The Dynamical Systems Toolbox (Part 2) : Getting Started

Workshop 11th October 2010

***Etienne Coetzee**, Phani Thota, and James Rankin*

University of BRISTOL

Contents

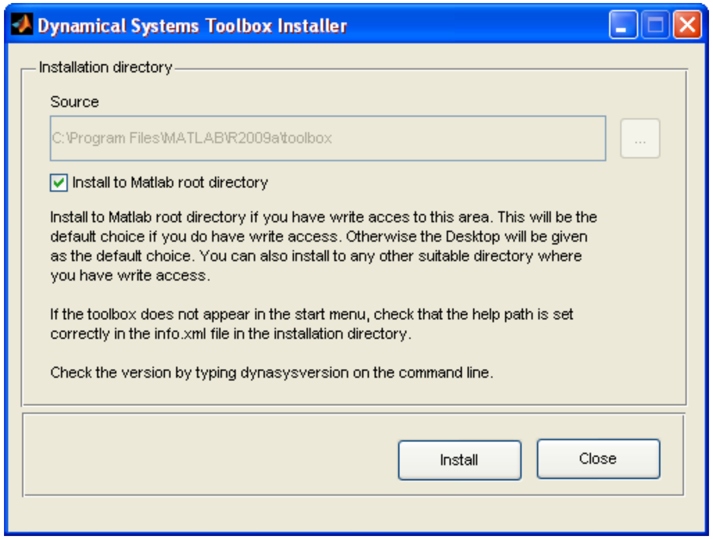
- Installation
- Help
  - Main Window
  - Other Information
- Function file
- Constants file
  - Converting current files
- Classes
  - autosimopts
  - autoconstants
  - autof7
  - autof8
- Demo ‘ab’
- Adding own examples
  - Template files

Installation

- Download from github page at

  https://github.com/ecoetzee/Dynamical-Systems-Toolbox

- Start Matlab and change to directory where zip file was unpacked.
- Will only work for R2009a or higher.
- Run the installer by running `installdynasys.m`



Dynamical Systems Toolbox Installer

Installation directory

Source

C:\Program Files\MATLAB\R2009a\toolbox

☑ Install to Matlab root directory

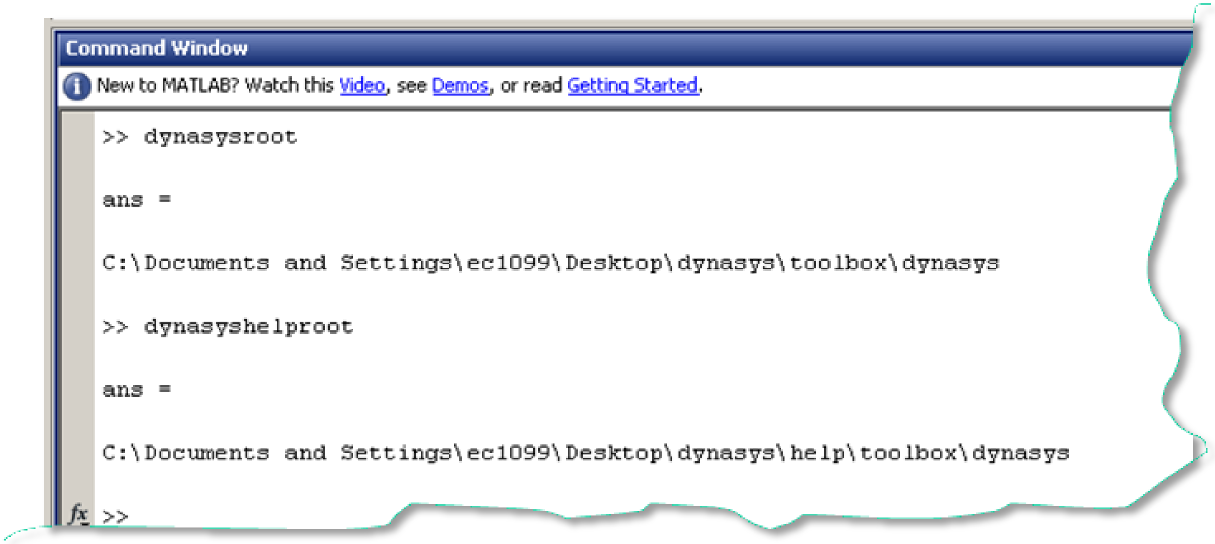
Install to Matlab root directory if you have write acces to this area. This will be the default choice if you do have write access. Otherwise the Desktop will be given as the default choice. You can also install to any other suitable directory where you have write access.

If the toolbox does not appear in the start menu, check that the help path is set correctly in the info.xml file in the installation directory.

Check the version by typing dynasysversion on the command line.

Installation

- Use default options if you have write access to matlab root directories. Otherwise, install somewhere where you have write access.
- Check root directories by typing



Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> dynasysroot

ans =

C:\Documents and Settings\ec1099\Desktop\dynasys\toolbox\dynasys

>> dynasyshelproot

ans =

C:\Documents and Settings\ec1099\Desktop\dynasys\help\toolbox\dynasys

>>
```

Installation Paths

- The following directories need to be on the path

```
$dynasysroot\utils\plaut
$dynasysroot\utils\autoconst
$dynasysroot\utils\autoobj
$dynasysroot\src
$dynasysroot\dynasysdemos
$dynasysroot\icons
$dynasysroot\cmds
$dynasysroot
$dynasyshelproot
```

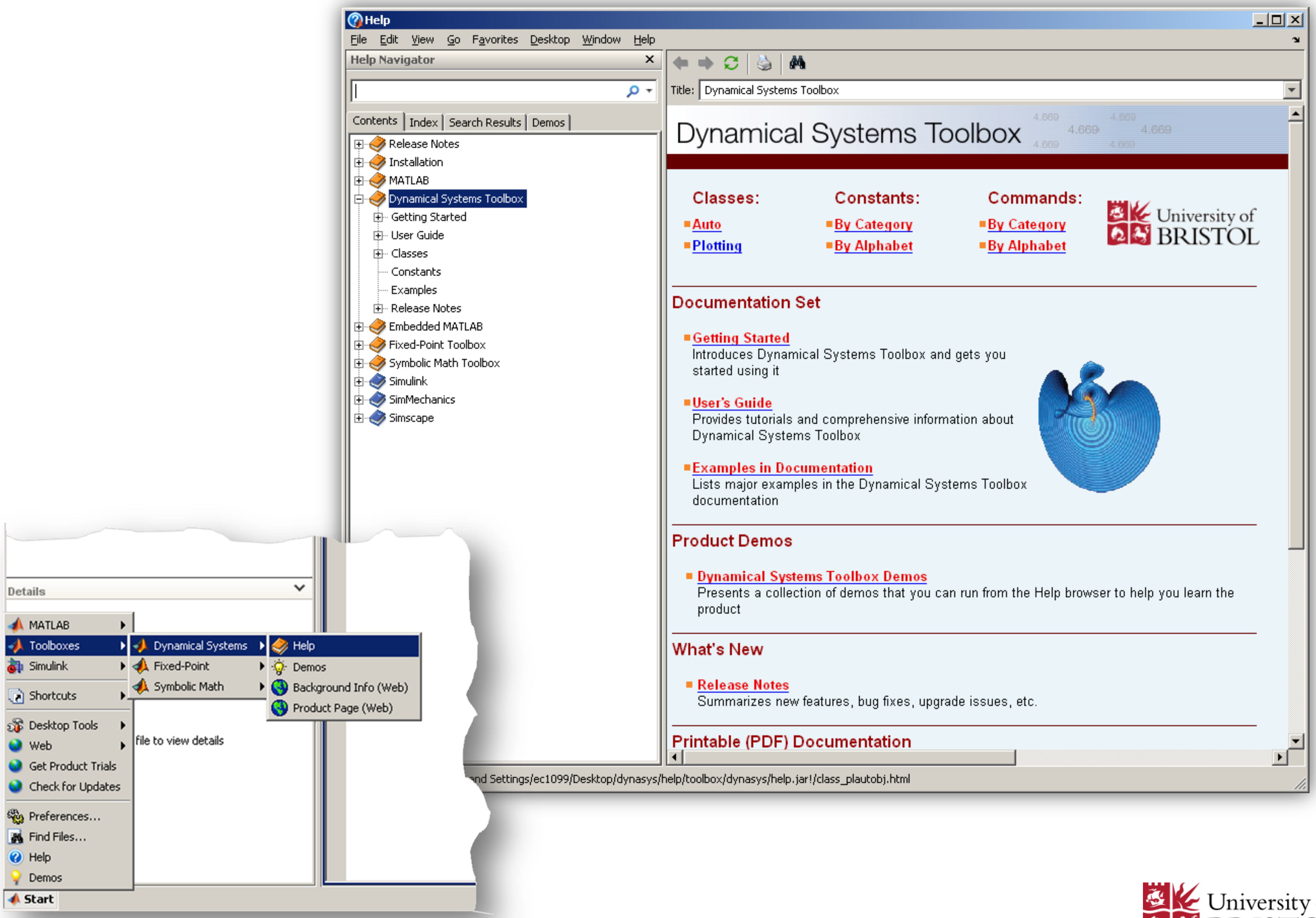
- `$dynasysroot` and `$dynasyshelproot` denote the root directories when you type these commands without the `$` sign.

Mex-Compilation

- AUTO is accessed via a mex-file, hence we had to compile AUTO with a FORTRAN compiler.
- Compiled with Intel Visual Fortran 9.1 on Windows.
- Also tested on Linux with gcc 4.4.
- Installations on Windows should work out of the box, but if in doubt, recompile.
- You will need to recompile on Linux or Unix.
- Make sure that mex-installation is correct before you try to compile. Check mex-installation by running yprime example.
- Run compile command

```
» compileauto07p('obj')
```

Opening Help



Dynamical Systems Toolbox

Classes:

- Auto
- Plotting

Constants:

- By Category
- By Alphabet

Commands:

- By Category
- By Alphabet

Documentation Set

- **Getting Started**
  Introduces Dynamical Systems Toolbox and gets you started using it
- **User's Guide**
  Provides tutorials and comprehensive information about Dynamical Systems Toolbox
- **Examples in Documentation**
  Lists major examples in the Dynamical Systems Toolbox documentation

Product Demos

- **Dynamical Systems Toolbox Demos**
  Presents a collection of demos that you can run from the Help browser to help you learn the product

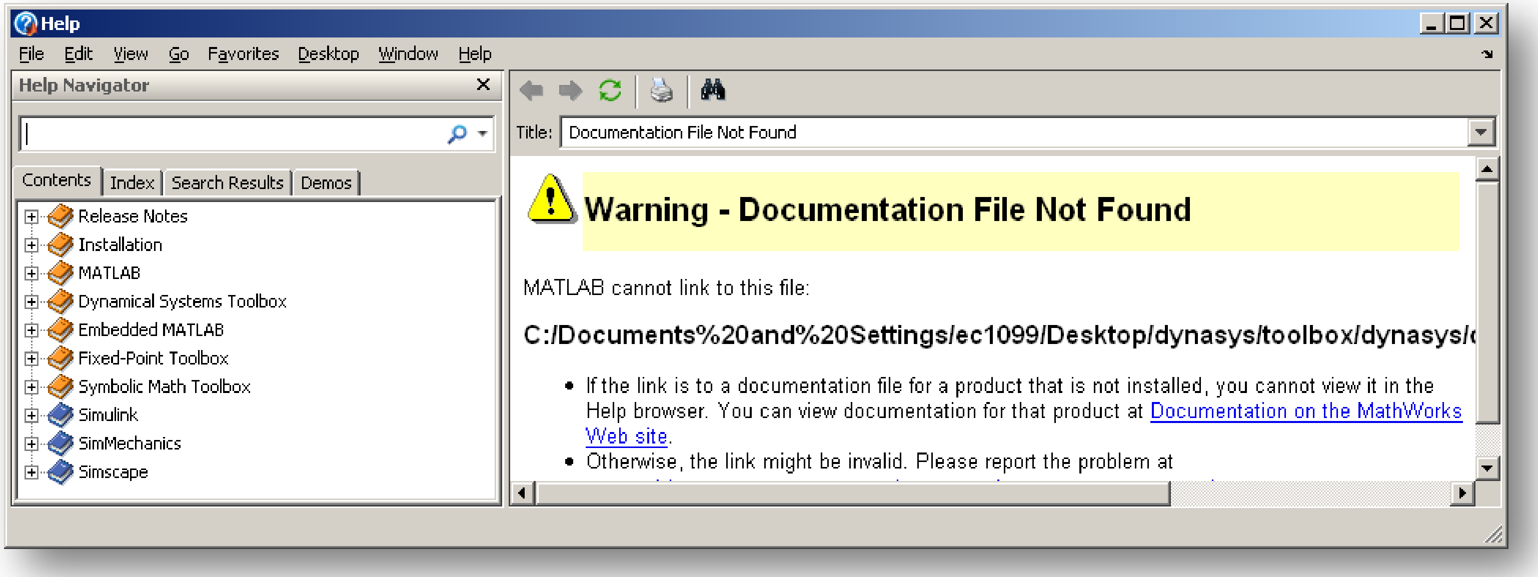
What's New

- **Release Notes**
  Summarizes new features, bug fixes, upgrade issues, etc.

Printable (PDF) Documentation

and Settings/ec1099/Desktop/dynasys/help/toolbox/dynasys/help.jar!/class_plautobj.html

What if help is not working?



Title: Documentation File Not Found

Warning - Documentation File Not Found

MATLAB cannot link to this file:

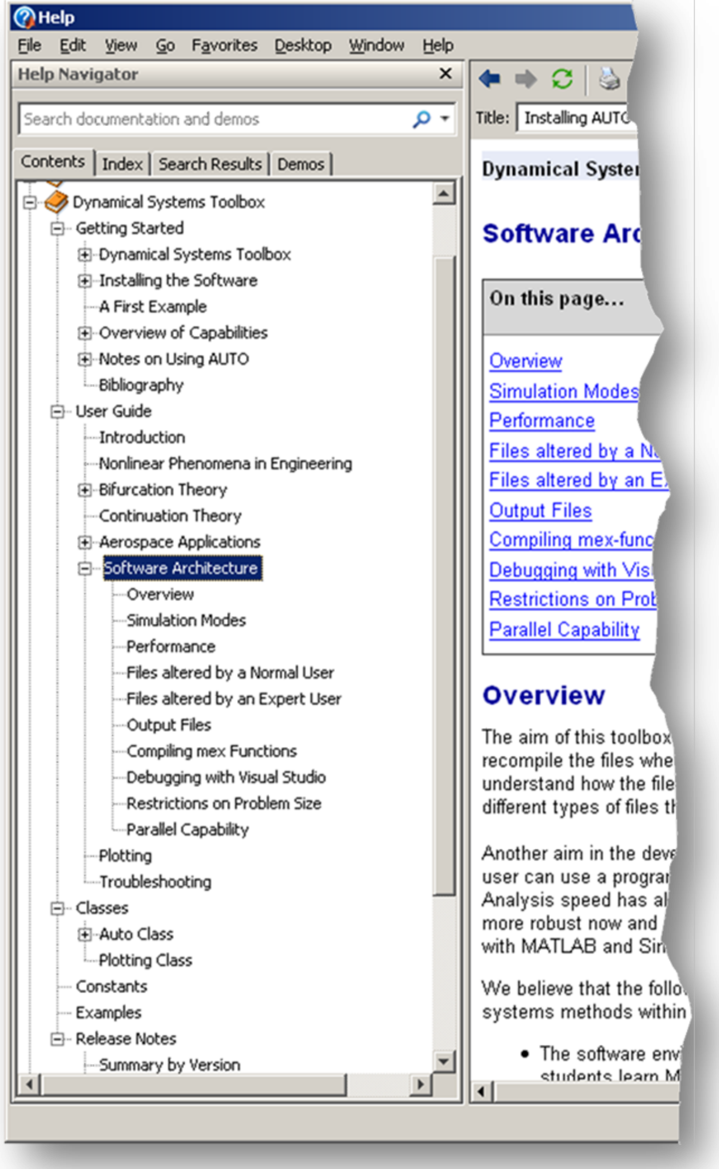
C:/Documents%20and%20Settings/ec1099/Desktop/dynasys/toolbox/dynasys/

- If the link is to a documentation file for a product that is not installed, you cannot view it in the Help browser. You can view documentation for that product at Documentation on the MathWorks Web site.
- Otherwise, the link might be invalid. Please report the problem at

- Make sure line 10 of the **`info.xml`** file in **`$dynasysroot`** should refer to:
  - **`$dynasyshelproot`** path if not installed to **`$matlabroot`**.
  - Defined as **`$docroot/toolbox/dyansys`** if installed to **`$matlabroot`** directory.

Help layout

- New tree structure of Matlab 2009b adopted.


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search documentation and demos
Contents
Index
Search Results
Demos
Title: Installing AUTO

- Dynamical Systems Toolbox
  - Getting Started
    - Dynamical Systems Toolbox
    - Installing the Software
    - A First Example
    - Overview of Capabilities
    - Notes on Using AUTO
    - Bibliography
  - User Guide
    - Introduction
    - Nonlinear Phenomena in Engineering
    - Bifurcation Theory
    - Continuation Theory
    - Aerospace Applications
    - Software Architecture
      - Overview
      - Simulation Modes
      - Performance
      - Files altered by a Normal User
      - Files altered by an Expert User
      - Output Files
      - Compiling mex Functions
      - Debugging with Visual Studio
      - Restrictions on Problem Size
      - Parallel Capability
    - Plotting
    - Troubleshooting
  - Classes
    - Auto Class
    - Plotting Class
  - Constants
  - Examples
  - Release Notes
    - Summary by Version

Dynamical Syste

Software Arc

On this page...

Overview
Simulation Modes
Performance
Files altered by a N
Files altered by an E
Output Files
Compiling mex-func
Debugging with Vis
Restrictions on Prob
Parallel Capability

Overview

The aim of this toolbox
recompile the files whe
understand how the file
different types of files th

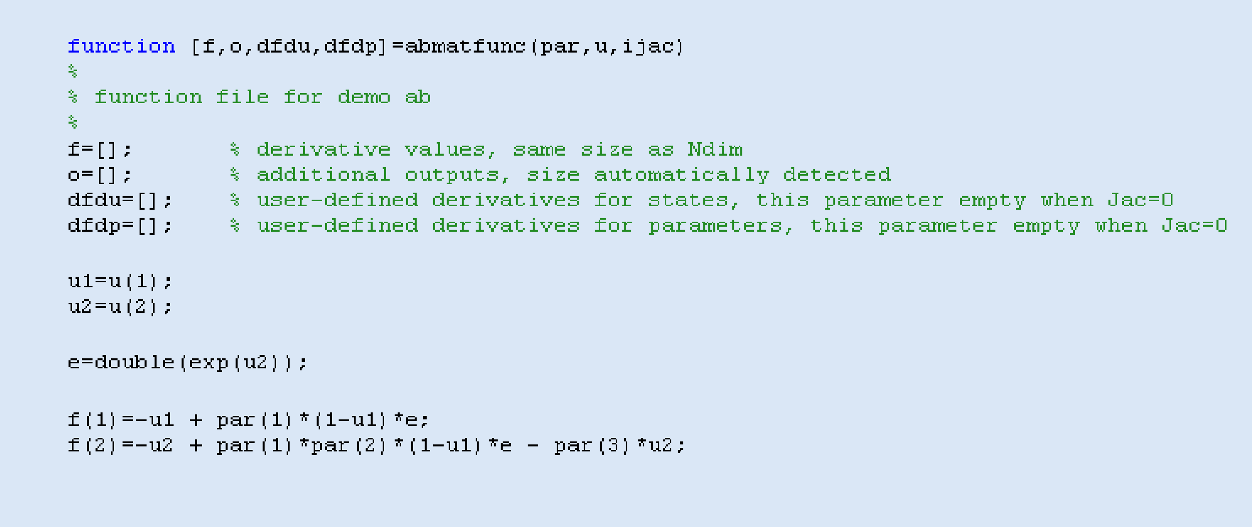
Another aim in the deve
user can use a prograr
Analysis speed has al
more robust now and
with MATLAB and Sin

We believe that the follo
systems methods within

- The software env
students learn M

Function file

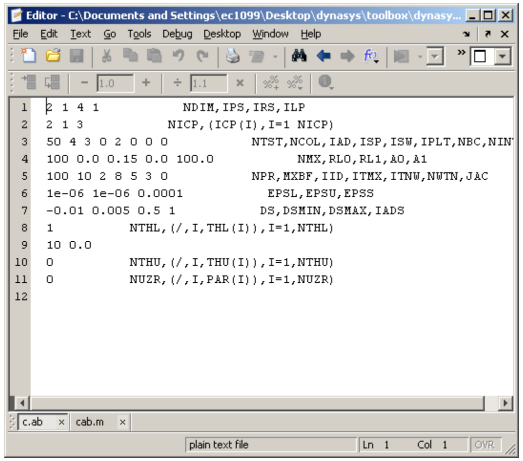
- Make sure number of arguments **in** and arguments **out** similar as below (for most cases). Different depending on problem type.
- Any arbitrary function name can be used.
- If arguments incorrect, or syntax errors, Matlab will throw an error, explaining what the problem is.



```
function [f,o,dfdu,dfdp]=abmatfunc(par,u,ijac)
%
% function file for demo ab
%
f=[];       % derivative values, same size as Ndim
o=[];       % additional outputs, size automatically detected
dfdu=[];    % user-defined derivatives for states, this parameter empty when Jac=0
dfdp=[];    % user-defined derivatives for parameters, this parameter empty when Jac=0

u1=u(1);
u2=u(2);

e=double(exp(u2));

f(1)=-u1 + par(1)*(1-u1)*e;
f(2)=-u2 + par(1)*par(2)*(1-u1)*e - par(3)*u2;
```

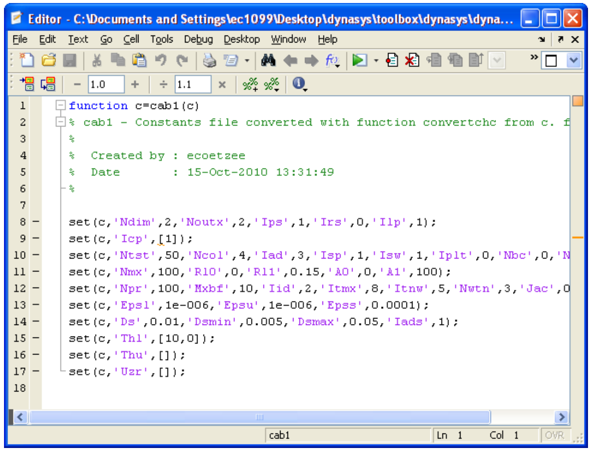
Constants file

- You do not need a constants file anymore, because the constants can be set directly in the constants object.
- You can however use an m-file equivalent to save the hassle of setting it up in the object itself, and to make it easier to port existing files.
- Use **`convertchc`** command to convert your old file into new format.


Editor - C:\Documents and Settings\ec1099\Desktop\dynasys\toolbox\dynasy...
File Edit Text Go Tools Debug Desktop Window Help
c.ab
cab.m
plain text file
Ln 1
Col 1
OVR

```
2 1 4 1             NDIM,IPS,IRS,ILP
2 1 3             NICP,(ICP(I),I=1 NICP)
50 4 3 0 2 0 0 0            NTST,NCOL,IAD,ISP,ISW,IPLT,NBC,NIN
100 0.0 0.15 0.0 100.0            NMX,RL0,RL1,A0,A1
100 10 2 8 5 3 0            NPR,MXBF,IID,ITMX,ITNW,NWTN,JAC
1e-06 1e-06 0.0001            EPSL,EPSU,EPSS
-0.01 0.005 0.5 1            DS,DSMIN,DSMAX,IADS
1          NTHL,(/,I,THL(I)),I=1,NTHL)
10 0.0
0          NTHU,(/,I,THU(I)),I=1,NTHU)
0          NUZR,(/,I,PAR(I)),I=1,NUZR)
```

c.ab

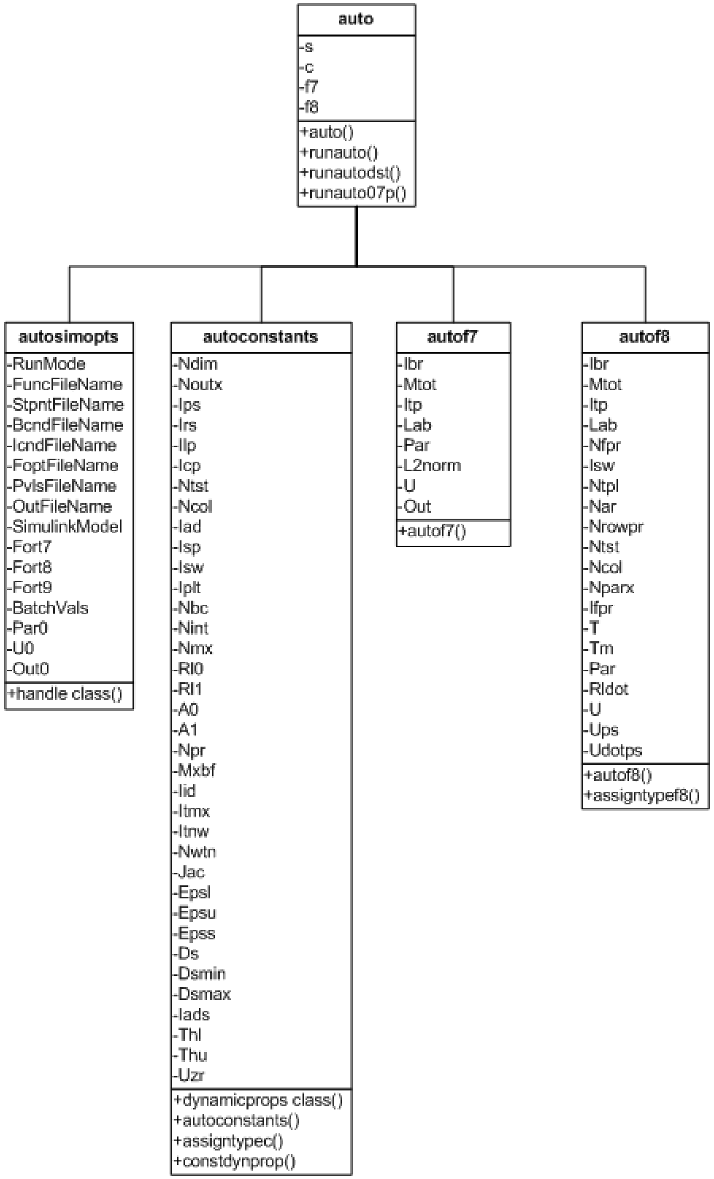

Editor - C:\Documents and Settings\ec1099\Desktop\dynasys\toolbox\dynasys\dyna...
File Edit Text Go Cell Tools Debug Desktop Window Help
cab1
Ln 1
Col 1
OVR

```
function c=cab1(c)
% cab1 - Constants file converted with function convertchc from c. f
%
%  Created by : ecoetzee
%  Date       : 15-Oct-2010 13:31:49
%

set(c,'Ndim',2,'Noutx',2,'Ips',1,'Irs',0,'Ilp',1);
set(c,'Icp',[1]);
set(c,'Ntst',50,'Ncol',4,'Iad',3,'Isp',1,'Isw',1,'Iplt',0,'Nbc',0,'N
set(c,'Nmx',100,'Rl0',0,'Rl1',0.15,'A0',0,'A1',100);
set(c,'Npr',100,'Mxbf',10,'Iid',2,'Itmx',8,'Itnw',5,'Nwtn',3,'Jac',0
set(c,'Epsl',1e-006,'Epsu',1e-006,'Epss',0.0001);
set(c,'Ds',0.01,'Dsmin',0.005,'Dsmax',0.05,'Iads',1);
set(c,'Thl',[10,0]);
set(c,'Thu',[]);
set(c,'Uzr',[]);
```

cab.m

Classes

- The auto class calls four other sub-classes:
  - **autosimopts:** simulation options
  - **autoconstants:** constants, similar to constants file in AUTO
  - **autof7:** continuation outputs similar to fort.7 file
  - **autof8:** special points similar to fort.8
- **Make sure you know the difference between Value and Handle classes in Matlab!**



auto
-s
-c
-f7
-f8
+auto()
+runauto()
+runautodst()
+runauto07p()

autosimopts
-RunMode
-FuncFileName
-StpntFileName
-BcndFileName
-IcndFileName
-FoptFileName
-PvlsFileName
-OutFileName
-SimulinkModel
-Fort7
-Fort8
-Fort9
-BatchVals
-Par0
-U0
-Out0
+handle class()

autoconstants
-Ndim
-Noutx
-Ips
-Irs
-Ilp
-Icp
-Ntst
-Ncol
-Iad
-Isp
-Isw
-Iplt
-Nbc
-Nint
-Nmx
-Rl0
-Rl1
-A0
-A1
-Npr
-Mxbf
-Iid
-Itmx
-Itnw
-Nwtn
-Jac
-Epsl
-Epsu
-Epss
-Ds
-Dsmin
-Dsmax
-Iads
-Thl
-Thu
-Uzr
+dynamicprops class()
+autoconstants()
+assigntypec()
+constdynprop()

autof7
-Ibr
-Mtot
-Itp
-Lab
-Par
-L2norm
-U
-Out
+autof7()

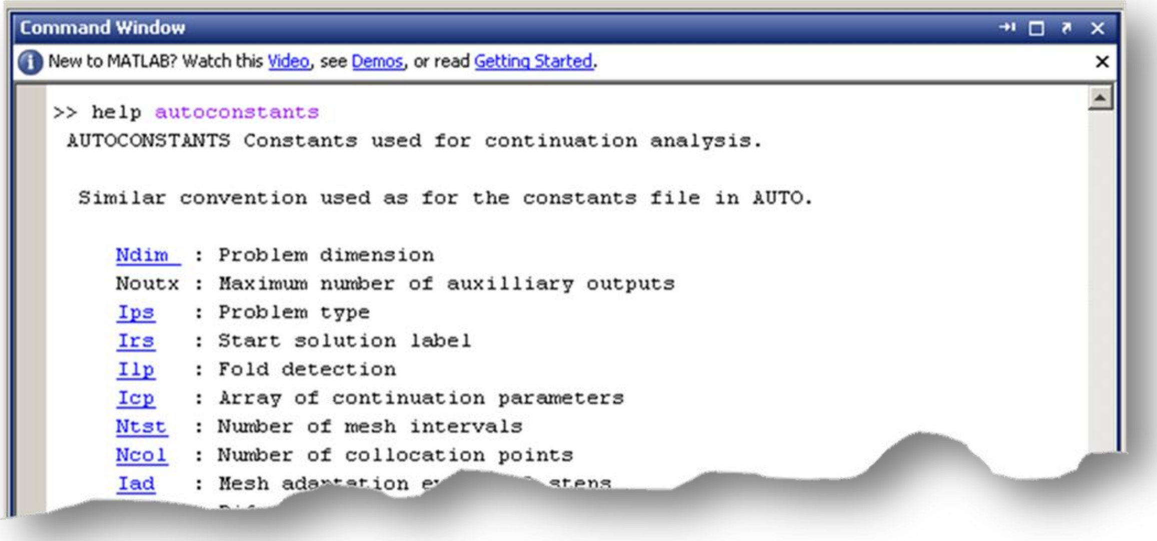
autof8
-Ibr
-Mtot
-Itp
-Lab
-Nfpr
-Isw
-Ntpl
-Nar
-Nrowpr
-Ntst
-Ncol
-Nparx
-Ifpr
-T
-Tm
-Par
-Rldot
-U
-Ups
-Udotps
+autof8()
+assigntypef8()

autosimopts – simulation options

Property	Value	Description
RunMode	{'DST'} | '07P'	Can either run model where outputs and inputs are obtained from objects ('DST'), or you can use the traditional way in which AUTO is used ('07P')
FuncFileName	'func'	Name of function file. This can be any name.
StpntFileName	'stpnt'	Starting conditions. Only needed for '07P' mode, or when calculating Parabolic PDE's. Can be any name.
BcndFileName	'bcnd'	Function containing boundary conditions. Can be any name.
IcndFileName	'icnd'	Function containing boundary value and integral constraints. Can be any name.
FoptFileName	'fopt'	Not used
PvlsFileName	'pvls'	Not used
OutFileName	any string	• The name of the extension for the constants file if using '07P' mode, i.e. **c.OutFileName**. • Or the extension name of the output files when files are required in 'DST' mode. The following files will be
SimulinkModel	model name	Name of simulink model. Model will be automatically opened and compiled.
Fort7	{'off'} | 'on'	Write fort.7 file if requested. File name will be **b.OutFileName**
Fort8	{'off'} | 'on'	Write fort.8 file if requested. File name will be **s.OutFileName**
Fort9	{'off'} | 'on'	Write fort.7 file if requested. File name will be **d.OutFileName**
BatchVals	[n x m]	Used for storing additional simulation information, i.e. tables for DOE etc, where n and m can be any value
Par0	[NPAX x 1]	Initial values for parameters. This is only used in 'DST' mode. n needs to be smaller than NPARX, and also remember that PAR(11) is reserved for limit cycle continuations.
U0	[NDIM x 1]	Initial values for continuation states.
Out0	[]	Initial values for additional Outputs. This does not have to be filled in, as runautodst method tries to determine initial outputs.

autoconstants – constants definition

- Removal of some dimensional constants: **NTHL**, **NTHU**, **NUZR**. Automatically detected.
- Redefined parameters **THL**, **THU**, **UZR**. Now defined as [nx2] vectors.
- New parameter called **Noutx**. Denotes maximum number of auxiliary outputs. This is a new parameter.
- Cross referenced help from command line and in documentation.



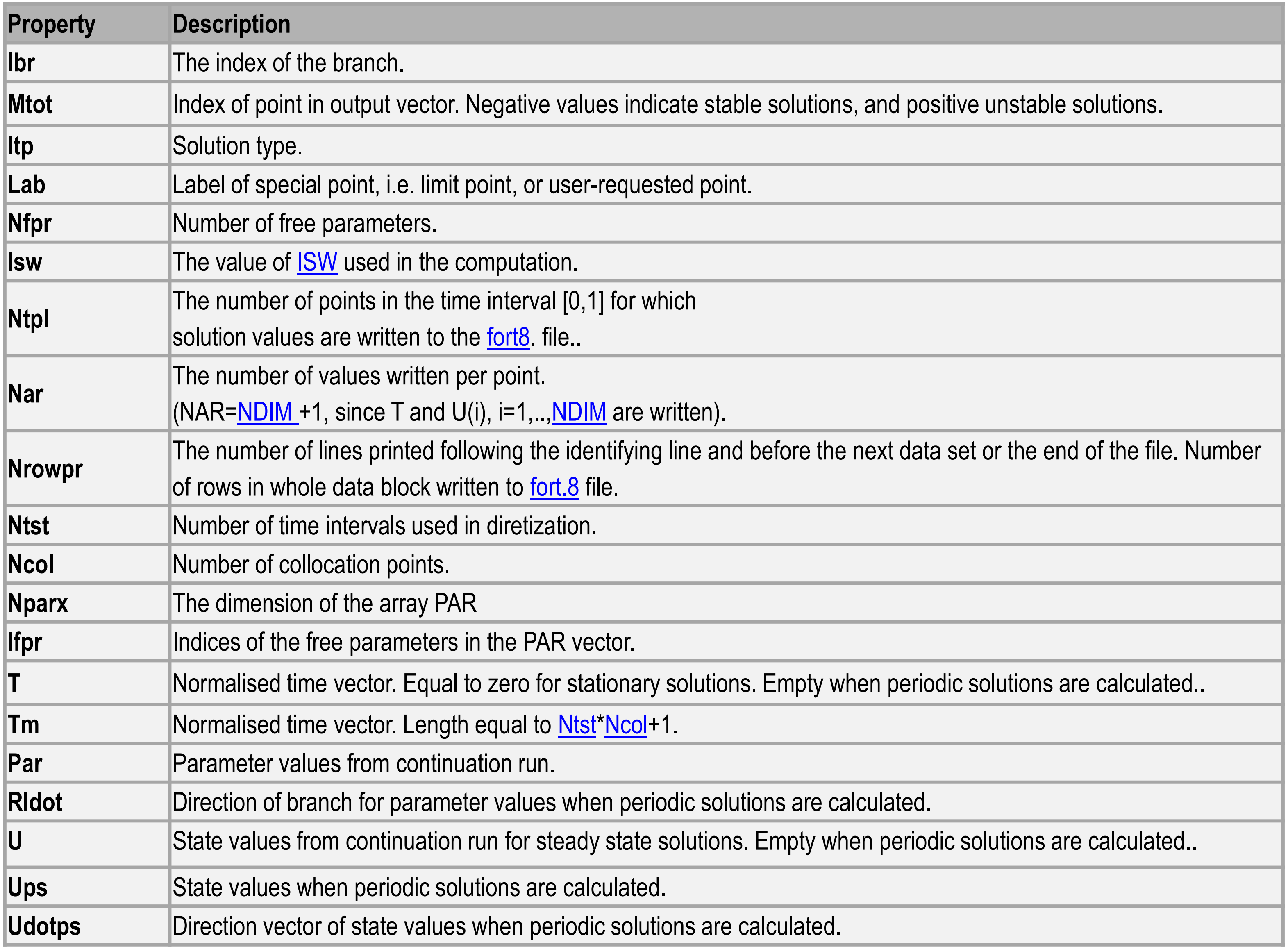
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> help autoconstants
 AUTOCONSTANTS Constants used for continuation analysis.

  Similar convention used as for the constants file in AUTO.

     Ndim  : Problem dimension
     Noutx : Maximum number of auxilliary outputs
     Ips   : Problem type
     Irs   : Start solution label
     Ilp   : Fold detection
     Icp   : Array of continuation parameters
     Ntst  : Number of mesh intervals
     Ncol  : Number of collocation points
     Iad   : Mesh adaptation e      steps
```

autof7 - outputs

Property	Description
Ibr	Branch number.
Mtot	Index of point in output vector. Negative values indicate stable solutions, and positive unstable solutions.
Itp	Solution type.
Lab	Label of special point, i.e. limit point, or user-requested point.
Par	Parameter values from continuation run.
L2norm	L2-norm value.
U	State values from continuation run.
Out	Additional outputs from continuation run.

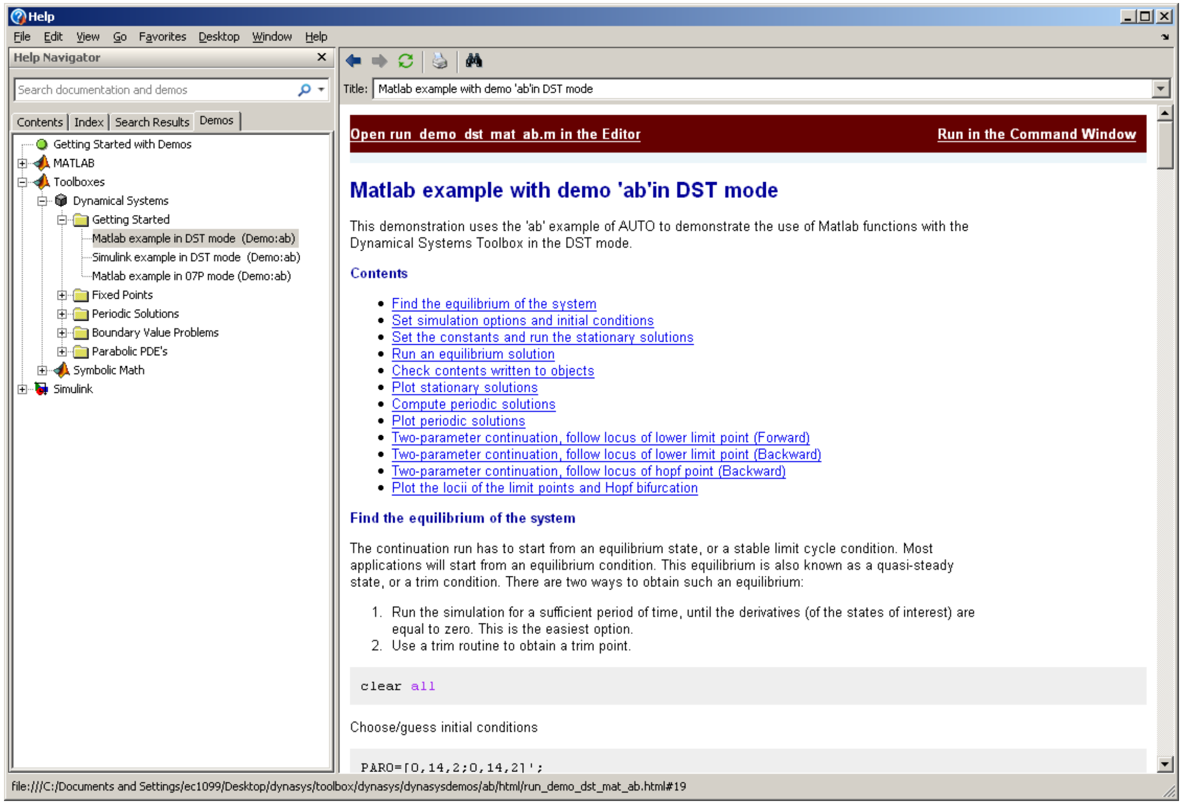
autof8 – special points



Property	Description
Ibr	The index of the branch.
Mtot	Index of point in output vector. Negative values indicate stable solutions, and positive unstable solutions.
Itp	Solution type.
Lab	Label of special point, i.e. limit point, or user-requested point.
Nfpr	Number of free parameters.
Isw	The value of ISW used in the computation.
Ntpl	The number of points in the time interval [0,1] for which solution values are written to the fort8. file..
Nar	The number of values written per point. (NAR=NDIM +1, since T and U(i), i=1,..,NDIM are written).
Nrowpr	The number of lines printed following the identifying line and before the next data set or the end of the file. Number of rows in whole data block written to fort.8 file.
Ntst	Number of time intervals used in diretization.
Ncol	Number of collocation points.
Nparx	The dimension of the array PAR
Ifpr	Indices of the free parameters in the PAR vector.
T	Normalised time vector. Equal to zero for stationary solutions. Empty when periodic solutions are calculated..
Tm	Normalised time vector. Length equal to Ntst*Ncol+1.
Par	Parameter values from continuation run.
Rldot	Direction of branch for parameter values when periodic solutions are calculated.
U	State values from continuation run for steady state solutions. Empty when periodic solutions are calculated..
Ups	State values when periodic solutions are calculated.
Udotps	Direction vector of state values when periodic solutions are calculated.

Demo ‘ab’

- Demos accessed via menu.
- Demos contained in `$dynasysroot/dynasysdemos`
- Script to publish in correct format.


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search documentation and demos
Contents | Index | Search Results | Demos
Getting Started with Demos
MATLAB
Toolboxes
Dynamical Systems
Getting Started
Matlab example in DST mode (Demo:ab)
Simulink example in DST mode (Demo:ab)
Matlab example in 07P mode (Demo:ab)
Fixed Points
Periodic Solutions
Boundary Value Problems
Parabolic PDE's
Symbolic Math
Simulink
Title: Matlab example with demo 'ab'in DST mode
Open run_demo_dst_mat_ab.m in the Editor
Run in the Command Window

Matlab example with demo 'ab'in DST mode

This demonstration uses the 'ab' example of AUTO to demonstrate the use of Matlab functions with the Dynamical Systems Toolbox in the DST mode.

Contents

- Find the equilibrium of the system
- Set simulation options and initial conditions
- Set the constants and run the stationary solutions
- Run an equilibrium solution
- Check contents written to objects
- Plot stationary solutions
- Compute periodic solutions
- Plot periodic solutions
- Two-parameter continuation, follow locus of lower limit point (Forward)
- Two-parameter continuation, follow locus of lower limit point (Backward)
- Two-parameter continuation, follow locus of hopf point (Backward)
- Plot the locii of the limit points and Hopf bifurcation

Find the equilibrium of the system

The continuation run has to start from an equilibrium state, or a stable limit cycle condition. Most applications will start from an equilibrium condition. This equilibrium is also known as a quasi-steady state, or a trim condition. There are two ways to obtain such an equilibrium:

1. Run the simulation for a sufficient period of time, until the derivatives (of the states of interest) are equal to zero. This is the easiest option.
2. Use a trim routine to obtain a trim point.

```
clear all
```

Choose/guess initial conditions

```
PAR0=[0,14,2;0,14,2]';
```

file:///C:/Documents and Settings/ec1099/Desktop/dynasys/toolbox/dynasys/dynasysdemos/ab/html/run_demo_dst_mat_ab.html#19

University of BRISTOL

Adding your own examples

- An html template file contained in `$dynasyshelproot/templates`
- Explanation of your work can be written here. Hyperlink to papers etc.
- Set up m-file in a similar way as in demonstration directory. We will then run a script to obtain correct format.
- At the moment the toolbox has a license that will time limit the use. This will be removed once we have enough examples.