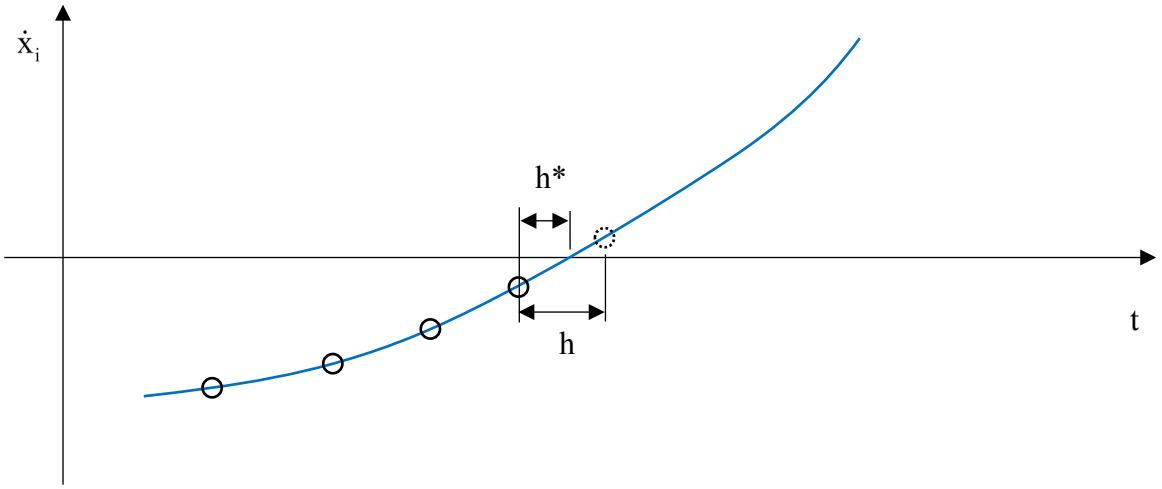
Integrating across Friction Reversal and Collision



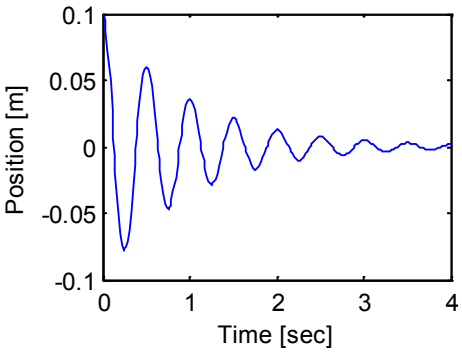
a) monitor $\dot{x}_i$ that causes Coulomb friction and check for zero crossing

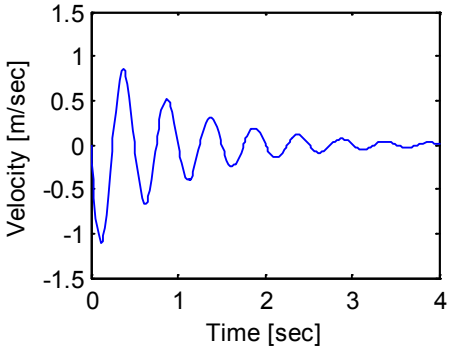
or

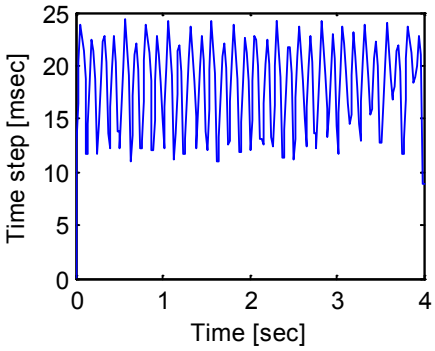
b) identify if collision occurs

interpolate to find h* when anomaly occurs

spring-mass-damper with viscous damping using ODE23

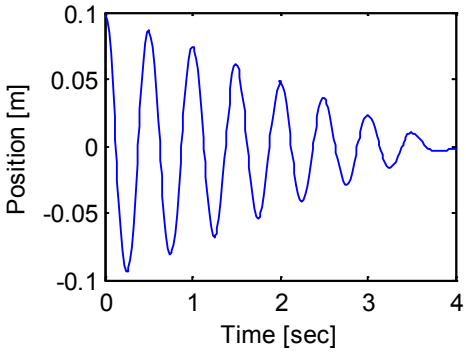


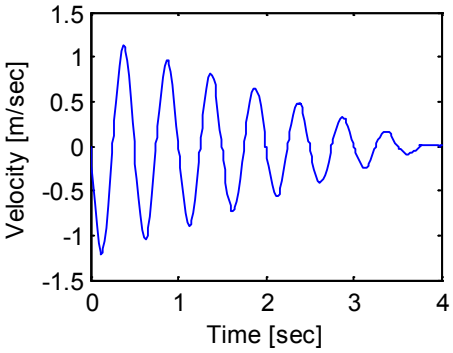


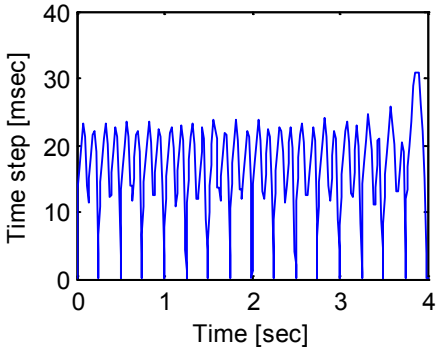


mean time step = 17.05 msec
CPU time = 0.0466 sec

spring-mass-damper with Coulomb friction using ODE23

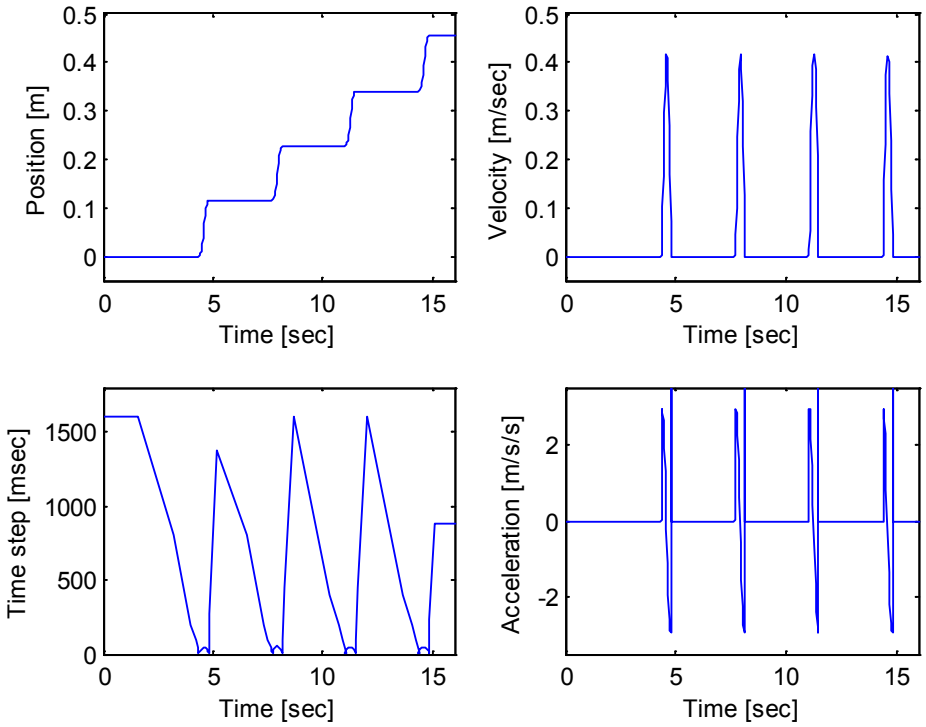






mean time step = 9.4787 msec
CPU time = 0.2339 sec

drag-sled with Coulomb friction using ODE23



Integrating Differential Algebraic Equations (DAE)

Forward Dynamics

know current state $\{y\}$ integrate $\{\dot{y}\} = f(\{y\}, t)$

DAE

initial values must be kinematically consistent $\{\Phi\} = \{0\}$ and $[\Phi_q]\{\dot{q}\} = \{\nu\}$

at each new time step $\{y\} = \begin{Bmatrix} \{q\} \\ \{\dot{q}\} \end{Bmatrix}$

$\{Q\}_{APPLIED}$ functions of $\{q\}$ $\{\dot{q}\}$ t

$[M]$ constant (typically)

$[\Phi_q]$ functions of $\{q\}$ t

$\{\gamma\}$ functions of $\{q\}$ $\{\dot{q}\}$ t

$$\begin{bmatrix} [M]_{nq \times nq} & [\Phi_q]^T_{nq \times nc} \\ [\Phi_q]_{nc \times nq} & [0]_{nc \times nc} \end{bmatrix} \begin{Bmatrix} \{\ddot{q}\}_{nq \times 1} \\ \{\lambda\}_{nc \times 1} \end{Bmatrix} = \begin{Bmatrix} \{Q\}_{APPLIED\ nq \times 1} \\ \{\gamma\}_{nc \times 1} \end{Bmatrix} \qquad [EOM] = \underset{(nc+nq) \times (nc+nq)}{\begin{bmatrix} [M] & [\Phi_q]^T \\ [\Phi_q] & [0] \end{bmatrix}}$$

solve for $\{\ddot{q}\}$

integrate $\{\dot{y}\} = \begin{Bmatrix} \{\dot{q}\} \\ \{\ddot{q}\} \end{Bmatrix}$

Kinematic consistency

Do new $\{q\}$ values satisfy $\{\Phi\} = \{0\}$?

Do new $\{\dot{q}\}$ values satisfy $[\Phi_q]\{\dot{q}\} = \{\nu\}$?

Methods for DAEs

1) Direct Integration

2) Coordinate Partitioning

3) Constraint Stabilization

Direct Integration

equations are numerically stiff

eigenvalues $\lambda_{MAX} >> \lambda_{MIN}$

for M = mobility, typically M relatively slow modes for rigid body motion

(nq-M) very fast modes for algebraic equations

requires VERY small h

relatively slow execution speed

more time steps allow more numerical error to accumulate - solution error

does not guarantee $\{\Phi\} = \{0\}$ position error = $\max(\text{abs}\{\Phi\})$

does not guarantee $\lfloor\Phi_q\rfloor\{\dot{q}\} = \{\nu\}$ velocity error = $\max\left(\text{abs}\left(\lfloor\Phi_q\rfloor\{\dot{q}\} - \{\nu\}\right)\right)$

Coordinate Partitioning

$$\underset{\text{nc x nq}}{\left[\Phi_q\right]} \quad \underset{\text{nq x 1}}{\{\dot{q}\}} = \{\nu\}$$

partition $\{\dot{q}\} = \begin{Bmatrix}\{\dot{u}\} \\ \{\dot{v}\}\end{Bmatrix}$ $\begin{matrix}\text{dependent} \\ \text{independent}\end{matrix}$ rearrange $\begin{bmatrix}\underset{\text{nc x nc}}{\left[\Phi_u\right]} & \underset{\text{nc x nv}}{\left[\Phi_v\right]}\end{bmatrix}\begin{Bmatrix}\underset{\text{nc x 1}}{\{\dot{u}\}} \\ \underset{\text{nv x 1}}{\{\dot{v}\}}\end{Bmatrix} = \{\nu\}$

apply Gaussian elimination with full pivoting to $\left[\Phi_q\right]$ to identify $\{u\}$ and $\{v\}$

(an example is provided at the end of these notes)

$$\left[\Phi_q\right]_{GEFP} = \begin{bmatrix}\begin{bmatrix}1 & * & * & & * \\ 0 & 1 & * & & * \\ 0 & 0 & 1 & & * \\ & & & \ddots & \\ 0 & 0 & 0 & & 1\end{bmatrix} & \begin{bmatrix}* & * & & * \\ * & * & & * \\ * & * & & * \\ & & & \\ * & * & & *\end{bmatrix}\end{bmatrix} = \begin{bmatrix}\underset{\text{nc x nc}}{\left[\Phi_u\right]} & \underset{\text{nc x nv}}{\left[\Phi_v\right]}\end{bmatrix}$$

$\left[\Phi_u\right]$ is not singular

$\{u\}$ corresponding to $\left[\Phi_u\right]$ can be considered dependent

$\{v\}$ corresponding to $\left[\Phi_v\right]$ can be considered independent

Simple Partitioning

compute all $\{\ddot{q}\}$ using $\begin{bmatrix}[M] & [\Phi_q]^T \\ [\Phi_q] & [0]\end{bmatrix}\begin{Bmatrix}\{\ddot{q}\} \\ \{\lambda\}\end{Bmatrix} = \begin{Bmatrix}\{Q\}_{APPLIED} \\ \{\hat{\gamma}\}\end{Bmatrix}$

HOWEVER only independent generalized accelerations $\{\dot{v}\}$ are integrated across time step h

provides new values for $\{v\}$ and $\{\dot{v}\}$

$\{v\}$ are used to determine $\{u\}$ by enforcing $\{\Phi\} = \{0\}$ with Newton-Raphson at each time step

$$\{u\}_{k+1} = \{u\}_k - \left[\Phi_u\right]_k^{-1}\{\Phi\}_k$$

$\{\dot{v}\}$ are used to determine $\{\dot{u}\}$ by enforcing $\{\dot{\Phi}\} = \{0\}$ with $\{\dot{u}\} = [\Phi_u]^{-1}(\{\nu\} - [\Phi_v]\{\dot{v}\})$

unfortunately coordinate partitioning does not always identify the same dependent and independent generalized coordinates at each time step which adds considerable bookkeeping

Full Partitioning

use partition $[\Phi_q]\{\dot{q}\} = [[\Phi_u][\Phi_v]]\begin{Bmatrix}\{\dot{u}\}\\ \{\dot{v}\}\end{Bmatrix} = \{\nu\}$

rearrange $[M] = \begin{bmatrix}[M]_{uu} & [M]_{uv}\\ [M]_{vu} & [M]_{vv}\end{bmatrix}$ $\{Q\}_{APPLIED} = \begin{Bmatrix}\{Q\}_{APPLIED\_u}\\ \{Q\}_{APPLIED\_v}\end{Bmatrix}$

$$[M]_{uu}\{\ddot{u}\} + [M]_{uv}\{\ddot{v}\} + [\Phi_u]^T\{\lambda\} = \{Q\}_{APPLIED\_u}$$
$$[M]_{vu}\{\ddot{u}\} + [M]_{vv}\{\ddot{v}\} + [\Phi_v]^T\{\lambda\} = \{Q\}_{APPLIED\_v}$$
$$[\Phi_u]\{\ddot{u}\} + [\Phi_v]\{\ddot{v}\} = \{\gamma\}$$

the third equation can be rearranged symbolically $\{\ddot{u}\} = [\Phi_u]^{-1}(\{\gamma\} - [\Phi_v]\{\ddot{v}\})$

and substituted into the first equation $\{\lambda\} = \left([\Phi_u]^T\right)^{-1}\left(\{Q\}_{APPLIED\_u} - [M]_{uu}\{\ddot{u}\} - [M]_{uv}\{\ddot{v}\}\right)$

finally providing differential equations in only independent generalized coordinates

$[\hat{M}]\{\ddot{v}\} = \{\hat{Q}\}$ $[\hat{M}]$ $\{\hat{Q}\}$ are NOT unit matrix/vector

$$[\hat{M}] = [M]_{vv} - [M]_{vu}[\Phi_u]^{-1}[\Phi_v] - [\Phi_v]^T\left([\Phi_u]^{-1}\right)^T\left([M]_{uv} - [M]_{uu}[\Phi_u]^{-1}[\Phi_v]\right)$$
$$\{\hat{Q}\} = \{Q\}_{APPLIED\_v} - [M]_{vu}[\Phi_u]^{-1}\{\gamma\} - [\Phi_v]^T\left([\Phi_u]^{-1}\right)^T\left(\{Q\}_{APPLIED\_u} - [M]_{uu}[\Phi_u]^{-1}\{\gamma\}\right)$$

only independent generalized accelerations $\{\ddot{v}\}$ are integrated across time step h

provides new values for $\{v\}$ and $\{\dot{v}\}$

$\{v\}$ are used to determine $\{u\}$ by enforcing $\{\Phi\} = \{0\}$ with Newton-Raphson at each time step

$\{\dot{u}\} = [\Phi_u]^{-1}(\{\nu\} - [\Phi_v]\{\dot{v}\})$

Constraint Stabilization

traditional acceleration constraint $\{\ddot{\Phi}\} = \{0\}$

Baumgarte's modified acceleration constraint $\{\ddot{\Phi}\} + 2\alpha\{\dot{\Phi}\} + \beta^2\{\Phi\} = \{0\}$ $\alpha > 0$ $\beta \neq 0$

$$\{\ddot{\Phi}\} = [\Phi_q]\{\ddot{q}\} - \{\gamma\} \qquad \{\dot{\Phi}\} = [\Phi_q]\{\dot{q}\} - \{\nu\}$$

$$[\Phi_q]\{\ddot{q}\} - \{\gamma\} + 2\alpha([\Phi_q]\{\dot{q}\} - \{\nu\}) + \beta^2\{\Phi\} = \{0\}$$

$$[\Phi_q]\{\ddot{q}\} = \{\gamma\} - 2\alpha([\Phi_q]\{\dot{q}\} - \{\nu\}) - \beta^2\{\Phi\}$$

$$[\Phi_q]\{\ddot{q}\} = \{\hat{\gamma}\} \qquad \{\hat{\gamma}\} = \{\gamma\} - 2\alpha([\Phi_q]\{\dot{q}\} - \{\nu\}) - \beta^2\{\Phi\}$$ $\{\hat{\gamma}\}$ is NOT unit vector

$$\begin{bmatrix} [M] & [\Phi_q]^T \\ [\Phi_q] & [0] \end{bmatrix} \begin{Bmatrix} \{\ddot{q}\} \\ \{\lambda\} \end{Bmatrix} = \begin{Bmatrix} \{Q\}_{APPLIED} \\ \{\hat{\gamma}\} \end{Bmatrix}$$

use direct integration

no general and uniformly valid method of selecting α and β have been found

can lead to divergence near singularities

Gaussian Elimination with Full Pivoting

$$\begin{bmatrix} 4 & 2 & -2 \\ 1 & 3 & 4 \\ 2 & 1 & 5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 6 \\ -1 \\ 2 \end{Bmatrix} \qquad \text{solution} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1.7667 \\ -0.7 \\ -0.1667 \end{Bmatrix}$$

first pivot position

$$\begin{bmatrix} 4 & 2 & -2 \\ 1 & 3 & 4 \\ 2 & 1 & 5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 6 \\ -1 \\ 2 \end{Bmatrix}$$

find largest absolute value below and to the right of pivot

$$\begin{bmatrix} 4 & 2 & -2 \\ 1 & 3 & 4 \\ 2 & 1 & 5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 6 \\ -1 \\ 2 \end{Bmatrix}$$

swap columns to get largest absolute value in pivot column (swap columns 1 and 3)
and swap corresponding rows in list of variables

$$\begin{bmatrix} -2 & 2 & 4 \\ 4 & 3 & 1 \\ 5 & 1 & 2 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 6 \\ -1 \\ 2 \end{Bmatrix}$$

swap rows to get largest absolute value in pivot row (swap rows 1 and 3)
and swap rows in right-hand side (RHS)

$$\begin{bmatrix} 5 & 1 & 2 \\ 4 & 3 & 1 \\ -2 & 2 & 4 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 2 \\ -1 \\ 6 \end{Bmatrix}$$

normalize pivot row including RHS

$$\begin{bmatrix} 1 & 0.2 & 0.4 \\ 4 & 3 & 1 \\ -2 & 2 & 4 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ -1 \\ 6 \end{Bmatrix}$$

multiply pivot row times 4 and subtract from row 2

$$\begin{bmatrix} 1 & 0.2 & 0.4 \\ 0 & 2.2 & -0.6 \\ -2 & 2 & 4 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ -2.6 \\ 6 \end{Bmatrix}$$

multiply pivot row times -2 and subtract from row 3

$$\begin{bmatrix} 1 & 0.2 & 0.4 \\ 0 & 2.2 & -0.6 \\ 0 & 2.4 & 4.8 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ -2.6 \\ 6.8 \end{Bmatrix}$$

second pivot position

$$\begin{bmatrix} 1 & 0.2 & 0.4 \\ 0 & 2.2 & -0.6 \\ 0 & 2.4 & 4.8 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ -2.6 \\ 6.8 \end{Bmatrix}$$

find largest absolute value below and to the right of pivot

$$\begin{bmatrix} 1 & 0.2 & 0.4 \\ 0 & 2.2 & -0.6 \\ 0 & 2.4 & 4.8 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_2 \\ x_1 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ -2.6 \\ 6.8 \end{Bmatrix}$$

swap columns to get largest absolute value in pivot column (swap columns 2 and 3)
and swap corresponding rows in list of variables

$$\begin{bmatrix} 1 & 0.4 & 0.2 \\ 0 & -0.6 & 2.2 \\ 0 & 4.8 & 2.4 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ -2.6 \\ 6.8 \end{Bmatrix}$$

swap rows to get largest absolute value in pivot row (swap rows 1 and 3)
and swap rows in RHS

$$\begin{bmatrix} 1 & 0.4 & 0.2 \\ 0 & 4.8 & 2.4 \\ 0 & -0.6 & 2.2 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ 6.8 \\ -2.6 \end{Bmatrix}$$

normalize pivot row including RHS

$$\begin{bmatrix} 1 & 0.4 & 0.2 \\ 0 & 1 & 0.5 \\ 0 & -0.6 & 2.2 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ 1.4167 \\ -2.6 \end{Bmatrix}$$

multiply pivot row times -0.6 and subtract from row 3

$$\begin{bmatrix} 1 & 0.4 & 0.2 \\ 0 & 1 & 0.5 \\ 0 & 0 & 2.5 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ 1.4167 \\ -1.75 \end{Bmatrix}$$

third pivot position

$$\begin{bmatrix} 1 & 0.4 & 0.2 \\ 0 & 1 & 0.5 \\ 0 & 0 & 2.5 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ 1.4167 \\ -1.75 \end{Bmatrix}$$

normalize pivot row including RHS

$$\begin{bmatrix} 1 & 0.4 & 0.2 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_3 \\ x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0.4 \\ 1.4167 \\ -0.7 \end{Bmatrix}$$

back-substitution

$x_2 = -0.7 \qquad x_1 = 1.7667 \qquad x_3 = -0.1667$

Results for full Jacobian (kinematically driven) for web cutter at phi2 = 85.5 deg

```
JAC =

    1.0000         0    1.9938         0         0         0         0         0         0
         0    1.0000   -0.1569         0         0         0         0         0         0
   -1.0000         0    1.9938    1.0000         0    8.6477         0         0         0
         0   -1.0000   -0.1569         0    1.0000   -6.4024         0         0         0
         0         0         0   -1.0000         0    4.3622    1.0000         0   -7.8080
         0         0         0         0   -1.0000    0.6374         0    1.0000   -7.0912
         0         0         0         0         0         0    1.0000         0   11.2196
         0         0         0         0         0         0         0    1.0000    0.0400
         0         0    1.0000         0         0         0         0         0         0


gamma_transpose =

   -6.1949  -78.7134  -13.4946  -88.5732   -1.7142   -2.2860    0.0138   -3.8620         0


a =

    1.0000         0         0    0.0891         0         0         0         0         0
         0    1.0000    0.2306         0   -0.1156         0         0         0    0.1156
         0         0    1.0000         0    0.5016         0         0         0         0
         0         0         0    1.0000    0.5949         0         0         0   -0.8871
         0         0         0         0    1.0000   -0.7133         0    0.7133   -0.5281
         0         0         0         0         0    1.0000   -0.8598   -0.1402   -0.3149
         0         0         0         0         0         0    1.0000   -0.0013   -0.0016
         0         0         0         0         0         0         0    1.0000    0.0623
         0         0         0         0         0         0         0         0    1.0000


b_transpose =

    0.0012   -1.5605   -3.1071    1.1661   67.3777  -11.1126   -3.9788  -87.9252 -147.8370


xlist_transpose =

     9     6     3     7     1     5     8     2     4


qdd_transpose =

   -6.1949  -78.7134         0 -147.8370  -72.4115   14.8187 -126.2931   -4.3123   11.2577
```

Results for reduced Jacobian (dynamically driven) for web cutter at phi2 = 85.5 deg

```
JAC8 =

    1.0000         0    1.9938         0         0         0         0         0         0
         0    1.0000   -0.1569         0         0         0         0         0         0
   -1.0000         0    1.9938    1.0000         0    8.6477         0         0         0
         0   -1.0000   -0.1569         0    1.0000   -6.4024         0         0         0
         0         0         0   -1.0000         0    4.3622    1.0000         0   -7.8080
         0         0         0         0   -1.0000    0.6374         0    1.0000   -7.0912
         0         0         0         0         0         0    1.0000         0   11.2196
         0         0         0         0         0         0         0    1.0000    0.0400


gamma8_transpose =

   -6.1949  -78.7134  -13.4946  -88.5732   -1.7142   -2.2860    0.0138   -3.8620


a =

    1.0000         0         0    0.0891         0         0         0         0         0
         0    1.0000    0.2306         0   -0.1156         0         0    0.1156         0
         0         0    1.0000         0    0.5016         0         0         0         0
         0         0         0    1.0000    0.5949         0         0   -0.8871         0
         0         0         0         0    1.0000   -0.7133         0   -0.5281    0.7133
         0         0         0         0         0    1.0000   -0.8598   -0.3149   -0.1402
         0         0         0         0         0         0    1.0000   -0.0016   -0.0013
         0         0         0         0         0         0         0    1.0000   16.0487


b_transpose =

  1.0e+003 *

    0.0000   -0.0016   -0.0031    0.0012    0.0674   -0.0111   -0.0040   -1.4111


xlist_transpose =

     9     6     3     7     1     5     8     4     2
```

```
% t_gefp.m - test Gaussian elimination with full pivoting
%   produces upper triangular form
% HJSIII, 12.03.12

% input

clear

% 3x3 square matrix in notes
a = [  4  2  -2 ;
       1  3   4 ;
       2  1   5 ];
b = [ 6 ; -1 ; 2 ];

% Jacobian and gamma for web cutter at phi2 = 85.5 deg when y3P - y4Q = 0
JAC = [ ...
    1.0000         0    1.9938         0         0         0         0         0         0 ;
         0    1.0000   -0.1569         0         0         0         0         0         0 ;
   -1.0000         0    1.9938    1.0000         0    8.6477         0         0         0 ;
         0   -1.0000   -0.1569         0    1.0000   -6.4024         0         0         0 ;
         0         0         0   -1.0000         0    4.3622    1.0000         0   -7.8080 ;
         0         0         0         0   -1.0000    0.6374         0    1.0000   -7.0912 ;
         0         0         0         0         0         0    1.0000         0   11.2196 ;
         0         0         0         0         0         0         0    1.0000    0.0400 ;
         0         0    1.0000         0         0         0         0         0         0 ];

gamma = [   -6.1949  -78.7134  -13.4946  -88.5732   -1.7142   -2.2860    0.0138   -3.8620  0 ]';


% call routine for full Jacobian (inverse dynamics)
qdd = inv(JAC) * gamma;
[ a, b, xlist ] = gefp( JAC, gamma );
JAC
gamma_transpose = gamma'
a
b_transpose = b'
xlist_transpose = xlist'
qdd_transpose = qdd'


% call routine for reduced Jacobian (forward dynamics)
JAC8 = JAC(1:8,:);
gamma8 = gamma(1:8);
[ a, b, xlist ] = gefp( JAC8, gamma8 );
JAC8
gamma8_transpose = gamma8'
a
b_transpose = b'
xlist_transpose = xlist'


% bottom of t_gefp
```

```
function [ a, b, xlist ] = gefp( amat, brhs )
% Gaussian elimination with full pivoting for linear model - amat * xvec = brhs
% produces upper triangular form - a * x = b
% HJSIII, 12.03.12
%
% USAGE
% [ a, b, xlist ] = gefp( amat, brhs )
%
% INPUT
% amat = nrow x ncol matrix of coefficients (may be non-square)
% brhs = nrow x 1 right-hand-side vector
%
% OUTPUT
% a = nrow x ncol upper triangular matrix
% b = nrow x 1 matching right-hand-side vector
% xlist = ncol x 1 vector of indices for shuffled entries after column pivoting
%
% nrow = ncol - linear solution but does not perform back-substitution
% nrow < ncol - triangulation of non-square matrix for DAE coordinate partitioning

% row/column order
a = amat;
b = brhs;
[ nrow, ncol ] = size( a );

% fill vector to sort variables
xlist = ( 1 : ncol )';

% loop through all rows - do not process last row
nrowm1 = nrow - 1;
for ipivot = 1 : nrowm1,

% find maximum absolute value below and right of pivot
% indices are relative to location within subset
  [ maxcol, indrow ] = max( abs( a( ipivot:nrow, ipivot:ncol ) ) );
  [ v, jcol ] = max( maxcol );
  irow = indrow( jcol );

% adjust indices to entire matrix
  irow = irow + ipivot - 1;
  jcol = jcol + ipivot - 1;

% swap columns - skip if already in pivot column
% must swap entire column
% also swap rows in variable list
  if jcol ~= ipivot,
    temp_col = a(:,ipivot);
    a(:,ipivot) = a(:,jcol);
    a(:,jcol) = temp_col;
    temp_x = xlist(ipivot);
    xlist(ipivot) = xlist(jcol);
    xlist(jcol) = temp_x;
  end

% swap rows - skip if already in pivot row
% right-hand-side handled by concatenation
  if irow ~= ipivot,
    temp_row = a( ipivot, ipivot:ncol );
    a( ipivot, ipivot:ncol ) = a( irow, ipivot:ncol );
    a( irow, ipivot:ncol ) = temp_row;
    temp_b = b(ipivot);
    b(ipivot) = b(irow);
    b(irow) = temp_b;
  end

% normalize pivot row
  d = a(ipivot,ipivot);
```

```
  a( ipivot, ipivot:ncol ) = a( ipivot, ipivot:ncol ) / d;
  b(ipivot) = b(ipivot) / d;

% factor rows below pivot
  ipp1 = ipivot + 1;
  for irow = ipp1 : nrow,
    d = a( irow, ipivot );
    a( irow, ipivot:ncol ) = a( irow, ipivot:ncol ) - d * a( ipivot, ipivot:ncol );
    b(irow) = b(irow) - d * b(ipivot);
  end

% bottom of loop for rows
end

% last row - only normalize
ipivot = nrow;
d = a(ipivot,ipivot);
a( ipivot, ipivot:ncol ) = a( ipivot, ipivot:ncol ) / d;
b(ipivot) = b(ipivot) / d;

return

% bottom of gefp
```