ME729 Advanced Robotics - Introduction to useful software tools

4/23/2018

Sangsin Park, Ph.D.

MotionGenesis

❑ Introduction

- Fast, compact symbolic math for forces and motion and code generation.
- See and manipulate symbolic equations for kinematics, statics, and dynamics.
- Exceptionally fast and compact multibody force/motion simulations.
- Expert tools for kinematics, forces, energy, mass and inertia properties.
- Made for engineers and scientists (physics and mathematics).
- Writes fast, compact MATLAB®, C, and Fortran codes for real-time application.
- Integrates with control-system and code generation for real-time tools.
- Highly efficient nonlinear and linearized equations for control system design.
- Symbolic optimization for high-speed, low-memory, in-the-loop hardware.
- Highly accurate symbolic and numeric computations and unit conversions.
- Vector (magnitude/direction) geometry and analysis (not just matrix algebra).
- Understands physical objects with built-in points, particles, rigid bodies, etc.
- Undergraduate and professional dynamics, simulation, and control.

MotionGenesis

- Introduction
  - Download a demo program & a tutorial.
    - http://www.motiongenesis.com/MGWebSite/MGFAQ/MGInstallAndLicense.html
    - http://www.motiongenesis.com/MGWebSite/MGGetStarted/MotionGenesisTutorial.pdf

F=ma www.MotionGenesis.com

MotionGenesis™ Kane: Version Comparison Chart

MotionGenesis™ Kane is a MATLAB(R) Connections Partner

MathWorks Partner

Version	Memory Limit	ODEs, linear & nonlinear equations ... Statics/dynamics via Newton/Euler/Kane ...	Code Generation MATLAB(R), C, Fortran, ...	Maintenance/support	Auto/Efficient variables (AutoZee)	Redistributable code	Automatic Efficient Linearization (AutoLinearize)
Student	1.5 MB	✓	MATLAB(R)				
Get Started	1.8 MB	✓	MATLAB(R)	Simple			
Vanilla*	2 MB	✓	MATLAB(R)	Basic Support	✓		
Research*	20 MB	✓	ALL	✓	✓	✓	
Professional*	None	✓	ALL	✓	✓	✓+	✓

1. Student software is for instructors and students at accredited institutions.
2. Get Started is for home/educational use, e.g., for individuals at organizations with a research/professional version.
3. Vanilla is for fiscally constrained research and generates non-redistributable MATLAB(R) code.
4. *Efficient Variables (Zees) provide significant speed and memory enhancements to both MotionGenesis and C, MATLAB(R), Fortran, or other codes produced by MotionGenesis.
5. Professional handles larger systems, makes use of all the computer's available RAM memory (e.g., multi-GigaBytes).
6. Redistributable code applies to C, MATLAB(R), Fortran codes generated by MotionGenesis (not MotionGenesis itself). Professional allows permanently re-distributable code (independent of license or maintenance expiration date).
7. The level of technical support depends on the purchased software version and maintenance/support policies.
8. Upgrades for the current 5.x release are available for all users (just download the latest version from our website). Licensees with paid-up maintenance/support are also provided major version upgrades (e.g., 6.x).
9. All terms and conditions subject to change without notice.

Click to compare: Professional / Research / Vanilla / Student

MotionGenesis Professional Purchase/Maintenance/Support policy

License	5 years	License	3 years	License	2 years	License	1 year	Extend license	
Maintenance	2 years	Maintenance	2 years	Maintenance	1 year	Maintenance	1/2 year	Maintenance	1 year
Price	$3725	Price	$2425	Price	$1745	Price	$995	Price	$885
Purchase		Purchase		Purchase		Purchase		Purchase	

MotionGenesis Research Purchase/Maintenance policy

License	2 years	License	1 year	Extend license	
Maintenance	1 year	Maintenance	1/2 year	Maintenance	1 year
Price	$995	Price	$645	Price	$425
Purchase		Purchase		Purchase	

MotionGenesis Vanilla

License	2 years	License	1 year
Support	Basic	Support	Basic
Price	$385	Price	$245
Purchase		Purchase	

MotionGenesis GetStarted

License	1 year
Maintenance	None
Price	$99
Purchase	

MotionGenesis Student (verified)

License	2 years	License	1 year	License	1/2 year
Maintenance	None	Maintenance	None	Maintenance	None
Price	$70	Price	$57	Price	$31
Purchase		Purchase		Purchase	

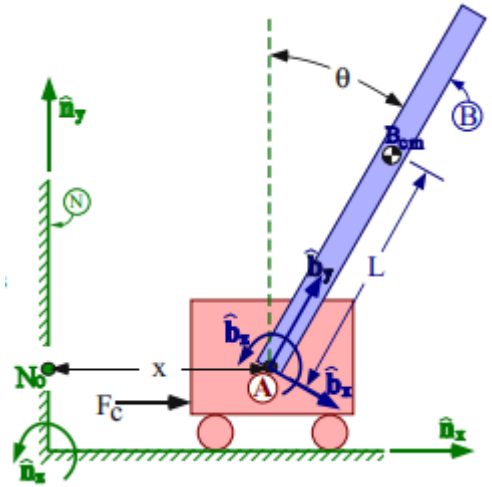
Late fee $9 Purchase

Transfer $15 Purchase

To avoid late fee, download, install, and request software license within 3 weeks of your purchase.
Transfer fee is to transfer remaining term of a student license to a computer owned by the same student (verified).
Fees also apply to software purchased with a new MotionGenesis-related textbook.

MotionGenesis

- Sample problems - an inverted pendulum on a cart
  - An inverted pendulum (a rigid body B) attached by an frictionless pin (revolute) joint to a cart A (modeled as a particle). The cart A slides on a horizontal frictionless track. The track is fixed in a Newtonian frame N.



Quantity	Symbol	Value
Mass of A	m^A	10.0 kg
Mass of B	m^B	1.0 kg
Distance between A and B_{cm} (B's center of mass)	L	0.5 m
B's moment of inertia about B_{cm} for $\hat{b}_z$	I_{zz}	0.08333 kg∗m^2
Earth's gravitational constant	g	9.8 m/s^2
$\hat{n}_x$ measure of feedback-control force applied to A	F_c	**Specified**
$\hat{n}_x$ measure of A's position vector from N_o (a point fixed in N)	x	Variable
Angle from $\hat{n}_y$ to $\hat{b}_y$ with $-\hat{n}_z$ sense	θ	Variable

MotionGenesis

- Sample problems - an inverted pendulum on a cart
  - Derive equations of motion.

```
% MotionGenesis file:  InvertedPendulumOnCartWithControl.txt
% Copyright (c) 2009 Motion Genesis LLC.  All rights reserved.
%---------------------------------------------------------------------
NewtonianFrame  N
Particle        A            % Cart
RigidBody       B            % Inverted pendulum
%---------------------------------------------------------------------
Variable   x''               % Distance between No to A
Variable   theta''           % Angle from local vertical to B's long axis
Constant   g+ = 9.81 m/s^2   % Gravitational constant
Constant   L+ = 0.5  m       % Distance between A and Bcm
Specified  Fc
A.SetMass( mA = 10 kg )
B.SetMassInertia( mB = 1 kg,  Izz = 1/12*mB*(2*L)^2,  0,  Izz )
%---------------------------------------------------------------------
%       Rotational and translational kinematics
B.RotateNegativeZ( N, theta )
A.Translate( No, x*Nx> )
Bcm.Translate( A, L*By> )
%---------------------------------------------------------------------
%       Relevant contact and distance forces
System.AddForceGravity( -g*Ny> )
A.AddForce( Fc*Nx> )
%---------------------------------------------------------------------
%       Form Kane's equations of motion
SetGeneralizedSpeed( x', theta' )
Zero = System.GetDynamicsKane()
```

```
   (37) %---------------------------------------------------------------------
   (38) % Form Kane's equations of motion
   (39) SetGeneralizedSpeed( x', theta' )
   (40) Zero = System.GetDynamicsKane()
-> (41) Zero[1] = (mA+mB)*x'' + mB*L*cos(theta)*theta'' - Fc - mB*L*sin(theta)*theta'^2
-> (42) Zero[2] = mB*L*cos(theta)*x'' + (mB*L^2+Izz)*theta'' - mB*g*L*sin(theta)
```

MotionGenesis

- Sample problems - an inverted pendulum on a cart
  - Control the system.

```
%*********************************************************************
%       CONTROL SYSTEM / STABILITY ANALYSIS
%*********************************************************************
%       Linearization: Perturbation variables
Variable   dx''            % Perturbations of x, x', x''
Variable   dtheta''        % Perturbations of theta, theta', theta''
Specified  dFc             % Perturbation of Fc.
Variable   FcNominal       % Nominal solution for Fc.
SetImaginaryNumber( i )
%---------------------------------------------------------------------
%       Check conditions for solution: x = x' = x'' = 0  and  theta = theta' = theta'' = 0.
Check = Evaluate( Zero,  x=0,  x'=0,  x''=0,  theta=0,  theta'=0,  theta''=0, Fc=FcNominal )
%---------------------------------------------------------------------
%       Linearize equations of motion about nominal solution
Perturb = Linearize1( Zero, x = 0:dx, x' = 0:dx', x'' = 0:dx'',        &
          theta = 0:dtheta,  theta' = 0:dtheta',  theta'' = 0:dtheta'', Fc=0:dFc )
Solve( Perturb,  dx'', dtheta'' )
%---------------------------------------------------------------------
%       Form matrix of peturbations and its time-derivative
Xm = [ dx ;  dtheta ;  dx' ;  dtheta' ]    % Matrix of perturbations
Xp = dt( Xm )
%---------------------------------------------------------------------
%       Form/simplify matrices A and B  so  Xm' = A * Xm  + B * Fc.
A = Expand( GetCoefficientMatrix( Xp,  Xm ) )
B = Expand( GetCoefficientMatrix( Xp,  dFc ) )
%---------------------------------------------------------------------
%       Stability when uncontrolled (Fc = 0) and with control.
RootsNoControl = GetEigen( EvaluateAtInput( A ) )
```

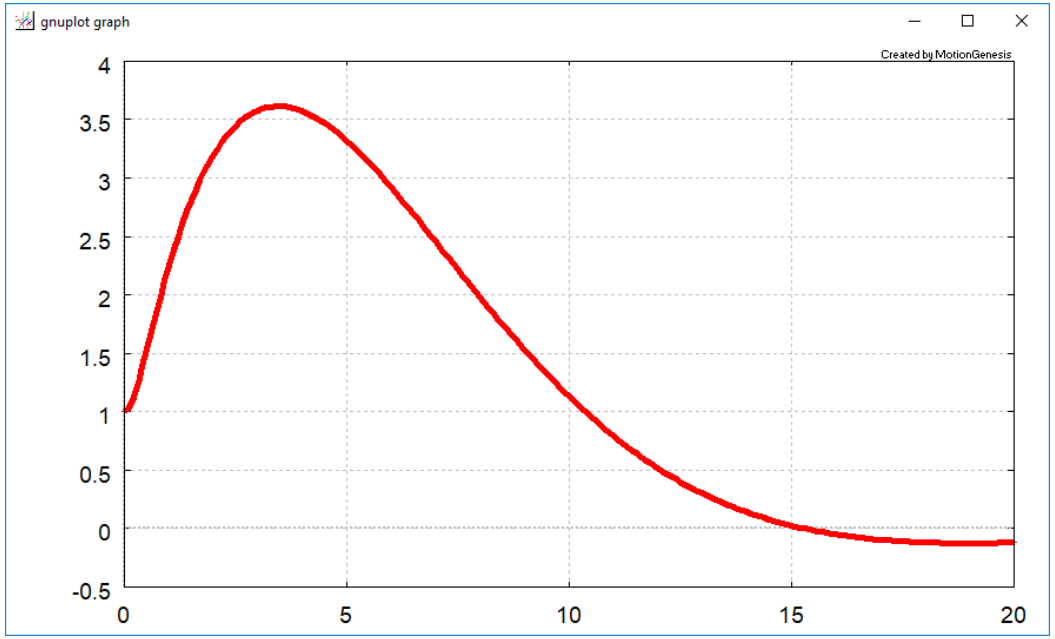
```
%---------------------------------------------------------------------
%       Stability with feedback control (k1, k2, k3, k4 are gains).
Constant  k1 = 1.0 N/m,  k2 = 244 N/rad,  k3 = 5.4 N*s/m,  k4 = 59  N*s/rad
K = [k1, k2, k3, k4]
RootsControlled = GetEigen(  EvaluateAtInput( A + B*K )  )
%---------------------------------------------------------------------
Fc = k1*x + k2*theta + k3*x' + k4*theta'
dFc = k1*dx + k2*dtheta + k3*dx' + k4*dtheta'
%---------------------------------------------------------------------
%       Integration parameters and initial values.
Input  tFinal = 20,  tStep = 0.1,  absError = 1.0E-08
Input   x = 1 m,   theta = 20 deg,   x' = 0 m/s,   theta' = 0 rad/s
Input  dx = 1 m,  dtheta = 20 deg,  dx' = 0 m/s,  dtheta' = 0 rad/s
%---------------------------------------------------------------------
%       Quantities to be output by ODE command.
Output  t sec,  x m,  dx m,  theta deg,  dtheta deg,  Fc Newtons
ODE( Zero,  x'',  theta''  ) InvertedPendulumOnCartWithControl
```

After run, type this.

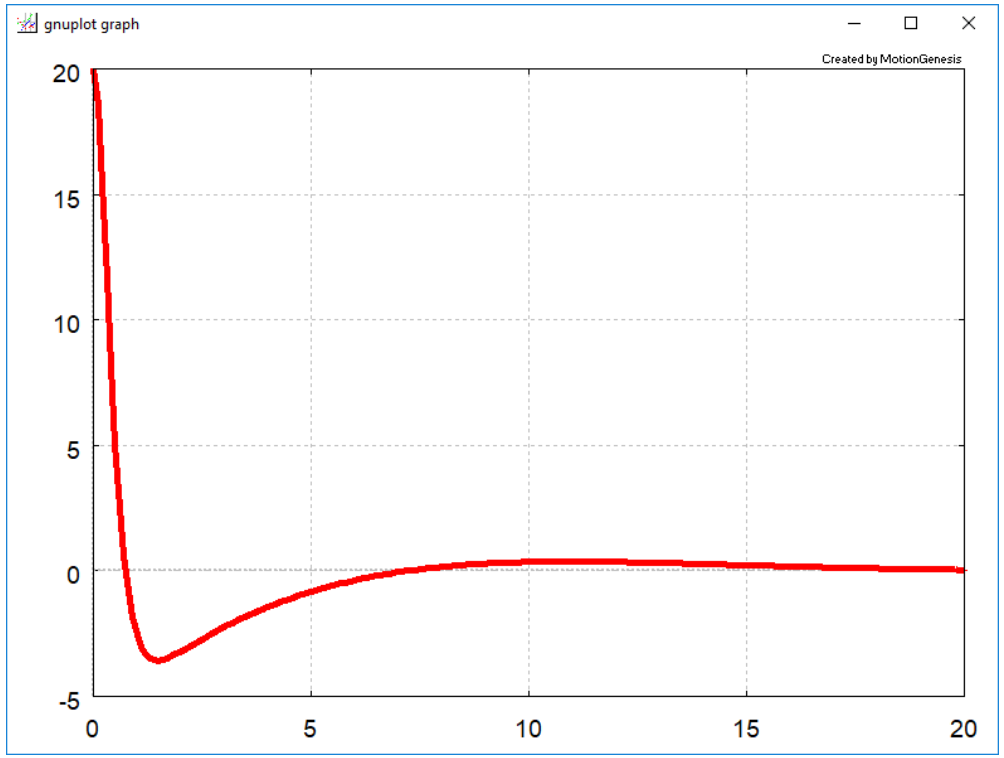
```
Plot  InvertedPendulumOnCartWithControl.1
```

MotionGenesis

- Sample problems - an inverted pendulum on a cart
  - Plot data.



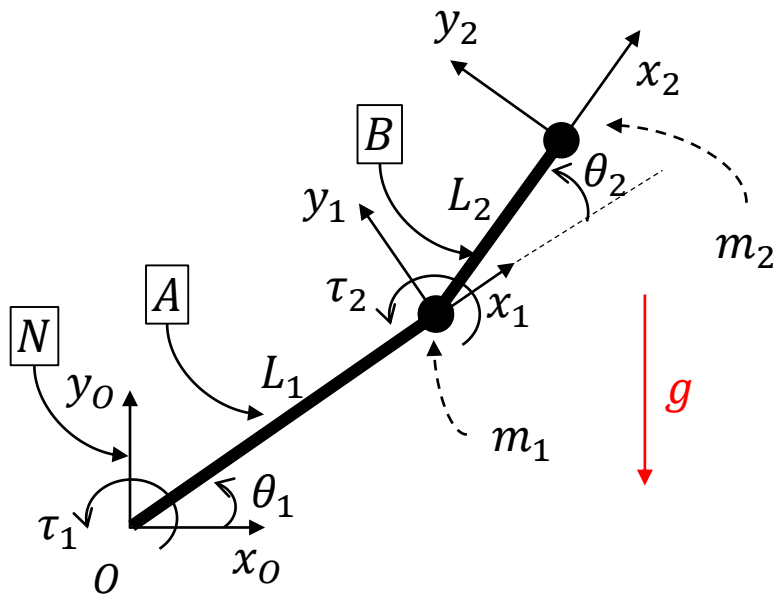
[Displacement of a cart A]



[Angle of an inverted pendulum]

MotionGenesis

- Sample problems – the two-link planar manipulator
  - Set the reference frame as NewtonianFrame N.
  - Set the link 1 as RigidBody A.
  - Set the link 2 as RigidBody B.
  - Let's find **kinematic equations** and **equations of motion**.



MotionGenesis

- Sample problems – the two-link planar manipulator
  - For kinematic equations.

```
%-----------------------------------------
NewtonianFrame N
RigidBody A % Link 1
RigidBody B % Link 2

%-----------------------------------------
Variable th1'', th2'' % Joint angles
Specified T1, T2      % Joint torques
Constant g, L1, L2
A.SetMassInertia(m1, 0, 0, 0)
B.SetMassInertia(m2, 0, 0, 0)

%-----------------------------------------
% Rotational and translational kinematics
A.RotateZ(N, th1)
Acm.Translate(No, L1*Ax>)
B.RotateZ(A, th2)
Bcm.Translate(Acm, L2*Bx>)

%-----------------------------------------
% Kinematics
P[1] = Dot(Bcm.GetPosition(No), Nx>)
P[2] = Dot(Bcm.GetPosition(No), Ny>)
Jacob = D(P, [th1, th2])
J_inv = GetInverse(Jacob)
```

```
   (31) %-----------------------------------------------------------------
   (32) % Kinematics
   (33) P[1] = Dot(Bcm.GetPosition(No), Nx>)
-> (34) P[1] = L1*cos(th1) + L2*cos(th1+th2)

   (35) P[2] = Dot(Bcm.GetPosition(No), Ny>)
-> (36) P[2] = L1*sin(th1) + L2*sin(th1+th2)

   (37) Jacob = D(P, [th1, th2])
-> (38) Jacob[1,1] = -L1*sin(th1) - L2*sin(th1+th2)
-> (39) Jacob[1,2] = -L2*sin(th1+th2)
-> (40) Jacob[2,1] = L1*cos(th1) + L2*cos(th1+th2)
-> (41) Jacob[2,2] = L2*cos(th1+th2)

   (42) J_inv = GetInverse(Jacob)
-> (43) J_inv[1,1] = cos(th1+th2)/(L1*sin(th2))
-> (44) J_inv[1,2] = sin(th1+th2)/(L1*sin(th2))
-> (45) J_inv[2,1] = -(L1*cos(th1)+L2*cos(th1+th2))/(L1*L2*sin(th2))
-> (46) J_inv[2,2] = -(L1*sin(th1)+L2*sin(th1+th2))/(L1*L2*sin(th2))
```

MotionGenesis

- Sample problems – the two-link planar manipulator
  - For equations of motion.

```
%----------------------------------------
% Add forces and torques
System.AddForceGravity(-g*Ny>)
A.AddTorque((T1 - T2)*Az>)
B.AddTorque(T2*Bz>)

%----------------------------------------
% Form Kane's equations of motion
SetGeneralizedSpeed(th1', th2')
Zero = System.GetDynamicsKane()
Solve(Zero, T1, T2)
```

```
   (62) Solve(Zero, T1, T2)
-> (63) T1 = m1*g*L1*cos(th1) + m2*g*(L1*cos(th1)+L2*cos(th1+th2)) + m2*L1*L2*sin(th2)*(th1'^2-(th1'+th2')^2)
 + m2*L2*(L2+L1*cos(th2))*th2'' + (m1*L1^2+m2*(L1^2+L2^2+2*L1*L2*cos(th2)))*th1''

-> (64) T2 = m2*L2*(g*cos(th1+th2)+L1*sin(th2)*th1'^2+L2*th2''+(L2+L1*cos(th2))*th1'')
```

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}(m_1+m_2)L_1^2 + m_2L_2^2 + 2m_2L_1L_2c_2 & m_2L_2^2 + m_2L_1L_2c_2\\ m_2L_2^2 + m_2L_1L_2c_2 & m_2L_2^2\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix}-m_2L_1L_2\left(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right)s_2\\ m_2L_1L_2\dot{\theta}_1^2 s_2\end{bmatrix}$$
$$+ \begin{bmatrix}(m_1+m_2)gL_1c_1 + m_2gL_2c_{12}\\ m_2gL_2c_{12}\end{bmatrix}$$