TSi ProPac

a Mathematica package for dynamics and control

Quick Start

by Techno–Sciences, Inc.

TSi ProPac

Software and documentation by

Techno-Sciences, Inc.

Techno–Sciences Incorporated

10001 Derekwood Lane, Suite 204

Lanham, MD 20706

(301) 577–6000

info@technosci.com

www.technosci.com

Table of Contents

1. Introduction

TSi ProPac is a *Mathematica* package that integrates and expands the capabilities of Techno-Sciences' *TSi Dynamics* and *TSi Controls*. It provides a comprehensive set of symbolic computing tools for modeling multibody mechanical systems as well as for linear and nonlinear control system design and analysis. New features include:

- the capability to model systems with nondifferentiable nonlinearities, such as static friction and backlash,
- the capability to compute equilibrium surfaces, to generate parameter-dependent linear families, and to construct parameter-dependent zero dynamics,
- tools for constructing nonlinear observers.
- tools for variable structure control system design.

In addition, ***ProPac*** includes a number of algorithm modifications designed to improve performance, particularly for large problems. Users of earlier versions should be aware that many functions can now be called with a simplified syntax that makes them easier to use.

TSi ProPac includes a revised and expanded set of tutorial and application notebooks. These include Dynamics and Controls which introduce the basic modeling and control tools available in ***ProPac***. For more information, notebooks and other documents visit our web site: www.technosci.com.

Using ***ProPac*** requires version 2, 3 or 4 of *Mathematica*. That is all that is required to develop the equations of motion, for conducting numerical simulations within *Mathematica*, and building the C source code required for simulations in SIMULINK. Use of the latter requires MATLAB/SIMULINK and a C compiler as recommended by the MathWorks for compiling MEX–files on the user's platform.

Installation

To install ***ProPac***, follow the two step procedure:

Step 1:

Put the entire ProPac directory in the *Mathematica's* AddOns/Applications directory. For the PC the full path is

C:\Program Files\Wolfram Research\Mathematica\4.0\AddOns\Applications

Step 2:

Start Mathematica 4.0 and rebuild the Help index: From the main menu choose: **Help ⇒ Rebuild Help Index …**

Once this is done, on line help is available. In the Help Browser select *Add-ons* and then *TSi ProPac*.

The Mex folder contains 3 C-source files that are need to be included when compiling MATLAB/SIMULINK MEX files. These may stored in any convenient location, but must be available at the time of compilation.

2. Package Content

Overview

TSi ***ProPac*** consists of seven packages: *Dynamics*, *ControlL*, *ControlN*, *GeoTools*, *MEXTools*, *NDTools*, and *VSCTools*. Once *ProPac* is loaded all of the functions in these packages are available for use and the appropriate packages will be automatically loaded as required. In general, a user does not have to be concerned about loading any particular package. To load ProPac in ***Mathematica 3.0 or 4.0*** simply enter <<ProPac`, and in ***Mathematica 2.2*** enter <<ProPac`Master`.

Dynamics contains the model building functions and ControlL and ControlN the linear and nonlinear control analysis functions, respectively. GeoTools includes basic functions used in differential geometry calculations. NDTools contains supporting functions for working with nondifferentiable nonlinearities and VSCTools contains functions for variable structure control. MEXTools includes functions for creating C-code files for both models and controllers that compile as S-functions for use with MATLAB/SIMULINK.

The following paragraphs contain a brief summary of the available functions. More details and numerous examples can be found in the help browser and in the notebooks. The notebooks **Dynamics.nb** and **Controls.nb** are tutorials that illustrate the basic data structures and tools.

Modeling

ProPac contains a comprehensive set of tools for assembling models of multibody mechanical systems. A few of these are listed in Table 1. The model building process has two distinctive features. First, the joints are defined in terms of their primitive action parameters from which all the required kinematic relations are derived. Thus, a user can contrive unusual joint configurations and is not restricted to a predefined set of standard joints. Second, the equations are formulated in Poincaré's form of Lagrange's equations[1] that admits the standard Lagrange equations as a special case [1, 3, 7]. However, Poincaré's form allows the exploitation of quasi–velocities which can greatly simplify the equations of motion.

The explicit models generated are of the form:

Kinematics: $$\dot{q} = V(q)\,p$$

Dynamics: $$M(q)\,\dot{p} + C(q,p)\,p + F(q,p,u) = 0$$

where q is a vector of configuration coordinates, p is a vector of quasi–velocities and u is a vector of exogenous inputs. They may be subjected to further symbolic processing for purposes such as nonlinear model reduction, nonlinear control system design or linearization. They may also be used for simulation or other numerical analysis procedures. To facilitate the latter applications, the package provides a direct interface to MATLAB/SIMULINK. In view of the complexity of models incorporating fully nonlinear kinematics, the C–code generated for this purpose, is organized to minimize the required numerical calculations.

To build a model, a user supplies defining data for individual joints and bodies, and the system structure. With this data, functions are available that can compute the kinetic energy function and inertia matrix as well as the gravitational potential energy function. It can also compute the strain potential energy and dissipation functions associated with deformations of flexible bodies. Various kinematic quantities can be obtained as well, e.g., end–effector configuration as a function of joint and deformation parameters. To complete a dynamic analysis, the user must supply the remaining parts of the potential energy

---

[1] *Poincaré's equations* [1-3] are also referred to as *Lagrange's equations in quasi-coordinates* [4, 5] or *pseudo-coordinates* [6].

function and definitions for any generalized forces. Functions are available to assist in developing these quantities.

Control

The control software can be grouped into three general categories: linear control, nonlinear control and geometric tools. Software tools are provided for the manipulation of linear controls systems in state space or frequency domain forms. Functions for the conversion of one form of model to the other are also included. Examples of the functions provided are listed in Table 2 and Table 3.

ProPac includes tools required to apply modern geometric methods of control system design to nonlinear affine systems [8, 9]. These methods play an important role in adaptive control system design [9-12] and variable structure control as well [13, 14]. Typical functions are given in Table 4. Table 5 illustrates functions for adaptive control system design and Table 6 shows the basic tools for variable structure control systems. Examples of geometric functions that support the control analysis constructions are given in Table 7.

Interfacing with MATLAB

MATLAB/SIMULINK are widely used tools for dynamic system simulation and control system analysis and design. Consequently, it is often convenient to implement the results of symbolic analysis in that environment. ***ProPac*** provides convenient interface tools. The function MatlabForm allows exporting numerical matrices constructed in ***Mathematica*** in a form readable by MATLAB. In addition, functions are included that construct 'optimized' C-source code files that compile as 'MEX files' defining S-functions for use as modules in the SIMULINK block diagram environment.

Separate functions are used to define models and controllers as illustrated in Figure 1. Models are defined in terms of Poincaré's equations (see above) and controllers are defined in terms of state descriptions:

$$\dot{x} = f(x,u)$$
$$y = h(x,u)$$

Controller modules do not require any MATLAB resources so they can be used for real time implementation via MATLAB's Real Time Workshop.

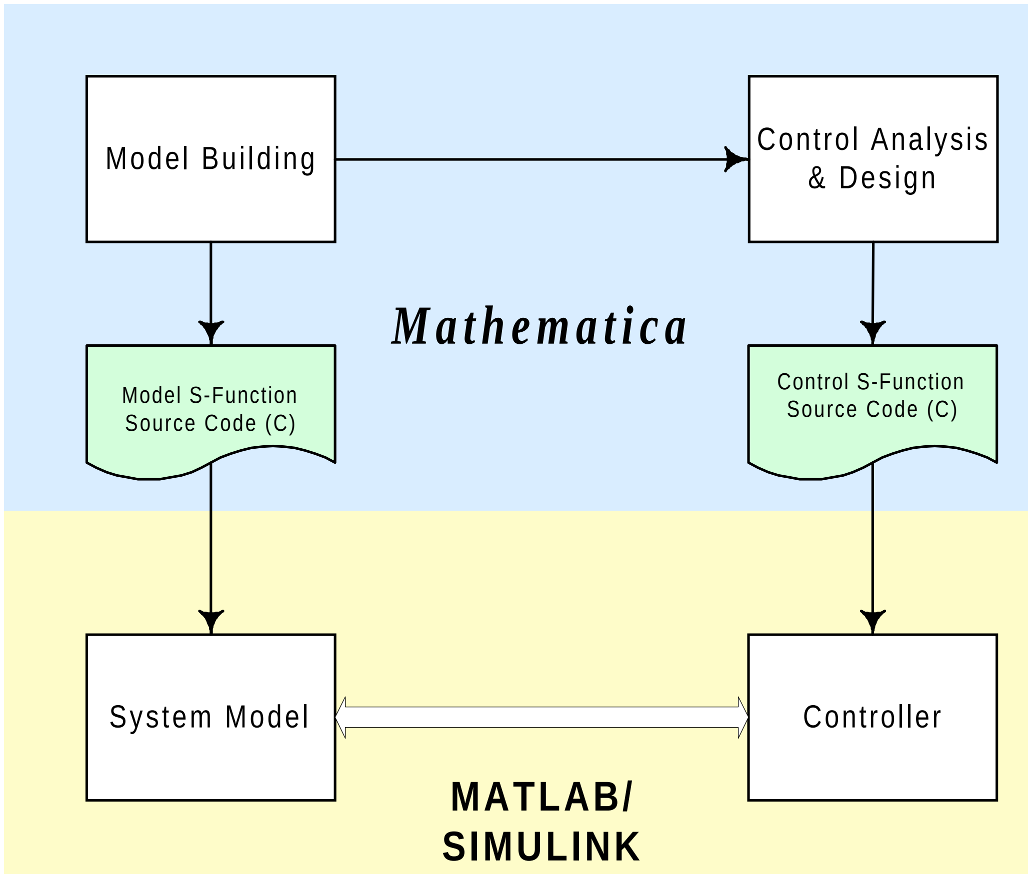

Figure 1. Interfacing with MATLAB/SIMULINK.

Table 1. **Multibody Dynamics**

Function Name	Operation
Joints	returns all of the kinematic quantities corresponding to a list of joint definitions
TreeInertia	computes the inertia matrix of a multibody system in a tree structure containing flexible and rigid bodies
EndEffector	returns the Euclidean Configuration Matrix of a body fixed frame at a specified node
NodeVelocity	returns the (6 dim) spatial velocity vector of a body fixed frame at a specified node
GeneralizedForce	computes the generalized force at specified node in terms of generalized coordinates
KinematicReplacements	sets up temporary replacement rules for repeated groups of expressions to simplify kinematic quantities
CreateModel	builds the kinematic and dynamic equations for tree structures
DifferentialConstraints	adds differential constraints to a tree configuration
AlgebraicConstraints	adds algebraic constraints to a tree configuration

Table 2. **Linear Systems: State Space**

Function Name	Operation
ControllablePair/ ObservablePair	tests for controllability and observability
ControllabilityMatrix **ObservabilityMatrix**	returns the controllability or observability matrices, respectively
PolePlace	state feedback pole placement based on Ackermann's formula with options
DecouplingConrol	state feedback and coordinate transformation that decouples input-output map
RelativeDegree	computes the vector relative degree
LQR, LQE	compute optimal quadratic regulator and estimator parameters

Table 3. **Linear Systems: Frequency Domain**

Function Name	Operation
LeastCommonDenominator	finds the least common denominator of the elements of a proper, rational G(s)
Poles	finds the roots of the least common denominator
LaurentSeries	computes the Laurent series up to specified order
AssociatedHankelMatrix	computes the Hankel matrix associated with Laurent expansion of G(s)
McMillanDegree	computes the degree of the minimal realization of G(s)
ControllableRealization **ObservableRealization**	compute, respectively, the controllable and observable realizations of a transfer function

Table 4. **Nonlinear systems: Geometric Control**

Function Name	Operation
VectorRelativeOrder	computes the relative degree vector
DecouplingMatrix	computes the decoupling matrix
IOLinearize	computes the linearizing control
NormalCoordinates	computes the partial state transformation,
LocalZeroDynamics	computes the local form of the zero dynamics
StructureAlgorithm	computes the parameters of an inverse system
DynamicExtension	applies dynamic extension as a remedy for singular decoupling matrix

Table 5. **Nonlinear systems: Adaptive Control**

Function Name	Operation
AdaptiveRegulator	generates an adaptive regulator for a class of linearizable systems
AdaptiveBackstepRegulator	computes an adaptive regulator by backstepping for SISO systems in PSFF form
AdaptiveTracking	computes an adaptive tracking controller
PSFFCond	tests a system to determine if it is reducible to PSFF form
PSFFSolve	transforms a system to PSFF form if possible

Table 6 **Nonlinear systems: Variable Structure Control**

Function Name	Operation
SlidingSurface	generates the sliding (switching) surface for feedback linearizable nonlinear systems
SwitchingControl	computes the switching functions – allows the inclusion of smoothing and moderating functions

Table 7. **Nonlinear systems: Geometric Tools**

Function Name	Operation
LieBracket	computes the Lie bracket of a given pair of vector fields
Ad	computes the iterated Lie bracket of specified order of a pair of vector fields
Involutive	tests a set of vector fields to determine if it is involutive
Span	generates a set of basis vector fields for a given set of vector fields
FlowComposition	generates a composite function from a given set of flows
ParametricManifold	computes a parametric representation for an imbedded manifold
StateTransformation	transforms nonlinear dynamic models in various forms

3. For Users of Earlier Versions

Users of earlier versions of ***TSi Dynamics*** and ***TSi Controls*** should be aware that some function names have been changed. This has been done in order to conform to WRI naming conventions and/or to enhance clarity. Table 8 and Table 9 summarize the function name changes.

Table 8 **Control Functions**

Old	*New*
`AdaptBackstepReg`	`AdaptiveBackstepRegulator`
`AlgRiccatiEq`	`AlgebraicRiccatiEquation`
`BodePlot`	`Bode`
`Controllable`	`ControllablePair`
`H2H1andH2`	`HToH1AndH2`
`H2NandM`	`HToNAndM`
`HankelMat/HankelMatrix`	`AssociatedHankelMatrix`
`IverseTrans`	`InverseTransformation`
`LocalInverseTrans`	`LocalInverseTransformation`
`MatRank`	`MatrixRank`
`NyquistPlot`	`Nyquist`
`Observable`	`ObservablePair`
`PartialTransSystem`	`PartialTransformSystem`
`RootLocusPlot`	`RootLocus`
`TransSystem`	`TransformSystem`

Table 9 **Dynamics Functions**

Old	***New***
AlgConstrainedSys	AlgebraicConstraints
Atil2a	ATildaToA
Atilda	AToATilda
Cmat	CMatrix
DiffConstrainedSys	DifferentialConstraints
EndEffectorVelocity	NodeVelocity
GamaKin	SimpleJointKinematics
GamCmpnd	CompoundJointKinematics
HCmpnd	CompoundJointMap
Leuler	RotationMatrixEuler
Lrot	JointRotation
PoincareFunc	PoincareFunctionCombined
PoincareFuncSim	PoincareFunction
PoinCoef	PoincareCoefficient
RotMat2Euler	RotationMatrixToEuler
Rtran	JointTranslation
Xeuler	ConfigurationMatrixEuler
XXCmpd	CompoundJointConfiguration
XXeuc	SimpleJointConfiguration

4. References

[1] V. I. Arnold, V. V. Kozlov, and A. I. Neishtadt, *Mathematical Aspects of Classical and Celestial Mechanics*, vol. 3. Heidelberg: Springer–Verlag, 1988.

[2] N. G. Chetaev, "On the Equations of Poincaré," *PMM (Applied Mathematics and Mechanics)*, pp. 253–262, 1941.

[3] N. G. Chetaev, *Theoretical Mechanics*. New York: Springer–Verlag, 1989.

[4] L. Meirovitch, *Methods of Analytical Dynamics*. New York: McGraw–Hill, Inc., 1970.

[5] J. I. Neimark and N. A. Fufaev, *Dynamics of Nonholonomic Systems*, vol. 33. Providence: American Mathematical Society, 1972.

[6] F. Gantmacher, *Lectures in Analytical Mechanics*, English Translation ed. Moscow: Mir, 1975.

[7] H. G. Kwatny and G. L. Blankenship, "Symbolic Construction of Models for Multibody Dynamics," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 271-281, 1995.

[8] A. Isidori, *Nonlinear Control Systems*, 3 ed. London: Springer-Verlag, 1995.

[9] H. Nijmeijer and H. J. van der Schaft, *Nonlinear Dynamical Control Systems*. New York: Springer–Verlag, 1990.

[10] G. L. Blankenship, R. Ghanadan, H. G. Kwatny, C. LaVigna, and V. Polyakov, "Integrated tools for Modeling and Design of Controlled Nonlinear Systems," *IEEE Control Systems*, vol. 15, pp. 65-79, 1995.

[11] I. Kanellakapoulos, P. V. Kokotovic, and A. S. Morse, "Systematic design of Adaptive Controllers for Feedback Linearizable Systems," *IEEE Transactions on Automatic Control*, vol. AC–36, pp. 1241–1253, 1991.

[12] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic, "Adaptive Nonlinear Control Without Overparameterization," *Systems and Control Letters*, vol. 19, pp. 177-185, 1992.

[13] H. G. Kwatny and H. Kim, "Variable Structure Regulation of Partially Linearizable Dynamics," *Systems & Control Letters*, vol. 15, pp. 67–80, 1990.

[14] H. G. Kwatny and G. L. Blankenship, "Symbolic Tools for Variable Structure Control System Design: The Zero Dynamics," presented at IFAC Symposium on Robust Control via Variable Structure and Lyapunov Techniques, Benevento, Italy, 1994.