DDENLP

Generated by Doxygen 1.8.11

Contents

Chapter 1

Documentation of the Toolbox DDENLP for robust steady state optimization of delayed systems.

1.1 Using this toolbox

This toolbox aids with the optimization of delayed systems using normal vector constraints for robust stability properties.

The code documentation can be found in this file.

Some common application/coding problems can be found in the Frequently Asked Questions.

Additionally, there are some example files.

1.2 Funding

The development of this toolbox was funded by Deutsche Forschungsgemeinschaft (grant MO 1086/13).

Author

Jonas Otten and Martin Moennigmann

Date

18 Jul 2017

Chapter 2

Frequently Asked Questions

2.1 Convergence of Initialization

Why does my initial value of for a (mod)Hopf converge to a (mod)fold?
Check if you have specified omega $\sim$= 0

2.2 DDE-BIFTOOL

Why does the calculation of eigenvalues using DDE-BIFTOOL throw the error message "`Input to EIG must not contain NaN or Inf.`"?
The DDE-BIFTOOL-Parameter `method.stability.minimal_real_part` is too big. Try reducing it.

Author

Jonas Otten and Martin Moennigmann

Date

20 Sep 2017

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

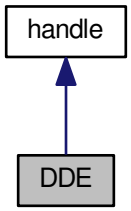
Chapter 6

Class Documentation

6.1 DDE Class Reference

The instances of this class contain information for a DDE used in DDENLP.

Inheritance diagram for DDE:



Public Member Functions

- function DDE (in rhsHandle, in delays, in xNomGuess, in uncParam, in certOptParam)

  Class constructor.
- function getHandles (in aDDE, in type)

  hands back function handles of a DDE instance

Public Attributes

- Property foldManiHandle

  function handle for fold bifurcation manifold
- Property modfoldManiHandle

  function handle for modified fold bifurcation manifold
- Property hopfManiHandle

function handle for hopf bifurcation manifold

- Property modhopfManiHandle

  function handle for modified hopf bifurcation manifold
- Property foldNVHandle

  function handle for fold bifurcation normal vector system
- Property modfoldNVHandle

  function handle for modified fold bifurcation normal vector system
- Property hopfNVHandle

  function handle for hopf bifurcation normal vector system
- Property modhopfNVHandle

  function handle for modified hopf bifurcation normal vector system

Protected Attributes

- Property rhs

  function handle for right-hand-side of DDE
- Property delays

  function handle of function for delays
- Property uncParam

  uncertain parameters: object of class VariableVector
- Property certOptParam

  certain optimization parameters: object of class VariableVector
- Property ntau

  number of delays

6.1.1 Detailed Description

The instances of this class contain information for a DDE used in DDENLP.

Those instances collect the relevant data of the DDE describing the system to be optimized

6.1.2 Constructor & Destructor Documentation

6.1.2.1 function DDE (in *rhsHandle*, in *delays*, in *xNomGuess*, in *uncParam*, in *certOptParam*)

Class constructor.

This function constructs instances of the class DDE

Parameters

rhsHandle	function handle for right hand side of DDE
delays	function handle for delays
xNomGuess	Guess for a nominal steady state vector of class VariableVector
uncParam	uncertain nominal paraemeter vector of class VariableVector
certOptParam	parameter vector with certain optimization variables of class VariableVector

Returns

instance of the DDE class.

6.1.3 Member Function Documentation

6.1.3.1 function getHandles (in *aDDE,* in *type*)

hands back function handles of a DDE instance

Those function handles come in handy for the formulation of normal vector constraints

Parameters

aDDE	function handle for right hand side of DDE
type	function handle for delays

Returns

handleMani function handle with function g(.)=0 describing a critical manifold
handleNV function handle with function h(.)=0 describing normal vectors on critical manifold

6.1.4 Member Data Documentation

6.1.4.1 Property certOptParam [protected]

certain optimization parameters: object of class VariableVector

6.1.4.2 Property delays [protected]

function handle of function for delays

6.1.4.3 Property foldManiHandle

function handle for fold bifurcation manifold

6.1.4.4 Property foldNVHandle

function handle for fold bifurcation normal vector system

6.1.4.5 Property hopfManiHandle

function handle for hopf bifurcation manifold

6.1.4.6 Property hopfNVHandle

function handle for hopf bifurcation normal vector system

6.1.4.7 Property modfoldManiHandle

function handle for modified fold bifurcation manifold

6.1.4.8 Property modfoldNVHandle

function handle for modified fold bifurcation normal vector system

6.1.4.9 Property modhopfManiHandle

function handle for modified hopf bifurcation manifold

6.1.4.10 Property modhopfNVHandle

function handle for modified hopf bifurcation normal vector system

6.1.4.11 Property ntau [protected]

number of delays

6.1.4.12 Property rhs [protected]

function handle for right-hand-side of DDE

6.1.4.13 Property uncParam [protected]

uncertain parameters: object of class VariableVector

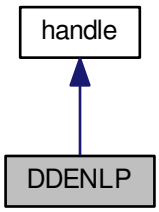
The documentation for this class was generated from the following file:

- @DDE/DDE.m

6.2 DDENLP Class Reference

this class desribes the whole optimization problem for delayed systems

Inheritance diagram for DDENLP:



Public Member Functions

- function DDENLP (in aCostFunction, in aDDE, in xNomGuess, in stateLB, in stateUB, in uncertParamLB, in uncertParamUB, in certOptParamLB, in certOptParamUB, in varargin)

  Class constructor.
- function initializeStSt (in aDDENLP)

  intialize steady state constraint within a DDENLP instance
- function initializeStStRot (in aDDENLP)

  intialize rotating steady state constraint within a DDENLP instance
- function addNVCon (in aDDENLP, in type, in augSysHandle, in nVSysHandle, in xGuess, in alphaGuess, in pGuess, in varargin)

  add a normal vector constraint to the optimization Problem
- function initNVCons (in aDDENLP, in nVflipmode, in varargin)

  initializes normal vector constraints by using a series of methods of the class NVConstraint
- function moveAwayFromManifolds (in aDDENLP, in steppingFactor, in distanceFactor, in)

  moves the nominal point away from known critical manifolds methods of the class NVConstraint
- function evaluateStatus (in aDDENLP)

  evaluates the status of DDENLP
- function concatConstraints (in aDDENLP, in otherEq, in otherIneq, in varargin)

  concatenates the various constraints in this DDENLP instance for later optimitzation
- function compareConnectionNV (in aDDENLP, in indexCrit)

  compares connection and normal vector
- function concatInitPoints (in aDDENLP)

  concatenates the various variables in this DDENLP instance for later optimitzation
- function checkConstraints (in aDDENLP)

  check if constraints hold for inital point
- function runOptim (in aDDENLP, in userDefinedOptions, in Aineq, in bineq, in varargin)

  run optimization of DDENLP instance
- function runOptimMultipleInitPoints (in aDDENLP, in furtherInput, in varargin)

  run optimization of DDENLP instance with many different initial points
- function runOptimWithStabChecks (in aDDENLP, in nIterBetweenStabChecks)

run optimization of DDENLP instance with intermediate stability checks.

- function runOptimAddingNewManifolds (in aDDENLP, in nIterBetweenStabChecks)

  run optimization of DDENLP instance with intermediate stability checks.
- function findManifoldPointOnLine (in ignoredArg, in type, in manifoldHandle, in point1, in point2, in maxEig)

  look for a manifold point between two given points
- function deconstructInit (in aDDENLP, in type, in varargin)

  reconstruct inital variables from single vector
- function deconstructOptimum (in aDDENLP)

  reconstruct optimal variables from single vector
- function checkStabilityPoint (in aDDENLP, in point)

  calculate eigenvalues at a given point in parameter space
- function checkStabilityAtVertices (in aDDENLP, in indexlist, in varargin)

  calculate stability an vertices of uncertianty region
- function checkStabilityAtRandom (in aDDENLP, in baseForNumberOfPoints, in seedForRandomNumbers, in varargin)

  calculate stability at random points within uncertianty region
- function ddesd (in aDDENLP, in point, in history, in tspan, in options)

  run a simulation of the optimum.

Public Attributes

- Property aCostFunction

  cost function of the main steady state optimization problem
- Property problemDDE

  object of class DDE, contains all relevant DDE information
- Property vars

  structure containing the cell array of variables, find entries with ind=find(ismember...
- Property algVarIndex

  indexes algebraic variables
- Property stStCon

  steady state constraint class StStConstraint
- Property lowerBoxCons

  lower boundaries of box constraints
- Property upperBoxCons

  lower boundaries of box constraints
- Property minDist

  parameter uncertainty
- Property maxAllowedRealPart

  the highest eigenvalue real part which is allowed
- Property allowedEigsInClosedRightHP

  accepted eigenvalues in the closed right halfplane (for eigenvalues on the imaginary axis)
- Property NVCon

  normal vector constraints, vector of objects of class NVConstraint
- Property allNLEqConstraints

  function handle that collects all nonlinear equality constraints
- Property allNLIneqConstraints

  function handle that collects all nonlinear inequality constraints
- Property fixedUncertParamIndex

  contains the indices of parameters that are uncertain, but fixed
- Property nlcon

function handle with the nonlinear constraints (inequality and equality)

- Property optionsInitEqCons

  options for initialization numerics (fsolve)

- Property optionsInitOptim

  options for initialization numerics (auxiliary optimization)

- Property optionsMainOptim

  options for the main optimization

- Property numMinEig

  real number, for DDE-BIFTOOL numerics

- Property verifyStabPoints

  $$ {verifyStabPoints}^{nAlpha} $$ random points are generated and evaluated

- Property useLHS

  flag for using latin hypercube sample for evaluating the uncertainty region

Protected Attributes

- Property nX

  number of states

- Property nAlpha

  number of uncertain parameters

- Property nP

  number of optimization parameters without uncertainty

- Property occupiedVars

  number of variables in whole optimization problem

- Property occupiedEqs

  number of equality constraints in whole optimization problem

- Property occupiedIneqs

  number of inequality constraints in whole optimization problem

- Property initVal

  initial value, length(initVal)==occupiedVars

- Property optimVal

  optimal value, should be a vector of length(optimVal)==occupiedVars

- Property optJ

  cost at optimum

- Property exitflag

  exitflag of main optimization

- Property optimOutput

  status message of main optimization

- Property lambda

  lagrange multiplier of main optimization

- Property grad

  gradient of cost function of main optimization

- Property hessian

  hessian of lagrange function of main optimization

- Property status

  status of this optimization problem, (key in main description)

6.2.1 Detailed Description

this class desribes the whole optimization problem for delayed systems

this class is intended to for the robust stable steady state of delayed systems

status: 1: input initial guess and might not solve the equations 2: parameter is on critical manifold 3: parameter is closest critical point 4: Normal Vector at closest critical point was found 5: connection of closest critical point and nominal point by normal vector verified 6: all nonlinear constraints have been concatenated >6: values were handed back by optimization algorithm

Author

Jonas Otten

Date

18 Jul 2017

6.2.2 Constructor & Destructor Documentation

6.2.2.1 function DDENLP (in *aCostFunction,* in *aDDE,* in *xNomGuess,* in *stateLB,* in *stateUB,* in *uncertParamLB,* in *uncertParamUB,* in *certOptParamLB,* in *certOptParamUB,* in *varargin*)

Class constructor.

This function constructs instances of the class DDENLP

Parameters

aCostFunction	function handle for cost function
aDDE	underlying DDE, instance of class DDE
xNomGuess	collection of instances of VariableVector, nominal state guess
stateLB	lower bounds for states
stateUB	upper bounds for states
uncertParamLB	lower bounds for uncertain optimization parameters
uncertParamUB	upper bounds for uncertain optimization parameters
certOptParamLB	lower bounds for certain optimization parameters (optional input)
certOptParamUB	upper bounds for certain optimization parameters (optional input)

Returns

instance of the DDENLP class.

6.2.3 Member Function Documentation

6.2.3.1 function addNVCon (in *aDDENLP,* in *type,* in *augSysHandle,* in *nVSysHandle,* in *xGuess,* in *alphaGuess,* in *pGuess,* in *varargin*)

add a normal vector constraint to the optimization Problem

Parameters

aDDENLP	instance of DDENLP
type	a string containing the requested manifold type
augSynVSysHandlesHandle	function handle describing the critical manifold
nVSysHandle	function handle describing the normal vectors of the critical manifold
xGuess	guess for critical state, instance of VariableVector
alphaGuess	guess for critical parameters, instance of VariableVector
pGuess	current certain parameters, instance of VariableVector

6.2.3.2 function checkConstraints (in *aDDENLP*)

check if constraints hold for inital point

Parameters

aDDENLP	instance of DDENLP

6.2.3.3 function checkStabilityAtRandom (in *aDDENLP,* in *baseForNumberOfPoints,* in *seedForRandomNumbers,* in *varargin*)

calculate stability at random points within uncertianty region

Parameters

aDDENLP	instance of DDENLP class
baseForNumberOfPoints	base for the number of random points
seedForRandomNumbers	(optional) seed for pseudo random number generator

Returns

maxRealPart maximal real part of all eigenvalues

6.2.3.4 function checkStabilityAtVertices (in *aDDENLP,* in *indexlist,* in *varargin*)

calculate stability an vertices of uncertianty region

Parameters

aDDENLP	instance of DDENLP class indexlist index list for selective stability calculation

Returns

maxRealPart maximal real part of all eigenvalues

6.2.3.5 function checkStabilityPoint (in *aDDENLP,* in *point*)

calculate eigenvalues at a given point in parameter space

Parameters

aDDENLP	instance of DDENLP class
point	string desribing at which point type the stability shall be evaluated

Returns

maxRealPart maximal real part of all eigenvalues
eigs some rightmost eigenvalues

6.2.3.6 function compareConnectionNV (in *aDDENLP,* in *indexCrit*)

compares connection and normal vector

Parameters

aDDENLP	instance of DDENLP
indexCrit	index of critical point for which comparison takes place

6.2.3.7 function concatConstraints (in *aDDENLP,* in *otherEq,* in *otherIneq,* in *varargin*)

concatenates the various constraints in this DDENLP instance for later optimitzation

Parameters

aDDENLP	instance of DDENLP
otherEq	additional equality constraints
otherIneq	additional inequality constraints

6.2.3.8 function concatInitPoints (in *aDDENLP*)

concatenates the various variables in this DDENLP instance for later optimitzation

Parameters

aDDENLP	instance of DDENLP

6.2.3.9 function ddesd (in *aDDENLP,* in *point,* in *history,* in *tspan,* in *options*)

run a simulation of the optimum.

Overloading of ddesd

Parameters

aDDENLP	instance of DDENLP class
point	parameters to use for simulation
history	for simulation (like inital point ODE, but for DDE)
tspan	time spaned by simulation
options	options for solver

Returns

sol is solution struct as known from ode45 etc

6.2.3.10 function deconstructInit (in *aDDENLP,* in *type,* in *varargin*)

reconstruct inital variables from single vector

Parameters

aDDENLP	instance of DDENLP class
type	(optional) string describing which points to reconstruct

Returns

(optional) return reconstruction to external variable

6.2.3.11 function deconstructOptimum (in *aDDENLP*)

reconstruct optimal variables from single vector

Parameters

aDDENLP	instance of DDENLP class

Returns

(optional) return reconstruction to external variable

6.2.3.12 function evaluateStatus (in *aDDENLP*)

evaluates the status of DDENLP

Parameters

aDDENLP	instance of DDENLP

6.2.3.13 function findManifoldPointOnLine (in *ignoredArg,* in *type,* in *manifoldHandle,* in *point1,* in *point2,* in *maxEig*)

look for a manifold point between two given points

Parameters

~	(ignore first input)
type	string of expected manifold type
manifoldHandle	function handle of manifold
point1	first point
point2	second point

Returns

intermediate Point

6.2.3.14 function initializeStSt (in *aDDENLP*)

intialize steady state constraint within a DDENLP instance

Parameters

aDDENLP	instance of DDENLP

6.2.3.15 function initializeStStRot (in *aDDENLP*)

intialize rotating steady state constraint within a DDENLP instance

Parameters

aDDENLP	instance of DDENLP

6.2.3.16 function initNVCons (in *aDDENLP,* in *nVflipmode,* in *varargin*)

initializes normal vector constraints by using a series of methods of the class NVConstraint

Parameters

aDDENLP	instance of DDENLP

6.2.3.17 function moveAwayFromManifolds (in *aDDENLP,* in *steppingFactor,* in *distanceFactor,* in)

moves the nominal point away from known critical manifolds methods of the class NVConstraint

Parameters

aDDENLP	instance of DDENLP
steppingFactor	(optional) step width of moving nominal point. Smaller value means slower convergence but better numerics, default is 0.7
distanceFactor	(optional) desired final minimal distance
iterations	(optional) number of iterations. Default is 20

6.2.3.18 function runOptim (in *aDDENLP,* in *userDefinedOptions,* in *Aineq,* in *bineq,* in *varargin*)

run optimization of DDENLP instance

Parameters

aDDENLP	instance of DDENLP
userDefinedOptions	(optional) user defined optimization options
Aineq	(optional) user defined linear inequality constraints
bineq	(optional) user defined linear inequality constraints

6.2.3.19 function runOptimAddingNewManifolds (in *aDDENLP,* in *nIterBetweenStabChecks*)

run optimization of DDENLP instance with intermediate stability checks.

Algorithm tries to add new critical manifold if one was crossed

Parameters

aDDENLP	instance of DDENLP
nIterBetweenStabChecks	number of iterations between stability checks and potential manifold adding

6.2.3.20 function runOptimMultipleInitPoints (in *aDDENLP,* in *furtherInput,* in *varargin*)

run optimization of DDENLP instance with many different initial points

Parameters

aDDENLP	instance of DDENLP
furtherInput	(optional) set of initial points to test

6.2.3.21 function runOptimWithStabChecks (in *aDDENLP,* in *nIterBetweenStabChecks*)

run optimization of DDENLP instance with intermediate stability checks.

Less iterations between stability checks lead to slower optimization, because optimizer has to cold start more frequently

Parameters

aDDENLP	instance of DDENLP
nIterBetweenStabChecks	number of iterations between stability checks

Returns

init initial point of optimization
final final point of optimization
maxEig biggest eigenvalue real part after optimization
eigs rightmost eigenvalues after stability loss

6.2.4 Member Data Documentation

6.2.4.1 Property aCostFunction

cost function of the main steady state optimization problem

6.2.4.2 Property algVarIndex

indexes algebraic variables

6.2.4.3 Property allNLEqConstraints

function handle that collects all nonlinear equality constraints

6.2.4.4 Property allNLIneqConstraints

function handle that collects all nonlinear inequality constraints

6.2.4.5 Property allowedEigsInClosedRightHP

accepted eigenvalues in the closed right halfplane (for eigenvalues on the imaginary axis)

6.2.4.6 Property exitflag [protected]

exitflag of main optimization

6.2.4.7 Property fixedUncertParamIndex

contains the indices of parameters that are uncertain, but fixed

6.2.4.8 Property grad [protected]

gradient of cost function of main optimization

6.2.4.9 Property hessian [protected]

hessian of lagrange function of main optimization

6.2.4.10 Property initVal [protected]

initial value, length(initVal)==occupiedVars

6.2.4.11 Property lambda [protected]

lagrange multiplier of main optimization

6.2.4.12 Property lowerBoxCons

lower boundaries of box constraints

6.2.4.13 Property maxAllowedRealPart

the highest eigenvalue real part which is allowed

6.2.4.14 Property minDist

parameter uncertainty

6.2.4.15 Property nAlpha [protected]

number of uncertain parameters

6.2.4.16 Property nlcon

function handle with the nonlinear constraints (inequality and equality)

6.2.4.17 Property nP [protected]

number of optimization parameters without uncertainty

6.2.4.18 Property numMinEig

real number, for DDE-BIFTOOL numerics

6.2.4.19 Property NVCon

normal vector constraints, vector of objects of class NVConstraint

6.2.4.20 Property nX [protected]

number of states

6.2.4.21 Property occupiedEqs [protected]

number of equality constraints in whole optimization problem

6.2.4.22 Property occupiedIneqs [protected]

number of inequality constraints in whole optimization problem

6.2.4.23 Property occupiedVars [protected]

number of variables in whole optimization problem

6.2.4.24 Property optimOutput [protected]

status message of main optimization

6.2.4.25 Property optimVal [protected]

optimal value, should be a vector of length(optimVal)==occupiedVars

6.2.4.26 Property optionsInitEqCons

options for initialization numerics (fsolve)

6.2.4.27 Property optionsInitOptim

options for initialization numerics (auxiliary optimization)

6.2.4.28 Property optionsMainOptim

options for the main optimization

6.2.4.29 Property optJ [protected]

cost at optimum

6.2.4.30 Property problemDDE

object of class DDE, contains all relevant DDE information

6.2.4.31 Property status [protected]

status of this optimization problem, (key in main description)

6.2.4.32 Property stStCon

steady state constraint class StStConstraint

6.2.4.33 Property upperBoxCons

lower boundaries of box constraints

6.2.4.34 Property useLHS

flag for using latin hypercube sample for evaluating the uncertainty region

6.2.4.35 Property vars

structure containing the cell array of variables, find entries with ind=find(ismember...

6.2.4.36 Property verifyStabPoints

$$ {verifyStabPoints}^{nAlpha} $$ random points are generated and evaluated

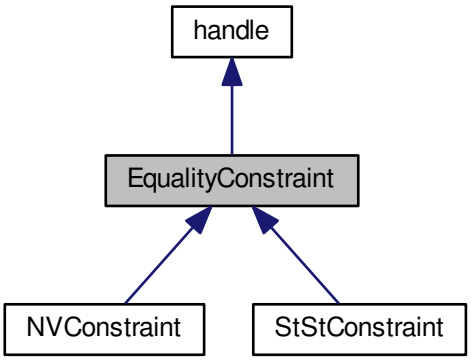
The documentation for this class was generated from the following file:

- @DDENLP/DDENLP.m

6.3 EqualityConstraint Class Reference

(abstract) class that inherits its properities to equality constraints

Inheritance diagram for EqualityConstraint:



Public Member Functions

- function EqualityConstraint (in conFunHandle, in nEqs, in vars, in eqOffset)

  Class constructor.
- function shiftIndex (in anEqCon, in eqShift, in varShift)

  shift the index of the equations stored in this equality constraint

Public Attributes

- Property vars

  collection of variables of class VariableVector on which the instance of EqualityConstraint depends on

Protected Attributes

- Property conFun

  function handle representin the constraint function
- Property eqIndex

  indices of the equation in the superordinate constraint optimization problem
- Property nEqs

  number of equation in this instance of EqualityConstraint
- Property status

  initialization status of this instance of EqualityConstraint

6.3.1 Detailed Description

(abstract) class that inherits its properities to equality constraints

The properties are necessary to manage mutliple classes of equality constraints

6.3.2 Constructor & Destructor Documentation

6.3.2.1 function EqualityConstraint (in *conFunHandle,* in *nEqs,* in *vars,* in *eqOffset*)

Class constructor.

This function constructs instances of the class EqualityConstraint

Parameters

conFunHandle	function handle for equality constraint
nEqs	number of equations in this equality constraint
vars	collection of instances of VariableVector
eqOffset	offset for equation indices

Returns

instance of the EqualityConstraint class.

6.3.3 Member Function Documentation

6.3.3.1 function shiftIndex (in *anEqCon,* in *eqShift,* in *varShift*)

shift the index of the equations stored in this equality constraint

Parameters

anEqCon	Instance of EqualityConstraint, where the variable equation will be shifted
eqShift	how far the index of equations will be shifted
varShift	how far the index of variables will be shifted

Returns

instance of the EqualityConstraint class.

6.3.4 Member Data Documentation

6.3.4.1 Property conFun [protected]

function handle representin the constraint function

6.3.4.2 Property eqIndex [protected]

indices of the equation in the superordinate constraint optimization problem

6.3.4.3 Property nEqs [protected]

number of equation in this instance of EqualityConstraint

6.3.4.4 Property status [protected]

initialization status of this instance of EqualityConstraint

6.3.4.5 Property vars

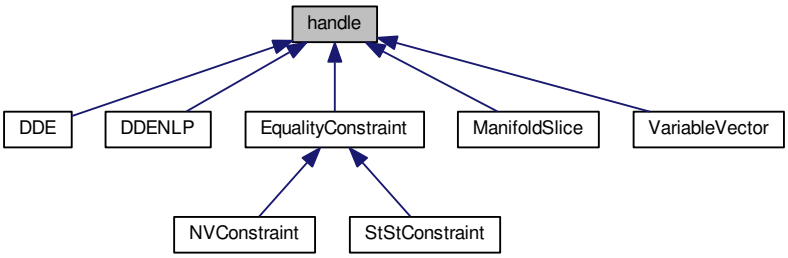
collection of variables of class VariableVector on which the instance of EqualityConstraint depends on

The documentation for this class was generated from the following file:

- @EqualityConstraint/EqualityConstraint.m

6.4 handle Class Reference

Inheritance diagram for handle:



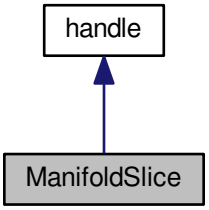
The documentation for this class was generated from the following file:

- @VariableVector/VariableVector.m

6.5 ManifoldSlice Class Reference

The instances of this class generate visualization data.

Inheritance diagram for ManifoldSlice:



Public Member Functions

- function ManifoldSlice (in aNVCon, in contin ParamsInd, in varargin)
  Class constructor.
- function maniContin2DbothDirections (in aManifoldSlice, in n)
  runs numerical continuation in both directions
- function manifoldContinuation2D (in aManifoldSlice, in direction)
  runs numerical continuation in one directions
- function plot (in aManifoldSlice)
  plots results of numerical continuation of critical manifolds
- function quiver (in aManifoldSlice)

Public Attributes

- Property freeParamIndices
  indeces for free parameters for numerical continuation
- Property nManiPoints
  number of steps to take during numerical continuation
- Property initStepLength
  initial step length
- Property stepLength
  step length during numerical continuation
- Property point
  the manifold points resulting form numerical continuation
- Property eqAugSys
  equations defining the manifold
- Property eqNVSys
  normal vector system of manifold
- Property maxStepLength
  bounds step length

- Property lowerBoxCons

  lower bounds to stop numerical continuation
- Property upperBoxCons

  upper bounds to stop numerical continuation
- Property showStepsFlag

  each step is displayed in a figure, if this is set to 1
- Property debugFlag

  enables warnings for debugging if set to 1

6.5.1 Detailed Description

The instances of this class generate visualization data.

Author

Jonas Otten

Date

18 Jul 2017 The instances of this class generate visualization data

This class simplyfies visualization of critical manifolds. Two dimensional numerical continuation calculates "slices" of the manifolds, which can be plotted using the methods of this class.

this class is allows easy visualization of optimization results by implementing a quasi-arclength numerical continuation of critical manifolds.

Author

Jonas Otten

Date

18 Jul 2017

6.5.2 Constructor & Destructor Documentation

6.5.2.1 function ManifoldSlice (in *aNVCon,* in *continParamsInd,* in *varargin*)

Class constructor.

This function constructs instances of the class ManifoldSlice

Parameters

aNVCon	an instance of the class NVConstraint. Its critical manifold will be visualized
continParamsInd	the index of the parameters used for continuation
varargin	enables to call constructor ommitting continParamsInd

Returns

instance of the ManifoldSlice class.

6.5.3 Member Function Documentation

6.5.3.1 function maniContin2DbothDirections (in *aManifoldSlice*, in *n*)

runs numerical continuation in both directions

Parameters

aManifoldSlice	instance of this class
n	number of steps to take during continuation

6.5.3.2 function manifoldContinuation2D (in *aManifoldSlice*, in *direction*)

runs numerical continuation in one directions

Parameters

aManifoldSlice	instance of this class
direction	determines direction for continuation

6.5.3.3 function plot (in *aManifoldSlice*)

plots results of numerical continuation of critical manifolds

Parameters

aManifoldSlice	instance of this class

Returns

handle vector of plot handles allowing later manipulation of plots

6.5.3.4 function quiver (in *aManifoldSlice*)

6.5.4 Member Data Documentation

6.5.4.1 Property debugFlag

enables warnings for debugging if set to 1

6.5.4.2 Property eqAugSys

equations defining the manifold

6.5.4.3 Property eqNVSys

normal vector system of manifold

6.5.4.4 Property freeParamIndices

indeces for free parameters for numerical continuation

6.5.4.5 Property initStepLength

initial step length

6.5.4.6 Property lowerBoxCons

lower bounds to stop numerical continuation

6.5.4.7 Property maxStepLength

bounds step length

6.5.4.8 Property nManiPoints

number of steps to take during numerical continuation

6.5.4.9 Property point

the manifold points resulting form numerical continuation

6.5.4.10 Property showStepsFlag

each step is displayed in a figure, if this is set to 1

6.5.4.11 Property stepLength

step length during numerical continuation

6.5.4.12 Property upperBoxCons

upper bounds to stop numerical continuation

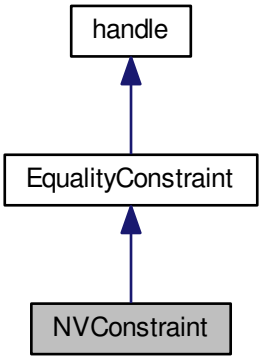
The documentation for this class was generated from the following file:

- @ManifoldSlice/ManifoldSlice.m

6.6 NVConstraint Class Reference

the instances of this class are objects representing normal vector constraints.

Inheritance diagram for NVConstraint:



Public Member Functions

- function NVConstraint (in aDDENLP, in type, in augSysHandle, in nVSysHandle, in nVvars)
  Class constructor.
- function prepareInitialGuess (in aNVCon, in aVarCollection)
  rotates complex eigenvector to make real and imaginary part orthogonal cf.
- function findManifoldPoint (in aNVCon, in aVarCollection)
  find a point on the critical manifold
- function findEigVector (in aNVCon, in aVarCollection)
  find an eigenvector of a critical manifold point candidate
- function findClosestCriticalPoint (in aNVCon, in alphaNom)
  find closest critical point
- function findNormalVector (in aNVCon, in alphaNom, in directionMode, in varargin)
  find normal vector at given closest critical point
- function findConnection (in aNVCon, in alphaNom)
  initialize connection constraint
- function checkSolution (in aNVCon)
  checks if a solution fits the requested manifold type
- function shiftIndex (in anEqCon, in eqShift, in varShift)
  shift the index of the equations stored in this equality constraint

Public Attributes

- Property type
  critical manifold type, string
- Property eqAugSys
  augmented system equations (manifold), a EqualityConstraint object
- Property eqNVSys
  normal vector system equations, a EqualityConstraint object
- Property eqConnect
  connection constraints, a EqualityConstraint object
- Property nVarAugSys
  number of variables in augemented system
- Property nVarNVSys
  number of variables in normal vector system
- Property inequalities
  function handle of inequalities
- Property optionsEqConsInit
  options for numerical solver
- Property optionsInitOptim
  options for auxiliary optimization
- Property vars
  collection of variables of class VariableVector on which the instance of EqualityConstraint depends on

Protected Attributes

- Property inequalityIndex
  index of the inequality constraint of this NVConstraint within all constaints
- Property problemDDE
  differential equation for calculation of stability
- Property numMinEig
  numerical constant for the ODE approximation of the DDE
- Property conFun
  function handle representin the constraint function
- Property eqIndex
  indices of the equation in the superordinate constraint optimization problem
- Property nEqs
  number of equation in this instance of EqualityConstraint
- Property status
  initialization status of this instance of EqualityConstraint

6.6.1 Detailed Description

the instances of this class are objects representing normal vector constraints.

The procedures initialize those constraints

6.6.2 Constructor & Destructor Documentation

6.6.2.1 function NVConstraint (in *aDDENLP,* in *type,* in *augSysHandle,* in *nVSysHandle,* in *nVvars*)

Class constructor.

This function constructs instances of the class NVConstraint

Parameters

aDDENLP	superordinate optimization problem
type	type of critical manifold (string)
augSysHandle	function handle to critical manifold
nVSysHandle	function handle to normal vector system
nVvars	collection of instances of VariableVector

Returns

instance of the NVConstraint class.

6.6.3 Member Function Documentation

6.6.3.1 function checkSolution (in *aNVCon*)

checks if a solution fits the requested manifold type

Parameters

aNVCon	instance of NVConstraint

Returns

6.6.3.2 function findClosestCriticalPoint (in *aNVCon,* in *alphaNom*)

find closest critical point

Parameters

aNVCon	instance of NVConstraint with a know point on the critical manifold
alphaNom	nominal point stored in an instance of VariableVector

Returns

instance of NVConstraint with potentially known closest critical point

6.6.3.3 function findConnection (in *aNVCon,* in *alphaNom*)

initialize connection constraint

Parameters

aNVCon	instance of NVConstraint with everything but connection constraint initialized
alphaNom	nominal point stored in an instance of VariableVector

Returns

instance of NVConstraint with potentially initialized connection constraint

6.6.3.4 function findEigVector (in *aNVCon,* in *aVarCollection*)

find an eigenvector of a critical manifold point candidate

Parameters

aNVCon	instance of NVConstraint that will initialized
aVarCollection	collection of instances of VariableVector containing numerical values for initial guess

Returns

instance of NVConstraint with potentially known

6.6.3.5 function findManifoldPoint (in *aNVCon,* in *aVarCollection*)

find a point on the critical manifold

Parameters

aNVCon	instance of NVConstraint that will initialized
aVarCollection	collection of instances of VariableVector containing numerical values for initial guess

Returns

instance of NVConstraint with potentially known critical point

6.6.3.6 function findNormalVector (in *aNVCon,* in *alphaNom,* in *directionMode,* in *varargin*)

find normal vector at given closest critical point

Parameters

aNVCon	instance of NVConstraint with known closest critical point
alphaNom	nominal point stored in an instance of VariableVector
directionMode	optional input to manipulate orientation of
varargin	

Returns

instance of NVConstraint with normal vectors etc. found

6.6.3.7 function prepareInitialGuess (in *aNVCon,* in *aVarCollection*)

rotates complex eigenvector to make real and imaginary part orthogonal cf.

Proof of Lemma 1 in [https://doi.org/10.1109/CDC.2016.7798469]

Parameters

aNVCon	instance of NVConstraint that will initialized
aVarCollection	collection of instances of VariableVector containing numerical values for initial guess

Returns

instance of NVConstraint with potentially known critical point
collection of instances of VariableVector with orthogonalized real part and imaginary part of eigenvector

6.6.3.8 function shiftIndex (in *anEqCon,* in *eqShift,* in *varShift*) [inherited]

shift the index of the equations stored in this equality constraint

Parameters

anEqCon	Instance of EqualityConstraint, where the variable equation will be shifted
eqShift	how far the index of equations will be shifted
varShift	how far the index of variables will be shifted

Returns

instance of the EqualityConstraint class.

6.6.4 Member Data Documentation

6.6.4.1 Property conFun [protected],[inherited]

function handle representin the constraint function

6.6.4.2 Property eqAugSys

augmented system equations (manifold), a EqualityConstraint object

6.6.4.3 Property eqConnect

connection constraints, a EqualityConstraint object

6.6.4.4 Property eqIndex [protected],[inherited]

indices of the equation in the superordinate constraint optimization problem

6.6.4.5 Property eqNVSys

normal vector system equations, a EqualityConstraint object

6.6.4.6 Property inequalities

function handle of inequalities

6.6.4.7 Property inequalityIndex [protected]

index of the inequality constraint of this NVConstraint within all constaints

6.6.4.8 Property nEqs [protected],[inherited]

number of equation in this instance of EqualityConstraint

6.6.4.9 Property numMinEig [protected]

numerical constant for the ODE approximation of the DDE

6.6.4.10 Property nVarAugSys

number of variables in augemented system

6.6.4.11 Property nVarNVSys

number of variables in normal vector system

6.6.4.12 Property optionsEqConsInit

options for numerical solver

6.6.4.13 Property optionsInitOptim

options for auxiliary optimization

6.6.4.14 Property problemDDE [protected]

differential equation for calculation of stability

6.6.4.15 Property status [protected], [inherited]

initialization status of this instance of EqualityConstraint

6.6.4.16 Property type

critical manifold type, string

6.6.4.17 Property vars [inherited]

collection of variables of class VariableVector on which the instance of EqualityConstraint depends on

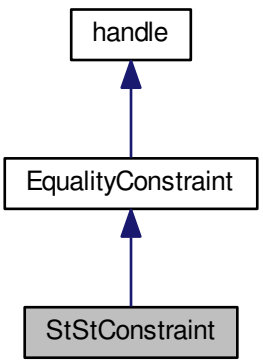
The documentation for this class was generated from the following file:

- @NVConstraint/NVConstraint.m

6.7 StStConstraint Class Reference

This class describes equality constraints representing the steady state condition for the nominal parameters.

Inheritance diagram for StStConstraint:



Public Member Functions

- function StStConstraint (in aDDE, in vars)

  Class constructor.
- function initStStConstraint (in aStStCon, in options)

  look for a steady state as initialization of this StStConstraint instance
- function initStStConstraintRot (in aStStCon, in options)

  look for a steady state as initialization of this StStConstraint instance while allowing rotating coordinates
- function shiftIndex (in anEqCon, in eqShift, in varShift)

  shift the index of the equations stored in this equality constraint

Public Attributes

- Property vars

  collection of variables of class VariableVector on which the instance of EqualityConstraint depends on

Protected Attributes

- Property conFun

  function handle representin the constraint function
- Property eqIndex

  indices of the equation in the superordinate constraint optimization problem
- Property nEqs

  number of equation in this instance of EqualityConstraint
- Property status

  initialization status of this instance of EqualityConstraint

6.7.1 Detailed Description

This class describes equality constraints representing the steady state condition for the nominal parameters.

It also contains a procedure to initialize the constraint

6.7.2 Constructor & Destructor Documentation

6.7.2.1 function StStConstraint (in *aDDE,* in *vars*)

Class constructor.

This function constructs instances of the class StStConstraint

Parameters

aDDE	instance of class DDE
vars	collection of instances of VariableVector (at least state, uncertain parameters and certain parameters)

Returns

instance of the StStConstraint class.

6.7.3 Member Function Documentation

6.7.3.1 function initStStConstraint (in *aStStCon,* in *options*)

look for a steady state as initialization of this StStConstraint instance

Parameters

aStStCon	instance of StStConstraint which will be initialized
options	options for numerical solver fsolve

Returns

initialized instance of the StStConstraint class.

6.7.3.2 function initStStConstraintRot (in *aStStCon,* in *options*)

look for a steady state as initialization of this StStConstraint instance while allowing rotating coordinates

Parameters

aStStCon	instance of StStConstraint which will be initialized
options	options for numerical solver fsolve

Returns

initialized instance of the StStConstraint class.

6.7.3.3 function shiftIndex (in *anEqCon,* in *eqShift,* in *varShift*) [inherited]

shift the index of the equations stored in this equality constraint

Parameters

anEqCon	Instance of EqualityConstraint, where the variable equation will be shifted
eqShift	how far the index of equations will be shifted
varShift	how far the index of variables will be shifted

Returns

instance of the EqualityConstraint class.

6.7.4 Member Data Documentation

6.7.4.1 Property conFun [protected],[inherited]

function handle representin the constraint function

6.7.4.2 Property eqIndex [protected],[inherited]

indices of the equation in the superordinate constraint optimization problem

6.7.4.3 Property nEqs [protected],[inherited]

number of equation in this instance of EqualityConstraint

6.7.4.4 Property status [protected],[inherited]

initialization status of this instance of EqualityConstraint

6.7.4.5 Property vars [inherited]

collection of variables of class VariableVector on which the instance of EqualityConstraint depends on

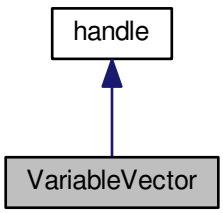
The documentation for this class was generated from the following file:

- @StStConstraint/StStConstraint.m

6.8 VariableVector Class Reference

The instances of this class contain information for a VariableVector used in DDENLP.

Inheritance diagram for VariableVector:



Public Member Functions

- function VariableVector (in values, in offset, in nameInput)

  Class constructor.
- function shiftIndex (in aVariableVec, in shift)

  shift the index of the variables stored in this vector
- function copy (in aVariableVec)

  this method creates a copy of an instance of VariableVector, necessary, because the superclass is the pointerlike handle-class

Public Attributes

- Property values

  column vector containing the numerical values of the Variables

Protected Attributes

- Property names

  column array of strings containing the description of the variables
- Property index

  indexes of variable within superordinate concatenation of variables
- Property nVar

  number of variables

6.8.1 Detailed Description

The instances of this class contain information for a VariableVector used in DDENLP.

Those instances collect human-readable information on the variables (their names and values) as well as the information which is used by other objects to adress those variables.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 function VariableVector (in *values,* in *offset,* in *nameInput*)

Class constructor.

This function constructs instances of the class VariableVector

Parameters

values	Description of first parameter
offset	Description of the second parameter
nameInput	Description of the second parameter

Returns

instance of the VariableVector class.

6.8.3 Member Function Documentation

6.8.3.1 function copy (in *aVariableVec*)

this method creates a copy of an instance of VariableVector, necessary, because the superclass is the pointerlike handle-class

Parameters

instance	of the VariableVector class.

Returns

new instance of the VariableVector class.

6.8.3.2 function shiftIndex (in *aVariableVec,* in *shift*)

shift the index of the variables stored in this vector

Parameters

aVariableVec	Instance of VariableVector, where the variable indices will be shifted
shift	how far the index will be shifted

Returns

instance of the VariableVector class.

6.8.4 Member Data Documentation

6.8.4.1 Property index [protected]

indexes of variable within superordinate concatenation of variables

6.8.4.2 Property names [protected]

column array of strings containing the description of the variables

6.8.4.3 Property nVar [protected]

number of variables

6.8.4.4 Property values

column vector containing the numerical values of the Variables

The documentation for this class was generated from the following file:

- @VariableVector/VariableVector.m

Chapter 7

File Documentation

7.1 @DDE/DDE.m File Reference

The instances of this class contain information for a DDE used in DDENLP.

Classes

- class DDE

  The instances of this class contain information for a DDE used in DDENLP.

7.1.1 Detailed Description

The instances of this class contain information for a DDE used in DDENLP.

Author

Jonas Otten

Date

18 Jul 2017

7.2 @DDENLP/DDENLP.m File Reference

Classes

- class DDENLP

  this class desribes the whole optimization problem for delayed systems

7.3 @EqualityConstraint/EqualityConstraint.m File Reference

(abstract) class that inherits its properities to equality constraints

Classes

- class EqualityConstraint

  (abstract) class that inherits its properities to equality constraints

7.3.1 Detailed Description

(abstract) class that inherits its properities to equality constraints

Author

Jonas Otten

Date

18 Jul 2017

7.4 @ManifoldSlice/ManifoldSlice.m File Reference

Classes

- class ManifoldSlice

  The instances of this class generate visualization data.

7.5 @NVConstraint/NVConstraint.m File Reference

the instances of this class are objects representing normal vector constraints.

Classes

- class NVConstraint

  the instances of this class are objects representing normal vector constraints.

7.5.1 Detailed Description

the instances of this class are objects representing normal vector constraints.

The procedures initialize those constraints

Author

Jonas Otten

Date

18 Jul 2017

7.6 @StStConstraint/StStConstraint.m File Reference

class that inherits its properities from EqualityConstraints.

Classes

- class StStConstraint

  This class describes equality constraints representing the steady state condition for the nominal parameters.

7.6.1 Detailed Description

class that inherits its properities from EqualityConstraints.

It represents steady state constraints.

Author

Jonas Otten

Date

18 Jul 2017

7.7 @VariableVector/VariableVector.m File Reference

The instances of this class contain information for a VariableVector used in DDENLP.

Classes

- class VariableVector

  The instances of this class contain information for a VariableVector used in DDENLP.

7.7.1 Detailed Description

The instances of this class contain information for a VariableVector used in DDENLP.

Author

Jonas Otten

Date

18 Jul 2017

7.8 checkStability.m File Reference

Functions

- function checkStability (in funcs, in parameter, in x0, in numMinEig, in freeParams)

7.8.1 Function Documentation

7.8.1.1 function checkStability (in *funcs,* in *parameter,* in *x0,* in *numMinEig,* in *freeParams*)

7.9 circle.m File Reference

Functions

- function circle (in radius, in x, in y)

7.9.1 Function Documentation

7.9.1.1 function circle (in *radius,* in *x,* in *y*)

7.10 demo/optimPopulationTestManifoldSlice.m File Reference

7.11 demo/populationModelModFoldMani.c File Reference

7.12 ExampleApplications/TDS2016Population/populationModelModFoldMani.c File Reference

7.13 demo/populationModelModFoldNV.c File Reference

7.14 ExampleApplications/TDS2016Population/populationModelModFoldNV.c File Reference

7.15 ExampleApplications/CDC2016SupplyChain/supplyChainDelays.m File Reference

Functions

- function supplyChainDelays (in xx, in alpha, in ignoredArg)

7.15.1 Function Documentation

7.15.1.1 function supplyChainDelays (in *xx,* in *alpha,* in *ignoredArg*)

7.16 ExampleApplications/CDC2016SupplyChain/supplyChainHopfMani.c File Reference

7.17 ExampleApplications/CDC2016SupplyChain/supplyChainHopfNV.c File Reference

7.18 ExampleApplications/CDC2016SupplyChain/supplyChainModel.m File Reference

Functions

- function supplyChainModel (in x, in xtau, in alpha, in ignoredArg)

7.18.1 Function Documentation

7.18.1.1 function supplyChainModel (in *x,* in *xtau,* in *alpha,* in *ignoredArg*)

7.19 ExampleApplications/CDC2016SupplyChain/supplyChainOptim.m File Reference

7.20 ExampleApplications/CDC2016SupplyChain/supplyChainOptimAcceptingTwo↩ EigsInRightHP.m File Reference

7.21 ExampleApplications/CoupledCSTRs/allNCSTROptim.m File Reference

7.22 ExampleApplications/CoupledCSTRs/defineROIWeightingNCSTR.m File Reference

7.23 ExampleApplications/CoupledCSTRs/NCSTRdelays.m File Reference

Functions

- function NCSTRdelays (in k, in ignoredArg, in p)

7.23.1 Function Documentation

7.23.1.1 function NCSTRdelays (in *k,* in *ignoredArg,* in *p*)

7.24 ExampleApplications/CoupledCSTRs/oneCSTRwithFlashSep/oneCSTRoneFlashSepDDE.c File Reference

7.25 ExampleApplications/CoupledCSTRs/oneCSTRwithFlashSep/oneCSTRoneFlashSepHopfMani.c File Reference

7.26 ExampleApplications/CoupledCSTRs/oneCSTRwithFlashSep/oneCSTRoneFlashSepHopfManiNV.c File Reference

7.27 ExampleApplications/CoupledCSTRs/oneCSTRwithFlashSep/oneCSTROptim.m File Reference

Functions

- function dbstack ()

7.27.1 Function Documentation

7.27.1.1 function dbstack ( )

7.28 ExampleApplications/CoupledCSTRs/oneCSTRwithFlashSep/returnOnInvest1.m File Reference

Functions

- function returnOnInvest1 (in x)

7.28.1 Function Documentation

7.28.1.1 function returnOnInvest1 (in *x*)

7.29 ExampleApplications/CoupledCSTRs/reactorCost.m File Reference

Functions

- function reactorCost (in V)

7.29.1 Function Documentation

7.29.1.1 function reactorCost (in *V*)

7.30 ExampleApplications/CoupledCSTRs/setNCSTRBoundaries.m File Reference

7.31 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/returnOnInvest3.m File Reference

Functions

- function returnOnInvest3 (in x)

7.31.1 Function Documentation

7.31.1.1 function returnOnInvest3 (in *x*)

7.32 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTRone↩ FlashSepDDE.c File Reference

7.33 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTRone↩ FlashSepFoldMani.c File Reference

7.34 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTRone↩ FlashSepFoldManiNV.c File Reference

7.35 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTRone↩ FlashSepHopfMani.c File Reference

7.36 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTRone↩ FlashSepHopfManiNV.c File Reference

7.37 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTROptim.m File Reference

Functions

- function dbstack ()

7.37.1 Function Documentation

7.37.1.1 function dbstack ( )

7.38 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/threeCSTRStab.m File Reference

7.39 ExampleApplications/CoupledCSTRs/threeCSTRwithFlashSep/tmp.m File Reference

7.40 ExampleApplications/CoupledCSTRs/twoCSTRwithFlashSep/returnOnInvest2.m File Reference

Functions

- function returnOnInvest2 (in x)

7.40.1 Function Documentation

7.40.1.1 function returnOnInvest2 (in *x*)

7.41 ExampleApplications/CoupledCSTRs/twoCSTRwithFlashSep/twoCSTRoneFlashSepDDE.c File Reference

7.42 ExampleApplications/CoupledCSTRs/twoCSTRwithFlashSep/twoCSTRoneFlashSepHopfMani.c File Reference

7.43 ExampleApplications/CoupledCSTRs/twoCSTRwithFlashSep/twoCSTRoneFlashSepHopfManiNV.c File Reference

7.44 ExampleApplications/CoupledCSTRs/twoCSTRwithFlashSep/twoCSTROptim.m File Reference

Functions

- function dbstack ()

7.44.1 Function Documentation

7.44.1.1 function dbstack ( )

7.45 ExampleApplications/CoupledLasers/10L_symm/DDE_10L_symm.c File Reference

7.46 ExampleApplications/CoupledLasers/10L_symm/Mod_Fold_10L_symm.c File Reference

7.47 ExampleApplications/CoupledLasers/10L_symm/NV_Mod_Fold_10L_symm.c File Reference

7.48 ExampleApplications/CoupledLasers/10L_symm/optim_10L_symm_ausnutzung_↩ symm.m File Reference

7.49 ExampleApplications/CoupledLasers/2L_symm/Laser_DDE_2L.c File Reference

7.50 ExampleApplications/CoupledLasers/2L_symmOldModel/Laser_DDE_2L.c File Reference

7.51 ExampleApplications/CoupledLasers/2L_symmOldParam/Laser_DDE_2L.c File Reference

7.52 ExampleApplications/CoupledLasers/2L_symm/maniPlotter2L.m File Reference

7.53 ExampleApplications/CoupledLasers/2L_symmOldModel/maniPlotter2L.m File Reference

7.54 ExampleApplications/CoupledLasers/2L_symmOldParam/maniPlotter2L.m File Reference

7.55 ExampleApplications/CoupledLasers/2L_symm/Mod_Fold_2L.c File Reference

7.56 ExampleApplications/CoupledLasers/2L_symmOldModel/Mod_Fold_2L.c File Reference

7.95 ExampleApplications/CoupledLasers/3L_asymm/NV_Mod_Hopf_3L_SETUP_1.c File Reference

7.96 ExampleApplications/CoupledLasers/3L_asymmOldParam/NV_Mod_Hopf_3L_SETUP_1.c File Reference

7.97 ExampleApplications/CoupledLasers/3L_asymm/optim_3L_Setup1_find_Start.m File Reference

7.98 ExampleApplications/CoupledLasers/3L_asymmOldParam/optim_3L_Setup1_find_Start.m File Reference

7.99 ExampleApplications/CoupledLasers/3L_asymm/prepareManifoldFig.m File Reference

7.100 ExampleApplications/CoupledLasers/3L_asymmOldParam/prepareManifoldFig.m File Reference

7.101 ExampleApplications/CoupledLasers/3L_asymm/Pump_3L_fixed_K_fsolve_Guete.m File Reference

7.102 ExampleApplications/CoupledLasers/3L_asymmOldParam/Pump_3L_fixed_K_fsolve_Guete.m File Reference

7.103 ExampleApplications/CoupledLasers/3L_asymm/Syncmanifold_3L_Setup1_fkt.m File Reference

Functions

- function Syncmanifold_3L_Setup1_fkt (in var, in pump, in omega)

7.103.1 Function Documentation

7.103.1.1 function Syncmanifold_3L_Setup1_fkt (in *var,* in *pump,* in *omega*)

7.104 ExampleApplications/CoupledLasers/3L_asymmOldParam/Syncmanifold_3L_Setup1_fkt.m File Reference

Functions

- function Syncmanifold_3L_Setup1_fkt (in var, in pump, in omega)

7.104.1 Function Documentation

7.104.1.1 function Syncmanifold_3L_Setup1_fkt (in *var,* in *pump,* in *omega*)

7.105 ExampleApplications/CoupledLasers/3L_asymmOldParam/optim_3L_Setup1_↩ Ausarbeitung_incl_plot.m File Reference

7.106 ExampleApplications/CoupledLasers/3L_symm/Bifurkationen_3L_symm_fsolve↩ _Schnittgeraden.m File Reference

7.107 ExampleApplications/CoupledLasers/3L_symmOldParam/Bifurkationen_3L_↩ symm_fsolve_Schnittgeraden.m File Reference

7.108 ExampleApplications/CoupledLasers/3L_symm/DDE_3L_symm.c File Reference

7.109 ExampleApplications/CoupledLasers/3L_symmOldParam/DDE_3L_symm.c File Reference

7.110 ExampleApplications/CoupledLasers/3L_symm/Mod_Fold_3L_symm.c File Reference

7.111 ExampleApplications/CoupledLasers/3L_symmOldParam/Mod_Fold_3L_symm.c File Reference

7.112 ExampleApplications/CoupledLasers/3L_symm/NV_Mod_Fold_3L_symm.c File Reference

7.113 ExampleApplications/CoupledLasers/3L_symmOldParam/NV_Mod_Fold_3L_↩ symm.c File Reference

7.114 ExampleApplications/CoupledLasers/3L_symm/optim_3L_symm.m File Reference

7.115 ExampleApplications/CoupledLasers/3L_symmOldParam/optim_3L_symm.m File Reference

7.116 ExampleApplications/CoupledLasers/3L_symm/Pump_3L_fsolve_Guete.m File Reference

7.117 ExampleApplications/CoupledLasers/3L_symmOldParam/Pump_3L_fsolve_↩ Guete.m File Reference

7.118 ExampleApplications/CoupledLasers/3L_symm/Syncmanifold_3L_fkt.m File Reference

Functions

- function Syncmanifold_3L_fkt (in var, in pump)

7.118.1 Function Documentation

7.118.1.1 function Syncmanifold_3L_fkt (in *var,* in *pump*)

7.119 ExampleApplications/CoupledLasers/3L_symmOldParam/Syncmanifold_3L_fkt.m File Reference

Functions

- function Syncmanifold_3L_fkt (in var, in pump)

7.119.1 Function Documentation

7.119.1.1 function Syncmanifold_3L_fkt (in *var,* in *pump*)

7.120 ExampleApplications/DiffLaser/CodeGeneration/diff11LaserDDE.c File Reference

7.121 ExampleApplications/DiffLaser/CodeGeneration/diff11LaserDelays.c File Reference

7.122 ExampleApplications/DiffLaser/CodeGeneration/diff11LaserHopfManifold.c File Reference

7.123 ExampleApplications/DiffLaser/CodeGeneration/diff11LaserHopfNV.c File Reference

7.124 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserDDE.c File Reference

7.125 ExampleApplications/DiffLaser/Diff21/Backup/diff21LaserDDE.c File Reference

7.126 ExampleApplications/DiffLaser/Diff21/diff21LaserDDE.c File Reference

7.127 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserDelays.c File Reference

7.128 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserHopfManifold.c File Reference

7.129 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserHopfNV.c File Reference

7.130 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserModFoldManifold.c File Reference

7.131 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserModFoldNV.c File Reference

7.132 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserModHopfManifold.c File Reference

7.133 ExampleApplications/DiffLaser/CodeGeneration/diff21LaserModHopfNV.c File Reference

7.134 ExampleApplications/DiffLaser/CodeGeneration/diff31LaserDDE.c File Reference

7.135 ExampleApplications/DiffLaser/CodeGeneration/diff31LaserDelays.c File Reference

7.136 ExampleApplications/DiffLaser/CodeGeneration/diff31LaserHopfManifold.c File Reference

7.137 ExampleApplications/DiffLaser/CodeGeneration/diff31LaserHopfNV.c File Reference

7.169 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2HopfEig.c File Reference

7.170 ExampleApplications/DiffLaser/Diff21/diff21Laser2HopfEig.c File Reference

7.171 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2HopfManifold.c File Reference

7.172 ExampleApplications/DiffLaser/Diff21/diff21Laser2HopfManifold.c File Reference

7.173 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2HopfNV.c File Reference

7.174 ExampleApplications/DiffLaser/Diff21/diff21Laser2HopfNV.c File Reference

7.175 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2ModFoldManifold.c File Reference

7.176 ExampleApplications/DiffLaser/Diff21/diff21Laser2ModFoldManifold.c File Reference

7.177 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2ModFoldNV.c File Reference

7.178 ExampleApplications/DiffLaser/Diff21/diff21Laser2ModFoldNV.c File Reference

7.179 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2ModHopfManifold.c File Reference

7.180 ExampleApplications/DiffLaser/Diff21/diff21Laser2ModHopfManifold.c File Reference

7.181 ExampleApplications/DiffLaser/Diff21/Backup/diff21Laser2ModHopfNV.c File Reference

7.215 ExampleApplications/DiffLaser/diffLaser61Optim2.m File Reference

7.216 ExampleApplications/DiffLaser/diffLaserCostPlot.m File Reference

7.217 ExampleApplications/DiffLaser/diffLaserCostPlotIntensity.m File Reference

7.218 ExampleApplications/DiffLaser/diffLaserCostPlotIntensityPTheta.m File Reference

7.219 ExampleApplications/DiffLaser/diffLaserNinitNVs.m File Reference

7.220 ExampleApplications/DiffLaser/diffLaserNOptim1.m File Reference

7.221 ExampleApplications/DiffLaser/diffLaserNOptim2.m File Reference

7.222 ExampleApplications/DiffLaser/diffLaserNOptim2WithoutCurrent.m File Reference

7.223 ExampleApplications/DiffLaser/diffLaserNStab.m File Reference

7.224 ExampleApplications/DiffLaser/diffLaserNstabRegions.m File Reference

7.225 ExampleApplications/DiffLaser/diffLaserOptim.m File Reference

7.226 ExampleApplications/DiffLaser/findHopf1.m File Reference

7.227 ExampleApplications/DiffLaser/findHopf41.m File Reference

7.228 ExampleApplications/DiffLaser/findHopfDiffLaserN.m File Reference

7.229 ExampleApplications/DiffLaser/findHopfDiffLaserNAlphaPhi.m File Reference

7.230 ExampleApplications/DiffLaser/findHopfDiffLaserNIntermediate.m File Reference

7.231 ExampleApplications/DiffLaser/findHopfDiffLaserNnoPlot.m File Reference

7.232 ExampleApplications/DiffLaser/findHopfDiffLaserNnoPlotFunction.m File Reference

Functions

- function findHopfDiffLaserNnoPlotFunction (in Ninput, in pCurrentIn, in varargin)

7.232.1 Function Documentation

7.232.1.1 function findHopfDiffLaserNnoPlotFunction (in *Ninput,* in *pCurrentIn,* in *varargin*)

7.233 ExampleApplications/DiffLaser/findHopfDiffLaserNnoPlotFunction2.m File Reference

Functions

- function findHopfDiffLaserNnoPlotFunction2 (in Ninput, in varargin)

7.233.1 Function Documentation

7.233.1.1 function findHopfDiffLaserNnoPlotFunction2 (in *Ninput,* in *varargin*)

7.234 ExampleApplications/DiffLaser/findHopfDiffLaserNnoPlotFunction2b.m File Reference

Functions

- function findHopfDiffLaserNnoPlotFunction2b (in Ninput, in etaIn, in varargin)

7.234.1 Function Documentation

7.234.1.1 function findHopfDiffLaserNnoPlotFunction2b (in *Ninput,* in *etaIn,* in *varargin*)

7.235 ExampleApplications/DiffLaser/findHopfDiffLaserNnoPlotFunction3.m File Reference

Functions

- function findHopfDiffLaserNnoPlotFunction3 (in Ninput, in varargin)

7.235.1 Function Documentation

7.235.1.1 function findHopfDiffLaserNnoPlotFunction3 (in *Ninput,* in *varargin*)

7.236 ExampleApplications/DiffLaser/findHopfDiffLaserNTest.m File Reference

7.237 ExampleApplications/DiffLaser/manifoldPlotter.m File Reference

7.238 ExampleApplications/DiffLaser/manifoldPlotter2.m File Reference

7.239 ExampleApplications/DiffLaser/manifoldPlotter2b.m File Reference

7.240 ExampleApplications/DiffLaser/manifoldPlotter3d.m File Reference

7.241 ExampleApplications/DiffLaser/manifoldPlotter3dSVG.m File Reference

7.242 ExampleApplications/DiffLaser/oneLaser/ddeGatewayPart1.h File Reference

7.243 ExampleApplications/DiffLaser/oneLaser/ddeGatewayPart2.h File Reference

7.244 ExampleApplications/DiffLaser/oneLaser/hopfManiGatewayPart1.h File Reference

7.245 ExampleApplications/DiffLaser/oneLaser/hopfManiGatewayPart2.h File Reference

7.246 ExampleApplications/DiffLaser/oneLaser/hopfNVGatewayPart1.h File Reference

7.247 ExampleApplications/DiffLaser/oneLaser/hopfNVGatewayPart2.h File Reference

7.248 ExampleApplications/DiffLaser/oneLaser/oneLaserDDE.c File Reference

7.249 ExampleApplications/DiffLaser/oneLaser/oneLaserFindHopf.m File Reference

7.250 ExampleApplications/DiffLaser/oneLaser/oneLaserHopfManifold.c File Reference

7.251 ExampleApplications/DiffLaser/oneLaser/oneLaserHopfNV.c File Reference

7.252 ExampleApplications/DiffLaser/oneLaser/oneLaserInitNV.m File Reference

7.253 ExampleApplications/DiffLaser/phaseCondition.m File Reference

Functions

- function phaseCondition (in point)

7.253.1 Function Documentation

7.253.1.1 function phaseCondition (in *point*)

7.254 ExampleApplications/DiffLaser/simLaser.m File Reference

7.255 ExampleApplications/DiffLaser/Test/alt/diff7LaserDDEBackup.c File Reference

7.256 ExampleApplications/DiffLaser/Test/alt/diff7LaserDDEwOmegaBackup.c File Reference

7.257 ExampleApplications/DiffLaser/Test/alt/eigsLaserDiodeBackup.m File Reference

7.258 ExampleApplications/DiffLaser/Test/checkScaling.m File Reference

7.259 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserDDEwOmega.c File Reference

7.260 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserWithOmegaHopf↩ Manifold.c File Reference

7.261 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserWithOmegaHopfNV.c File Reference

7.262 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserWithOmegaModFold↩ Manifold.c File Reference

7.263 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserWithOmegaModFold↩ NV.c File Reference

7.264 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserWithOmegaModHopf↩ Manifold.c File Reference

7.265 ExampleApplications/DiffLaser/Test/diff01Laser/diff1LaserWithOmegaModHopf↩ NV.c File Reference

7.290 ExampleApplications/DiffLaser/Test/diff31Laser/diff31LaserWithOmegaHopfNV.c File Reference

7.291 ExampleApplications/DiffLaser/Test/diff31Laser/Backup/diff31LaserWithOmegaModFoldManifold.c File Reference

7.292 ExampleApplications/DiffLaser/Test/diff31Laser/diff31LaserWithOmegaModFoldManifold.c File Reference

7.293 ExampleApplications/DiffLaser/Test/diff31Laser/Backup/diff31LaserWithOmegaModFoldNV.c File Reference

7.294 ExampleApplications/DiffLaser/Test/diff31Laser/diff31LaserWithOmegaModFoldNV.c File Reference

7.295 ExampleApplications/DiffLaser/Test/diff31Laser/Backup/diff31LaserWithOmegaModHopfManifold.c File Reference

7.296 ExampleApplications/DiffLaser/Test/diff31Laser/diff31LaserWithOmegaModHopfManifold.c File Reference

7.297 ExampleApplications/DiffLaser/Test/diff31Laser/Backup/diff31LaserWithOmegaModHopfNV.c File Reference

7.298 ExampleApplications/DiffLaser/Test/diff31Laser/diff31LaserWithOmegaModHopfNV.c File Reference

7.299 ExampleApplications/DiffLaser/Test/diff41Laser/diff41LaserDDEwOmega.c File Reference

7.300 ExampleApplications/DiffLaser/Test/diff41Laser/diff41LaserWithOmegaHopfManifold.c File Reference

7.301 ExampleApplications/DiffLaser/Test/diff41Laser/diff41LaserWithOmegaHopfNV.c File Reference

7.326 ExampleApplications/DiffLaser/Test/diff71Laser/diff71LaserWithOmegaModHopfNV.c File Reference

7.327 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserDDEwOmega.c File Reference

7.328 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserWithOmegaHopfManifold.c File Reference

7.329 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserWithOmegaHopfNV.c File Reference

7.330 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserWithOmegaModFoldManifold.c File Reference

7.331 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserWithOmegaModFoldNV.c File Reference

7.332 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserWithOmegaModHopfManifold.c File Reference

7.333 ExampleApplications/DiffLaser/Test/diff81Laser/diff81LaserWithOmegaModHopfNV.c File Reference

7.334 ExampleApplications/DiffLaser/Test/eigenvalueZero.m File Reference

7.335 ExampleApplications/DiffLaser/Test/Systemerschliessung/diff7LaserDDEvisual.c File Reference

7.336 ExampleApplications/DiffLaser/Test/Systemerschliessung/efieldCond.m File Reference

Functions

- function efieldCond (in point)

7.336.1 Function Documentation

7.336.1.1 function efieldCond (in *point*)

7.337 ExampleApplications/DiffLaser/Test/Systemerschliessung/eigsLaserDiode.m File Reference

7.338 ExampleApplications/DiffLaser/Test/Systemerschliessung/eigsLaserDiode11.m File Reference

7.339 ExampleApplications/DiffLaser/Test/Systemerschliessung/omegaAEasODE.m File Reference

7.340 ExampleApplications/DiffLaser/Test/Systemerschliessung/simLaserDiodeVisual.m File Reference

7.341 ExampleApplications/DiffLaser/Test/Systemerschliessung/symbolicComplex.m File Reference

7.342 ExampleApplications/DiffLaser/testDiscretization.m File Reference

7.343 ExampleApplications/DiffLaser/testTruncation.m File Reference

7.344 ExampleApplications/DiffLaser/truncManifoldData.m File Reference

Functions

- function truncManifoldData (in data, in lower, in upper)

7.344.1 Function Documentation

7.344.1.1 function truncManifoldData (in *data,* in *lower,* in *upper*)

7.345 ExampleApplications/DiffLaser/uniformlyDiscretizeCurve.m File Reference

Functions

- function uniformlyDiscretizeCurve (in curve, in nPoints)

7.345.1 Function Documentation

7.345.1.1 function uniformlyDiscretizeCurve (in *curve,* in *nPoints*)

7.346 ExampleApplications/DiffLaser/uniformlyDiscretizeCurveFixedStep.m File Reference

Functions

- function uniformlyDiscretizeCurveFixedStep (in curve, in pieceLength)

7.346.1 Function Documentation

7.346.1.1 function uniformlyDiscretizeCurveFixedStep (in *curve,* in *pieceLength*)

7.347 ExampleApplications/DiffLaser/Wochenendskript.m File Reference

7.348 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15DDE.c File Reference

7.349 ExampleApplications/FCC/l15/pcontrolledFCC15DDE.c File Reference

7.350 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15Delays.c File Reference

7.351 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15FoldManifold.c File Reference

7.352 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15FoldNV.c File Reference

7.353 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15HopfManifold.c File Reference

7.354 ExampleApplications/FCC/l15/pcontrolledFCC15HopfManifold.c File Reference

7.355 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15HopfNV.c File Reference

7.356 ExampleApplications/FCC/l15/pcontrolledFCC15HopfNV.c File Reference

7.357 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15ModFoldManifold.c File Reference

7.358 ExampleApplications/FCC/l15/pcontrolledFCC15ModFoldManifold.c File Reference

7.359 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15ModFoldNV.c File Reference

7.360 ExampleApplications/FCC/l15/pcontrolledFCC15ModFoldNV.c File Reference

7.361 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15ModHopfManifold.c File Reference

7.362 ExampleApplications/FCC/l15/pcontrolledFCC15ModHopfManifold.c File Reference

7.363 ExampleApplications/FCC/CodeGeneration/pcontrolledFCC15ModHopfNV.c File Reference

7.364 ExampleApplications/FCC/l15/pcontrolledFCC15ModHopfNV.c File Reference

7.365 ExampleApplications/FCC/l15/compareResults.m File Reference

7.366 ExampleApplications/FCC/l15/convertFold2ModFold.m File Reference

7.367 ExampleApplications/FCC/l15/FCC15HopfNV.c File Reference

7.368 ExampleApplications/FCC/l15/fixedAndFreeAlpha.m File Reference

Functions

- function fixedAndFreeAlpha (in yMaAlpha, in alphaIn, in contInd1, in contInd2)

7.368.1 Function Documentation

7.368.1.1 function fixedAndFreeAlpha (in *yMaAlpha,* in *alphaIn,* in *contInd1,* in *contInd2*)

7.369 ExampleApplications/FCC/l15/JuliaSimulation/importSim2Plot.m File Reference

7.370 ExampleApplications/FCC/l15/noStabOptPControlledFCC15.m File Reference

7.371 ExampleApplications/FCC/l15/noStabOptPControlledFCC15IneqScaled.m File Reference

7.372 ExampleApplications/FCC/l15/pcontrolledFCC15DDE2.c File Reference

7.373 ExampleApplications/FCC/l15/pcontrolledFCC15foldManifold.c File Reference

7.374 ExampleApplications/FCC/l15/pcontrolledFCC15foldManifold2.c File Reference

7.375 ExampleApplications/FCC/l15/pcontrolledFCC15foldNV.c File Reference

7.376 ExampleApplications/FCC/l15/pcontrolledFCC15foldNV2.c File Reference

7.377 ExampleApplications/FCC/l15/pcontrolledFCC15HopfNVprint.c File Reference

7.378 ExampleApplications/FCC/l15/pcontrolledFCC15ModFoldManifold2.c File Reference

7.379 ExampleApplications/FCC/l15/pcontrolledFCC15ModFoldNV2.c File Reference

7.380 ExampleApplications/FCC/l15/pcontrolledFCCContFold.m File Reference

7.381 ExampleApplications/FCC/l15/pcontrolledFCCContFoldFromOpt.m File Reference

7.382 ExampleApplications/FCC/l15/pcontrolledFCCContHopf.m File Reference

7.383 ExampleApplications/FCC/l15/pcontrolledFCCContinuationFindExp.m File Reference

creates variable collections for different manifold types

Functions

- function varCollection (in type, in offset, in x, in alpha, in p, in algVars)

  creates variable collections for different manifold types

7.416.1 Detailed Description

creates variable collections for different manifold types

Author

Jonas Otten

Date

18 Jul 2017

7.416.2 Function Documentation

7.416.2.1 function varCollection (in *type,* in *offset,* in *x,* in *alpha,* in *p,* in *algVars*)

creates variable collections for different manifold types

the structs are allways created the same, but for some manifolds with empty entries

Parameters

type	requested manifold type
offset	index offset
x	state vector, instance of class VariableVector
alpha	uncertain variables, instance of class VariableVector
p	certain optimization variables, instance of class VariableVector

Return values

vars	variables vector to return

Index