_1 - 2\pi = 0\]
\[\ddot{\Phi}_3 = \ddot{\theta}_1 = 0\]

Coordinate Partitioning
- At \(t = 0\), the driver constraints yield \(\theta_1 = \frac{\pi}{2}\), \(\dot{\theta}_1 = 2\pi\), \(\ddot{\theta}_1 = 0\)
- Position constraints for \(\theta_3 = \frac{\pi}{2}\)
  \[\Phi_1 = 3.0 \cos \theta_2 - 2.2 \cos \theta_3 + 1.0 \cos \frac{\pi}{2} - 2.0 = 0\]
  \[\Phi_2 = 3.0 \sin \theta_2 - 2.2 \sin \theta_3 + 1.0 \sin \frac{\pi}{2} - 0.5 = 0\]
  - Solve to get \(\theta_2 = 0.58\), \(\theta_3 = 1.34\)
- Velocity constraints for \(\dot{\theta}_3 = 2\pi\)
  \[
  \begin{bmatrix}
  -1.64 & 2.14 \\
  2.51 & -0.51
  \end{bmatrix}
  \begin{bmatrix}
  \dot{\theta}_2 \\
  \dot{\theta}_3
  \end{bmatrix}
  =
  \begin{bmatrix}
  6.28 \\
  0
  \end{bmatrix}
  \]
  - Solve to get \(\dot{\theta}_2 = 0.71\), \(\dot{\theta}_3 = 3.50\)
- Acceleration constraints for \(\ddot{\theta}_3 = 0\)
  \[
  \begin{bmatrix}
  -1.64 & 2.14 \\
  2.51 & -0.51
  \end{bmatrix}
  \begin{bmatrix}
  \ddot{\theta}_2 \\
  \ddot{\theta}_3
  \end{bmatrix}
  =
  \begin{bmatrix}
  -4.94 \\
  14.38
  \end{bmatrix}
  \]
  - Solve to get \(\ddot{\theta}_2 = 6.23\), \(\ddot{\theta}_3 = 2.46\)

Appended Constraint
- At \(t = 0\), the constraints are arranged as:
  - Position constraints
Numerical/Computational Methods

- Complete Kinematic Analysis
  - Solving acceleration constraints (linear equations)
    \[ \Phi_1 = \cos \theta_1 + 3 \cos \theta_2 - 2.2 \cos \theta_3 - 2 = 0 \]
    \[ \Phi_2 = \sin \theta_1 + 3 \sin \theta_2 - 2.2 \sin \theta_3 - 0.5 = 0 \]
    \[ \Phi_3 = \theta_1 - \frac{\pi}{2} = 0 \]
  - Solve to get \( \theta_1 = 1.57 \), \( \theta_2 = 0.58 \), \( \theta_3 = 1.34 \)

- Velocity constraints
  \[
  \begin{bmatrix}
  -\sin \theta_1 & -3 \sin \theta_2 & 2.2 \sin \theta_3 \\
  \cos \theta_1 & 3 \cos \theta_2 & -2.2 \cos \theta_3 \\
  1 & 0 & 0
  \end{bmatrix}
  \begin{bmatrix}
  \dot{\theta}_1 \\
  \dot{\theta}_2 \\
  \dot{\theta}_3
  \end{bmatrix}
  =
  \begin{bmatrix}
  0 \\
  0 \\
  2\pi
  \end{bmatrix}
  \]
  - Substituting known values for the coordinates
    \[
    \begin{bmatrix}
    -1 & -1.64 & 2.14 \\
    0 & 2.51 & -0.51 \\
    1 & 0 & 0
    \end{bmatrix}
    \begin{bmatrix}
    \dot{\theta}_1 \\
    \dot{\theta}_2 \\
    \dot{\theta}_3
    \end{bmatrix}
    =
    \begin{bmatrix}
    0 \\
    0 \\
    2\pi
    \end{bmatrix}
    \]
  - Solve to get \( \dot{\theta}_1 = 6.28 \), \( \dot{\theta}_2 = 0.71 \), \( \dot{\theta}_3 = 3.50 \)

- Acceleration constraints
  \[
  \begin{bmatrix}
  -\sin \theta_1 & -3.0 \sin \theta_2 & 2.2 \sin \theta_3 \\
  \cos \theta_1 & 3.0 \cos \theta_2 & -2.2 \cos \theta_3 \\
  1 & 0 & 0
  \end{bmatrix}
  \begin{bmatrix}
  \ddot{\theta}_1 \\
  \ddot{\theta}_2 \\
  \ddot{\theta}_3
  \end{bmatrix}
  =
  \begin{bmatrix}
  -\dot{\theta}_1 \cos \theta_1 & -3.0\dot{\theta}_2 \cos \theta_2 & 2.2\dot{\theta}_3 \cos \theta_3 \\
  -\dot{\theta}_1 \sin \theta_1 & -3.0\dot{\theta}_2 \sin \theta_2 & 2.2\dot{\theta}_3 \sin \theta_3 \\
  0 & 0 & 0
  \end{bmatrix}
  \begin{bmatrix}
  \dot{\theta}_1 \\
  \dot{\theta}_2 \\
  \dot{\theta}_3
  \end{bmatrix}
  \]
  - Substituting known values for the coordinates and velocities
    \[
    \begin{bmatrix}
    -1 & -1.64 & 2.14 \\
    0 & 2.51 & -0.51 \\
    1 & 0 & 0
    \end{bmatrix}
    \begin{bmatrix}
    \dot{\theta}_1 \\
    \dot{\theta}_2 \\
    \dot{\theta}_3
    \end{bmatrix}
    =
    \begin{bmatrix}
    -4.94 \\
    14.38 \\
    0
    \end{bmatrix}
    \]
  - Solve to get \( \ddot{\theta}_1 = 0 \), \( \ddot{\theta}_2 = 6.23 \), \( \ddot{\theta}_3 = 2.46 \)

Complete Kinematic Analysis

- The kinematic constraints can be solved using either method at different time, \( t \). The variable \( t \) is incremented from an initial time to a final time with increments \( \Delta t \). If the four-bar is Grashof, then the period of simulation could cover a complete revolution of the crank.

Numerical/Computational Methods

- Kinematic analysis requires solving linear and nonlinear algebraic equations. Velocity and acceleration constraints are linear equations but position constraints are nonlinear equations.
  - Two procedures can be implemented to solve the complete kinematic equations:
    - Solving algebraic equations
      1. Applying Newton-Raphson method to solve position constraints
      2. Solving velocity constraints (linear equations)
      3. Solving acceleration constraints (linear equations)
    - Integration
      1. Solving acceleration constraints (linear equations)
2. Finding coordinates and velocities by integration

Solving Linear Algebraic Equations

- Consider the following example:

\[
\begin{bmatrix}
3 & 1 & -1 \\
-1 & 2 & 1 \\
2 & -3 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
2 \\
6 \\
-1 \\
\end{bmatrix}
\]

This set of 3 equations in 3 unknowns can be solved in different ways. Here are some Matlab examples.

- Matrix inversion command

```matlab
% Solving a set of linear algebraic equations
% Method: Matrix inversion
A   = [ 3  1  -1 
         -1  2   1 
         2  -3  1 ];
rhs = [ 2  6  -1 ];
x   = inv(A)*rhs
```

If we execute this program we find the solution to be

\[
x = \\
1.0000 \\
2.0000 \\
3.0000 \\
\]

However, matrix inversion is the method to avoid if the number of equations is large due to computational inefficiency.

- Backslash command

```matlab
% Solving a set of linear algebraic equations
% Method: Backslash command
A   = ... (as in the first example)
rhs = ... (as in the first example)
x   = A\rhs
```

This is one of the recommended methods—it is much more efficient than matrix inversion. This method applies Gaussian elimination type procedures and pivoting to solve the equations.

Refer to Sections 3.3.1 and 3.3.2 of the Textbook for a brief review of these methods.

- L-U factorization
% Solving a set of linear algebraic equations
% Method: L-U factorization

A   = ... (as in the first example)
rhs = ... (as in the first example)

[L, U, P] = lu(A)
y = L\(P*rhs)
x = U\y

% L: A lower triangular matrix
% U: An upper triangular matrix
% P: A permutation matrix for row interchanges

Executing this program provide the following intermediate and final results:

L =
   1.0000         0         0
   0.6667    1.0000         0
  -0.3333   -0.6364    1.0000

U =
   3.0000    1.0000   -1.0000
   0   -3.6667    1.6667
   0         0    1.7273

P =
   1     0     0
   0     0     1
   0     1     0

y =
   2.0000
  -2.3333
   5.1818

x =
   1.0000
   2.0000
   3.0000

This is another recommended method—it is efficient and it provides useful information about
the coefficient matrix. We will apply this method in special circumstances.

Refer to Sections 3.3.3 and 3.3.4 of the Textbook for a brief review of this method.

Solving Nonlinear Algebraic equations

• Newton-Raphson method is applied to solve nonlinear algebraic equations.

Refer to Sections 3.4.1 and 3.4.2 of the Textbook for a brief review of this method.

• Consider the constraints from the fourbar example:

  \[ 1.0 \cos \theta_1 + 3.0 \cos \theta_2 - 2.2 \cos \theta_3 - 2.0 = 0 \]
  \[ 1.0 \sin \theta_1 + 3.0 \sin \theta_2 - 2.2 \sin \theta_3 - 0.5 = 0 \]
For $\theta_1 = \frac{\pi}{2}$ we solve these equations with the Coordinate partitioning formulation.

- Matlab program
  
  We assume initial estimates of $\theta_2 = 45^\circ$ and $\theta_3 = 80^\circ$ for the unknown angles.

```matlab
% Solving nonlinear algebraic equations
% Method: Newton-Raphson

global L1 L2 L3 a b
global theta1

% Constants
L1 = 1.0; L2 = 3.0; L3 = 2.2; a = 2.2; b = 0.5;

% Known coordinate(s)
theta1 = 90*pi/180;

% Initial estimates
theta2 = 45*pi/180;
theta3 = 80*pi/180;

% Move unknown coordinates to array x
x = [theta2; theta3];

for n = 1:20
    % Compute 2 x 1 array of functions
    f = constraints(x);
    % condition for termination:
    normf = norm(f);
    if ( normf <= 1e-7 ) break; end;
    % Construct 2 x 2 Jacobian
    D = jacobian(x);
    % Compute corrections
    delta_x = D\f;
    % Correct the coordinates
    x = x - delta_x;
end

% Report solution in degrees
theta2 = x(1)*180/pi
theta3 = x(2)*180/pi

function f = constraints(x)
% Evaluate the constraints

global L1 L2 L3 a b
global theta1

theta2 = x(1);
theta3 = x(2);

f1 = L1*cos(theta1) + L2*cos(theta2) - L3*cos(theta3) - a;

f2 = L1*sin(theta1) + L2*sin(theta2) - L3*sin(theta3) - b;

f = [f1 f2];
```

```matlab
function f = constraints(x)
% Evaluate the constraints

global L1 L2 L3 a b
global theta1

theta2 = x(1);
theta3 = x(2);

f1 = L1*cos(theta1) + L2*cos(theta2) - L3*cos(theta3) - a;

f2 = L1*sin(theta1) + L2*sin(theta2) - L3*sin(theta3) - b;

f = [f1 f2];
```
function D = jacobian(x)
% Evaluate the Jacobian
global  L1 L2 L3 a b
theta2 = x(1);
theta3 = x(2);
d11 = -L2*sin(theta2);
d12 = L3*sin(theta3);
d21 = L2*cos(theta2);
d22 = -L3*cos(theta3);
D = [d11 d12; d21 d22];

If we execute this program we find the solution to be
theta2 =
  34.0912

theta3 =
  82.5714

This Matlab program shows the functions and the corresponding Jacobian matrix in expanded form with all the details. Later on in other programs we will take advantage of Matlab’s capabilities and will write the programs in more compact and modular form.

A Complete Kinematic Analysis Program (Algebraic Equations)

• A Matlab program for complete kinematic analysis of the fourbar example with the Appended Constraint formulation is provided in the file “Kinem_AE”. Download the file and execute the main program “Kinem”. Review the structure and contents of the main program and the functions.
• You can add your own reporting statements.