#### CAPDDDEs

0.2.1

Generated by Doxygen 1.8.17

1 Nonrigorous, non-autonomous demo	1
2 Exemplary programs	3
3 Namespace Index	5
3.1 Namespace List	5
4 Hierarchical Index	7
4.1 Class Hierarchy	7
5 Class Index	9
5.1 Class List	9
6 Namespace Documentation	11
6.1 capd Namespace Reference	11
6.1.1 Detailed Description	11
7 Class Documentation	13
7.1 capd::ddeshelper::ArgumentParser Class Reference	13
7.1.1 Detailed Description	13
7.1.2 Friends And Related Function Documentation	14
7.1.2.1 operator<<	14
7.2 capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class Template Reference	14
7.2.1 Detailed Description	16
7.2.2 Member Typedef Documentation	17
7.2.2.1 const_iterator	17
7.2.2.2 DelayStorageType	17
7.2.2.3 iterator	17
7.2.3 Constructor & Destructor Documentation	17
7.2.3.1 ∼BasicDiscreteDelaysFunctionalMap()	18
7.2.4 Member Function Documentation	18
7.2.4.1 begin() [1/2]	18
<b>7.2.4.2 begin()</b> [2/2]	18
7.2.4.3 collectComputationData()	18
7.2.4.4 computeDDECoefficients() [1/8]	19
<b>7.2.4.5</b> computeDDECoefficients() [2/8]	19
7.2.4.6 computeDDECoefficients() [3/8]	19
7.2.4.7 computeDDECoefficients() [4/8]	19
<b>7.2.4.8 computeDDECoefficients()</b> [5/8]	20
7.2.4.9 computeDDECoefficients() [6/8]	20
<b>7.2.4.10 computeDDECoefficients()</b> [7/8]	20
7.2.4.11 computeDDECoefficients() [8/8]	21
7.2.4.12 delaysCount()	21
7.2.4.13 dimension()	21

7.2.4.14 end() [1/2]	 . 21
<b>7.2.4.15 end()</b> [2/2]	 . 22
7.2.4.16 imageDimension()	 . 22
7.2.4.17 operator()()	 . 22
$7.3\ capd:: ddes:: Basic Doubleton < Matrix Spec,\ Policies Spec > Class\ Template\ Reference \ .\ .\ .\ .$	 . 22
7.3.1 Detailed Description	 . 26
7.3.2 Constructor & Destructor Documentation	 . 26
7.3.2.1 BasicDoubleton() [1/11]	 . 26
<b>7.3.2.2</b> BasicDoubleton() [2/11]	 . 27
<b>7.3.2.3 BasicDoubleton()</b> [3/11]	 . 27
7.3.2.4 BasicDoubleton() [4/11]	 . 27
<b>7.3.2.5</b> BasicDoubleton() [5/11]	 . 27
7.3.2.6 BasicDoubleton() [6/11]	 . 27
7.3.2.7 BasicDoubleton() [7/11]	 . 28
7.3.2.8 BasicDoubleton() [8/11]	 . 28
<b>7.3.2.9 BasicDoubleton()</b> [9/11]	 . 28
7.3.2.10 BasicDoubleton() [10/11]	 . 28
7.3.2.11 BasicDoubleton() [11/11]	 . 28
$7.3.2.12 \sim Basic Doubleton() \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots$	 . 29
7.3.3 Member Function Documentation	 . 29
7.3.3.1 add() [1/2]	 . 29
<b>7.3.3.2</b> add() [2/2]	 . 29
7.3.3.3 affineTransform()	 . 29
7.3.3.4 common_B()	 . 29
7.3.3.5 common_C()	 . 30
7.3.3.6 common_r()	 . 30
7.3.3.7 common_r0()	 . 30
7.3.3.8 common_x()	 . 30
7.3.3.9 dimension()	 . 31
7.3.3.10 get_B()	 . 31
7.3.3.11 get_C()	 . 31
7.3.3.12 get_r()	 . 31
7.3.3.13 get_r0()	 . 31
7.3.3.14 get_x()	 . 31
7.3.3.15 hull()	 . 31
7.3.3.16 midPoint()	 . 32
7.3.3.17 mul()	 . 32
7.3.3.18 mulThenAdd()	 . 32
7.3.3.19 operator=()	 . 32
7.3.3.20 reinitialize()	 . 32
7.3.3.21 sanityCheck()	 . 32
7.3.3.22 set_B()	 . 33

7.3.3.23 set_C()	33
7.3.3.24 set_Cr0()	33
7.3.3.25 set_r()	33
7.3.3.26 set_r0()	34
7.3.3.27 set_x() [1/3]	34
7.3.3.28 set_x() [2/3]	34
<b>7.3.3.29 set_x()</b> [3/3]	34
7.3.3.30 setupDimension()	34
7.3.3.31 setupFromData()	35
7.3.3.32 setupFromOther() [1/2]	35
7.3.3.33 setupFromOther() [2/2]	35
7.3.3.34 show()	35
7.3.3.35 storageDimension()	35
7.3.3.36 storageN0()	35
7.3.3.37 translate()	36
$7.4\ capd:: ddeshelper:: Binary Data < T > Union\ Template\ Reference\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\$	36
$7.5\ capd:: ddeshelper:: Coordinate Frame < Matrix Spec,\ Vector Spec,\ Scalar Spec > Class\ Template\ Reference = Coordinate Frame < Coordinate$	
ence	36
7.5.1 Detailed Description	37
7.6 capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::CoordinateSystem Class Reference	37
7.6.1 Detailed Description	38
$\textbf{7.7 capd::} \textbf{ddeshelper::} \textbf{CubeSet} < \textbf{VectorSpec} > \textbf{Class Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	38
7.7.1 Member Function Documentation	39
7.7.1.1 insert_cover()	39
7.7.1.2 subcubeset()	40
7.7.2 Friends And Related Function Documentation	40
7.7.2.1 operator<<	40
7.7.2.2 operator>>	40
7.8 capd::ddeshelper::CubeSet< VectorSpec >::CubeSetIterator Class Reference	40
$\textbf{7.9 capd::} \textbf{ddes::} \textbf{CubicIkeda} < \textbf{ScalarSpec}, \textbf{ParamSpec} > \textbf{Class Template Reference} \ \dots \ \dots \ \dots$	41
7.9.1 Detailed Description	42
7.9.2 Member Function Documentation	42
7.9.2.1 dimension()	42
7.9.2.2 getParamsCount()	42
7.9.2.3 imageDimension()	42
$7.10\ capd:: ddes:: DDEBasicFunctional Map < Solution CurveSpec,\ JetSpec > Class\ Template\ Reference\ .$	42
7.10.1 Detailed Description	44
7.10.2 Member Typedef Documentation	44
7.10.2.1 JacobianStorageType	44
7.10.2.2 ValueStorageType	44
7.10.2.3 VariableStorageType	45
7.10.3 Constructor & Destructor Documentation	45

7.10.3.1 ~DDEBasicFunctionalMap()	45
7.10.4 Member Function Documentation	45
7.10.4.1 checkCurveDimension()	45
7.10.4.2 collectComputationData()	46
7.10.4.3 computeDDECoefficients() [1/6]	46
7.10.4.4 computeDDECoefficients() [2/6]	46
7.10.4.5 computeDDECoefficients() [3/6]	47
7.10.4.6 computeDDECoefficients() [4/6]	47
7.10.4.7 computeDDECoefficients() [5/6]	48
7.10.4.8 computeDDECoefficients() [6/6]	48
7.10.4.9 convert()	48
7.10.4.10 deconvert()	49
7.10.4.11 imageDimension()	49
7.10.4.12 operator()() [1/2]	49
7.10.4.13 operator()() [2/2]	49
$7.11\ capd:: ddes:: DDEBasicPoincare Map < \ DynSysSpec,\ SectionSpec > Class\ Template\ Reference \qquad . \ .$	50
7.11.1 Detailed Description	51
7.11.2 Member Function Documentation	51
7.11.2.1 detectCrossingDirection()	51
7.11.2.2 findCrossingTime()	52
7.11.2.3 integrateUntilSectionCrossing()	52
7.11.2.4 operator()() [1/3]	52
<b>7.11.2.5</b> operator()() [2/3]	52
<b>7.11.2.6 operator()()</b> [3/3]	53
7.11.2.7 setCurrentV()	54
<b>7.11.2.8 setInitialV()</b> [1/2]	54
<b>7.11.2.9 setInitialV()</b> [2/2]	54
$7.12\ capd:: ddeshelper:: DDECompare Helper < \ Vector Spec > Class\ Template\ Reference\ .\ .\ .\ .\ .\ .$	54
7.12.1 Detailed Description	55
7.12.2 Member Function Documentation	55
7.12.2.1 getUnknownsCount()	55
$7.13\ capd:: ddes:: DDEForward Taylor Curve Piece < Time Point Spec,\ Set Spec,\ is Interval > Class\ Template$	
Reference	56
7.13.1 Detailed Description	60
7.13.2 Constructor & Destructor Documentation	60
7.13.2.1 DDEForwardTaylorCurvePiece() [1/11]	60
7.13.2.2 DDEForwardTaylorCurvePiece() [2/11]	61
7.13.2.3 DDEForwardTaylorCurvePiece() [3/11]	61
7.13.2.4 DDEForwardTaylorCurvePiece() [4/11]	61
7.13.2.5 DDEForwardTaylorCurvePiece() [5/11]	61
7.13.2.6 DDEForwardTaylorCurvePiece() [6/11]	61
7.13.2.7 DDEForwardTaylorCurvePiece() [7/11]	62

7.13.2.8 DDEForward Taylor Curve Piece() [8/11]	 . 62
7.13.2.9 DDEForwardTaylorCurvePiece() [9/11]	 . 62
7.13.2.10 DDEForwardTaylorCurvePiece() [10/11]	 . 62
7.13.2.11 DDEForwardTaylorCurvePiece() [11/11]	 . 63
$7.13.2.12 \sim \! DDEForwardTaylorCurvePiece() \ \dots \ $	 . 63
7.13.3 Member Function Documentation	 . 63
7.13.3.1 affineTransform()	 . 63
7.13.3.2 allocate()	 . 63
7.13.3.3 allocateJet()	 . 64
7.13.3.4 allocateR0()	 . 64
7.13.3.5 allocateXi()	 . 64
7.13.3.6 backJet() [1/2]	 . 64
7.13.3.7 backJet() [2/2]	 . 64
7.13.3.8 badge()	 . 65
7.13.3.9 beginJet() [1/2]	 . 65
7.13.3.10 beginJet() [2/2]	 . 65
7.13.3.11 deallocate()	 . 65
7.13.3.12 deallocateJet()	 . 65
7.13.3.13 deallocateR0()	 . 65
7.13.3.14 deallocateXi()	 . 66
7.13.3.15 dimCheck()	 . 66
7.13.3.16 dimCheckB()	 . 66
7.13.3.17 dimCheckC()	 . 66
7.13.3.18 dimension()	 . 66
7.13.3.19 dot()	 . 67
7.13.3.20 dt()	 . 67
7.13.3.21 endJet() [1/2]	 . 67
7.13.3.22 endJet() [2/2]	 . 67
7.13.3.23 eval() [1/2]	 . 67
7.13.3.24 eval() [2/2]	 . 68
7.13.3.25 evalAtDelta() [1/2]	 . 68
7.13.3.26 evalAtDelta() [2/2]	 . 68
7.13.3.27 evalCoeff() [1/2]	 . 68
7.13.3.28 evalCoeff() [2/2]	 . 68
7.13.3.29 evalCoeffAtDelta() [1/2]	 . 69
7.13.3.30 evalCoeffAtDelta() [2/2]	 . 69
7.13.3.31 get_B()	 . 69
7.13.3.32 get_C()	 . 69
7.13.3.33 get_r()	 . 69
7.13.3.34 get_r0()	 . 70
7.13.3.35 get_x()	 . 70
7.13.3.36 get_Xi()	 . 70

7.13.3.37 getT0()
7.13.3.38 isMidCurve()
7.13.3.39 jetAt() [1/2]
7.13.3.40 jetAt() [2/2]
7.13.3.41 makeStorage_x()
7.13.3.42 midCurve()
7.13.3.43 mul()
7.13.3.44 operator=()
7.13.3.45 operator[]() [1/2]
7.13.3.46 operator[]() [2/2]
7.13.3.47 order()
7.13.3.48 reallocate()
7.13.3.49 reallocateJet()
7.13.3.50 reallocateR0()
7.13.3.51 reallocateXi()
7.13.3.52 reinitialize()
7.13.3.53 set_B()
7.13.3.54 set_Binv()
7.13.3.55 set_C()
7.13.3.56 set_Cr0()
7.13.3.57 set_r()
7.13.3.58 set_r0() [1/2]
7.13.3.59 set_r0() [2/2] 74
7.13.3.60 set_x()
7.13.3.61 set_Xi() [1/2]
7.13.3.62 set_Xi() [2/2]
7.13.3.63 setAsConstant() [1/4]
7.13.3.64 setAsConstant() [2/4]
7.13.3.65 setAsConstant() [3/4]
7.13.3.66 setAsConstant() [4/4]
7.13.3.67 setT0()
7.13.3.68 setupFromData() [1/2]
7.13.3.69 setupFromData() [2/2]
7.13.3.70 show()
7.13.3.71 storageDimension()
7.13.3.72 storageN0()
7.13.3.73 summa()
7.13.3.74 summaAtDelta()
7.13.3.75 t0()
7.13.3.76 take_r0()
7.13.3.77 take_Xi()
7.13.3.78 taylor() [1/2]

7.13.3.79 taylor() [2/2]	79
7.13.3.80 taylorAtDelta() [1/2]	79
7.13.3.81 taylorAtDelta() [2/2]	79
7.13.3.82 translate()	79
7.13.3.83 updateCommonR0()	80
7.14 capd::ddes::DDEJetSection< CurveSpec, isInterval > Class Template Reference	80
7.14.1 Constructor & Destructor Documentation	81
7.14.1.1 DDEJetSection() [1/2]	81
7.14.1.2 DDEJetSection() [2/2]	81
7.14.2 Member Function Documentation	81
7.14.2.1 getGradient()	82
$7.15\ capd:: ddes:: DDEN on rigorous Taylor Solver < Functional Map Spec > Class\ Template\ Reference \ . \ . \ .$	82
7.15.1 Member Function Documentation	83
7.15.1.1 oneStep() [1/2]	83
7.15.1.2 oneStep() [2/2]	83
7.15.1.3 operator()() [1/2]	84
7.15.1.4 operator()() [2/2]	84
$7.16\ capd:: ddes:: DDEPiecewise Polynomial Curve < Grid Spec,\ Jet Spec > Class\ Template\ Reference  . \ .$	84
7.16.1 Detailed Description	87
7.16.2 Constructor & Destructor Documentation	87
7.16.2.1 DDEPiecewisePolynomialCurve() [1/5]	87
7.16.2.2 DDEPiecewisePolynomialCurve() [2/5]	87
7.16.2.3 DDEPiecewisePolynomialCurve() [3/5]	87
7.16.2.4 DDEPiecewisePolynomialCurve() [4/5]	88
7.16.2.5 DDEPiecewisePolynomialCurve() [5/5]	88
7.16.2.6 ∼DDEPiecewisePolynomialCurve()	88
7.16.3 Member Function Documentation	88
7.16.3.1 add() [1/2]	88
7.16.3.2 add() [2/2]	88
7.16.3.3 addPastPiece() [1/2]	89
7.16.3.4 addPastPiece() [2/2]	89
7.16.3.5 addPiece()	89
7.16.3.6 affineTransform()	89
7.16.3.7 at() [1/2]	89
7.16.3.8 at() [2/2]	89
7.16.3.9 badge()	90
7.16.3.10 begin() [1/2]	90
7.16.3.11 begin() [2/2]	90
7.16.3.12 clear()	90
7.16.3.13 copyPieces()	90
7.16.3.14 currentTime()	90
7.16.3.15 deallocatePieces()	91

7.16.3.16 dimension()
7.16.3.17 dot() [1/2]
7.16.3.18 dot() [2/2] 91
7.16.3.19 end() [1/2] 91
7.16.3.20 end() [2/2] 91
7.16.3.21 epsilonShift() [1/2] 92
7.16.3.22 epsilonShift() [2/2] 92
7.16.3.23 eval()
7.16.3.24 extend()
7.16.3.25 get_x()
7.16.3.26 getCurrentTime()
7.16.3.27 getGrid()
7.16.3.28 getPiece()
7.16.3.29 getStep()
7.16.3.30 getT0()
7.16.3.31 getValueAtCurrent() [1/2]
7.16.3.32 getValueAtCurrent() [2/2]
7.16.3.33 grid()
7.16.3.34 j() [1/4] 94
7.16.3.35 j() [2/4] 94
7.16.3.36 j() [3/4] 94
7.16.3.37 j() [4/4] 95
7.16.3.38 jet() [1/3] 95
7.16.3.39 jet() [2/3] 95
7.16.3.40 jet() [3/3]
7.16.3.41 jetCommonMaximalOrder()
7.16.3.42 jetOrderAt()
7.16.3.43 jetPtr()
7.16.3.44 length()
7.16.3.45 mul()
7.16.3.46 mulThenAdd()
7.16.3.47 operator*=()
7.16.3.48 operator=()
7.16.3.49 pastTime()
7.16.3.50 pointToIndex()
7.16.3.51 rbegin() [1/2]
7.16.3.52 rbegin() [2/2]
7.16.3.53 readFrom()
7.16.3.54 rend() [1/2]
7.16.3.55 rend() [2/2]
7.16.3.56 set_x()
7.16.3.57 setCurrentTime()

7.16.3.58 setT0()	98
7.16.3.59 setValueAtCurrent()	98
7.16.3.60 show()	98
7.16.3.61 storageDimension()	99
7.16.3.62 subcurve() [1/4]	99
7.16.3.63 subcurve() [2/4]	99
7.16.3.64 subcurve() [3/4]	99
7.16.3.65 subcurve() [4/4]	99
7.16.3.66 t0()	99
7.16.3.67 translate()	100
7.16.3.68 writeTo()	100
7.16.4 Friends And Related Function Documentation	100
7.16.4.1 operator <<	100
7.16.4.2 operator>>	100
$7.17\ capd:: ddes:: DDEPoincare Map < DynSysSpec,\ Section Spec > Class\ Template\ Reference \qquad . \ . \ .$	100
7.17.1 Detailed Description	102
7.17.2 Member Function Documentation	102
7.17.2.1 detectCrossingDirection()	102
7.17.2.2 findCrossingTime()	102
7.17.2.3 integrateUntilSectionCrossing()	102
7.17.2.4 operator()() [1/2]	103
7.17.2.5 operator()() [2/2]	103
$7.18\ capd:: ddes:: DDERigorous Functional Map < Solution Curve Spec, Jet Spec > Class\ Template\ Reference for the control of the control $	ence103
7.18.1 Detailed Description	105
7.18.2 Constructor & Destructor Documentation	105
7.18.2 Constructor & Destructor Documentation	
	105
7.18.2.1 ∼DDERigorousFunctionalMap()	105
7.18.2.1 ∼DDERigorousFunctionalMap()	105 105 105
7.18.2.1 ∼DDERigorousFunctionalMap()	105 105 106
7.18.2.1 ~DDERigorousFunctionalMap()	105 105 105 106 106
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation	105 105 106 106 106
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]	105 105 106 106 107
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]	105 105 106 106 107
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]	105 105 106 106 107 107 107 107
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]  7.18.3.7 computeDDECoefficients() [4/6]	105 105 106 106 107 107 108 108
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]  7.18.3.7 computeDDECoefficients() [4/6]  7.18.3.8 computeDDECoefficients() [5/6]	105 105 106 106 107 107 107 108 108
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]  7.18.3.7 computeDDECoefficients() [4/6]  7.18.3.8 computeDDECoefficients() [5/6]  7.18.3.9 computeDDECoefficients() [5/6]	105 105 106 106 107 107 108 108 108
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]  7.18.3.7 computeDDECoefficients() [4/6]  7.18.3.8 computeDDECoefficients() [5/6]  7.18.3.9 computeDDECoefficients() [5/6]  7.18.3.10 convert()	105 105 106 106 107 107 108 108 108 108
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]  7.18.3.7 computeDDECoefficients() [4/6]  7.18.3.8 computeDDECoefficients() [5/6]  7.18.3.9 computeDDECoefficients() [6/6]  7.18.3.10 convert()  7.18.3.11 imageDimension()	105 105 106 106 107 107 108 108 108 109 109
7.18.2.1 ~DDERigorousFunctionalMap()  7.18.3 Member Function Documentation  7.18.3.1 checkCurveDimension()  7.18.3.2 collectComputationData() [1/2]  7.18.3.3 collectComputationData() [2/2]  7.18.3.4 computeDDECoefficients() [1/6]  7.18.3.5 computeDDECoefficients() [2/6]  7.18.3.6 computeDDECoefficients() [3/6]  7.18.3.7 computeDDECoefficients() [4/6]  7.18.3.8 computeDDECoefficients() [5/6]  7.18.3.9 computeDDECoefficients() [6/6]  7.18.3.10 convert()  7.18.3.11 imageDimension()  7.19 capd::ddes::DDESolutionCurve< SetSpec > Class Template Reference	105 105 106 106 107 107 108 108 108 108 109 109

7.19.2.2 DDESolutionCurve() [2/9]	 115
<b>7.19.2.3</b> DDESolutionCurve() [3/9]	 115
7.19.2.4 DDESolutionCurve() [4/9]	 115
<b>7.19.2.5</b> DDESolutionCurve() [5/9]	 115
7.19.2.6 DDESolutionCurve() [6/9]	 115
<b>7.19.2.7 DDESolutionCurve()</b> [7/9]	 116
7.19.2.8 DDESolutionCurve() [8/9]	 116
<b>7.19.2.9 DDESolutionCurve()</b> [9/9]	 116
7.19.3 Member Function Documentation	 116
<b>7.19.3.1 add()</b> [1/2]	 117
<b>7.19.3.2 add()</b> [2/2]	 117
7.19.3.3 addPastPiece() [1/2]	 117
7.19.3.4 addPastPiece() [2/2]	 117
7.19.3.5 affineTransform()	 117
7.19.3.6 copyPieces()	 118
7.19.3.7 dot() [1/2]	 118
7.19.3.8 dot() [2/2]	 118
7.19.3.9 dt()	 118
7.19.3.10 epsilonShift() [1/2]	 118
7.19.3.11 epsilonShift() [2/2]	 119
7.19.3.12 eval() [1/2]	 119
7.19.3.13 eval() [2/2]	 119
7.19.3.14 extend()	 119
7.19.3.15 get_B()	 120
7.19.3.16 get_Binv()	 120
7.19.3.17 get_C()	 120
7.19.3.18 get_r()	 120
7.19.3.19 get_r0()	 120
7.19.3.20 get_x()	 120
7.19.3.21 get_Xi()	 120
7.19.3.22 getLastEnclosure()	 121
7.19.3.23 getPiece()	 121
7.19.3.24 getValueAtCurrent() [1/2]	 121
7.19.3.25 getValueAtCurrent() [2/2]	 121
7.19.3.26 jetOrderAt()	 121
7.19.3.27 jetPtr()	 121
7.19.3.28 length()	 122
7.19.3.29 makeStorage_x()	 122
7.19.3.30 move()	 122
7.19.3.31 mul()	 122
7.19.3.32 mulThenAdd()	 122
7.19.3.33 operator=()	 123

7.19.3.34 pointToIndex()	. 123
7.19.3.35 reduce()	. 123
7.19.3.36 reinitialize()	. 123
7.19.3.37 set_B() [1/2]	. 123
7.19.3.38 set_B() [2/2]	. 124
7.19.3.39 set_Binv() [1/2]	. 124
7.19.3.40 set_Binv() [2/2]	. 124
7.19.3.41 set_C() [1/2]	. 124
7.19.3.42 set_C() [2/2]	. 124
7.19.3.43 set_Cr0() [1/2]	. 124
7.19.3.44 set_Cr0() [2/2]	. 125
7.19.3.45 set_r() [1/2]	. 125
7.19.3.46 set_r() [2/2]	. 125
7.19.3.47 set_r0() [1/2]	. 125
7.19.3.48 set_r0() [2/2]	. 126
7.19.3.49 set_x() [1/2]	. 126
7.19.3.50 set_x() [2/2]	. 126
7.19.3.51 set_Xi()	. 126
7.19.3.52 setValueAtCurrent() [1/2]	. 126
7.19.3.53 setValueAtCurrent() [2/2]	. 127
7.19.3.54 show()	. 127
7.19.3.55 storageDimension()	. 127
7.19.3.56 storageN0()	. 127
7.19.3.57 subcurve() [1/4]	. 127
7.19.3.58 subcurve() [2/4]	. 127
<b>7.19.3.59 subcurve()</b> [3/4]	. 128
7.19.3.60 subcurve() [4/4]	. 128
7.19.3.61 take_B()	. 128
7.19.3.62 take_Binv()	. 128
7.19.3.63 take_C()	. 128
7.19.3.64 take_r()	. 129
7.19.3.65 take_r0()	. 129
7.19.3.66 take_x()	. 129
7.19.3.67 translate()	. 129
7.19.3.68 writeTo()	. 130
7.19.4 Friends And Related Function Documentation	. 130
7.19.4.1 operator<<	. 130
7.19.5 Member Data Documentation	. 130
7.19.5.1 m_r0	. 130
$7.20\ capd:: ddes:: DDE Taylor Solver < Functional Map Spec > Class\ Template\ Reference\ .\ .\ .\ .\ .\ .$	. 131
7.20.1 Member Function Documentation	. 131
7.20.1.1 encloseSolution() [1/2]	. 132

7.20.1.2 encloseSolution() [2/2]	132
7.21 capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class Template Reference	133
7.21.1 Detailed Description	135
7.21.2 Member Typedef Documentation	135
7.21.2.1 const_iterator	
7.21.2.2 DelayStorageType	
7.21.2.3 iterator	
7.21.3 Constructor & Destructor Documentation	
7.21.3.1 ~DiscreteDelaysFunctionalMap()	
7.21.4 Member Function Documentation	
7.21.4.1 begin() [1/2]	
7.21.4.2 begin() [2/2]	
7.21.4.3 collectComputationData() [1/2]	
7.21.4.4 collectComputationData() [2/2]	
7.21.4.5 computeDDECoefficients() [1/2]	
7.21.4.6 computeDDECoefficients() [2/2]	
7.21.4.7 delaysCount()	
7.21.4.8 dimension()	
7.21.4.9 end() [1/2]	
7.21.4.10 end() [2/2]	
7.21.4.11 findRoughEnclosure()	
7.21.4.12 imageDimension()	
7.21.4.13 operator()()	
7.22 capd::ddes::DiscreteTimeGrid< RealSpec > Class Template Reference	
7.22.1 Detailed Description	
7.22.2 Member Function Documentation	140
7.22.2.1 badge()	141
7.23 capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec > Class Template Reference	141
7.23.1 Detailed Description	143
7.23.2 Constructor & Destructor Documentation	143
7.23.2.1 ~DoubletonInterface()	144
7.23.3 Member Function Documentation	144
7.23.3.1 add() [1/2]	144
7.23.3.2 add() [2/2]	144
7.23.3.3 affineTransform()	144
7.23.3.4 common_B()	145
7.23.3.5 common_C()	145
7.23.3.6 common_r()	145
7.23.3.7 common_r0()	145
7.23.3.8 common_x()	146
7.23.3.9 dimension()	146

7.23.3.10 dot()
7.23.3.11 get_B()
7.23.3.12 get_Binv()
7.23.3.13 get_C()
7.23.3.14 get_r()
7.23.3.15 get_r0()
7.23.3.16 get_x()
7.23.3.17 hull()
7.23.3.18 makeStorage_B()
7.23.3.19 makeStorage_C()
7.23.3.20 makeStorage_r()
7.23.3.21 makeStorage_r0()
7.23.3.22 makeStorage_x()
7.23.3.23 midPoint()
7.23.3.24 mul()
7.23.3.25 mulThenAdd()
7.23.3.26 operator VectorType()
7.23.3.27 operator*=()
7.23.3.28 reinitialize()
7.23.3.29 set_B() [1/2]
7.23.3.30 set_B() [2/2]
7.23.3.31 set_Binv() [1/2]
7.23.3.32 set_Binv() [2/2]
7.23.3.33 set_C() [1/2]
7.23.3.34 set_C() [2/2]
7.23.3.35 set_Cr0() [1/2]
7.23.3.36 set_Cr0() [2/2]
7.23.3.37 set_r() [1/2]
7.23.3.38 set_r() [2/2]
7.23.3.39 set_r0() [1/2]
7.23.3.40 set_r0() [2/2]
7.23.3.41 set_x() [1/2]
7.23.3.42 set_x() [2/2]
7.23.3.43 storageDimension()
7.23.3.44 storageN0()
7.23.3.45 take_B()
7.23.3.46 take_Binv()
7.23.3.47 take_C()
7.23.3.48 take_r()
7.23.3.49 take_r0()
7.23.3.50 take_x()
7.23.3.51 translate()

7.23.4 Friends And Related Function Documentation	156
7.23.4.1 operator<<	156
7.23.4.2 operator>>	156
7.24 ElNino < ScalarSpec, ParamSpec > Class Template Reference	156
7.24.1 Detailed Description	157
7.24.2 Member Function Documentation	157
7.24.2.1 dimension()	157
7.24.2.2 getParamsCount()	157
7.24.2.3 imageDimension()	158
7.24.2.4 operator()()	158
$7.25 \ capd:: ddeshelper:: Generator Range < IType > Class \ Template \ Reference \ \dots $	158
7.25.1 Member Function Documentation	159
7.25.1.1 increment()	159
7.26 capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > Class Template Reference	159
7.26.1 Detailed Description	161
7.26.2 Constructor & Destructor Documentation	162
7.26.2.1 GenericJet() [1/6]	162
<b>7.26.2.2 GenericJet()</b> [2/6]	162
<b>7.26.2.3 GenericJet()</b> [3/6]	162
7.26.2.4 GenericJet() [4/6]	162
<b>7.26.2.5 GenericJet()</b> [5/6]	163
7.26.2.6 GenericJet() [6/6]	163
7.26.2.7 ~GenericJet()	163
7.26.3 Member Function Documentation	163
7.26.3.1 allocateCoeffs()	163
<b>7.26.3.2 at()</b> [1/2]	164
<b>7.26.3.3 at()</b> [2/2]	164
<b>7.26.3.4 back()</b> [1/2]	164
<b>7.26.3.5 back()</b> [2/2]	164
7.26.3.6 badge()	164
<b>7.26.3.7 begin()</b> [1/2]	165
<b>7.26.3.8 begin()</b> [2/2]	165
7.26.3.9 coeff()	165
7.26.3.10 deallocateCoeffs()	165
7.26.3.11 decreasedOrder()	165
7.26.3.12 dimension()	166
7.26.3.13 dt()	166
7.26.3.14 end() [1/2]	166
<b>7.26.3.15 end()</b> [2/2]	166
<b>7.26.3.16 eval()</b> [1/2]	166
7.26.3.17 eval() [2/2]	167

7.26.3.18 evalAtDelta() [1/2]	
7.26.3.19 evalAtDelta() [2/2]	
7.26.3.20 evalCoeff() [1/2]	
7.26.3.21 evalCoeff() [2/2]	
7.26.3.22 evalCoeffAtDelta() [1/2]	
7.26.3.23 evalCoeffAtDelta() [2/2]	
7.26.3.24 get_x()	
7.26.3.25 getCoeffs()	
7.26.3.26 getT0()	
7.26.3.27 hull()	
7.26.3.28 increasedOrder()	9
7.26.3.29 operator VectorType()	9
7.26.3.30 operator=()	9
7.26.3.31 operator[]() [1/2]	9
7.26.3.32 operator[]() [2/2]	0
7.26.3.33 order()	0
7.26.3.34 reallocateCoeffs()	0
7.26.3.35 set_x()	0
7.26.3.36 setCoeffs()	0
7.26.3.37 setOrder()	1
7.26.3.38 setT0()	1
7.26.3.39 show()	1
7.26.3.40 storageDimension()	1
7.26.3.41 t0()	1
7.26.4 Friends And Related Function Documentation	2
7.26.4.1 operator <<	2
7.26.4.2 operator>>	2
7.27 capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec > Class Template Reference 172	2
7.27.1 Detailed Description	3
7.27.2 Member Function Documentation	3
7.27.2.1 dimension()	3
7.27.2.2 getParamsCount()	4
7.27.2.3 imageDimension()	4
7.27.2.4 operator()()	4
7.28 capd::ddes::LinearMap< MatrixSpec, ParamSpec > Class Template Reference	4
7.28.1 Detailed Description	5
7.28.2 Member Function Documentation	5
7.28.2.1 dimension()	5
7.28.2.2 imageDimension()	6
7.28.2.3 operator()()	6
7.29 capd::ddes::MackeyGlass < ScalarSpec, ParamSpec > Class Template Reference	6
7.29.1 Detailed Description	7

7.29.2 Member Function Documentation
7.29.2.1 dimension()
7.29.2.2 getParamsCount()
7.29.2.3 imageDimension()
7.29.2.4 operator()()
7.29.3 Member Data Documentation
7.29.3.1 gamma
7.29.3.2 n
$7.30\; capd:: ddeshelper:: Nonrigorous Helper < EqSpec, \; delays Spec > Class\; Template \; Reference \; . \; . \; . \; . \; 1799 = 1000 \; MeV \; . \; . \; . \; . \; . \; . \; . \; . \; . \; $
7.30.1 Detailed Description
7.30.2 Constructor & Destructor Documentation
7.30.2.1 NonrigorousHelper() [1/3]
7.30.2.2 NonrigorousHelper() [2/3]
<b>7.30.2.3 NonrigorousHelper()</b> [3/3]
7.30.3 Member Function Documentation
7.30.3.1 integrate() [1/3]
7.30.3.2 integrate() [2/3]
<b>7.30.3.3 integrate()</b> [3/3]
7.30.3.4 loadData()
7.30.3.5 makeSection()
7.30.3.6 makeSolver()
7.30.3.7 poincare() [1/3]
7.30.3.8 poincare() [2/3]
7.30.3.9 poincare() [3/3]
7.30.3.10 refinePeriodic()
7.30.3.11 setParam()
7.30.3.12 setParams()
7.31 capd::ddes::ODEPendulum Class Reference
7.31.1 Detailed Description
7.31.2 Member Function Documentation
7.31.2.1 dimension()
7.31.2.2 imageDimension()
7.32 capd::ddeshelper::PathConfig Class Reference
7.32.1 Detailed Description
7.32.2 Constructor & Destructor Documentation
7.32.2.1 PathConfig()
7.32.3 Member Function Documentation
7.32.3.1 filename()
7.32.3.2 filepath()
7.32.3.3 fullpath()
7.32.3.4 mkdir_p()
7.32.3.5 suffix()

$7.33\ capd:: ddes:: Perez Malta Cout in ho < Scalar Spec,\ Param Spec > Class\ Template\ Reference \\ \ .\ .\ .\ .$	188
7.33.1 Detailed Description	189
7.33.2 Member Function Documentation	189
7.33.2.1 dimension()	189
7.33.2.2 getParamsCount()	190
7.33.2.3 imageDimension()	190
7.33.2.4 operator()()	190
7.34 capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > $\leftarrow$ ::rebind< OtherSolutionSpec, OtherJetSpec > Struct Template Reference	190
7.35 capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind< OtherSolutionSpec, OtherJetSpec > Struct Template Reference	191
7.36 capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::rebind< OtherTimePointSpec, OtherDataSpec, OtherVectorSpec, OtherMatrixSpec, OtherIsInterval > Struct Template Reference	191
7.37 capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rebind< OtherJetTypeSpec > Struct Template Reference	191
$7.38\ capd:: ddeshelper:: Rigorous Helper < EqSpec,\ delays Spec,\ Policies Spec > Class\ Template\ Reference + Class\ Template\ R$	191
7.38.1 Detailed Description	194
7.38.2 Member Function Documentation	194
7.38.2.1 constantInitialSolution() [1/2]	194
7.38.2.2 constantInitialSolution() [2/2]	194
7.38.2.3 dataToSolution() [1/2]	195
7.38.2.4 dataToSolution() [2/2]	195
7.38.2.5 dumpData()	195
7.38.2.6 functionToSolution() [1/2]	195
7.38.2.7 functionToSolution() [2/2]	196
<b>7.38.2.8 poincare()</b> [1/3]	196
<b>7.38.2.9 poincare()</b> [2/3]	196
<b>7.38.2.10 poincare()</b> [3/3]	196
7.38.2.11 r0ToSolution() [1/2]	197
7.38.2.12 r0ToSolution() [2/2]	197
7.38.2.13 setParam()	197
7.38.2.14 setParams()	197
7.38.2.15 timemap()	198
$7.39\ capd:: ddes:: Rossler Delay < Scalar Spec,\ Param Spec > Class\ Template\ Reference \\ \ \ldots \\ \ \ldots$	198
7.39.1 Detailed Description	199
7.39.2 Member Function Documentation	199
7.39.2.1 getParamsCount()	199
7.39.2.2 imageDimension()	199
7.39.3 Member Data Documentation	199
7.39.3.1 a	200
7.39.3.2 b	200
7 39 3 3 c	200

7.39.3.4 epsi	200
7.40 capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec > Class Template Reference	200
7.40.1 Detailed Description	201
7.40.2 Member Function Documentation	201
7.40.2.1 dimension()	201
7.40.2.2 imageDimension()	202
7.41 capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec > Class Template Reference	202
7.41.1 Detailed Description	206
7.41.2 Constructor & Destructor Documentation	206
7.41.2.1 SharedDoubleton() [1/11]	206
7.41.2.2 SharedDoubleton() [2/11]	206
<b>7.41.2.3 SharedDoubleton()</b> [3/11]	207
7.41.2.4 SharedDoubleton() [4/11]	207
<b>7.41.2.5 SharedDoubleton()</b> [5/11]	207
7.41.2.6 SharedDoubleton() [6/11]	207
7.41.2.7 SharedDoubleton() [7/11]	207
7.41.2.8 SharedDoubleton() [8/11]	208
7.41.2.9 SharedDoubleton() [9/11]	208
7.41.2.10 SharedDoubleton() [10/11]	208
7.41.2.11 SharedDoubleton() [11/11]	208
7.41.2.12 $\sim$ SharedDoubleton()	208
7.41.3 Member Function Documentation	209
7.41.3.1 add() [1/2]	209
7.41.3.2 add() [2/2]	209
7.41.3.3 affineTransform()	209
7.41.3.4 allocate()	209
7.41.3.5 assureOwner()	210
7.41.3.6 common_B()	210
7.41.3.7 common_Binv()	210
7.41.3.8 common_C()	210
7.41.3.9 common_r()	210
7.41.3.10 common_r0()	211
7.41.3.11 common_x()	211
7.41.3.12 deallocate()	211
7.41.3.13 dimension()	211
7.41.3.14 get_B()	211
7.41.3.15 get_Binv()	212
7.41.3.16 get_C()	212
7.41.3.17 get_r()	212
7.41.3.18 get_r0()	212
7.41.3.19 get_x()	212
7.41.3.20 mul()	212

7.41.3.21 mulThenAdd()	213
7.41.3.22 operator=()	213
7.41.3.23 rawSetup()	213
7.41.3.24 reallocate()	213
7.41.3.25 reinitialize()	213
7.41.3.26 sanityCheck()	214
7.41.3.27 set_B() [1/2]	214
7.41.3.28 set_B() [2/2]	214
7.41.3.29 set_Binv() [1/2]	214
7.41.3.30 set_Binv() [2/2]	215
7.41.3.31 set_C() [1/2]	215
7.41.3.32 set_C() [2/2]	215
7.41.3.33 set_Cr0() [1/2]	215
7.41.3.34 set_Cr0() [2/2]	216
7.41.3.35 set_r() [1/2]	216
7.41.3.36 set_r() [2/2]	216
7.41.3.37 set_r0() [1/2]	216
7.41.3.38 set_r0() [2/2]	217
7.41.3.39 set_x() [1/2]	217
7.41.3.40 set_x() [2/2]	217
7.41.3.41 storageDimension()	217
7.41.3.42 storageN0()	217
7.41.3.43 take_B()	218
7.41.3.44 take_Binv()	218
7.41.3.45 take_C()	218
7.41.3.46 take_r()	218
7.41.3.47 take_r0()	218
7.41.3.48 take_x()	219
7.41.3.49 translate()	219
7.41.3.50 updateBinv()	219
$7.42\ capd:: ddes:: Discrete Time Grid < Real Spec > :: Time Point Type\ Class\ Reference \\ \ .\ .\ .\ .\ .\ .$	219
7.42.1 Member Function Documentation	220
7.42.1.1 badge()	221
7.43 capd::ddes::ToyModel Class Reference	221
7.43.1 Detailed Description	221
7.43.2 Member Function Documentation	221
7.43.2.1 dimension()	222
7.43.2.2 imageDimension()	222
7.44 capd::ddes::ToyModelSq Class Reference	222
7.44.1 Detailed Description	222
7.44.2 Member Function Documentation	223
7.44.2.1 dimension()	223

7.44.2.2 imageDimension()	223
$7.45\ capd:: ddes:: Toy Model SqA < Scalar Spec,\ Param Spec > Class\ Template\ Reference\ .\ .\ .\ .\ .\ .$	223
7.45.1 Detailed Description	224
7.45.2 Member Function Documentation	224
7.45.2.1 dimension()	224
7.45.2.2 getParamsCount()	224
7.45.2.3 imageDimension()	225
$7.46\ capd:: des:: Vector With Jac Data < Vector Spec,\ Matrix Spec > Class\ Template\ Reference \qquad . \ . \ . \ . \ .$	225
7.46.1 Detailed Description	226
7.47 capd::ddes::Wischert< ScalarSpec, ParamSpec > Class Template Reference	226
7.47.1 Detailed Description	227
7.47.2 Member Function Documentation	227
7.47.2.1 dimension()	227
7.47.2.2 getParamsCount()	228
7.47.2.3 imageDimension()	228
7.47.2.4 operator()()	228
Index	229

## Nonrigorous, non-autonomous demo

- before compiling this demo, you need to compile extrenal libraries first, go to the root directory fo the project and read README.md there.
- When you have compiled the external libraries, you can return here and compile this demo
- · to compile this demo you need to invoke in this directory:

```
``` make all ```
```

• it will create ./bin subdirectory

```
``` cd ./bin ```
```

• to run the program, simply type

```
"... /draw-nonrig "...
```

• to see possible options of the program, type

```
".' /draw-nonrig -help "
```

• you may also try to run 'run-demos.sh' from this demo root directory:

```
" bash run-demos.sh
```

· have a nice day!

## **Exemplary programs**

This folder contains exemplary programs to show how to use the library.

As an example consider **demo-elninio** folder. You can compile it by invoking make all inside the **demo-elninio** folder, or by invoking make examples/demo-elninio in the root directory of the repository.

The executables will be created in demo-elninio/bin folder, and you can execute bash run-demos.sh inside the demo-elninio folder to run some demonstrations. The output will go into demo-elninio/bin folder.

All other examples will provide similar functionality.

You can copy any example folder into \*\*../devel\*\* folder and experiment with it/modify it to your liking.

See documentation in \*\*../devel\*\* folder to learn how to make your own programs.

Exemplary programs

# Namespace Index

### 3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:	
capd	1

6 Namespace Index

## **Hierarchical Index**

### 4.1 Class Hierarchy

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

capd::ddeshelper::ArgumentParser
capd::ddeshelper::BinaryData< T >
capd::ddeshelper::CoordinateFrame < MatrixSpec, VectorSpec, ScalarSpec >
capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::CoordinateSystem
capd::ddeshelper::CubeSet< VectorSpec >
capd::ddeshelper::CubeSet< VectorSpec >::CubeSetIterator
capd::ddes::Cubiclkeda < ScalarSpec, ParamSpec >
capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >
${\tt capd::ddes::DDEBasicFunctionalMap} < {\tt SolutionCurveSpec}, \ {\tt typename SolutionCurveSpec::JetType} > \ .  {\tt 42} \\ {\tt capd::ddes::DDEBasicFunctionalMap} < {\tt SolutionCurveSpec}, \ {\tt typename SolutionCurveSpec::JetType} > \ .  {\tt 42} \\ {\tt capd::ddes::DDEBasicFunctionalMap} < {\tt SolutionCurveSpec}, \ {\tt typename SolutionCurveSpec}, \ {$
capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > 14
capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >
capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::Jet↔
Type >
capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > 133
capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >
capd::ddeshelper::DDECompareHelper< VectorSpec >
capd::ddes::DDEJetSection < CurveSpec, isInterval >
capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec >
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >
capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >
${\it capd::} {\it ddes::} {\it DDETaylorSolver} < {\it FunctionalMapSpec} > \dots                                  $
capd::ddes::DiscreteTimeGrid < RealSpec >
${\sf capd::ddes::DiscreteTimeGrid} < {\sf Real} > \dots $
${\sf ElNino}{<}\ {\sf ScalarSpec},\ {\sf ParamSpec}{>}\ \dots$
$capd:: ddeshelper:: Generator Range < IType > \dots $
$capd:: ddes:: Generic Jet < Time Point Spec, \ Data Spec, \ Vector Spec, \ Matrix Spec, \ is Interval > . \ . \ . \ . \ . \ . \ . \ . \ . \ .$
IdQRPolicy
capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >
capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >
capd::ddes::BasicDoubleton < MatrixSpec, PoliciesSpec >
capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >
capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >
capd::ddes::LinearMap< MatrixSpec, ParamSpec >
${\sf capd::} {\sf ddes::} {\sf MackeyGlass} {<  {\sf ScalarSpec},  {\sf ParamSpec} >  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots  \dots $
${\sf capd::} {\sf ddeshelper::} {\sf NonrigorousHelper} {\sf < EqSpec}, {\sf delaysSpec} {\sf > \dots $

8 Hierarchical Index

capd::ddes::ODEPendulum	185
capd::ddeshelper::PathConfig	186
capd::ddes::PerezMaltaCoutinho < ScalarSpec, ParamSpec >	188
Policy	
capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >	141
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >	56
capd::ddes::DDESolutionCurve< SetSpec >	109
capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind<	
OtherSolutionSpec, OtherJetSpec >	190
capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > ~	
::rebind< OtherSolutionSpec, OtherJetSpec >	191
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::rebind<	
OtherTimePointSpec, OtherDataSpec, OtherVectorSpec, OtherMatrixSpec, OtherIsInterval > .	191
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rebind< OtherJetTypeSpec >	191
capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >	191
capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >	198
capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >	200
capd::ddes::DiscreteTimeGrid< RealSpec >::TimePointType	219
capd::ddes::ToyModel	221
capd::ddes::ToyModelSq	222
capd::ddes::ToyModelSqA < ScalarSpec, ParamSpec >	223
VectorSpec	
capd::ddes::VectorWithJacData < VectorSpec, MatrixSpec >	225
capd::ddes::Wischert< ScalarSpec. ParamSpec >	226

## **Class Index**

#### 5.1 Class List

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

capd::ddeshelper::ArgumentParser	13
capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >	14
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >	22
capd::ddeshelper::BinaryData< T >	36
capd::ddeshelper::CoordinateFrame < MatrixSpec, VectorSpec, ScalarSpec >	36
capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::CoordinateSystem	37
capd::ddeshelper::CubeSet< VectorSpec >	38
capd::ddeshelper::CubeSet< VectorSpec >::CubeSetIterator	40
capd::ddes::Cubiclkeda < ScalarSpec, ParamSpec >	41
capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >	42
capd::ddes::DDEBasicPoincareMap < DynSysSpec, SectionSpec >	50
capd::ddeshelper::DDECompareHelper< VectorSpec >	54
capd::ddes::DDEForwardTaylorCurvePiece < TimePointSpec, SetSpec, isInterval >	56
capd::ddes::DDEJetSection < CurveSpec, isInterval >	80
capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec >	82
capd::ddes::DDEPiecewisePolynomialCurve < GridSpec, JetSpec >	84
capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >	100
$capd:: ddes:: DDER igorous Functional Map < Solution Curve Spec, Jet Spec > \dots $	103
${\sf capd::} {\sf ddes::} {\sf DDESolutionCurve} {\sf < SetSpec} > \dots $	109
capd::ddes::DDETaylorSolver< FunctionalMapSpec >	131
$capd:: ddes:: Discrete Delays Functional Map < Fin Dim Map Spec, Solution Curve Spec, Jet Spec > \dots \dots$	133
${\sf capd::ddes::DiscreteTimeGrid} < {\sf RealSpec} > \dots $	
capd::ddes::DoubletonInterface < MatrixSpec, PoliciesSpec >	141
ElNino < ScalarSpec, ParamSpec >	156
capd::ddeshelper::GeneratorRange < IType >	158
${\it capd::} {\it ddes::} {\it GenericJet} {\it < TimePointSpec},  {\it DataSpec},  {\it VectorSpec},  {\it MatrixSpec},  {\it isInterval} > \dots \dots \dots$	
${\sf capd::ddes::LasotaWazewska} < {\sf ScalarSpec}, \ {\sf ParamSpec} > \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	172
capd::ddes::LinearMap< MatrixSpec, ParamSpec >	174
${\sf capd::} {\sf ddes::} {\sf MackeyGlass} {< ScalarSpec}, {\sf ParamSpec} {> } \ldots $	176
${\sf capd::} {\sf ddeshelper::} {\sf NonrigorousHelper} {\sf < EqSpec},  {\sf delaysSpec} > \dots $	179
capd::ddes::ODEPendulum	185
capd::ddeshelper::PathConfig	
${\sf capd::ddes::PerezMaltaCoutinho} < {\sf ScalarSpec}, \ {\sf ParamSpec} > \ \ldots \ldots$	
capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind< Of	therSolutionSpec, Othe

10 Class Index

capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind 191	d< OtherSolutionSpec
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::rebind< Other 191	TimePointSpec, Other
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rebind< OtherJetTypeSpec >	191
capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >	191
capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >	198
capd::ddes::ScalarLinearMap < ScalarSpec, ParamSpec >	
capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >	
capd::ddes::DiscreteTimeGrid < RealSpec >::TimePointType	
capd::ddes::ToyModel	
capd::ddes::ToyModelSq	
capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >	223
capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec >	225
cand: ddes: Wischert < ScalarSpec ParamSpec >	226

### **Namespace Documentation**

#### 6.1 capd Namespace Reference

#### 6.1.1 Detailed Description

This file constitutes part of DDEs rigorous integration framework developed in PhD Thesis:

```
"Rigorous Integration of Delay Differential Equations", Jagiellonian University, 2015
```

When using it in any scientific work please consider citing my articles (preferred), PhD thesis and/or my web page. For the publications describing the source code please refer to <a href="http://scirsc.org/p/papers">http://scirsc.org/p/papers</a>.

This work would not be possible without aid and expertise of people involved in developing CAPD library (Computer Assisted Proofs in Dynamics). Please refer to <a href="http://capd.ii.uj.edu.pl">http://capd.ii.uj.edu.pl</a> and consider citing also the CAPD library when using those codes in any scientific work.

Author: Robert Szczelina, PhD (former) Małopolska Center of Biotechnology, Jagiellonian University AND (currently) Faculty of Mathematics and Computer Science, Jagiellonian University email: robert.szczelina@uj.←edu.pl www: scirsc.org

This source code is provided under GNU GPL license (v.2 or whatever)

### **Class Documentation**

#### 7.1 capd::ddeshelper::ArgumentParser Class Reference

```
#include <ddeshelperlib.h>
```

#### **Public Member Functions**

- **ArgumentParser** (ArgcType &argc, ArgvType &argv)
- std::string getCommandLine (std::string sep=" \\\n# ")
- std::string getHelp ()
- bool parse (std::string const &param, std::string const &help\_s="")
- template<typename T >

bool parse (std::string const &param, T &out, std::string const &help\_s="")

• template<typename T >

bool parse (std::string const &param, T &out, const char \*const help\_s)

• template<typename T >

bool **parse** (std::string const &param, T &out, std::initializer\_list< T > allowed\_values, std::string const &help\_s="")

• template<typename T , typename F >

bool parse (std::string const &param, T &out, F predicate, std::string const &help\_s="")

#### **Friends**

```
    template < typename T >
        ArgumentParser & operator < < (ArgumentParser & out, T const & item)</li>
```

#### 7.1.1 Detailed Description

This is for a nice parsing of arguments from command line

This is not so optimal ('m' \* 'argc' \* 'string comparison' operations) Where m is number of "parse" parameters. It could be done better, but I decided to keep it simple and produce decent help string automatically.

For now it only supports params in the following form: param=value and param You can do -param=value and -param if you wih For now it does not support multiple forms of params, ie. -f and -file meaning the same, itp.

Example use:

14 Class Documentation

```
capd::ddeshelper::ArgumentParser args(argc, argv);
```

std::string outdir = "."; std::string prefix = "attractor"; int p = 32; int n = 3; std::string plot\_mode = "poincare"; args. parse("outdir=", outdir); args.parse("prefix=", prefix, "A prefix of all the filenames"); args.parse("p=", p); args. parse("n=", n, "order of the method"); args.parse("plot=", plot\_mode, {std::string("poincare"), std::string("xpx"), std::string("phasespace")}, "Kind of plot to produce and this is a very long explanation what happens in that parameter lorem ipsum sit dolor"); int itest = 5; std::string stest; args.parse("stest=", stest, {std::string("poincare"), std::string("xpx"), std::string("phasespace")}); args.parse("itest=", itest, {-1,0,1}); bool flag = args.parse(std ::string("flag"));

TODO: (not important) add support for variation in params, eg. args.parse({"-h", "--help"}), itp. should be easy TODO: (not important) extract those from the project and send to github as a small library TODO: (not important) add option to handle separated values, ie. "--param data"

#### 7.1.2 Friends And Related Function Documentation

#### 7.1.2.1 operator <<

outputs with a nice pad

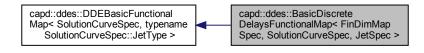
The documentation for this class was generated from the following file:

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/ddeshelperlib.
 h

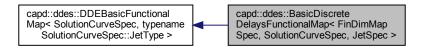
# 7.2 capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class Template Reference

#include <BasicDiscreteDelaysFunctionalMap.h>

 $Inheritance\ diagram\ for\ capd:: ddes:: Basic Discrete Delays Functional Map < Fin Dim Map Spec,\ Solution Curve Spec,\ Jet Spec >:$ 



Collaboration diagram for capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >:



#### **Classes**

· struct rebind

# **Public Types**

- typedef DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec > BaseClass
- typedef BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class
- typedef SolutionCurveSpec CurveType
- typedef JetSpec JetType
- typedef FinDimMapSpec MapType
- typedef BaseClass::RealType RealType
- typedef BaseClass::TimePointType TimePointType
- typedef BaseClass::MatrixType MatrixType
- typedef BaseClass::VectorType VectorType
- typedef BaseClass::ScalarType ScalarType
- typedef BaseClass::size type size type
- typedef BaseClass::DataType DataType
- typedef BaseClass::ValueStorageType ValueStorageType
- typedef BaseClass::VariableStorageType VariableStorageType
- typedef BaseClass::JacobianStorageType JacobianStorageType
- typedef std::vector< TimePointType > DelayStorageType
- typedef DelayStorageType::const\_iterator const\_iterator
- typedef DelayStorageType::iterator iterator

# **Public Member Functions**

- template<typename OtherDiscreteDelaysFunctionalMap >
  - BasicDiscreteDelaysFunctionalMap (const OtherDiscreteDelaysFunctionalMap &orig)
- BasicDiscreteDelaysFunctionalMap (const BasicDiscreteDelaysFunctionalMap &orig)
- BasicDiscreteDelaysFunctionalMap (const TimePointType &tau)
- BasicDiscreteDelaysFunctionalMap (const MapType &f)
- BasicDiscreteDelaysFunctionalMap (const MapType &f, const TimePointType &tau)
- BasicDiscreteDelaysFunctionalMap (const MapType &f, std::vector< TimePointType > delays)
- BasicDiscreteDelaysFunctionalMap (const MapType &f, int n, TimePointType delays[])
- virtual ~BasicDiscreteDelaysFunctionalMap ()
- iterator begin ()
- iterator end ()
- · const iterator begin () const
- const\_iterator end () const

- size\_type imageDimension () const
- · size\_type dimension () const
- size\_type delaysCount () const
- VectorType operator() (const TimePointType &t, const CurveType &x) const
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out\_u, size\_type &out\_admissible\_order) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const

computeDDECoefficients

- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const=0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const=0
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs, Variable
   — StorageType &u, JacobianStorageType &Du) const

#### **Protected Member Functions**

- · void checkDimension (size\_type d) const
- template<typename AnyVector >
   void checkDimension (AnyVector const &v) const
- · void checkMapSignature () const

# **Protected Attributes**

- MapType m map
- DelayStorageType m delays

#### **Additional Inherited Members**

# 7.2.1 Detailed Description

 $template < typename\ FinDimMapSpec,\ typename\ SolutionCurveSpec,\ typename\ JetSpec = typename\ SolutionCurveSpec:: Jet \leftarrow Type >$ 

class capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >

A class to represent right hand side of the DDE of the form:

```
x' = f(t, x(t), x(t-tau_1), ..., x(t-tau_m))
```

NOTE: This is a demo version of what I want to achieve with the DDEs code interface I assume it should be similar to ODEs in original CAPD. I assume that any RHS map f of the equation  $x'(t) = f(x_tau)$  should be representable with similar interface (i.e. the function accepts some SolutionCurve and can produce Jet at current time in the solution defined with the equation (function computeDDECoefficients) I do not assume anything about Solution curve except, it can produce Forward Taylor Jets of itself at a given point (that is used in the equation) and that it can return its value at a current time.

In the implementation below, I assume that I can ask about SOlutionCUrve and its Jet at \$t-\tau\$, \$t\$ is current time. Then I construct an AD jet of x(t-\tau) and pass it to standard CAPD map.

NOTE: I do not know if integral equations (i.e.  $x'(t) = F(\int_{t-1}^t du)^{t} g(x(s)) ds$ ) could be described in this manner... But I hope it could be done, at least if g is linear in x (e.g. g = Id). NOTE: in fact, functional map could have operators that are templates, but we assume that the argument curve might have some requirements when computing, for example it could determine only the delays at the specific grid points (because of the loss of regularity, we are not able to allways give jets over some general intervals, see new notes). Therefore, we supply the Map with appropriate CurveType that it can manipulate.

# 7.2.2 Member Typedef Documentation

# 7.2.2.1 const iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef DelayStorageType::const_iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< Fin←
DimMapSpec, SolutionCurveSpec, JetSpec >::const_iterator
```

iterator over this map will iterate over delays!

# 7.2.2.2 DelayStorageType

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef std::vector< TimePointType > capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDim←
MapSpec, SolutionCurveSpec, JetSpec >::DelayStorageType
```

storage type for discrete number of delays

# 7.2.2.3 iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef DelayStorageType::iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMap↔
Spec, SolutionCurveSpec, JetSpec >::iterator
```

iterator over this map will iterate over delays!

### 7.2.3 Constructor & Destructor Documentation

#### 7.2.3.1 ∼BasicDiscreteDelaysFunctionalMap()

```
\label{template} $$ \end{template} $$$ \end{template} $$ \end{template} $$$ \end{template} $$ \end{template} $$$ \end{template} $$$ \end{template} $$$ \end{template} $$$ \e
```

standard

# 7.2.4 Member Function Documentation

# 7.2.4.1 begin() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↔
Spec >::begin ( ) [inline]
```

iterator over delays

### 7.2.4.2 begin() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::begin ( ) const [inline]
```

iterator over delays

#### 7.2.4.3 collectComputationData()

Implementation of virual func. See Interface docs.

TODO: (NOT URGENT) explain what is stored in output for DiscreteDelaysMap...

Warning: dt should be enclosure for the step [0, h]

Implements capd::ddes::DDEBasicFunctionalMap < SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

#### 7.2.4.4 computeDDECoefficients() [1/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients [inline]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

# 7.2.4.5 computeDDECoefficients() [2/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE←
Coefficients [inline]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

# 7.2.4.6 computeDDECoefficients() [3/8]

Implementation of virual func. See Interface docs. for k loop

Implements capd::ddes::DDEBasicFunctionalMap < SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

#### 7.2.4.7 computeDDECoefficients() [4/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE←
Coefficients
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true. For non-developer users of the library it is possible to use versions of the function for specific CurveType.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out\_admissible\_order from collectComputationData() is used for this purpose.

# 7.2.4.8 computeDDECoefficients() [5/8]

computeDDECoefficients

see Interface docs.

Implements capd::ddes::DDEBasicFunctionalMap < SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

### 7.2.4.9 computeDDECoefficients() [6/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It also computes  $\frac{\pi}{\pi}$  (partial coeffs[i]}{\partial u}\$ (in Du). It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out\_admissible\_order from collectComputationData() is used for this purpose.

# 7.2.4.10 computeDDECoefficients() [7/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE
Coefficients [inline]
```

computes recursively the Jet at time t for a given curve for a DDE of the form:

```
x'(t) = F(t, x)
```

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. If order is 0 or order is too high for a given curve then the function should alter the order to highest possible. (use coeffs. resize(order) for this purpose)

THERE IS DEFAULT IMPLEMENTATION OF THIS WITH collectComputationData() and computeDDE ← Coefficients(..., u, ...) call to pure virtual function.

# 7.2.4.11 computeDDECoefficients() [8/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients [inline]
```

Same as computeDDECoefficients(const TimePointType&, const CurveType&, JetType&), but also computes the partial derivative:

```
Du[k] = \frac{\pi u}{\pi u}
```

where u are all the variables used to evaluate the map. In u[j] is a d-dimensional set used in computation. In Du[k][j] is a matrix of dimension M(d,d) and of course it is

```
Du[k][j] = \frac{\sqrt{partial coeffs[k]}{\langle u[j]}}
```

It is done this way to allow Solvers to use this structure to reduce wrapping effect of interval arithmetics with the help of Lohner algorithm.

#### 7.2.4.12 delaysCount()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::delaysCount ( ) const [inline]
```

number of delays

# 7.2.4.13 dimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

# 7.2.4.14 end() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet
Spec >::end ( ) [inline]
```

iterator over delays

# 7.2.4.15 end() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::end ( ) const [inline]
```

iterator over delays

# 7.2.4.16 imageDimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::imageDimension ( ) const [inline]
```

output dimension of the internal map

# 7.2.4.17 operator()()

Implementation of virual func. See Interface docs.

Implements capd::ddes::DDEBasicFunctionalMap < SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

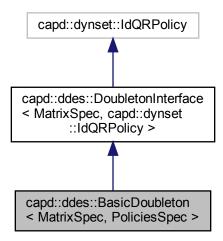
The documentation for this class was generated from the following file:

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Basic
 DiscreteDelaysFunctionalMap.h

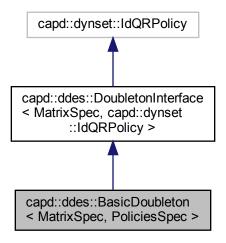
# 7.3 capd::ddes::BasicDoubleton < MatrixSpec, PoliciesSpec > Class Template Reference

#include <BasicDoubleton.h>

Inheritance diagram for capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >:



Collaboration diagram for capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >:



# **Public Types**

- typedef MatrixSpec MatrixType
- typedef MatrixType::RowVectorType VectorType
- typedef MatrixType::ScalarType ScalarType
- typedef MatrixType::size\_type size\_type
- typedef BasicDoubleton Class

- typedef DoubletonInterface < MatrixSpec > BaseClass
- typedef VectorType \* VectorTypePtr
- typedef MatrixType \* MatrixTypePtr
- typedef std::bitset< 5 > OwnershipType
- typedef PoliciesSpec Policy
- typedef PoliciesSpec QRPolicy

#### **Public Member Functions**

- Class & operator= (Class const &orig)
- BasicDoubleton ()
- BasicDoubleton (VectorType const &x, VectorType \*common\_r0=0, bool passOwnership=false)
- BasicDoubleton (Class const &orig)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, VectorType const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, MatrixType const &Binv, VectorType const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType \*r0, MatrixType const &B, Vector
   —
   Type const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType \*r0, MatrixType const &B, Matrix
   —
   Type const &Binv, VectorType const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType \*r0)
- BasicDoubleton (VectorType \*x, MatrixType \*C, VectorType \*r0, MatrixType \*B, VectorType \*r)
- BasicDoubleton (size\_type d, size\_type N0=-1)
- virtual ∼BasicDoubleton ()
- size\_type storageN0 () const
  - < import dimension from interface
- size\_type storageDimension () const
- VectorType get x () const
- MatrixType get C () const
- VectorType get\_r0 () const
- MatrixType get\_B () const
- VectorType get\_r () const
- BaseClass & set\_x (VectorType const &x)
- BaseClass & set\_C (MatrixType const &C)
- BaseClass & set\_r0 (VectorType const &r0)
- BaseClass & set\_B (MatrixType const &B)
- BaseClass & set\_r (VectorType const &r)
- BaseClass & set\_Cr0 (MatrixType const &C, VectorType const &r0)
- VectorType midPoint () const
- VectorType hull () const
- virtual bool common\_x (VectorType const \*x) const
  - < using the base class take\_\* method...
- virtual bool common C (MatrixType const \*C) const
- virtual bool common r0 (VectorType const \*r0) const
- virtual bool common\_B (MatrixType const \*B) const
- virtual bool common\_r (VectorType const \*r) const
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- std::string show () const
- virtual void reinitialize (size\_type d, size\_type N0)
- virtual size\_type dimension () const

```
• virtual Class & set_x (VectorType const &x)=0

    virtual Class & set_x (VectorType *x, bool passOwnership=false)

    virtual Class & set_C (MatrixType const &C)=0

      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set C (MatrixType *C, bool passOwnership=false)

      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set_r0 (VectorType const &r0)=0

      < using the base class set *(pointer, ownershhip) method...

    virtual Class & set r0 (VectorType *r0, bool passOwnership=false)

      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set_B (MatrixType const &B)=0

      < using the base class set *(pointer, ownershhip) method...

    virtual Class & set B (MatrixType *B, bool passOwnership=false)

      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set_r (VectorType const &r)=0

      < using the base class set *(pointer, ownershhip) method...

    virtual Class & set_r (VectorType *r, bool passOwnership=false)

      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set_Cr0 (MatrixType const &C, VectorType const &r0)=0

      < using the base class set_*(pointer, ownershhip) method...
   virtual Class & set Cr0 (MatrixType *C, VectorType *r0, bool passOwnership1=false, bool pass

→
  Ownership2=false)
      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set_Binv (MatrixType *Binv, bool passOwnership=false)

      < using the base class set_*(pointer, ownershhip) method...

    virtual Class & set Binv (MatrixType const &Binv)

      < using the base class set_*(pointer, ownershhip) method...
virtual VectorType * take_x ()
      < using the base class set_*(pointer, ownershhip) method...

    virtual MatrixType * take C ()

      < using the base class take_* method...

    virtual VectorType * take_r0 ()

      < using the base class take * method...

    virtual MatrixType * take_B ()

      < using the base class take_* method...
virtual VectorType * take_r ()
      < using the base class take_* method...

    virtual MatrixType * take Binv ()

      < using the base class take_* method...

    virtual Class & add (VectorType const &v)

· virtual Class & add (Class const &set)

    virtual Class & mul (ScalarType const &c)

    virtual Class & mulThenAdd (ScalarType const &c, Class const &set)
```

# **Protected Member Functions**

- void sanityCheck (std::string const &what="") const
- void setupDimension (size type d, size type N0=0)
- void setupFromData (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, VectorType const &r)
- void setupFromOther (Class const &other)
- void setupFromOther (BaseClass const &other)

#### **Protected Attributes**

- VectorType m\_x
- MatrixType m\_C
- VectorType m\_r0
- MatrixType m\_B
- VectorType m\_r

#### **Additional Inherited Members**

# 7.3.1 Detailed Description

template<typename MatrixSpec, typename PoliciesSpec = capd::dqnset::ldQRPolicy> class capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >

This class stores a set  $X \in \mathbb{R}^{n}$  of the form:

```
X = x + C * r0 + B * r
```

with  $x \in \mathbb{R}^d$ , \$r0, r\$ - closed intervals centered at 0, \$r0 \subset \R^{\N\_0}\$, \$C \in Lin(\R^{\N\_0}, \R^d)\$, \$B \in Lin(\R^d, \R^d)\$, \$r \in \R^d\$, \$B\$ being a matrix easy to invert, e.g. B = ID (Interval form of the remainder) or \$B\$ orthogonal (Doubleton set with QR decomposition).

NOTE: compare to the classic Lohner Doubleton Set in CAPD, where authors assume  $N_0 = d$  NOTE: This is the most basic implementation, I use bare Vectors and Matrices and all the data is stored inside the set. For DDEs as described in notes it is somehow innefficient, as each component will contain its copy of r0 vector which in the computation in fact is common to all the components. Therefore we will store O(size of repr.) elements of size dim(r0) instead of O(1). SharedDoubleton is designed for this purpose, but is is highly complicated as of now (and I have some problems with memory leak - o be corrected later). I have decided to implement simple solution to test things without need to worry about the problems with dynamical memory allocation. The code is also simpler to read.

NOTE: This is quite long and tedious code in C++ because of C++ This should be heavily tested for correctedness and memory leaks Reader (e.g. reviewer of the manuscript for publication) should not worry to check that code

NOTE: THIS PARTICULAR IMPLEMENTATION does not use shared memory, everything is static (from the point of view of implkemnentation, afaik CAPD uses shared ptr inside the library) Its main purpose is testing, as it will be (probably) slower and (for sure) more memory consuming (2x at least) than a SharedDoubleton version. SharedDoubleton will be used in applications.

TODO: (NOT URGENT) implement QR policy in BinvB (now IdPolicy is hardcoded). This is not urgent, as I use SharedDoubleton in my computations.

#### 7.3.2 Constructor & Destructor Documentation

### 7.3.2.1 BasicDoubleton() [1/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton ( ) [inline]
```

default constructor makes a 1D point-set at 0

#### 7.3.2.2 BasicDoubleton() [2/11]

setup set with a given vector. You can pass common\_r0 and passOwnership = false (default) if you want to set externally controlled r0 element. If passOwnership = true then the set will take responsibility for deallocating item.

#### 7.3.2.3 BasicDoubleton() [3/11]

copies the original set (ownership is preserved)

#### 7.3.2.4 BasicDoubleton() [4/11]

setup this set with a given data, set owns everything

# 7.3.2.5 BasicDoubleton() [5/11]

setup this set with a given data, set owns everything

# 7.3.2.6 BasicDoubleton() [6/11]

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

#### 7.3.2.7 BasicDoubleton() [7/11]

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

# 7.3.2.8 BasicDoubleton() [8/11]

setup this set with a given data, set is owner of everything

#### 7.3.2.9 BasicDoubleton() [9/11]

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

# 7.3.2.10 BasicDoubleton() [10/11]

setup this set with a given data but the set does not own the data (user is responsible for deleting)

#### 7.3.2.11 BasicDoubleton() [11/11]

setup this set as zero vector, but the structure of given dimensions. If second arg is < 0 then d is used instead.

# 7.3.2.12 ∼BasicDoubleton()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::~BasicDoubleton ( ) [inline],
[virtual]
```

standard thing

#### 7.3.3 Member Function Documentation

# 7.3.3.1 add() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::add [inline]
```

add a Set or Vector to this representation. It takes the representation into consideration.

# 7.3.3.2 add() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::add [inline]
```

add a Set or Vector to this representation. It takes the representation into consideration.

#### 7.3.3.3 affineTransform()

applies in a smart way to this set X the affine transform f(y) = M \* (X - v)

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.3.3.4 common B()

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

# 7.3.3.5 common\_C()

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.3.3.6 common\_r()

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dqnset::IdQRPolicy >.

#### 7.3.3.7 common\_r0()

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dqnset::IdQRPolicy >.

# 7.3.3.8 common\_x()

< using the base class take\_\* method...

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.3.3.9 dimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dimension [inline]
```

by default dimension is equal to storage dimension

# 7.3.3.10 get\_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_B ( ) const [inline]
```

see interface

# 7.3.3.11 get\_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_C ( ) const [inline]
```

see interface

# 7.3.3.12 get\_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_r ( ) const [inline]
```

see interface

# 7.3.3.13 get\_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_r0 () const [inline]
```

see interface

#### 7.3.3.14 get\_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_x ( ) const [inline]
```

see interface

# 7.3.3.15 hull()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::hull () const [inline]
```

see interface

#### 7.3.3.16 midPoint()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::midPoint ( ) const [inline]
```

see interface

#### 7.3.3.17 mul()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::mul [inline]
```

multiply set by a scalar. Should take set structure into consideration.

# 7.3.3.18 mulThenAdd()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::des::DoubletonInterface< MatrixSpec, PoliciesSpec >::mulThenAdd [inline]
```

multiply set by a scalar, then add another set. Very simple basic implementation by first mul then add.

# 7.3.3.19 operator=()

assign operator - it needs to deallocate memory if neccessery, then setup set anew

# 7.3.3.20 reinitialize()

see base class

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

# 7.3.3.21 sanityCheck()

helper function to check if the object represents sane data (all dimensions compatible and all pointers set if neccesary ) should be called after constructing or manipulating object

#### 7.3.3.22 set B()

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

# 7.3.3.23 set\_C()

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

# 7.3.3.24 set\_Cr0()

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

# 7.3.3.25 set\_r()

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.3.3.26 set r0()

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

### 7.3.3.27 set\_x() [1/3]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_x [inline]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the preceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

# 7.3.3.28 set\_x() [2/3]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

# 7.3.3.29 set\_x() [3/3]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_x
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

#### 7.3.3.30 setupDimension()

makes all vector and matrices to be of appropriate dimension

# 7.3.3.31 setupFromData()

helper in setup

#### 7.3.3.32 setupFromOther() [1/2]

helper in copying from base class (interface)

# 7.3.3.33 setupFromOther() [2/2]

helper in copying

#### 7.3.3.34 show()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
std::string capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::show ( ) const [inline]
```

show info on this set

# 7.3.3.35 storageDimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::storageDimension ( ) const
[inline]
```

see interface

# 7.3.3.36 storageN0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::storageN0 ( ) const [inline]
```

< import dimension from interface

see interface

#### 7.3.3.37 translate()

applies in a smart way to this set X the transform f(y) = X - v

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

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

# 7.4 capd::ddeshelper::BinaryData< T > Union Template Reference

#### **Public Member Functions**

• BinaryData (T v)

#### **Public Attributes**

- T value
- char repr [sizeof(T)]

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

# 7.5 capd::ddeshelper::CoordinateFrame< MatrixSpec, VectorSpec, ScalarSpec > Class Template Reference

#include <DDECoordinateFrameHelper.h>

# **Public Types**

- typedef MatrixSpec MatrixType
- typedef VectorSpec VectorType
- typedef ScalarSpec ScalarType

### **Public Member Functions**

- **CoordinateFrame** (VectorType x, VectorType sec, ScalarType secval, capd::poincare::CrossingDirection d, MatrixType C, MatrixType invC)
- **CoordinateFrame** (VectorType x, MatrixType C, MatrixType invC, capd::poincare::CrossingDirection d=capd::poincare::MinusPlus)
- CoordinateFrame (int M)
- void **setCrossingDirection** (capd::poincare::CrossingDirection d)
- VectorType inCoords (VectorType &x)
- VectorType inGlobal (VectorType &x)
- VectorType **x0** ()
- MatrixType C ()
- MatrixType invC ()
- VectorType sectionVector ()
- ScalarType sectionValue ()

#### **Public Attributes**

- int M
- VectorType reference
- VectorType vsection
- ScalarType secvalue
- capd::poincare::CrossingDirection crossingDirection
- MatrixType coords
- MatrixType inverseCoords
- · std::string filepath

# **Friends**

- std::istream & operator>> (std::istream &in, CoordinateFrame &coords)
- bool operator== (CoordinateFrame &first, CoordinateFrame &second)
- bool operator!= (CoordinateFrame &first, CoordinateFrame &second)

# 7.5.1 Detailed Description

template < typename MatrixSpec, typename VectorSpec = typename MatrixSpec::VectorType, typename ScalarSpec = typename VectorSpec::ScalarType >

class capd::ddeshelper::CoordinateFrame< MatrixSpec, VectorSpec, ScalarSpec >

Internal class to hold a affine coordinate change on a (p,n)-fset.

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

# 7.6 capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::CoordinateSystem Class Reference

#include <DDEHelperRigorous.h>

#### **Public Member Functions**

- CoordinateSystem (Vector x, Vector sec, Scalar secval, capd::poincare::CrossingDirection d, Matrix C, Matrix invC)
- CoordinateSystem (Vector x, Matrix C, Matrix invC, capd::poincare::CrossingDirection d=capd::poincare ← ::MinusPlus)
- CoordinateSystem (int M)
- void **setCrossingDirection** (capd::poincare::CrossingDirection d)
- Vector x0 ()
- Matrix **C** ()
- · Matrix invC ()
- Vector sectionVector ()
- Vector sectionValue ()

### **Public Attributes**

- int M
- · Vector reference
- · Vector vsection
- · Scalar secvalue
- capd::poincare::CrossingDirection crossingDirection
- Matrix coords
- Matrix inverseCoords
- · std::string filepath

#### **Friends**

- std::istream & operator>> (std::istream &in, CoordinateSystem &coords)
- bool operator== (CoordinateSystem &first, CoordinateSystem &second)
- bool operator!= (CoordinateSystem &first, CoordinateSystem &second)

# 7.6.1 Detailed Description

template<typename EqSpec, int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11Rect2Policies>class capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec > ::CoordinateSystem

Internal class to hold a affine coordinate change on a (p,n)-fset.

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

• /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D ← DEHelperRigorous.h

# 7.7 capd::ddeshelper::CubeSet< VectorSpec > Class Template Reference

#### Classes

· class CubeSetIterator

# **Public Types**

- typedef VectorSpec VectorType
- typedef VectorType::ScalarType ScalarType
- typedef VectorType::size type size type
- typedef std::vector< int > CubeSignatureType
- typedef std::set< CubeSignatureType > CSContainerType
- typedef CSContainerType::const iterator CubeSignatureConstIterator
- typedef CSContainerType::iterator CubeSignatureIterator

#### **Public Member Functions**

- CubeSet (VectorType const &x0, VectorType const &r0)
- CubeSet (VectorType const &r0)
- CubeSet & insert (CubeSignatureType const &cube)
- int insert cover (VectorType box, int max cuts=-1)
- CubeSetIterator begin ()
- CubeSetIterator end ()
- CubeSignatureIterator csbegin ()
- CubeSignatureIterator csend ()
- · CubeSignatureConstIterator csbegin () const
- · CubeSignatureConstIterator csend () const
- VectorType get\_x0 () const
- VectorType **get\_r0** () const
- VectorType get\_dx (CubeSignatureType const &signature) const
- bool operator== (CubeSet const &other) const
- · bool subcubeset (CubeSet const &other) const
- template<typename lterator >
   CubeSet & insert (Iterator from, Iterator to)
- int size () const

# **Friends**

- std::ostream & operator<< (std::ostream &out, CubeSet const &items)
- std::istream & operator>> (std::istream &in, CubeSet &items)

#### 7.7.1 Member Function Documentation

# 7.7.1.1 insert\_cover()

Max\_cuts == -1 means we mean cut in all neccessary dimensions. This is default behaviour. returns the last index at what the cut was needed. If it is smaller than max\_cuts requirement then we are ok with the set. If it is greater on equal, then we need to probably change the  $m_r0$ , as the box cannot be covered with the current small boxes.

#### 7.7.1.2 subcubeset()

this only check if x0's, r0's are equal (same cube structure), and that all cubes are in the other

# 7.7.2 Friends And Related Function Documentation

#### **7.7.2.1** operator<<

this only outputs cube signatures and add to the set

# 7.7.2.2 operator>>

this only reads cube signatures and adds to the set

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

• /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D ← DECubeSet.h

# 7.8 capd::ddeshelper::CubeSet< VectorSpec >::CubeSetIterator Class Reference

#### **Public Member Functions**

- CubeSetIterator (CubeSetIterator const &orig)
- operator VectorType () const
- CubeSignatureType const & operator\* () const
- CubeSetIterator & operator++ ()
- CubeSetIterator operator++ (int)

#### **Friends**

- · class CubeSet
- bool operator== (CubeSetIterator const &lhs, CubeSetIterator const &rhs)
- bool operator!= (CubeSetIterator const &lhs, CubeSetIterator const &rhs)

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

# 7.9 capd::ddes::Cubiclkeda< ScalarSpec, ParamSpec > Class Template Reference

#include <SampleEqns.h>

# **Public Types**

- typedef ScalarSpec ScalarType
- typedef unsigned int size\_type
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarType, 0 > VectorType

# **Public Member Functions**

- Cubiclkeda (ParamType a)
- Cubiclkeda (Cubiclkeda const &orig)
- Cubiclkeda (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- Cubiclkeda & operator= (Cubiclkeda const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

# **Static Public Member Functions**

- static size\_type imageDimension ()
- static size type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

# **Protected Attributes**

• ParamType a

# 7.9.1 Detailed Description

```
template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class capd::ddes::CubicIkeda < ScalarSpec, ParamSpec >
```

A model map of what I expect from a Map  $R^{\wedge}m \rightarrow R^{\wedge}n$  to be good for use with DDE codes.

# 7.9.2 Member Function Documentation

#### 7.9.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >::dimension ( ) [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

#### 7.9.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >::getParamsCount ( ) [inline],
[static]
```

number of parameters to fully configure equation

# 7.9.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >::imageDimension ( ) [inline],
[static]
```

output dimension of the internal map

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

# 7.10 capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec > Class Template Reference

#include <FunctionalMap.h>

# **Public Types**

- typedef SolutionCurveSpec CurveType
- typedef JetSpec JetType
- typedef CurveType::RealType RealType
- typedef CurveType::TimePointType
- typedef CurveType::MatrixType MatrixType
- typedef CurveType::VectorType VectorType
- typedef CurveType::ScalarType ScalarType
- typedef CurveType::size\_type size\_type
- typedef CurveType::DataType DataType
- typedef std::vector< DataType > VariableStorageType
- typedef std::vector< VectorType > ValueStorageType
- typedef std::vector< std::vector< MatrixType >> JacobianStorageType

#### **Public Member Functions**

- virtual size type imageDimension () const =0
- virtual VectorType operator() (const TimePointType &t0, const CurveType &x) const =0
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out\_u, size\_type &out\_admissible\_order) const =0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const =0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const =0
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual VectorType operator() (const CurveType &x) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual ~DDEBasicFunctionalMap ()

#### Static Public Member Functions

- static void convert (VariableStorageType const &u, ValueStorageType &v)
- static void deconvert (ValueStorageType const &u, VariableStorageType &v)

#### **Protected Member Functions**

virtual void checkCurveDimension (CurveType const &x, std::string extra="") const

# 7.10.1 Detailed Description

template<typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::JetType> class capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >

• TODO: rewrite DOCS this is the interface of a Functional map, that can be evaluated only by "querying" the curve about its jets at various points in the past. it can produce jet at a given time t together with partial derivatives w.r.t. to internal variables from curve used in the computation by recurrent formula for a functional equation:

```
x'(t) = f(t, x)
```

where x is the solution curve in \R^d defined for times smaller than t (up to some maximal past time usually).

A good example of such a functional map is discrete delay case, e.g. F(t, x) = f(t, x(t), x(t-tau)),  $f : R \times R^{2d} > R$  (easily extended to more delays). (Probably) the functionals that include integral over the past: F(t, x) = f(t, x) (int  $\{t-tau\}^{t} x(s) = t(t, x)$ ) will also fall into this category.

The concrete implementations will be available later.

Implementation note: when implementing concrete class of this interface, do not forget to put using BaseClass.::operator(); using BaseClass::computeDDECoefficients; in the public part of your class to gain access to default implementations of some functions.

TODO: (NOT URGENT, FUTURE, RETHINK) it would be better not to pass containers to computeDDECoefficients, but maybe iterators to given types? It could be used to compute in a given place, i.e. at an already created Jet?

# 7.10.2 Member Typedef Documentation

# 7.10.2.1 JacobianStorageType

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> typedef std::vector< std::vector<MatrixType> > capd::ddes::DDEBasicFunctionalMap< Solution← CurveSpec, JetSpec >::JacobianStorageType

This type will store partial derivatives with respect to variables from input curve used in the computations. Each matrix will be of M(d, d) type.

#### 7.10.2.2 ValueStorageType

 $\label{template} $$ \textbf{typename SolutionCurveSpec::} \textbf{JetType>typedef std::} \textbf{vector< VectorType> capd::} \textbf{ddes::} \textbf{DDEBasicFunctionalMap} < \textbf{SolutionCurveSpec, Jet} \leftarrow \textbf{Spec>::} \textbf{ValueStorageType} $$$ 

similar to VariableStorageType but does not care about the internal representation. It will be used to store the Jet coefficients only in output (we do not need to care about the time point and we can freely change order (i.e. size of the ValueStorageType)

# 7.10.2.3 VariableStorageType

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef std::vector< DataType > capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec
>::VariableStorageType

This type will store all variables from the input curve used to evaluate the map and to produce recurrently the jet at a given time. It will be stored in native SetType to allow usage of Lohner-type algorithms for controlling wrapping effect later. Each element of this collection would be a set in \R^d d dimensional space.

#### 7.10.3 Constructor & Destructor Documentation

#### 7.10.3.1 ∼DDEBasicFunctionalMap()

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::~DDEBasicFunctionalMap
( ) [inline], [virtual]

virtual destructor for warning suppresion

#### 7.10.4 Member Function Documentation

# 7.10.4.1 checkCurveDimension()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::checkCurve \leftarrow Dimension (

CurveType const & x,

std::string extra = "") const [inline], [protected], [virtual]
```

helper function to check. TODO: (NOT URGENT, FUTURE) make a switch to turn this off for speed.

#### 7.10.4.2 collectComputationData()

Collects all data from curve necessary for the computation of the value. This does not collect ENCLOSURE DATA required in rigorous computations.

We use collection of DataType items as the output, we ASSUME that DataType elements are used to describe function by CurveType. Therefore, in the output, we could relate how inputs (VariableStorage) corresponds to outputs (ValueStorage) mainly by Jacobian of the map at Variables (JacobianStorageType).

Then, the main function that needs to be implemented in the Map are those depending on the VariableStorageType. Other can have default implementations and are for users convenience.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Note: dt might seem redundant here, but for some reasons it might be helpful, i.e. for epsilon steps (see paper) Note: dt should be by definition dt = th - t0. Note: This function is for optimization purposes. I could assume that the whole Curve goes into computation, but then the computing cost would grow, as Curve grows in time. Also, if some coefficients are not used in rhs of the equation, we would carry out the unnecessary computations (i.e. Jac Phi would be 0 and we would do 0 \* C \* r0). There is one problem with this however: in rigorous code we need to check if the section is not crossed between steps in Poincare map. Please see comments in JetSection / DDEPoincareMap classes.

Param: out\_u a finite-dimensional collection of data that is used to compute the map Param: admissible\_order will tell how high order of expansion is able to produce with data in u

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

# 7.10.4.3 computeDDECoefficients() [1/6]

this is for a current time in the solution. It provides basic implementation by call to the other function.

# 7.10.4.4 computeDDECoefficients() [2/6]

this is for a current time in the solution. It provides basic implementation by call to the other function.

#### 7.10.4.5 computeDDECoefficients() [3/6]

```
\verb|virtual void capd::ddes::DDEBasicFunctionalMap| < SolutionCurveSpec, | JetSpec > :: computeDDE \leftarrow | SolutionCurveSpec | JetSpec > :: computeDDE | JetSpec > :: computeDDE | JetSpec | J
Coefficients (
                                                                                           const RealType & t0,
                                                                                            const ValueStorageType & u,
                                                                                             ValueStorageType & coeffs ) const [pure virtual]
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true. For non-developer users of the library it is possible to use versions of the function for specific CurveType.

#### THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out\_admissible\_order from collectComputationData() is used for this purpose.

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

#### 7.10.4.6 computeDDECoefficients() [4/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↔
Coefficients (
            const RealType & t0,
            const ValueStorageType & u,
            ValueStorageType & coeffs,
            JacobianStorageType & Du ) const [pure virtual]
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It also computes \$\frac{\partial coeffs[i]}{\partial u}\$ (in Du). It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true.

#### THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out admissible order from collectComputationData() is used for this purpose.

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

# 7.10.4.7 computeDDECoefficients() [5/6]

computes recursively the Jet at time t for a given curve for a DDE of the form:

```
x'(t) = F(t, x)
```

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. If order is 0 or order is too high for a given curve then the function should alter the order to highest possible. (use coeffs. resize(order) for this purpose)

THERE IS DEFAULT IMPLEMENTATION OF THIS WITH collectComputationData() and computeDDE ← Coefficients(..., u, ...) call to pure virtual function.

#### 7.10.4.8 computeDDECoefficients() [6/6]

Same as computeDDECoefficients(const TimePointType&, const CurveType&, JetType&), but also computes the partial derivative:

```
Du[k] = \frac{\pi u}{\pi u}
```

where u are all the variables used to evaluate the map. In u[j] is a d-dimensional set used in computation. In Du[k][j] is a matrix of dimension M(d,d) and of course it is

It is done this way to allow Solvers to use this structure to reduce wrapping effect of interval arithmetics with the help of Lohner algorithm.

#### 7.10.4.9 convert()

helper to convert from collection of sets to collection of vectors

# 7.10.4.10 deconvert()

```
static void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::deconvert (
        ValueStorageType const & u,
        VariableStorageType & v ) [inline], [static]
```

helper to convert from collection of sets to collection of vectors NOTE: name is changed, as ValueStorageType might be VariableStorageType in some instances, and this would make compilation ambiguous.

#### 7.10.4.11 imageDimension()

```
virtual size_type capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::image↔
Dimension ( ) const [pure virtual]
```

output dimension of the map. THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implemented in capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

# 7.10.4.12 operator()() [1/2]

```
virtual VectorType capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::operator()
(
       const CurveType & x ) const [inline], [virtual]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

# 7.10.4.13 operator()() [2/2]

```
virtual VectorType capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::operator()
        const TimePointType & t0,
        const CurveType & x ) const [pure virtual]
```

computes value of the map at a given time point for a given solution curve. THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

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

 /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Functional Map.h

# 7.11 capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec > Class Template Reference

#include <DDEBasicPoincareMap.h>

# **Public Types**

- typedef DDEBasicPoincareMap< DynSysSpec, SectionSpec > Class
- typedef DynSysSpec DynSysType
- typedef SectionSpec SectionType
- typedef DynSysSpec::CurveType CurveType
- typedef CurveType::TimePointType TimePointType
- typedef CurveType::VectorType VectorType
- typedef CurveType::MatrixType MatrixType
- typedef CurveType::ScalarType ScalarType
- typedef CurveType::RealType RealType
- typedef DynSysType::JacobianStorageType JacobianStorageType
- typedef CurveType::size\_type size\_type

#### **Public Member Functions**

- DDEBasicPoincareMap (DDEBasicPoincareMap const &other)
- DDEBasicPoincareMap (DynSysType &dynsys, SectionType &section, CrossingDirection direction=Crossing 

  Direction::Both, int reqSteps=0, int maxSteps=-1, double binsearchEpsilon=DEFAULT\_BINSEARCH\_EP 

  SILON)
- void operator() (CurveType &curve, CurveType &on\_section, RealType &out\_approachTime)
- void operator() (CurveType &in\_out\_curve, CurveType &in\_out\_Pcurve, RealType &out\_approachTime, VectorType &out\_x, VectorType &out\_Px, VectorType &out\_fPx, MatrixType &out\_V, MatrixType &out\_DP)
- void setInitialV (CurveType &in\_out\_curve)
- void setInitialV (CurveType &in\_out\_curve, MatrixType const &V)
- void setCurrentV (CurveType &in\_out\_curve, MatrixType const &V)
- CurveType operator() (CurveType const &X)
- DDEBasicPoincareMap & **setDirection** (CrossingDirection direction)
- CrossingDirection getDirection ()
- DDEBasicPoincareMap & setMaxSteps (int maxSteps)
- int getMaxSteps ()
- int getMaximumSteps ()
- DDEBasicPoincareMap & setRequiredSteps (int requiredSteps)
- int getRequiredSteps ()
- void integrateUntilSectionCrossing (CurveType &curve, RealType &timeBeforeSection, SingleStepFn stepFn)
- void findCrossingTime (CurveType const &curve, CurveType &requested, RealType &crossingTime, int test
   —
   Direction)
- CrossingDirection detectCrossingDirection (CurveType const &curve)
- RealType getLastEpsilonTime () const
- RealType getLastTimeBeforeSection () const
- RealType getLastReachTime () const
- JacobianStorageType **getLastVariational** () const
- int getLastStepsAfterSection ()
- bool isNormalizeVariational () const
- Class & setNormalizeVariational (bool value)

#### **Protected Member Functions**

- void checkSteps ()
- void extractVariationalMatrix (CurveType const &curve, MatrixType &fullV, MatrixType &reducedV, std
   ::vector < size\_type > reducedShape, int p\_howFar=-1) const
- void extractVariationalMatrix (CurveType const &curve, MatrixType &fullV) const

# **Protected Attributes**

- DynSysType & m\_dynsys
- SectionType & m\_section
- · CrossingDirection m direction
- · int m\_requiredSteps
- int m\_maxSteps
- · int m steps
- double m\_binsearchEpsilon
- RealType m\_lastTimeBeforeSection
- RealType m\_lastEpsilonTime
- JacobianStorageType m\_variational
- bool m\_normalizeVariational
- std::vector< size\_type > m\_storedInitialShape

# 7.11.1 Detailed Description

```
template<typename DynSysSpec, typename SectionSpec> class capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >
```

Implementation of Poincare Map for use with nonrigorous DDESolvers.

In theory should work with any compatible DDEDynSys and Section.

# 7.11.2 Member Function Documentation

## 7.11.2.1 detectCrossingDirection()

warning: might be time consuming, it (might) integrate the solution... it assumes curve is on section.

#### 7.11.2.2 findCrossingTime()

testDirection = -1 for before section testDirection = +1 for after section binsearch for now

#### 7.11.2.3 integrateUntilSectionCrossing()

it extends the solution until the crossing with the section. Also, it takes into consideration the required number of steps (should be set-up before) It also moves away of section is initially curve crosses the section.

WARNING: it does not check for sanity of argument as of now, so it might cause runtime-errors if the proof is not well prepared.

## 7.11.2.4 operator()() [1/3]

in curve there is a curve that reaches first full step after the section in on\_section there is the image on the section, as close to section as possible on\_section must be initialized with the desired "length" (time interval) and "order" of the representation. on section must be compatible (length, order) with the section

#### 7.11.2.5 operator()() [2/3]

For other parameters and base description, see the other operator()

This is more complicated than the case of ODEs, but for technical reasons it is most desirable to work with vector/matrix representations of the objects when dealing with variational equation on the coefficients.

NOTE: CurveType must be defined with DataType that allows to handle extra Matrix data. I have special structures for this. CHeck examples and helper classes. Thanks to C++ lazy evaluation of templates this also works on the basic data structure as long as you do not call the function in the program.

out\_x is vector representation of input curve out\_Px is vector representation of the image on the section (Pcurve) out\_fPx is the value of the "Vector Field" of the equation as defined in the Banach space in practice, if c = Px is a curve in C([-tau, 0], \R^d), then out\_fPx = c'. This is theoretically correct when c is on the C^1 solutions manifold, i.e.  $c(0^{-}) = f(c)$  where f is the r.h.s. of the DDE. This is true for any initial curve, if the reachTime > delay. See works for more explanation. out\_V is the variational equation solution on the coefficients, i.e. d\varphi(t\_p, x)/dx (in sense of the coefficients defining x) therefore it is of dimension out\_x.dimension() x out\_x.dimension() out\_DP is the dP/dx (x) (derivative of P w.r.t. initial data, computed at point x) out\_DP is obtained from out\_V by some correction on the reach time t\_p(x) in the neighbourhood of x. i.e. dP/dx (x) =  $d(varphi(t_p(x), x))/dx = varial(varphi)/varial x (t_p(x), x) +$ 

• \partial(\varphi)\partial t (t\_p(x), x) \* d(t\_p)/dx (x) = V + f(Px) \* dt\_p/dx (x) Please note that f(Px) is "vertical" and dt\_p/dx (x) is "horizontal" vector so their product is a full matrix of desired dimension! We see that f(Px) is in out fPx, V in out V, then the last term is obtained by differentiation of the section condition equation:

```
s(\operatorname{varphi}(t_p(x), x)) == 0
```

so we get (\* means scalar product in  $\R^M$  where needed,  $M = out_x.dimension()$ )

```
\label{eq:continuous} $$ \left(V + f(Px) * dt_p/dx (x)\right) == 0 \text{ so } dt_p/dx (x) = - \left(\left\{ gradient s \right\} * V \right) / \left(\left\{ gradient s \right\} * f(Px) \right) $$
```

Note, that ( $\{\gradient\ s\} * f(Px)$ ) is simply a scalar, and it needs to be not equal 0 for the formula to make sense. This is usual notion of the section to be transversal to the flow at x.

Finally, note that x, Px, fPx can be used in .set\_x() to a curve of a proper structure (important!) to get interesting data. The proper structure is stored in Pcurve, i.e. usually, one can do:  $curve_x = Pcurve$ ;  $curve_x.set_x(out_x)$ ; // guaranteed:  $curve_x = Curve_x.set_x(out_px)$ ; // guaranteed:  $curve_px = Pcurve_x.set_x(out_px)$ ; // guaranteed:  $curve_x.set_x(out_px)$ 

IMPORTANT: curve\_fPx will be of one order higher than simply trying to compute Pcurve.dt()! IMPORTANT: as .dt() loses one order. So Pcurve.dt() is not a proper way to get "force field" IMPORTANT: on curve\_Px, as the dimensions will be different (orders different). IMPORTANT: Therefore, use curve\_fPx to get the value of the "force field" on curve Px.

## 7.11.2.6 operator()() [3/3]

to be compatible with standard notion and CAPD interface, It makes following assumptions:

• X and PX will have the same structure (i.e. over same grid points and jets of the same order).

NOTE: you can retrieve information on reach time and epsilon step, but you cannot acquire the solution on the whole time. If you need solution over the full integration time, then use other operator().

# 7.11.2.7 setCurrentV()

Experimental...

# 7.11.2.8 setInitialV() [1/2]

helper function, sets to Id the Variational matrix w.r.t. initial coefficients in curve TODO: DRY

#### 7.11.2.9 setInitialV() [2/2]

sets initial V to a given Matrix. Matrix must be of a good shape TODO: DRY

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

# 7.12 capd::ddeshelper::DDECompareHelper< VectorSpec > Class Template Reference

```
#include <DDECompareHelper.h>
```

# **Public Types**

- typedef VectorSpec VectorType
- typedef VectorType::size\_type size\_type
- typedef VectorType::ScalarType ScalarType

#### **Public Member Functions**

- **DDECompareHelper** (VectorType const &orig, VectorType const &other)
- void printRelation (int i, std::ostream &out) const
- · void printRelations (std::ostream &out) const
- · void printSummary (std::ostream &out) const
- std::string inlineSummary () const
- · operator std::string () const
- · int getSubsetsCount () const
- int getSupsetsCount () const
- int getMissLeftCount () const
- int getMissRightCount () const
- · int getMissCount () const
- · int getIntersectionLeftCount () const
- int getIntersectionRightCount () const
- int getIntersectionsCount () const
- int getUnknownsCount () const
- bool isSubset (size\_type i) const
- bool isSupset (size\_type i) const
- bool isMissLeft (size\_type i) const
- bool isMissRight (size\_type i) const
- bool isMiss (size type i) const
- bool isIntersectLeft (size\_type i) const
- bool isIntersectRight (size\_type i) const
- bool isIntersection (size\_type i) const
- · bool isUnknown (size type i) const

#### **Friends**

• std::ostream & operator<< (std::ostream &out, DDECompareHelper< VectorSpec > const &comp)

# 7.12.1 Detailed Description

```
template<typename VectorSpec> class capd::ddeshelper::DDECompareHelper< VectorSpec >
```

allows to easily check how the other vector relates to the original.

# 7.12.2 Member Function Documentation

## 7.12.2.1 getUnknownsCount()

```
template<typename VectorSpec >
int capd::ddeshelper::DDECompareHelper< VectorSpec >::getUnknownsCount ( ) const [inline]
```

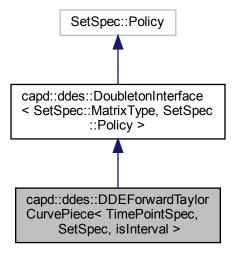
this should always return 0, otherwise some serious errors happen

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

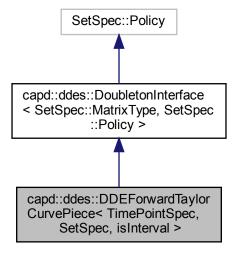
# 7.13 capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > Class Template Reference

#include <DDEForwardTaylorCurvePiece.h>

Inheritance diagram for capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >:



Collaboration diagram for capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >:



# **Public Types**

- typedef SetSpec DoubletonStorageType
- typedef SetSpec SetType
- typedef SetType::MatrixType MatrixType
- typedef SetType::Policy Policy
- typedef SetType::QRPolicy QRPolicy
- typedef DDEForwardTaylorCurvePiece < TimePointSpec, SetSpec, isInterval > Class
- typedef DoubletonInterface < MatrixType, Policy > BaseClass
- typedef SetType::VectorType VectorType
- typedef SetType::ScalarType ScalarType
- typedef SetType::size type size type
- typedef ScalarType RealType
- typedef TimePointSpec TimePointType
- typedef Class JetType
- typedef DoubletonStorageType \* iterator
- typedef capd::ddes::GenericJet < TimePointType, VectorType, VectorType, MatrixType > ExternalJetType

# **Public Member Functions**

- DDEForwardTaylorCurvePiece & operator= (DDEForwardTaylorCurvePiece const &orig)
- DDEForwardTaylorCurvePiece (TimePointType t0=TimePointType())
- DDEForwardTaylorCurvePiece (DDEForwardTaylorCurvePiece const &orig)
- DDEForwardTaylorCurvePiece (TimePointType t0, size type dimension, size type order)
- DDEForwardTaylorCurvePiece (TimePointType t0, size\_type order, const VectorType &v, size\_type N0=0)
- DDEForwardTaylorCurvePiece (TimePointType t0, size\_type order, const VectorType &v, VectorType \*r0, bool passOwnership=false)
- DDEForwardTaylorCurvePiece (TimePointType t0, VectorType const \*it, VectorType const \*itEnd)
- DDEForwardTaylorCurvePiece (TimePointType t0, std::vector< VectorType > coeffs)
- DDEForwardTaylorCurvePiece (TimePointType t0, size\_type order, SetType &value)
- DDEForwardTaylorCurvePiece (TimePointType t0, size\_type order, const SetType &value)
- DDEForwardTaylorCurvePiece (TimePointType t0, std::vector< VectorType > coeffs, std::vector< Matrix← Type > Cs, VectorType \*r0, bool passR0Ownership, std::vector< MatrixType > Bs, std::vector< VectorType > rs, VectorType \*Xi, bool passXiOwnership)
- DDEForwardTaylorCurvePiece (TimePointType t0, std::vector< VectorType > coeffs, std::vector< Matrix← Type > Cs, VectorType \*r0, bool passR0Ownership, std::vector< MatrixType > Bs, std::vector< MatrixType > invBs, std::vector< VectorType > rs, VectorType \*Xi, bool passXiOwnership)
- size\_type dimension () const
- size\_type order () const
- size\_type storageDimension () const
- size\_type storageN0 () const
- VectorType get\_x () const
- MatrixType get\_C () const
- MatrixType get\_B () const
- VectorType get r () const
- VectorType get r0 () const
- BaseClass & set x (VectorType const &x)
- BaseClass & set\_C (MatrixType const &C)
- BaseClass & set\_r0 (VectorType const &r0)
- BaseClass & set Cr0 (MatrixType const &C, VectorType const &r0)
- BaseClass & set\_Binv (MatrixType const &Binv)
- BaseClass & set\_B (MatrixType const &B)
- BaseClass & set\_r (VectorType const &r)
- virtual Class & mul (ScalarType const &c)

- < see Doubleton interface
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- ScalarType dot (ExternalJetType const &s) const
- DDEForwardTaylorCurvePiece dt (size\_type n=1) const
- VectorType evalAtDelta (const RealType &delta\_t) const
- VectorType eval (const RealType &t) const
- VectorType evalCoeffAtDelta (size\_type n, const RealType &delta\_t) const
- VectorType evalCoeff (size\_type n, const RealType &t) const
- VectorType taylorAtDelta (const RealType &delta\_t) const
- VectorType taylor (const RealType &t) const
- VectorType summaAtDelta (const RealType &delta\_t) const
- VectorType summa (const RealType &t) const
- JetType jetAt (const TimePointType &t) const
- JetType jetAt (const RealType &t) const
- void taylorAtDelta (const RealType &delta t, SetType &out) const
- void taylor (const RealType &t, SetType &out) const
- void evalAtDelta (const RealType &delta\_t, SetType &out) const
- void eval (const RealType &t, SetType &out) const
- void evalCoeffAtDelta (size\_type n, const RealType &delta\_t, SetType &out) const
- void evalCoeff (size type n, const RealType &t, SetType &out) const
- DDEForwardTaylorCurvePiece midCurve () const
- bool isMidCurve () const
- std::string show () const
- iterator beginJet ()
- iterator endJet ()
- iterator backJet ()
- const\_iterator beginJet () const
- const iterator endJet () const
- · const iterator backJet () const
- DoubletonStorageType & operator[] (size\_type k)
- DoubletonStorageType const & operator[] (size type k) const
- Class & setAsConstant (SetType &value)
- Class & setAsConstant (const SetType &value, VectorType \*overwrite r0=NULL, bool passOwnership=false)
- Class & setAsConstant (VectorType const &value, size type N0=0)
- Class & setAsConstant (VectorType const &value, VectorType \*r0, bool passOwnership=false)
- Class & setT0 (TimePointType const &t0)
- TimePointType getT0 () const
- TimePointType t0 () const
- void set\_Xi (VectorType const &xi)
- void set\_Xi (VectorType \*xi, bool passOwnership=false)
- VectorType get\_Xi () const
- VectorType \* take\_Xi ()
- VectorType \* take r0 ()
- Class & set r0 (VectorType \*r0, bool passOwnership=false)
- virtual ~DDEForwardTaylorCurvePiece ()
- virtual void reinitialize (size\_type d, size\_type N0)
- virtual VectorType makeStorage\_x () const
- virtual MatrixType makeStorage C () const
  - < see Doubleton interface
- virtual VectorType makeStorage r0 () const
  - < see Doubleton interface
- virtual MatrixType makeStorage\_B () const
  - < see Doubleton interface

- virtual VectorType makeStorage\_r () const
  - < see Doubleton interface
- virtual VectorType hull () const
  - < see Doubleton interface
- virtual VectorType midPoint () const
  - < see Doubleton interface
- virtual ScalarType dot (VectorType const &v) const
  - < see Doubleton interface

# **Static Public Member Functions**

• static std::string badge ()

#### **Public Attributes**

const typedef DoubletonStorageType \* const\_iterator

# **Protected Member Functions**

- void dimCheck (const VectorType \*const v)
- void dimCheckC (const MatrixType \*const C)
- void dimCheckB (const MatrixType \*const B)
- void deallocateJet ()
- void deallocateXi ()
- void deallocateR0 ()
- void deallocate ()
- · void allocateJet ()
- void allocateXi ()
- void allocateR0 (size\_type N0=0)
- void allocate (size\_type N0=0)
- void reallocateJet ()
- void reallocateXi ()
- void reallocateR0 (size\_type N0=0)
- void reallocate (size\_type N0=0)
- void updateCommonR0 ()
- template<typename lteratorType > void setupFromData (IteratorType it, IteratorType itEnd)
- template < typename XItSpec , typename CItSpec , typename BItSpec , typename RItSpec >
   void setupFromData (XItSpec xit, XItSpec xend, CItSpec Cit, CItSpec Cend, VectorType \*r0, BItSpec Bit,
   BItSpec Bend, BItSpec invBit, BItSpec invBend, RItSpec rit, RItSpec rend, VectorType \*Xi)

#### **Protected Attributes**

- TimePointType m\_t0
- size\_type m\_dimension
- size\_type m\_order
- DoubletonStorageType \* m\_jet\_at\_t0
- VectorType \* m\_r0
- bool m\_r0\_owner
- VectorType \* m\_Xi
- bool m\_Xi\_owner

#### **Friends**

- std::ostream & operator<< (std::ostream &out, DDEForwardTaylorCurvePiece const &jet)
- std::istream & operator>> (std::istream &in, DDEForwardTaylorCurvePiece &jet)

#### 7.13.1 Detailed Description

template<typename TimePointSpec, typename SetSpec, bool isInterval = capd::TypeTraits<typename SetSpec::MatrixType::
ScalarType>::isInterval>

 ${\it class\ capd::} {\it ddes::} {\it DDEForwardTaylorCurvePiece} < {\it TimePointSpec,\ SetSpec,\ isInterval} >$ 

A class to represent a part of some curve \$x\$ over \$[t0, t1)\$ given by:

```
(J^{\{N\}}_{t0}x)(t) + (n+1) * \int_{t0}^{t} xi(s) (t0-s)^{N} ds
```

where  $xi(s) \in R \quad \text{mathbb}{R}^d$ 

in the code below: d = dimension n = order

If the interval version is used, then the class can represent sets of Curves, so that  $J^{[N]}_{10} \le 10 \le r$ , (x is here a middle point of the set and Jet is interpreted as a collection of coefficients). \$r0\$ is some vector of any given dimension, compatible with \$A\$. We denote the dimension of \$r0\$ as \$N0\$. \$Q\$ is a diagonal block matrix, so that \$Q\_ii\$ is orthonormal \$d \times matrix and and \$Q\$ is 0 elsewhere. Vector \$r\$ is of dimension \$d \* n\$. we store matrices row-wise, ie in the code below m\_A[k] represents a \$d \times N0\$ matrix and m\_Q[k] is \$Q\_kk\$ a \$d \times matrix. Finally, we store \$r\$ and \$x\$ again component-wise, so that in code below m\_r[k] and \$m\_x[k]\$ is a \$d\$-dimensional vector. Therefore we have: \$(J^{[N]}\_{10}x)\_{10} \times J^{[k]} \times m\_x[k] + m\_A[k] \* r0 + m\_Q[k] \* m\_r[k]\$

NOTE: we store \$m\_r0\$ as a one \$N0\$-dimensional vector, but for optimization purposes we hold the variable as a pointer - to allow sharing of this vector among many instances (important in DDESolutionCurve)

TODO: (NOT URGENT) extract abstract interface that has eval, taylor, summa, etc? TODO: (NOT URGENT) base interface is already done as GenericJet, it has taylor (or eval). TODO: (NOT URGENT) This should inherit Jet structure from there and add summa and taylor, and reload eval.

#### 7.13.2 Constructor & Destructor Documentation

# 7.13.2.1 DDEForwardTaylorCurvePiece() [1/11]

creates an empty curve of order 0 and dimension 0, by default at the point \$t\_0=0\$

# 7.13.2.2 DDEForwardTaylorCurvePiece() [2/11]

copy constructor. if orig has a shared r0 and/or Xi then it will share it with the copy (this) Otherwise, it assures that the r0 / Xi is copied accordingly and the jet updated if necessary.

#### 7.13.2.3 DDEForwardTaylorCurvePiece() [3/11]

makes a constant function of the vector value 0

# 7.13.2.4 DDEForwardTaylorCurvePiece() [4/11]

makes a constant function of the vector value v

# 7.13.2.5 DDEForwardTaylorCurvePiece() [5/11]

makes a constant function of the vector value v

# 7.13.2.6 DDEForwardTaylorCurvePiece() [6/11]

fills vector with data. Determine order from the size of data.

# 7.13.2.7 DDEForwardTaylorCurvePiece() [7/11]

fills vector with data. Determine order from the size of data.

#### 7.13.2.8 DDEForwardTaylorCurvePiece() [8/11]

makes a curve piece of constant function, located at t0, of a given order and of given value. The r0 will be taken from the sets r0 (and no ownership transfer occurs).

#### 7.13.2.9 DDEForwardTaylorCurvePiece() [9/11]

makes a curve piece of constant function, located at t0, of a given order and of given value. The r0 will be taken from the sets r0 (as copy).

#### 7.13.2.10 DDEForwardTaylorCurvePiece() [10/11]

almost complete constructor, setups everything from scratch (computes invB)

most complete constructor, setups everything from scratch

# 7.13.2.12 ~DDEForwardTaylorCurvePiece()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
virtual capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::~DDEForwardTaylorCurvePiece
( ) [inline], [virtual]
```

standard thing

# 7.13.3 Member Function Documentation

## 7.13.3.1 affineTransform()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename SetSpec::MatrixType::ScalarType>::isInterval>

BaseClass& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↔

::affineTransform (

    MatrixType const & M,

    VectorType const & v ) [inline]
```

see doubleton interface

# 7.13.3.2 allocate()

technical, manages allocation of resources

#### 7.13.3.3 allocateJet()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::allocateJet
( ) [inline], [protected]
```

technical, manages deallocation of resources

#### 7.13.3.4 allocateR0()

technical, manages allocation of resources

#### 7.13.3.5 allocateXi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::allocateXi
( ) [inline], [protected]
```

technical, manages allocation of resources

#### 7.13.3.6 backJet() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::backJet
( ) [inline]
```

iterator to the last element in coefficients. Standard thing in C++

# 7.13.3.7 backJet() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
const_iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >
::backJet ( ) const [inline]
```

iterator to the last element in coefficients. Standard thing in C++

#### 7.13.3.8 badge()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
static std::string capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval
>::badge ( ) [inline], [static]
```

should be one word

#### 7.13.3.9 beginJet() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::begin←
Jet ( ) [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

# 7.13.3.10 beginJet() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename SetSpec::MatrixType::ScalarType>::isInterval> const_iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > ← ::beginJet ( ) const [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

#### 7.13.3.11 deallocate()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate
( ) [inline], [protected]
```

technical, manages deallocation of resources

#### 7.13.3.12 deallocateJet()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate
Jet ( ) [inline], [protected]
```

technical, manages deallocation of resources

# 7.13.3.13 deallocateR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate
R0 () [inline], [protected]
```

technical, manages deallocation of resources

#### 7.13.3.14 deallocateXi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate
Xi () [inline], [protected]
```

technical, manages deallocation of resources

# 7.13.3.15 dimCheck()

checks if the dimension of v is compatible with current set

# 7.13.3.16 dimCheckB()

checks if the dimension of B is compatible with current set B matrix

# 7.13.3.17 dimCheckC()

checks if the dimension of C matrix is compatible with current set C matrix (might be swapped)

## 7.13.3.18 dimension()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::dimension
( ) const [inline]
```

dimension of the value space of the Jet (i.e.  $J : R \to R^{(dimension)}$ )

#### 7.13.3.19 dot()

computes dot product of this with JetType (does not take Xi part into consideration!)

This is a simple dot product in vector space. Consider implementing some other like integral (but it should be equivalent more or less becouse of polynomials)

#### 7.13.3.20 dt()

returns Forward Taylor Jet of the derivative of order n w.r.t. time, n must be smaller than the order

#### 7.13.3.21 endJet() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::endJet
( ) [inline]
```

iterator to the past the order place. Standard thing in C++

#### 7.13.3.22 endJet() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
const_iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval>
::endJet ( ) const [inline]
```

iterator to the past the order place. Standard thing in C++

# 7.13.3.23 eval() [1/2]

evaluates the jet at time t (please note that it is true time, not some small h w.r.t. jet.t0().

#### 7.13.3.24 eval() [2/2]

evaluates the jet at time t0 + delta\_t, delta\_t should be positive! Note: out should be set representing  $[0,0]^{\hat{}}d$ . Procedure do not test for it!

#### 7.13.3.25 evalAtDelta() [1/2]

evaluates the jet at time t0 + delta t, delta t should be positive!

#### 7.13.3.26 evalAtDelta() [2/2]

evaluates the jet at time  $t0 + delta_t$ ,  $delta_t$  should be positive. Note: out should be set representing  $[0,0]^{\wedge}d$ . Procedure do not test for it!

#### 7.13.3.27 evalCoeff() [1/2]

helper. Evaluates n-th derivative (more precisely  $j^{n}[n] = j^{n}(n)/n!$  - it is used in the algorithm most often) of this jet at t. It should be faster than jet.dt(n).eval(t)/n!

# 7.13.3.28 evalCoeff() [2/2]

evaluates the jet at time t0 + delta\_t, delta\_t should be positive! Note: out should be set representing  $[0,0]^{\hat{}}d$ . Procedure do not test for it!

# 7.13.3.29 evalCoeffAtDelta() [1/2]

helper. Evaluates n-th derivative (more precisely  $j^{n}[n] = j^{n}(n)/n!$  - it is used in the algorithm most often) of this jet at t. It should be faster than jet.dt(n).evalAtDelta(t)/n!. The delta\_t should be positive!

# 7.13.3.30 evalCoeffAtDelta() [2/2]

evaluates the jet at time t0 + delta\_t, delta\_t should be positive!

• Note: out should be set representing [0,0]<sup>^</sup>d. Procedure do not test for it!

#### 7.13.3.31 get\_B()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::MatrixType capd::ddes::DDEForwardTaylorCurvePiece<, isInterval >::get_B
```

see doubleton interface

# 7.13.3.32 get\_C()

see doubleton interface

# 7.13.3.33 get\_r()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurveTimePointSpec, SetSpec, isInterval >::get_r
```

see doubleton interface

#### 7.13.3.34 get r0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurvePiece, SetSpec, isInterval >::get_r0
```

see doubleton interface

# 7.13.3.35 get\_x()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurvePiece<, setSpec, isInterval >::get_x
```

see doubleton interface

# 7.13.3.36 get Xi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::get 
_Xi ( ) const [inline]
```

returns a value of Xi

## 7.13.3.37 getT0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
TimePointType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >
::getT0 () const [inline]
```

returns time at which this jet is located

# 7.13.3.38 isMidCurve()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
bool capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::isMidCurve
( ) const [inline]
```

returns true, if the diamater of Taylor part is 0

#### 7.13.3.39 jetAt() [1/2]

returns jet at t for a function represented by this. It is rigorous, so it returns validated estimates (overestimates)

#### 7.13.3.40 jetAt() [2/2]

returns jet at t for a function represented by this. It is rigorous, so it returns validated estimates (overestimates)

# 7.13.3.41 makeStorage\_x()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_x
[inline]
```

makes a vector that can store x part

#### 7.13.3.42 midCurve()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > capd::ddes::DDEForwardTaylorCurvePiece<
TimePointSpec, SetSpec, isInterval >::midCurve
```

returns jet with Taylor part of diameter 0 and the same Xi

## 7.13.3.43 mul()

< see Doubleton interface

multiply set by a scalar. Should take set structure into consideration.

#### 7.13.3.44 operator=()

copy operator. Works as in case of copy constructor (see doc there)

# 7.13.3.45 operator[]() [1/2]

returns n-th order Taylor coefficient.

# 7.13.3.46 operator[]() [2/2]

returns n-th order Taylor coefficient.

#### 7.13.3.47 order()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::order
( ) const [inline]
```

order of the Taylor part of the Jet

#### 7.13.3.48 reallocate()

technical, manages re-allocation of resources

#### 7.13.3.49 reallocateJet()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::reallocate←
Jet ( ) [inline], [protected]
```

technical, manages allocation of resources

#### 7.13.3.50 reallocateR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
 \verb|void capd::ddes::DDEForwardTaylorCurvePiece| < TimePointSpec, SetSpec, isInterval >::reallocate \leftrightarrow the capability of the content of the capability of the
R0 (
                                                                                                      size_type N0 = 0) [inline], [protected]
```

technical, manages re-allocation of resources

# 7.13.3.51 reallocateXi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
 SetSpec::MatrixType::ScalarType>::isInterval>
\verb|void capd::des::DDEForwardTaylorCurvePiece| < TimePointSpec, SetSpec, isInterval >::reallocate \leftarrow | Continuous | Contin
 Xi ( ) [inline], [protected]
```

technical, manages re-allocation of resources

#### 7.13.3.52 reinitialize()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
::reinitialize (
        size type d,
        size_type NO ) [inline], [virtual]
```

see base class

# 7.13.3.53 set\_B()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece < TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_B (
            MatrixType const & B )
```

see doubleton interface

WARNING: we assume additionaly, that B has a block-diagonal form, with d x d diagonal blocks (d = dimension()). the code will raise exception if off-block-diagonal element is found non zero.

# 7.13.3.54 set Binv()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece < TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_Binv (
            MatrixType const & Binv )
```

see doubleton interface

WARNING: we assume additionaly, that B has a block-diagonal form, with d x d diagonal blocks (d = dimension()). the code will raise exception if off-block-diagonal element is found non zero.

#### 7.13.3.55 set C()

see doubleton interface

#### 7.13.3.56 set Cr0()

see doubleton interface

#### 7.13.3.57 set r()

see doubleton interface

# 7.13.3.58 set\_r0() [1/2]

set the estimate on the r0 part of the set, by default id does not pass ownership. Updates jet pointers! Throws exception if r0 is of bad dimension.

# 7.13.3.59 set\_r0() [2/2]

see doubleton interface

# 7.13.3.60 set x()

see doubleton interface

# 7.13.3.61 set\_Xi() [1/2]

set the estimate on the \xi part of the jet, by default id does not pass ownership. Throws exception if Xi is of bad dimension.

# 7.13.3.62 set\_Xi() [2/2]

set the estimate on the \xi part of the jet, makes object to own it.

#### 7.13.3.63 setAsConstant() [1/4]

set this vector to represent a constant function f(t) = value for all t > t0, the r0 vector is taken from the value set.

#### 7.13.3.64 setAsConstant() [2/4]

set this vector to represent a constant function f(t) = value for all t > t0, the r0 vector is taken from the value set.

# 7.13.3.65 setAsConstant() [3/4]

set this vector to represent a constant function f(t) = value for all t > t0

# 7.13.3.66 setAsConstant() [4/4]

set this vector to represent a constant function f(t) = value for all t > t0

#### 7.13.3.67 setT0()

set the base time of the jet

# 7.13.3.68 setupFromData() [1/2]

setup data from a generic form of Iterator (to collection of Vectors)

# 7.13.3.69 setupFromData() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
{\tt template}{<}{\tt typename~XItSpec~,~typename~CItSpec~,~typename~BItSpec~,~typename~RItSpec~>}
\verb|void capd::ddes::DDEForwardTaylorCurvePiece| < TimePointSpec, SetSpec, isInterval >::setupFrom \leftrightarrow to the content of the con
                                                             XItSpec xit,
                                                             XItSpec xend,
                                                              CItSpec Cit,
                                                               CItSpec Cend,
                                                               VectorType * r0,
                                                              BItSpec Bit,
                                                              BItSpec Bend,
                                                              BItSpec invBit,
                                                              BItSpec invBend,
                                                               RItSpec rit,
                                                               RItSpec rend,
                                                               VectorType * Xi ) [inline], [protected]
```

Setup set from a rich set of data. Please note that you pass r0 and Xi as pointers, but not pass ownership flags You need to manage flags manually, outside the procedure. This is highly internal function, not meant to be used by the end-user.

# 7.13.3.70 show()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
std::string capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::show
```

more verbose output, more human-friendly. Not rigorous-friendly.

#### 7.13.3.71 storageDimension()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::storage
Dimension ( ) const [inline]
```

DoubletonStorageInterface: returns a dimension of the Jet as a Vector (sequence) to store all the coefficients (without Xi part)

#### 7.13.3.72 storageN0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::storage
NO ( ) const [inline]
```

DoubletonStorageInterface: returns a dimension of the r0 vector

#### 7.13.3.73 summa()

evaluates the integral of Xi part of the jet at t

#### 7.13.3.74 summaAtDelta()

evaluates the integral of Xi part of the jet at t0 + delta\_t

#### 7.13.3.75 t0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
TimePointType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::t0
( ) const [inline]
```

naming convention. See getT0()

#### 7.13.3.76 take r0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType* capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >
::take_r0 ( ) [inline], [virtual]
```

returns r0 and forgets that object is owner (user is responsible for deallocating r0)

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

#### 7.13.3.77 take Xi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType* capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >
::take_Xi ( ) [inline]
```

returns Xi and forgets that object is owner (user is responsible for deallocating Xi)

```
\label{template} $$ \text{typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename SetSpec::MatrixType::ScalarType>::isInterval>$$ \text{VectorType capd::ddes::DDEForwardTaylorCurvePiece} < TimePointSpec, SetSpec, isInterval>$$ \text{::taylor (} $$ const RealType & $t$ ) const [inline]$
```

evaluates the Taylor part of the jet at t

#### 7.13.3.79 taylor() [2/2]

evaluates the Taylor part of the jet at t, delta t should be positive.

• Note: out should be set representing [0,0]<sup>^</sup>d. Procedure do not test for it!

#### 7.13.3.80 taylorAtDelta() [1/2]

evaluates the Taylor part of the jet at t0 + delta\_t

# 7.13.3.81 taylorAtDelta() [2/2]

evaluates the Taylor part of the jet at  $t0 + delta_t$ ,  $delta_t$  should be positive. Note: out should be set representing  $[0,0]^{\wedge}d$ . Procedure do not test for it!

#### 7.13.3.82 translate()

see doubleton interface

#### 7.13.3.83 updateCommonR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::update↔
CommonR0 () [inline], [protected]
```

technical, takes care that all jets share the same r0

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

# 7.14 capd::ddes::DDEJetSection< CurveSpec, isInterval > Class Template Reference

# **Public Types**

- typedef CurveSpec CurveType
- typedef CurveType::size type size type
- typedef CurveType::TimePointType TimePointType
- typedef CurveType::VectorType VectorType
- typedef CurveType::MatrixType MatrixType
- typedef CurveType::ScalarType ScalarType
- typedef GenericJet< TimePointType, VectorType, VectorType, MatrixType, isInterval > JetType
- typedef std::vector< JetType > JetStorageType
- typedef JetStorageType::const\_iterator const\_iterator
- typedef JetStorageType::iterator iterator
- typedef JetStorageType::const\_reverse\_iterator const\_reverse\_iterator
- typedef JetStorageType::reverse iterator reverse iterator

# **Public Member Functions**

- DDEJetSection (size\_type d, size\_type i, ScalarType c=0.)
- DDEJetSection (size\_type d, size\_type p, size\_type n, VectorType const &vec, ScalarType c)
- ScalarType operator() (CurveSpec const &curve) const
- VectorType getGradient (CurveSpec curve) const
- operator VectorType () const
- size\_type storageDimension () const
- · size type dimension () const
- DDEJetSection & set\_c (ScalarType c)
- ScalarType get\_c () const
- DDEJetSection & set\_s (VectorType s)
- VectorType const & get\_s () const
- DDEJetSection & extend (JetType const &jet)
- reverse\_iterator rbegin ()
- · const\_reverse\_iterator rbegin () const
- reverse iterator **rend** ()
- const\_reverse\_iterator rend () const
- iterator begin ()
- · const iterator begin () const
- iterator end ()
- const\_iterator end () const

#### **Protected Attributes**

- ScalarType m c
- VectorType m\_s
- JetStorageType m\_jets
- VectorType m\_orig\_s

#### 7.14.1 Constructor & Destructor Documentation

#### 7.14.1.1 DDEJetSection() [1/2]

```
template<typename CurveSpec , bool isInterval = capd::TypeTraits<typename CurveSpec::Scalar \leftarrow Type>::isInterval> capd::ddes::DDEJetSection < CurveSpec, isInterval >::DDEJetSection ( size_type d, size_type i, ScalarType c = 0. ) [inline]
```

makes a section checking value at t=0 of a given direction, i.e. coordinate section:  $x(0)_i == c$ , where x(0) is d dimensional.

# 7.14.1.2 DDEJetSection() [2/2]

makes a section with p Jets of order n each and expected scalar value c. vector vec must be d \* (1 + p \* (n+1)) dimensional and will propagate the jets.

## 7.14.2 Member Function Documentation

#### 7.14.2.1 getGradient()

This returns gradient of the section d/dx s(x) evaluated at x = j(curve)

NOTE: Since in our case section is a hypersurface, the gradient is constant and equal (z(s), j(s)), extended to the structure of curve, se next NOTE.

NOTE: this function uses structure in curve to deliver higher order jet elements needed for computations. They will be set simply to 0. Therefore, it would be possible to multiply as a Vectors, i.e. we can simply do the vector-vector dot product:

```
getGradient(curve) . (z(curve), j(curve))
```

without worrying about (z(s), j(s)) having bad dimensions.

Note: curve passed by copy, as I need it inside

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

# 7.15 capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec > Class Template Reference

# **Public Types**

- typedef FunctionalMapSpec FunctionalMapType
- typedef FunctionalMapType::CurveType SolutionCurveType
- typedef FunctionalMapType::CurveType CurveType
- typedef FunctionalMapType::JetType JetType
- typedef FunctionalMapType::DataType DataType
- typedef FunctionalMapType::VectorType VectorType
- typedef FunctionalMapType::MatrixType MatrixType
- typedef FunctionalMapType::TimePointType
- typedef FunctionalMapType::RealType
- typedef FunctionalMapType::size\_type size\_type
- typedef FunctionalMapType::VariableStorageType VariableStorageType
- typedef FunctionalMapType::JacobianStorageType JacobianStorageType
- typedef FunctionalMapType::ValueStorageType ValueStorageType

#### **Public Member Functions**

- DDENonrigorousTaylorSolver (DDENonrigorousTaylorSolver const &solver)
- DDENonrigorousTaylorSolver (FunctionalMapType const &map, size\_type maxOrder=20)
- void operator() (CurveType &in out curve)
- void operator() (CurveType &in out curve, JacobianStorageType &D)
- void oneStep (TimePointType const &in\_t0, TimePointType const &in\_th, RealType const &in\_h, Value
   StorageType const &in\_u, ValueStorageType &out\_Phi\_coeffs\_t0, JacobianStorageType &out\_JacPhi\_
   coeffs\_t0, ValueStorageType &out\_Phi\_z, JacobianStorageType &out\_JacPhi\_z)
- void oneStep (TimePointType const &in\_t0, TimePointType const &in\_th, RealType const &in\_h, Value 
  StorageType const &in\_u, ValueStorageType &out\_Phi\_coeffs\_t0, ValueStorageType &out\_Phi\_z)
- FunctionalMapType const & getMap () const
- FunctionalMapType & getMap ()

#### 7.15.1 Member Function Documentation

# 7.15.1.1 oneStep() [1/2]

this function produces approximation to the solution after time h = distance between grid points in the input curve. The var out\_Phi\_coeffs\_t0 will hold the Jet coefficients at point t0, then out\_JacPhi\_coeffs\_t0 will be the Jacobian of coeffs w.r.t. variables in out\_u. Then out\_Phi\_x[0] is the value at t = t0+h computed by the Taylor method, i.e. evaluation of out\_Phi\_coeffs\_t0(t0+h). out\_JacPhi\_x[0] is like in Taylor method for ODEs.

It is designed to be used internally. Operators() and epsilonShift() use this.

#### 7.15.1.2 oneStep() [2/2]

same as the other oneStep, but without computing JacPhi

#### 7.15.1.3 operator()() [1/2]

a nice operator form of the one step solver

#### 7.15.1.4 operator()() [2/2]

a nice operator form of the one step solver NOTE IMPORTANT: we assume that DataType of Jet can hold value and Matrix we provide candidate for this: VectorWithJacobianData you cannot use this function with the basic data structure for SolutionCurve this is also motivated with the optimization in computation of rhs (if we do not use all past data, but only e.g. single discrete delay) the VectorWithJacobianData is to bind value of a variable with the derivative of coefficients w.r.t. initial data. Please check examples on how to use this in your code. TODO: (FUTURE): make this more user friendly...

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

# 7.16 capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec > Class Template Reference

#include <DDEPiecewisePolynomialCurve.h>

#### **Classes**

· struct rebind

#### **Public Types**

- typedef GridSpec GridType
- typedef JetSpec JetType
- typedef JetType CurvePieceType
- typedef DDEPiecewisePolynomialCurve< GridType, JetType > Class
- typedef JetType::DataType
- typedef JetType::MatrixType MatrixType
- typedef JetType::VectorType VectorType
- typedef JetType::ScalarType ScalarType
- typedef ScalarType RealType
- typedef GridType::TimePointType TimePointType
- typedef JetType::size\_type size\_type
- typedef std::deque< CurvePieceType \* > PiecesStorageType
- typedef PiecesStorageType::iterator iterator
- typedef PiecesStorageType::const\_iterator const\_iterator
- typedef PiecesStorageType::reverse\_iterator reverse\_iterator
- typedef PiecesStorageType::const\_reverse\_iterator const\_reverse\_iterator

#### **Public Member Functions**

- virtual std::string badge () const
- DDEPiecewisePolynomialCurve (const DDEPiecewisePolynomialCurve &orig)
- Class & operator= (const Class & orig)
- DDEPiecewisePolynomialCurve (const GridType &grid, size type d=0)
- DDEPiecewisePolynomialCurve (const GridType &grid, const TimePointType &t0, size\_type d=0)
- DDEPiecewisePolynomialCurve (const GridType &grid, const TimePointType &t0, const DataType &value)
- DDEPiecewisePolynomialCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size\_type order, const DataType &value)
- size\_type length () const
- Class subcurve (size\_type index1, size\_type index2) const
- Class subcurve (size\_type index1) const
- Class subcurve (TimePointType const &t0, TimePointType const &t1) const
- Class subcurve (TimePointType const &t0) const
- reverse iterator rbegin ()
- const reverse iterator rbegin () const
- reverse iterator rend ()
- const\_reverse\_iterator rend () const
- iterator begin ()
- · const iterator begin () const
- iterator end ()
- · const\_iterator end () const
- iterator at (TimePointType const &t)
- const\_iterator at (TimePointType const &t) const
- GridType const & grid () const
- size\_type dimension () const
- size\_type storageDimension () const
- Class & setCurrentTime (TimePointType const &t0)
- TimePointType getCurrentTime () const
- TimePointType currentTime () const
- TimePointType t0 () const
- TimePointType getT0 () const
- Class & setT0 (TimePointType const &t0)
- TimePointType pastTime () const
- VectorType get\_x () const
- Class & set\_x (VectorType const &x)
- Class & affineTransform (MatrixType const &M, VectorType const &v)
- Class & translate (VectorType const &v)
- Class & add (VectorType const &v)
- Class & add (Class const &set)
- Class & mulThenAdd (ScalarType const &c, Class const &set)
- Class & mul (ScalarType const &c)
- Class & operator\*= (ScalarType const &c)
- size\_type pointToIndex (TimePointType const &t) const
- CurvePieceType & getPiece (TimePointType const &t) const
- CurvePieceType & getPiece (size\_type const &index) const
- VectorType eval (TimePointType t) const
- VectorType eval (RealType t) const
- void eval (TimePointType t, DataType &out) const
- · void eval (RealType t, DataType &out) const
- Class dt (DataType const &valueAtCurrent, size type n=1) const
- template<typename FunctionalSpec >
  - Class dt (FunctionalSpec const &f, size\_type n=1) const
- Class increasedOrder (size\_type r=1)

- Class decreasedOrder (size type r=1)
- Class & addPiece (CurvePieceType const &newPiece)
- Class & addPiece (CurvePieceType \*newPiece)
- Class & addPastPiece (const CurvePieceType &newPiece)
- Class & addPastPiece (CurvePieceType \*newPiece)
- Class & setValueAtCurrent (const DataType &value)
- DataType & getValueAtCurrent ()
- const DataType & getValueAtCurrent () const
- JetType jet (RealType t0, RealType t1) const
- JetType jet (TimePointType t0, TimePointType t1) const
- JetType jet (TimePointType t0) const
- JetType \* jetPtr (TimePointType t0) const
- DataType j (size\_type i, size\_type k) const
- DataType j (TimePointType t0, size\_type k) const
- JetType const & j (TimePointType t0) const
- JetType & j (TimePointType t0)
- size\_type jetOrderAt (TimePointType t0)
- size\_type jetCommonMaximalOrder ()
- const GridType & getGrid () const
- const TimePointType getStep () const
- std::string show () const
- template<typename DynSysSpec >
   Class & extend (DynSysSpec &solver)
- template<typename DynSysSpec >
   void epsilonShift (DynSysSpec const &solver, RealType const &epsilon, Class &out\_result) const
- template<typename DynSysSpec >
   void epsilonShift (DynSysSpec const &solver, TimePointType const &at\_t0, RealType const &epsilon, Class
   &out result) const
- ScalarType dot (VectorType const &v) const
- template<typename JetSection >
  ScalarType dot (JetSection const &v) const
- · Class & clear ()
- virtual ~DDEPiecewisePolynomialCurve ()

# **Protected Member Functions**

- void deallocatePieces ()
- template<typename IteratorSpec > void copyPieces (IteratorSpec from, IteratorSpec to)
- void copyPieces (iterator from, iterator to)
- void copyPieces (const iterator from, const iterator to)
- void writeTo (std::ostream &out) const
- · void readFrom (std::istream &out) const

## **Protected Attributes**

- const GridType & m\_grid
- TimePointType m\_t\_current
- VectorType m\_lastEnclosure
- DataType m\_valueAtCurrent
- size\_type m\_dimension
- PiecesStorageType m\_pieces

### Friends

- std::ostream & operator<< (std::ostream &out, Class const &curve)</li>
- std::istream & operator>> (std::istream &in, Class const &curve)

# 7.16.1 Detailed Description

```
template < typename \ GridSpec, \ typename \ JetSpec > \\ class \ capd:: ddes:: DDEPiecewisePolynomialCurve < GridSpec, \ JetSpec > \\
```

TODO: docs

# 7.16.2 Constructor & Destructor Documentation

# 7.16.2.1 DDEPiecewisePolynomialCurve() [1/5]

copy constructor, standard thing

# 7.16.2.2 DDEPiecewisePolynomialCurve() [2/5]

makes a Curve that is d dimensional, over given grid, located initially at TimePoint 0

# 7.16.2.3 DDEPiecewisePolynomialCurve() [3/5]

makes a Curve that is d dimensional, over given grid starting at a given TimePoint

### 7.16.2.4 DDEPiecewisePolynomialCurve() [4/5]

creates a Curve that is a single point of given value at time point t0

# 7.16.2.5 DDEPiecewisePolynomialCurve() [5/5]

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration).

### 7.16.2.6 ~DDEPiecewisePolynomialCurve()

```
template<typename GridSpec , typename JetSpec >
virtual capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::~DDEPiecewisePolynomialCurve
( ) [inline], [virtual]
standard C++ thing
```

# 7.16.3 Member Function Documentation

template<typename GridSpec , typename JetSpec >

VectorType const & v ) [inline]

### 7.16.3.1 add() [1/2]

Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::add (

# 7.16.3.3 addPastPiece() [1/2]

same as addPiece but adds in the past (before existing pieces)

### 7.16.3.4 addPastPiece() [2/2]

adds directly by pointer, faster than by copying from reference. The curve will be resp. for deleting.

# 7.16.3.5 addPiece()

adds directly by pointer, faster than by copying from reference. The curve will be resp. for deleting.

# 7.16.3.6 affineTransform()

todo: docs

# 7.16.3.7 at() [1/2]

todo: docs

### 7.16.3.8 at() [2/2]

### 7.16.3.9 badge()

```
template<typename GridSpec , typename JetSpec >
virtual std::string capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::badge ( )
const [inline], [virtual]
```

for identifying in exceptions and output eventual derived classes

### 7.16.3.10 begin() [1/2]

```
template<typename GridSpec , typename JetSpec >
iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::begin ( ) [inline]
```

todo: docs

# 7.16.3.11 begin() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::begin ( ) const
[inline]
```

todo: docs

### 7.16.3.12 clear()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::clear ( ) [inline]
```

clears the Curve, removing all pieces and setting it to be a point of value 0. at current t0

# 7.16.3.13 copyPieces()

low level, does not care about changing time points, does not clear m\_pieces, etc. just copies from to

# 7.16.3.14 currentTime()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::currentTime ( )
const [inline]
```

### 7.16.3.15 deallocatePieces()

```
template<typename GridSpec , typename JetSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::deallocatePieces ( ) [inline],
[protected]
```

safely deallocates pointers in m\_pieces and clears the container

### 7.16.3.16 dimension()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::dimension ( ) const
[inline]
```

todo: docs

# 7.16.3.17 dot() [1/2]

computes dot taking appropriate jets into account. TODO: only template with a given type of section (based on JetType)?

### 7.16.3.18 dot() [2/2]

computes dot taking appropriate number of first components from Curve

# 7.16.3.19 end() [1/2]

```
template<typename GridSpec , typename JetSpec >
iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::end ( ) [inline]
```

todo: docs

### 7.16.3.20 end() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::end ( ) const
[inline]
```

### 7.16.3.21 epsilonShift() [1/2]

it computes epsilon shift of in\_curve and store it in out\_result. out\_result must be setup so that all desired Time ← Points are presented and Jets must be initialized to a desired order. Most probable scenario: out\_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in\_curve starting at t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out\_result at -tau, .., 0, respectively! The t0 is taken from in\_curve.t0() in this procedure. For a general t0 see other function.

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance (?)

### 7.16.3.22 epsilonShift() [2/2]

it computes epsilon shift of in\_curve and store it in out\_result. out\_result must be setup so that all desired Time ← Points are presented and Jets must be initialized to a desired order. Most probable scenario: out\_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in\_curve starting at at\_t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out\_result at -tau, .., 0, respectively!

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance

# 7.16.3.23 eval()

```
\label{template} $$ \ensuremath{\mathsf{Lindes::DDEPiecewisePolynomialCurve} < GridSpec, JetSpec >::eval ( RealType $t$ ) const [inline] }
```

WARNING: THIS FUNCTION IS NOT RIGOROUS FOR INTERVALS! DO NOT USE IN PROOFS WARNING: C← OMPUTE RIGOROUSLY ONLY USING eval(TimePointType t) VERSION

### 7.16.3.24 extend()

Naming convention. To be a pair of extend <-> epsilonShift See move() for more docs.

# 7.16.3.25 get\_x()

```
template<typename GridSpec , typename JetSpec >
VectorType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::get_x ( ) const
[inline]
```

todo: docs

# 7.16.3.26 getCurrentTime()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getCurrentTime ( )
const [inline]
```

todo: docs

### 7.16.3.27 getGrid()

```
template<typename GridSpec , typename JetSpec >
const GridType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getGrid ( )
const [inline]
```

returns grid object for this curve

# 7.16.3.28 getPiece()

it assumes that index is already processed (see pointToIndex())

# 7.16.3.29 getStep()

```
template<typename GridSpec , typename JetSpec >
const TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getStep ( )
const [inline]
```

returns the grid step for this curve

# 7.16.3.30 getT0()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getT0 ( ) const
[inline]
```

```
7.16.3.31 getValueAtCurrent() [1/2]
```

```
template<typename GridSpec , typename JetSpec >
DataType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getValueAtCurrent ( )
[inline]
```

get the value at current

### 7.16.3.32 getValueAtCurrent() [2/2]

```
template<typename GridSpec , typename JetSpec >
const DataType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getValueAt←
Current ( ) const [inline]
```

get the value at current (const object)

# 7.16.3.33 grid()

```
template<typename GridSpec , typename JetSpec >
GridType const& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::grid ( ) const
[inline]
```

todo: docs

# 7.16.3.34 j() [1/4]

k-th coeff at grid point i

# 7.16.3.35 j() [2/4]

jet at grid point t0 (reference)

### 7.16.3.36 j() [3/4]

jet at grid point t0 (reference)

# 7.16.3.37 j() [4/4]

k-th coeff at grid point t0

### 7.16.3.38 jet() [1/3]

jet at grid point t0 and valid until t1 (copy, see j() for reference)

# 7.16.3.39 jet() [2/3]

jet at grid point t0, and valid for t0 to t0 + grid\_step (copy, see j() for reference)

# 7.16.3.40 jet() [3/3]

jet at grid point t0 and valid until t1 (copy, see j() for reference)

### 7.16.3.41 jetCommonMaximalOrder()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jetCommonMaximalOrder
( ) [inline]
```

returns maximal common jet of all jets in this curve

# 7.16.3.42 jetOrderAt()

returns order of jet available at point t. This is for optimization purposes (do not copy whole Jet just to know its order)

```
7.16.3.43 jetPtr()
```

use with caution! Mainly for internal use for speed purposes

### 7.16.3.44 length()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::length ( ) const
[inline]
```

number of pieces

# 7.16.3.45 mul()

todo: docs

# 7.16.3.46 mulThenAdd()

todo: docs

# 7.16.3.47 operator\*=()

todo: docs

# 7.16.3.48 operator=()

assign operator, standard thing

### 7.16.3.49 pastTime()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::pastTime ( ) const
[inline]
```

todo: docs

# 7.16.3.50 pointToIndex()

returns index in the array of the Jets for a given TimePoint

# 7.16.3.51 rbegin() [1/2]

```
template<typename GridSpec , typename JetSpec >
reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rbegin ( )
[inline]
```

todo: docs

### 7.16.3.52 rbegin() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rbegin (
) const [inline]
```

todo: docs

# 7.16.3.53 readFrom()

internal function to make short impl. of >> operator and the long impl. in .hpp file (see comment in >> operator)

# 7.16.3.54 rend() [1/2]

```
template<typename GridSpec , typename JetSpec >
reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rend ( ) [inline]
```

```
7.16.3.55 rend() [2/2]
```

```
template<typename GridSpec , typename JetSpec >
const_reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rend ( )
const [inline]
todo: docs
7.16.3.56 set_x()
template<typename GridSpec , typename JetSpec >
{\tt Class\&\ capd::ddes::DDEPiecewisePolynomialCurve} < {\tt GridSpec},\ {\tt JetSpec} > :: {\tt set\_x}\ (
              VectorType const & x ) [inline]
todo: docs
7.16.3.57 setCurrentTime()
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::setCurrentTime (
             TimePointType const & t0 ) [inline]
todo: docs
7.16.3.58 setT0()
\label{lem:condition} \verb|template| < typename GridSpec |, typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::setT0 (
              TimePointType const & t0 ) [inline]
todo: docs
7.16.3.59 setValueAtCurrent()
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::setValueAtCurrent (
             const DataType & value ) [inline]
set the value at a current time
7.16.3.60 show()
```

```
template<typename GridSpec , typename JetSpec >
std::string capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::show
```

show human readable representation

# 7.16.3.61 storageDimension()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::storageDimension ( )
const [inline]
```

todo: docs

# 7.16.3.62 subcurve() [1/4]

Returns subcurve from idex1 to current time. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

# 7.16.3.63 subcurve() [2/4]

returns subcurve from index1 to index2 (in indexaction of internal data. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

# 7.16.3.64 subcurve() [3/4]

returns subcurve of the segment between t0 and currentTime()

### 7.16.3.65 subcurve() [4/4]

returns subcurve of the segment between t0 and t1

# 7.16.3.66 t0()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::t0 ( ) const
[inline]
```

### 7.16.3.67 translate()

todo: docs

### 7.16.3.68 writeTo()

internal function to make short impl. of << operator and the long impl. in .hpp file (see comment in << operator)

### 7.16.4 Friends And Related Function Documentation

### 7.16.4.1 operator <<

friend operator must be inline, or the linker gets confused, so we use writeTo/readFrom

# 7.16.4.2 operator>>

friend operator must be inline, or the linker gets confused, so we use writeTo/readFrom

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

# 7.17 capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec > Class Template Reference

```
#include <DDEPoincareMap.h>
```

# **Public Types**

- typedef DynSysSpec DynSysType
- typedef SectionSpec SectionType
- typedef DynSysSpec::CurveType CurveType
- typedef CurveType::TimePointType TimePointType
- typedef CurveType::VectorType VectorType
- typedef CurveType::MatrixType MatrixType
- typedef CurveType::ScalarType ScalarType
- typedef CurveType::RealType RealType

### **Public Member Functions**

- DDEPoincareMap (DDEPoincareMap const &other)
- **DDEPoincareMap** (DynSysType &dynsys, SectionType &section, CrossingDirection direction=Crossing 

  Direction::Both, int reqSteps=0, int maxSteps=-1, double binsearchEpsilon=DEFAULT\_BINSEARCH\_EPS 

  ILON)
- void operator() (CurveType &curve, CurveType &on\_section, RealType &out\_approachTime)
- CurveType operator() (CurveType const &X)
- DDEPoincareMap & setDirection (CrossingDirection direction)
- CrossingDirection getDirection ()
- DDEPoincareMap & setMaxSteps (int maxSteps)
- int getMaxSteps ()
- DDEPoincareMap & setRequiredSteps (int requiredSteps)
- int getRequiredSteps ()
- void integrateUntilSectionCrossing (CurveType &curve, RealType &timeBeforeSection, bool &isExtraStep
   — Needed)
- void findCrossingTime (CurveType const &curve, CurveType &requested, RealType &crossingTime, int test
   —
   Direction)
- CrossingDirection detectCrossingDirection (CurveType const &curve)
- RealType **getLastEpsilonTime** () const
- RealType getLastTimeBeforeSection () const
- RealType getLastReachTime () const

### **Protected Member Functions**

void checkSteps ()

# **Protected Attributes**

- DynSysType & m\_dynsys
- SectionType & m section
- CrossingDirection m\_direction
- int m\_requiredSteps
- int m maxSteps
- · int m steps
- $\bullet \ \ \text{double} \ m\_binsearchEpsilon$
- RealType m\_lastTimeBeforeSection
- RealType m\_lastEpsilonTime

# 7.17.1 Detailed Description

```
{\it template}{<} {\it typename DynSysSpec, typename SectionSpec}{>} \\ {\it class capd::} {\it ddes::} {\it DDEPoincareMap}{<} \\ {\it DynSysSpec, SectionSpec}{>} \\ {\it class capd::} \\ {\it des::} \\ {\it class capd::} \\
```

Implementation of Rigorous Poincare Map for use with DDESolvers.

In theory should work with any compatible DDEDynSys and Section.

### 7.17.2 Member Function Documentation

### 7.17.2.1 detectCrossingDirection()

```
\label{template} $$ \ensuremath{\mathsf{template}}$ $$ \ensuremath{\mathsf{curveType}}$ const & $curve$ ) [inline] $$
```

warning: might be time consuming, it (might) integrate the solution... it assumes curve is on section.

# 7.17.2.2 findCrossingTime()

testDirection = -1 for before section testDirection = +1 for after section binsearch for now

### 7.17.2.3 integrateUntilSectionCrossing()

it extends the solution until the crossing with the section. Also, it takes into consideration the required number of steps (should be set-up before) It also moves away of section is initially curve crosses the section.

WARNING: it does not check for sanity of argument as of now, so it might cause runtime-errors if the proof is not well prepared.

### 7.17.2.4 operator()() [1/2]

in curve there is a curve that reaches just after the section in on\_section there is the image on the section, as close to section as possible on\_section must be initialized with the desired "length" (time interval) and "order" of the representation.

# 7.17.2.5 operator()() [2/2]

to be compatible with standard notion and CAPD interface, It makes following assumptions:

• X and PX will have the same structure (i.e. over same grid points and jets of the same order).

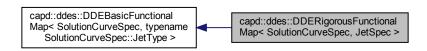
NOTE: you can retrieve information on reach time and epsilon step, but you cannot acquire the solution on the whole time. If you need solution over the full integration time, then use other operator().

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

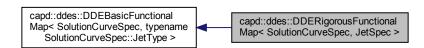
# 7.18 capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec > Class Template Reference

#include <FunctionalMap.h>

Inheritance diagram for capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >:



Collaboration diagram for capd::ddes::DDERigorousFunctionalMap < SolutionCurveSpec, JetSpec >:



# **Public Types**

- typedef DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec > BaseClass
- typedef BaseClass::CurveType CurveType
- typedef BaseClass::JetType JetType
- typedef BaseClass::RealType RealType
- typedef BaseClass::TimePointType TimePointType
- typedef BaseClass::MatrixType MatrixType
- typedef BaseClass::VectorType VectorType
- typedef BaseClass::ScalarType ScalarType
- typedef BaseClass::size\_type size\_type
- typedef BaseClass::DataType DataType
- typedef BaseClass::VariableStorageType VariableStorageType
- typedef BaseClass::ValueStorageType ValueStorageType
- typedef BaseClass::JacobianStorageType JacobianStorageType

### **Public Member Functions**

- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out\_u, ValueStorageType &out\_encl, size\_type &out\_← admissible order) const =0
- virtual ~DDERigorousFunctionalMap ()
- virtual size\_type imageDimension () const=0
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out\_u, size\_type &out\_admissible\_order) const=0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const=0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const=0
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs, Variable
   — StorageType &u, JacobianStorageType &Du) const

# **Static Public Member Functions**

• static void convert (VariableStorageType const &u, ValueStorageType &v)

# **Protected Member Functions**

virtual void checkCurveDimension (CurveType const &x, std::string extra="") const

# 7.18.1 Detailed Description

template<typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::JetType> class capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >

TODO: rewrite DOCS this is the interface of a Functional map, that can be evaluated only by "querying" the curve about its jets at various points in the past. it can produce jet at a given time t together with partial derivatives w.r.t. to internal variables from curve used in the computation by recurrent formula for a functional equation:

```
x'(t) = f(t, x)
```

where x is the solution curve in \R^d defined for times smaller than t (up to some maximal past time usually).

A good example of such a functional map is discrete delay case, e.g. F(t, x) = f(t, x(t), x(t-tau)),  $f : \R \times \R^{2d} > \R$  (easily extended to more delays). (Probably) the functionals that include integral over the past: F(t, x) = f(t, x) int\_{t-tau}^{t} x(s) ds) will also fall into this category.

The concrete implementations will be available later.

Implementation note: when implementing concrete class of this interface, do not forget to put using BaseClass::operator(); using BaseClass::computeDDECoefficients; in the public part of your class to gain access to default implementations of some functions.

TODO: (NOT URGENT, FUTURE, RETHINK) it would be better not to pass containers to computeDDECoefficients, but maybe iterators to given types? It could be used to compute in a given place, i.e. at an already created Jet?

# 7.18.2 Constructor & Destructor Documentation

### 7.18.2.1 ∼DDERigorousFunctionalMap()

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >::~DDERigorousFunctionalMap
( ) [inline], [virtual]

virtual destructor for warning suppresion

# 7.18.3 Member Function Documentation

### 7.18.3.1 checkCurveDimension()

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::checkCurve← Dimension [inline], [protected]

helper function to check. TODO: (NOT URGENT, FUTURE) make a switch to turn this off for speed.

### 7.18.3.2 collectComputationData() [1/2]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::collectComputation← Data
```

Collects all data from curve necessary for the computation of the value. This does not collect ENCLOSURE DATA required in rigorous computations.

We use collection of DataType items as the output, we ASSUME that DataType elements are used to describe function by CurveType. Therefore, in the output, we could relate how inputs (VariableStorage) corresponds to outputs (ValueStorage) mainly by Jacobian of the map at Variables (JacobianStorageType).

Then, the main function that needs to be implemented in the Map are those depending on the VariableStorageType. Other can have default implementations and are for users convenience.

### THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Note: dt might seem redundant here, but for some reasons it might be helpful, i.e. for epsilon steps (see paper) Note: dt should be by definition dt = th - t0. Note: This function is for optimization purposes. I could assume that the whole Curve goes into computation, but then the computing cost would grow, as Curve grows in time. Also, if some coefficients are not used in rhs of the equation, we would carry out the unnecessary computations (i.e. Jac Phi would be 0 and we would do 0 \* C \* r0). There is one problem with this however: in rigorous code we need to check if the section is not crossed between steps in Poincare map. Please see comments in JetSection / DDEPoincareMap classes.

Param: out\_u a finite-dimensional collection of data that is used to compute the map Param: admissible\_order will tell how high order of expansion is able to produce with data in u

# 7.18.3.3 collectComputationData() [2/2]

Extra method needed by the rigorous code

Collects all data from curve necessary for the computation of the value. We use collection of DataType items as the output, we ASSUME that SetType elements are used to describe function by CurveType. Therefore, in the output, we could relate how inputs (VariableStorage) corresponds to outputs (ValueStorage) mainly by Jacobian of the map at Variables (JacobianStorageType).

Then, the main function that needs to be implemented in the Map are those depending on the VariableStorageType. Other can have default implementations and are for users convenience.

### THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Note: dt might seem redundant here, but for some reasons it might be helpful, i.e. for epsilon steps (see paper) Note: dt should be by definition dt = th - t0. Note: This function is for optimization purposes. I could assume that the

whole Curve goes into computation, but then the computing cost would grow, as Curve grows in time. Also, if some coefficients are not used in rhs of the equation, we would carry out the unnecesary computations (i.e. Jac Phi would be 0 and we would do 0 \* C \* r0). There is one problem with this however: in rigorous code we need to check if the section is not crossed between steps in Poincare map. Please see comments in JetSection / DDEPoincareMap classes.

Param: out\_u a finite-dimensional collection of data that is used to compute the map Param: out\_encl first entries corresponds to enclosures over [t0, t0+dt] of the variables in u, then, it might contain more entries, relevant to computation. For example higher order enclosures used in Taylor solver. Param: admissible\_order will tell how high order of expansion is able to produce with data in u

### 7.18.3.4 computeDDECoefficients() [1/6]

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients [inline]

this is for a current time in the solution. It provides basic implementation by call to the other function.

### 7.18.3.5 computeDDECoefficients() [2/6]

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients [inline]

this is for a current time in the solution. It provides basic implementation by call to the other function.

# 7.18.3.6 computeDDECoefficients() [3/6]

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true. For non-developer users of the library it is possible to use versions of the function for specific CurveType.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out\_admissible\_order from collectComputationData() is used for this purpose.

### 7.18.3.7 computeDDECoefficients() [4/6]

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It also computes  $\frac{\pi}{\pi}$  (partial coeffs[i]}{\partial u} (in Du). It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out\_admissible\_order from collectComputationData() is used for this purpose.

# 7.18.3.8 computeDDECoefficients() [5/6]

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients [inline]

computes recursively the Jet at time t for a given curve for a DDE of the form:

```
x'(t) = F(t, x)
```

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. If order is 0 or order is too high for a given curve then the function should alter the order to highest possible. (use coeffs. cresize(order) for this purpose)

THERE IS DEFAULT IMPLEMENTATION OF THIS WITH collectComputationData() and computeDDE  $\leftarrow$  Coefficients(..., u, ...) call to pure virtual function.

## 7.18.3.9 computeDDECoefficients() [6/6]

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType> virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE← Coefficients [inline]

Same as computeDDECoefficients(const TimePointType&, const CurveType&, JetType&), but also computes the partial derivative:

```
\texttt{Du[k] = \{partial coeffs[k]\}\{partial u\}}
```

where u are all the variables used to evaluate the map. In u[j] is a d-dimensional set used in computation. In Du[k][j] is a matrix of dimension M(d,d) and of course it is

It is done this way to allow Solvers to use this structure to reduce wrapping effect of interval arithmetics with the help of Lohner algorithm.

### 7.18.3.10 convert()

template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
static void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::convert [inline],
[static]

helper to convert from collection of sets to collection of vectors

### 7.18.3.11 imageDimension()

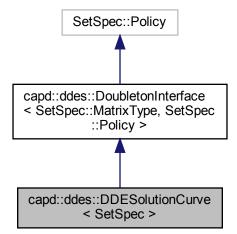
output dimension of the map. THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

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

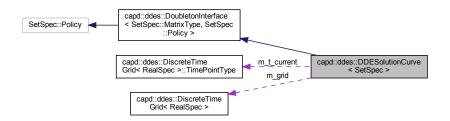
# 7.19 capd::ddes::DDESolutionCurve< SetSpec > Class Template Reference

#include <DDESolutionCurve.h>

Inheritance diagram for capd::ddes::DDESolutionCurve< SetSpec >:



Collaboration diagram for capd::ddes::DDESolutionCurve< SetSpec >:



# **Public Types**

- typedef SetSpec SetType
- typedef SetType DataType
- $\bullet \ \ typedef \ \ \ Doubleton Interface < typename \ \ SetSpec:: Matrix Type, \ typename \ \ SetSpec:: Policy > \textbf{BaseClass}$
- typedef DDESolutionCurve < SetSpec > Class
- typedef SetType::MatrixType MatrixType
- typedef MatrixType::RowVectorType VectorType
- typedef MatrixType::ScalarType
- typedef ScalarType RealType
- typedef DiscreteTimeGrid < RealType > GridType
- typedef GridType::TimePointType TimePointType
- typedef BaseClass::size\_type size\_type
- typedef DDEForwardTaylorCurvePiece
   TimePointType, SetType
   CurvePieceType
- typedef SetType::Policy QRPolicy
- typedef SetType::Policy Policy
- typedef CurvePieceType JetType
- typedef std::deque < CurvePieceType \* > PiecesStorageType
- typedef std::deque< bool > OwnerStorageType
- typedef PiecesStorageType::iterator iterator
- typedef PiecesStorageType::const\_iterator const\_iterator
- typedef PiecesStorageType::reverse\_iterator reverse\_iterator
- typedef PiecesStorageType::const\_reverse\_iterator const\_reverse\_iterator

# **Public Member Functions**

- DDESolutionCurve (const DDESolutionCurve &orig)
- DDESolutionCurve & operator= (const DDESolutionCurve & orig)
- DDESolutionCurve (const GridType &grid, size type d=0, size type N0=0)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, size\_type d=0, size\_type N0=0)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const SetType &value)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const VectorType &value, size\_type N0=0)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size\_type order, const SetType &value)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size\_type order, SetType &value)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size\_type order, const VectorType &value, size\_type N0=0)

- 111 template<typename AnyMatrixSpec > DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size type order, const capd::map::Map< AnyMatrixSpec > &f, size\_type N0=0) • size type length () const • DDESolutionCurve subcurve (size type index1, size type index2) const DDESolutionCurve subcurve (size type index1) const DDESolutionCurve subcurve (TimePointType const &t0, TimePointType const &t1) const DDESolutionCurve subcurve (TimePointType const &t0) const • reverse\_iterator rbegin () · const\_reverse\_iterator rbegin () const • reverse iterator **rend** () · const reverse iterator rend () const • iterator begin () · const iterator begin () const iterator end () • const\_iterator end () const iterator at (TimePointType const &t) const\_iterator at (TimePointType const &t) const
- GridType const & grid () const
- · size\_type dimension () const
- TimePointType t0 () const
- TimePointType currentTime () const
- TimePointType pastTime () const
- size type storageDimension () const
- size\_type storageN0 () const
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- VectorType get\_x () const
- MatrixType get\_C () const
- VectorType get r0 () const
- MatrixType get B () const
- VectorType get\_r () const
- MatrixType get Binv () const
- BaseClass & set x (VectorType const &x)
- BaseClass & set\_C (MatrixType const &C)
- BaseClass & set\_r0 (VectorType const &r0)
- BaseClass & set\_Cr0 (MatrixType const &C, VectorType const &r0)
- BaseClass & set\_B (MatrixType const &B)
- BaseClass & set\_r (VectorType const &r)
- BaseClass & set Binv (MatrixType const &Binv)
- BaseClass & set x (VectorType \*x, bool passOwnership=false)
- BaseClass & set C (MatrixType \*C, bool passOwnership=false)
- BaseClass & set\_r0 (VectorType \*r0, bool passOwnership=false)
- BaseClass & set\_Cr0 (MatrixType \*C, VectorType \*r0, bool passCOwnership=false, bool passr0← Ownership=false)
- BaseClass & set B (MatrixType \*B, bool passOwnership=false)
- BaseClass & set\_r (VectorType \*r, bool passOwnership=false)
- BaseClass & set\_Binv (MatrixType \*B, bool passOwnership=false)
- VectorType \* take x ()
- MatrixType \* take\_C ()
- VectorType \* take r0 ()
- MatrixType \* take B ()
- VectorType \* take r ()
- MatrixType \* take\_Binv ()
- BaseClass & add (VectorType const &v)

### < see DoubletonInterface

- BaseClass & add (BaseClass const &set)
- BaseClass & mulThenAdd (ScalarType const &c, BaseClass const &set)
- BaseClass & mul (ScalarType const &c)
- BaseClass & set Xi (VectorType const &Xi)
- VectorType get Xi () const
- size\_type pointToIndex (TimePointType const &t) const
- CurvePieceType & getPiece (TimePointType const &t) const
- CurvePieceType & getPiece (size\_type const &index) const
- VectorType eval (TimePointType t) const
- VectorType eval (RealType t) const
- void eval (TimePointType t, SetType &out) const
- void eval (RealType t, SetType &out) const
- DDESolutionCurve & addPiece (CurvePieceType \*newPiece, bool passOwnership=false)
- DDESolutionCurve & addPiece (CurvePieceType const &newPiece)
- DDESolutionCurve & addPastPiece (CurvePieceType \*newPiece, bool passOwnership=false)
- DDESolutionCurve & addPastPiece (const CurvePieceType &newPiece)
- DDESolutionCurve & setValueAtCurrent (const SetType &value)
- DDESolutionCurve & setValueAtCurrent (const VectorType &value)
- SetType & getValueAtCurrent ()
- const SetType & getValueAtCurrent () const
- JetType jet (RealType t0, RealType t1) const
- JetType jet (TimePointType t0, TimePointType t1) const
- JetType jet (TimePointType t0) const
- JetType \* jetPtr (TimePointType t0) const
- SetType j (size\_type i, size\_type k) const
- SetType j (TimePointType t0, size type k) const
- const JetType & j (TimePointType t0) const
- JetType & j (TimePointType t0)
- size type jetOrderAt (TimePointType t0)
- · const GridType & getGrid () const
- const TimePointType getStep () const
- Class midCurve () const
- Class dt (int k=1) const
- std::string show () const
- template<typename DynSysSpec >

DDESolutionCurve & move (DynSysSpec &solver)

- template<typename DynSysSpec >
  - DDESolutionCurve & extend (DynSysSpec &solver)
- VectorType getLastEnclosure () const
- template<typename DynSysSpec >

void epsilonShift (DynSysSpec const &solver, RealType const &epsilon, DDESolutionCurve &out\_result) const

- $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{DynSysSpec} \mathord{>}$ 
  - void epsilonShift (DynSysSpec const &solver, TimePointType const &at\_t0, RealType const &epsilon, DDESolutionCurve &out\_result) const
- virtual void reinitialize (size\_type d, size\_type N0)
- ScalarType dot (VectorType const &v) const
- $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{JetSection} >$ 
  - ScalarType dot (JetSection const &v) const
- void reduce (DDESolutionCurve &out\_result) const
- template<typename SetSpec >
  - DDESolutionCurve< SetSpec >::ScalarType **dot** (DDESolutionCurve< SetSpec >::VectorType const &v) const

- template<typename DynSysSpec >
   DDESolutionCurve< SetSpec > & move (DynSysSpec &solver)
- virtual VectorType makeStorage\_x () const
- virtual MatrixType makeStorage\_C () const
  - < see DoubletonInterface
- virtual VectorType makeStorage\_r0 () const
  - < see DoubletonInterface
- virtual MatrixType makeStorage\_B () const
  - < see DoubletonInterface
- virtual VectorType makeStorage\_r () const
  - < see DoubletonInterface
- virtual VectorType hull () const
  - < see DoubletonInterface
- virtual VectorType midPoint () const
  - < see DoubletonInterface

# **Protected Member Functions**

- void allocate (size\_type N0=0)
- void deallocatePieces ()
- void deallocateR0 ()
- void deallocate ()
- void reallocateR0 (size\_type N0)
- void reallocateR0 ()
- void reallocate (size\_type N0)
- · void reallocate ()
- void updateCommonR0 ()
- template<typename IteratorSpec > void copyPieces (IteratorSpec from, IteratorSpec to)
- · void copyPieces (iterator from, iterator to)
- void copyPieces (const\_iterator from, const\_iterator to)
- · void writeTo (std::ostream &out) const

# **Protected Attributes**

- · const GridType & m\_grid
- TimePointType m t current
- VectorType m\_lastEnclosure
- VectorType \* m\_r0
- SetType m\_valueAtCurrent
- size\_type m\_dimension
- PiecesStorageType m\_pieces
- OwnerStorageType m\_pieceOwner

### **Friends**

• std::ostream & operator<< (std::ostream &out, DDESolutionCurve< SetSpec > const &curve)

### **Additional Inherited Members**

# 7.19.1 Detailed Description

```
template<typename SetSpec> class capd::ddes::DDESolutionCurve< SetSpec >
```

A class to represent the solution x(t) of the DDE. It provides methods To ask for the DDECurvePiece over specific intervals (time intervals) The basic implementation allow to ask about intervals  $I_i = [i * h, i* h + h)$ , where  $h = \frac{1}{2} \exp\{p\}$  is the grid size as in FoCM article / PhD thesis. The DDECurvePiece is just a Jet of order n of this solution x at a given time and valid on the specified interval + estimates on the n+1-st derivative. Please consult DDE(ForwardTaylor)CurvePiece for more information how this is organized.

DDESolutionCurve also contains the value of the solution at a maximal time \$t\_{current}\$ - to be used later in the process of solving DDEs. Usually, we will start with some candidate DDESolutionCurve defined over [-\tau, 0], with \$t\_{current} = 0\$. Then we propagate solution by Integrator (see DDESolver) to some \$t\_{future}\$, so that the solution is estimated over [-\tau, t\_{future}]. Please note that the history is preserved for theoretical and technical purposes.

From the theoretical point of view, the history could be used to design solvers for varying delay (as long as it is bounded). From technical point of view, there might be some ways to improve current methods using the history (out of scope as of now).

TODO: (FAR FUTURE) see notes below this line TODOs / DevNOTEs:

Newer note: As of now, class is designed to work with my own implementation of Lohner sets (see 'storage/' subdirectory). They are just Doubletons with some restrictions of classical CAPD sets removed to allow m\_C and m\_r0 parts to be more freely defined. Ideally I would like to use standard sets from CAPD, but this might be not possible / not optimal (becouse of the part r\_0 in definitions, which is ideally shared between sets). Alternatively, I might consider using a big CAPD set and some proxies that can extract subcolumns/subrows of various elements to be passed down to ForwardTaylorCurvePieces or Taylor coefficients of given order.

Older note: The class is templated with a classical CAPD Lohner Set Type, but we will forcefully alter it behaviour (I hope). I will probably use its QR policy etc., but I will alter the matrix  $m_C$  and vector  $m_r$  - they will be stored from within the set more or less, and might not be related to the space dimension. What is required that the product  $m_r$  has dimension \$d\$, i.e.  $m_r$  has dimension \$d\$, i.e.  $m_r$  has dimension \$0.

### 7.19.2 Constructor & Destructor Documentation

### 7.19.2.1 DDESolutionCurve() [1/9]

copy constructor, standard thing

### 7.19.2.2 **DDESolutionCurve()** [2/9]

makes a Curve that is d dimensional, over given grid, located initially at TimePoint 0, and its r0 vector is N0 dimensional

# 7.19.2.3 DDESolutionCurve() [3/9]

makes a Curve that is d dimensional, over given grid starting at a given TimePoint, and its r0 vector is N0 dimensional

### 7.19.2.4 DDESolutionCurve() [4/9]

creates a Curve that is a single point of given value at time point t0,m the r0 part of value is used as r0 of the Curve

### 7.19.2.5 **DDESolutionCurve()** [5/9]

creates a Curve that is a single point of given value at time point t0 and with given dimension of r0 part.

# 7.19.2.6 **DDESolutionCurve()** [6/9]

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration). The r0 part will be COPIED from the set and will be common to all elements.

### 7.19.2.7 DDESolutionCurve() [7/9]

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration). The r0 part will be transferred (take from, the pointer - of possible / the set implementation allows that) from the set and will be common to all elements.

# 7.19.2.8 DDESolutionCurve() [8/9]

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a set in integration). The r0 part will be N0 dimensional (default = 0, i.e. no C\*r0 set part).

## 7.19.2.9 **DDESolutionCurve()** [9/9]

creates a representation of the function f over the interval [t0,t1] The r0 part will be N0 dimensional (default = 0, i.e. no C\*r0 set part).

TODO: rethink: move to a ddeshelper?

### 7.19.3 Member Function Documentation

# 7.19.3.1 add() [1/2]

### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

### 7.19.3.2 add() [2/2]

< see DoubletonInterface

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

### 7.19.3.3 addPastPiece() [1/2]

same as addPiece but adds in the past (before existing pieces)

### 7.19.3.4 addPastPiece() [2/2]

same as addPiece but adds in the past (before existing pieces)

# 7.19.3.5 affineTransform()

```
template<typename SetSpec > DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::affine \leftarrow Transform (

MatrixType const & M,

VectorType const & v)
```

### see DoubletonInterface

### 7.19.3.6 copyPieces()

low level, does not care about changing time points, does not clear m pieces, etc. just copies from to

# 7.19.3.7 dot() [1/2]

computes dot taking appropriate jets into account

### 7.19.3.8 dot() [2/2]

computes dot taking appropriate number of first components from Curve

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

# 7.19.3.9 dt()

compute derivative of the whole curve w.r.t. t of order k

### 7.19.3.10 epsilonShift() [1/2]

it computes epsilon shift of in\_curve and store it in out\_result. out\_result must be setup so that all desired Time ← Points are presented and Jets must be initialized to a desired order. Most probable scenario: out\_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in\_curve starting at t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out\_result at -tau, ..., 0, respectively! The t0 is taken from in\_curve.t0() in this procedure. For a general t0 see other function.

The function does not check if the result is theoretically rigorous, i.e. it does not check if the solution curve is smooth enough to produce rigorous enclosure of a given order. Other classes of the library takes this responsibility (i.e. PoincareMap).

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance (?)

### 7.19.3.11 epsilonShift() [2/2]

it computes epsilon shift of in\_curve and store it in out\_result. out\_result must be setup so that all desired Time← Points are presented and Jets must be initialized to a desired order. Most probable scenario: out\_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in\_curve starting at at\_t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out\_result at -tau, .., 0, respectively!

The function does not check if the result is theoretically rigorous, i.e. it does not check if the solution curve is smooth enough to produce rigorous enclosure of a given order. Other classes of the library takes this responsibility (i.e. PoincareMap).

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance

# 7.19.3.12 eval() [1/2]

WARNING: THIS FUNCTION IS NOT RIGOROUS FOR INTERVALS! DO NOT USE IN PROOFS WARNING: C← OMPUTE RIGOROUSLY ONLY USING eval(TimePointType t) VERSION

### 7.19.3.13 eval() [2/2]

this is rigorous, it returns the value stored in the Jet at a given time point of the grid.

# 7.19.3.14 extend()

Naming convention. To be a pair of extend <-> epsilonShift See move() for more docs.

```
7.19.3.15 get_B()
template<typename SetSpec >
DDESolutionCurve< SetSpec >::get_B
see DoubletonInterface
7.19.3.16 get_Binv()
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType capd::ddes::DDESolutionCurve< SetSpec >::get_Binv
see DoubletonInterface
7.19.3.17 get_C()
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType capd::ddes::DDESolutionCurve< SetSpec >::get_C
see DoubletonInterface
7.19.3.18 get_r()
{\tt template}{<}{\tt typename SetSpec}>
DDESolutionCurve< SetSpec >::get_r
see DoubletonInterface
7.19.3.19 get_r0()
template<typename SetSpec >
DDESolutionCurve< SetSpec >::YectorType capd::ddes::DDESolutionCurve< SetSpec >::get_r0
see DoubletonInterface
7.19.3.20 get x()
template<typename SetSpec >
DDESolutionCurve< SetSpec >::YectorType capd::ddes::DDESolutionCurve< SetSpec >::get_x
see DoubletonInterface
7.19.3.21 get Xi()
template<typename SetSpec >
VectorType capd::ddes::DDESolutionCurve< SetSpec >::get_Xi ( ) const [inline]
```

allows to retrieve Xi for all Jets

### 7.19.3.22 getLastEnclosure()

```
template<typename SetSpec >
VectorType capd::ddes::DDESolutionCurve< SetSpec >::getLastEnclosure ( ) const [inline]
```

returns last enclosure of the value of the function over [t0, t0+h] obtained in extend() / move() procedure.

TODO: (NOT URGENT) make a function that returns enclosure of the whole Jet?

### 7.19.3.23 getPiece()

it assumes that index is already processed (see pointToIndex())

# 7.19.3.24 getValueAtCurrent() [1/2]

```
template<typename SetSpec >
SetType& capd::ddes::DDESolutionCurve< SetSpec >::getValueAtCurrent ( ) [inline]
```

get the value at current

### 7.19.3.25 getValueAtCurrent() [2/2]

```
template<typename SetSpec >
const SetType& capd::ddes::DDESolutionCurve< SetSpec >::getValueAtCurrent ( ) const [inline]
```

get the value at current (const object)

# 7.19.3.26 jetOrderAt()

returns order of jet available at point t. This is for optimization purposes (do not copy whole Jet just to know its order)

# 7.19.3.27 jetPtr()

use with caoution! Mainly for internal use for speed purposes

# 7.19.3.28 length()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::length ( ) const [inline]
```

number of pieces

# 7.19.3.29 makeStorage\_x()

```
template<typename SetSpec >
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_x
[inline]
```

makes a vector that can store x part

# 7.19.3.30 move()

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance

### 7.19.3.31 mul()

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

# 7.19.3.32 mulThenAdd()

# see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

#### 7.19.3.33 operator=()

assign operator, standard thing

#### 7.19.3.34 pointToIndex()

returns index in the array of the Jets for a given TimePoint

# 7.19.3.35 reduce()

it reduces the representation so that the maximal order of the jets is the same as in out\_result.

It throws exception if the out\_result order is higher than this solution at some jet.

# 7.19.3.36 reinitialize()

see base class DoubletonInterface

Implements capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

# 7.19.3.37 set\_B() [1/2]

see DoubletonInterface

```
7.19.3.38 set_B() [2/2]
```

see DoubletonInterface

```
7.19.3.39 set Binv() [1/2]
```

see DoubletonInterface

## 7.19.3.40 set\_Binv() [2/2]

see DoubletonInterface

# 7.19.3.41 set\_C() [1/2]

see DoubletonInterface

## 7.19.3.42 set\_C() [2/2]

see DoubletonInterface

#### 7.19.3.43 set\_Cr0() [1/2]

see DoubletonInterface

## 7.19.3.44 set\_Cr0() [2/2]

Set C \* r0 part of this set (as a doubleton). Because this set is defined by many sub-sets then we need to assume some order. The order is that, the value at current time is in first d (d = dimension() of the curve) rows of C, the the first jet past the current time is in the following d rows,..., finally last d rows is the Jet at pastTime(). see also DoubletonInterface.

## 7.19.3.45 set\_r() [1/2]

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.46 set\_r() [2/2]

## see DoubletonInterface

Implements capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.47 set\_r0() [1/2]

## see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.48 set\_r0() [2/2]

```
\label{local-condition} $$ $$ \end{template} $$ $$ \end{template} $$ $$ \end{template} $$ \end{template} $$ \end{template} $$ $$ \end{template} $$$ \end{tem
```

#### see DoubletonInterface

Implements capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.49 set\_x() [1/2]

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.50 set\_x() [2/2]

#### see DoubletonInterface

Implements capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.51 set\_Xi()

allows to set Xi for all Jets

#### 7.19.3.52 setValueAtCurrent() [1/2]

#### set the value at a current time

## 7.19.3.53 setValueAtCurrent() [2/2]

set the value at a current time

#### 7.19.3.54 show()

```
template<typename SetSpec >
std::string capd::ddes::DDESolutionCurve< SetSpec >::show
```

show detailed info

#### 7.19.3.55 storageDimension()

```
template<typename SetSpec >
size_type capd::ddes::DESolutionCurve< SetSpec >::storageDimension ( ) const [inline]
```

DoubletonStorageInterface: returns a dimension of the Jet as a Vector (sequence) to store all the coefficients (without Xi part)

#### 7.19.3.56 storageN0()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::storageN0 ( ) const [inline]
```

DoubletonStorageInterface: returns a dimension of the r0 vector

# 7.19.3.57 subcurve() [1/4]

Returns subcurve from idex1 to current time. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

### 7.19.3.58 subcurve() [2/4]

returns subcurve from index1 to index2 (in indexaction of internal data. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

## 7.19.3.59 subcurve() [3/4]

returns subcurve of the segment between t0 and currentTime()

## 7.19.3.60 subcurve() [4/4]

returns subcurve of the segment between t0 and t1

#### 7.19.3.61 take B()

```
template<typename SetSpec >

DDESolutionCurve< SetSpec >::MatrixType * capd::ddes::DDESolutionCurve< SetSpec >::take_←

B [virtual]
```

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

#### 7.19.3.62 take\_Binv()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType * capd::ddes::DDESolutionCurve< SetSpec >::take_Binv
[virtual]
```

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

# 7.19.3.63 take\_C()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType * capd::ddes::DDESolutionCurve< SetSpec >::take_←
C [virtual]
```

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

# 7.19.3.64 take\_r()

```
\label{local_continuous_continuous} $$ $$ \end{substrain_curve} > ::VectorType * capd::ddes::DDESolutionCurve< SetSpec >::take\_{\leftarrow} r [virtual]
```

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.65 take\_r0()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType * capd::ddes::DDESolutionCurve< SetSpec >::take_r0
[virtual]
```

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

#### 7.19.3.66 take x()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType * capd::ddes::DDESolutionCurve< SetSpec >::take_←
x [virtual]
```

#### see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

## 7.19.3.67 translate()

#### see DoubletonInterface

Implements capd::ddes::DoubletonInterface < SetSpec::MatrixType, SetSpec::Policy >.

#### 7.19.3.68 writeTo()

internal function to make short impl. of << operator and the long impl. in .hpp file (see comment in << operator)

#### 7.19.4 Friends And Related Function Documentation

#### 7.19.4.1 operator <<

friend operator must be inline, or the linker gets confused

#### 7.19.5 Member Data Documentation

#### 7.19.5.1 m r0

```
template<typename SetSpec >
VectorType* capd::ddes::DDESolutionCurve< SetSpec >::m_r0 [protected]
```

this r0 will always be common for all elements. THIS IS IMPORTANT (because how the Lohner algorithm is implemented). Solution is always the owner of r0, all copies must copy it by creating new instance. Each instance must assure that m\_r0 is always copied as a pointer (not owned) to the pieces and the current value. This variable must be before other dependent variables, because I use this fact to allocate it before others, so I can sometimes pass it to later dependable objects in the constructor list!

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

# 7.20 capd::ddes::DDETaylorSolver< FunctionalMapSpec > Class Template Reference

## **Public Types**

- typedef FunctionalMapSpec FunctionalMapType
- typedef FunctionalMapType::CurveType SolutionCurveType
- typedef FunctionalMapType::CurveType CurveType
- typedef FunctionalMapType::JetType JetType
- typedef SolutionCurveType::SetType SetType
- typedef FunctionalMapType::DataType DataType
- typedef FunctionalMapType::VectorType VectorType
- typedef FunctionalMapType::MatrixType MatrixType
- typedef FunctionalMapType::TimePointType TimePointType
- typedef FunctionalMapType::RealType RealType
- typedef FunctionalMapType::size\_type size\_type
- typedef FunctionalMapType::VariableStorageType

  VariableStorageType
- typedef FunctionalMapType::JacobianStorageType JacobianStorageType
- typedef FunctionalMapType::ValueStorageType ValueStorageType

## **Public Member Functions**

- DDETaylorSolver (DDETaylorSolver const &solver)
- **DDETaylorSolver** (FunctionalMapType const &map, size type maxOrder)
- void encloseSolution (SolutionCurveType const &in\_curve, TimePointType &out\_th, RealType &out\_←
  HH, VariableStorageType &out\_u, ValueStorageType &out\_u\_encl, ValueStorageType &out\_Phi\_coeffs\_t0,
  JacobianStorageType &out\_JacPhi\_coeffs\_t0, ValueStorageType &out\_Rem\_coeffs\_t0, ValueStorage←
  Type &out\_CoeffsEnclosure, ValueStorageType &out\_Phi\_z, JacobianStorageType &out\_JacPhi\_z, Value←
  StorageType &out\_Rem\_z)
- const FunctionalMapType & getMap () const
- FunctionalMapType & getMap ()

#### 7.20.1 Member Function Documentation

#### 7.20.1.1 encloseSolution() [1/2]

see docs for encloseSolution(TimePointType const& t0, ...). It does the same but for t0 = in\_curve.t0()

#### 7.20.1.2 encloseSolution() [2/2]

this function produces estimates that could be used to produce solution after time h = distance between grid points in the input curve. The solution will be valid on [t0, t0+h], its enclosure will be as interval set in out\_CoeffsEnclosure up to order one higher than the out\_Phi\_coeffs\_t0 The var out\_Phi\_coeffs\_t0 will hold the Jet coefficients at point t0 computed for the middle point of the curve, out\_JacPhi\_coeffs\_t0 will be the Jacobian of coeffs w.r.t. variables in out\_u, over the whole set of the curve. The value out\_Rem\_coeffs\_t0 is the remainder, but it will be 0 always in this implementation, as Jets are computed exactly for grid points. Then out\_Phi\_x[0] is the value at t = t0+h computed by the Taylor method, i.e. evaluation of out\_Phi\_coeffs\_t0(t0+h). out\_JacPhi\_x[0] and out\_Rem\_x[0] are like in Taylor method for ODEs.

Curve.move() procedure then should be able to extract information needed to construct a succesive CurvePiece of the solution over [t0, t0+h) and to set current value by use of out\_{Phi|JacPhi|Rem}\_x as it is customary in ODEs code.

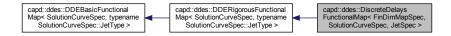
TODO: (FAR FUTURE) rethink that all data in and out encapsulated in some other structure?

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

# 7.21 capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class Template Reference

#include <DiscreteDelaysFunctionalMap.h>

Inheritance diagram for capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet $\sim$  Spec>:



 $\label{lem:collaboration} \mbox{Collaboration diagram for capd::ddes::DiscreteDelaysFunctionalMap$< FinDimMapSpec, SolutionCurveSpec, Jet$$\subset >:$ 



## **Classes**

· struct rebind

## **Public Types**

- $\bullet \ \ type def \ DDE Rigorous Functional Map < Solution Curve Spec, \ Jet Spec > \textbf{Base Class} \\$
- $\bullet \ \ typedef \ {\tt DiscreteDelaysFunctionalMap} < {\tt FinDimMapSpec}, \ Solution CurveSpec}, \ {\tt JetSpec} > {\tt Class} \\$
- typedef SolutionCurveSpec CurveType
- typedef JetSpec JetType
- typedef FinDimMapSpec MapType
- typedef BaseClass::RealType RealType
- typedef BaseClass::TimePointType TimePointType
- typedef BaseClass::MatrixType MatrixType
- typedef BaseClass::VectorType VectorType
- typedef BaseClass::ScalarType ScalarType
- typedef BaseClass::size type size type
- typedef BaseClass::DataType DataType
- typedef BaseClass::ValueStorageType ValueStorageType
- typedef BaseClass::VariableStorageType VariableStorageType
- typedef BaseClass::JacobianStorageType JacobianStorageType
- $\bullet \ \ typedef \ std:: vector < TimePointType > DelayStorageType \\$
- typedef DelayStorageType::const\_iterator const\_iterator
- typedef DelayStorageType::iterator iterator

#### **Public Member Functions**

- $\bullet \quad template {<} typename\ Other Discrete Delays Functional Map > \\$ 
  - DiscreteDelaysFunctionalMap (const OtherDiscreteDelaysFunctionalMap &orig)
- DiscreteDelaysFunctionalMap (const DiscreteDelaysFunctionalMap &orig)
- DiscreteDelaysFunctionalMap (const TimePointType &tau)
- DiscreteDelaysFunctionalMap (const MapType &f)
- **DiscreteDelaysFunctionalMap** (const MapType &f, const TimePointType &tau)
- **DiscreteDelaysFunctionalMap** (const MapType &f, std::vector< TimePointType > delays)
- **DiscreteDelaysFunctionalMap** (const MapType &f, int n, TimePointType delays[])
- virtual ~DiscreteDelaysFunctionalMap ()
- iterator begin ()
- iterator end ()
- · const iterator begin () const
- · const iterator end () const
- size type imageDimension () const
- · size\_type dimension () const
- · size\_type delaysCount () const
- VectorType operator() (const TimePointType &t, const CurveType &x) const
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out\_u, ValueStorageType &out\_encl, size\_type &out\_encl, admissible\_order) const
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out\_u, size\_type &out\_admissible\_order) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const

computeDDECoefficients

#### **Protected Member Functions**

- void checkDimension (size\_type d) const
- template<typename AnyVector >
  void checkDimension (AnyVector const &v) const
- void checkMapSignature () const
- void findRoughEnclosure (RealType const &t0, RealType const &dt, ValueStorageType const &pastData, VectorType const &z, VectorType &out\_Z, VectorType &out\_Y) const

#### **Protected Attributes**

- MapType m map
- DelayStorageType m\_delays

## **Static Protected Attributes**

- static const double **ROUGH MULFACT** = 1.05
- static const double **ROUGH REFINE FACTOR** = 0.8
- static const double **ROUGH\_TRIALSTEP** = 0.1
- static const size\_type ROUGH\_LIMIT = 32

#### **Additional Inherited Members**

## 7.21.1 Detailed Description

template<typename FinDimMapSpec, typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::Jet← Type>

class capd::ddes::DiscreteDelaysFunctionalMap < FinDimMapSpec, SolutionCurveSpec, JetSpec >

A class to represent right hand side of the DDE of the form:

```
x' = f(t, x(t), x(t-tau_1), ..., x(t-tau_m))
```

NOTE: This is a demo version of what I want to achieve with the DDEs code interface I assume it should be similar to ODEs in original CAPD. I assume that any RHS map f of the equation  $x'(t) = f(x_{tau})$  should be representable with similar interface (i.e. the function accepts some SolutionCurve and can produce Jet at current time in the solution defined with the equation (function computeDDECoefficients) I do not assume anything about Solution  $\leftarrow$  Curve except, it can produce Forward Taylor Jets of itself at a given point (that is used in the equation) and that it can return its value at a current time.

In the implementation below, I assume that I can ask about SOlutionCUrve and its Jet at \$t-\tau\$, \$t\$ is current time. Then I construct an AD jet of x(t-\tau) and pass it to standard CAPD map.

NOTE: I do not know if integral equations (i.e.  $x'(t) = F(\int_{t-1}^t g(x(s))) ds$ ) could be described in this manner... But I hope it could be done, at least if g is linear in x (e.g. g = Id). NOTE: in fact, functional map could have operators that are templates, but we assume that the argument curve might have some requirements when computing, for example it could determine only the delays at the specific grid points (because of the loss of regularity, we are not able to allways give jets over some general intervals, see new notes). Therefore, we supply the Map with appropriate CurveType that it can manipulate.

# 7.21.2 Member Typedef Documentation

#### 7.21.2.1 const iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef DelayStorageType::const_iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMap↔
Spec, SolutionCurveSpec, JetSpec >::const_iterator
```

iterator over this map will iterate over delays!

#### 7.21.2.2 DelayStorageType

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef std::vector< TimePointType > capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec,
SolutionCurveSpec, JetSpec >::DelayStorageType
```

storage type for discrete number of delays

#### 7.21.2.3 iterator

template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef DelayStorageType::iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec,
SolutionCurveSpec, JetSpec >::iterator

iterator over this map will iterate over delays!

### 7.21.3 Constructor & Destructor Documentation

## 7.21.3.1 ~DiscreteDelaysFunctionalMap()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >
::~DiscreteDelaysFunctionalMap ( ) [inline], [virtual]
```

standard

# 7.21.4 Member Function Documentation

## 7.21.4.1 begin() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::begin ( ) [inline]
```

iterator over delays

# 7.21.4.2 begin() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::begin ( ) const [inline]
```

iterator over delays

### 7.21.4.3 collectComputationData() [1/2]

Implementation of virual func. See Interface docs.

TODO: (NOT URGENT) explain what is stored in output for DiscreteDelaysMap...

Warning: dt should be enclosure for the step [0, h] < this is used in computation of rough enclosure, it only need to be the enclosures of value of solution in the past.

## 7.21.4.4 collectComputationData() [2/2]

Implementation of virual func. See Interface docs.

TODO: (NOT URGENT) explain what is stored in output for DiscreteDelaysMap...

Warning: dt should be enclosure for the step [0, h] < this is used in computation of rough enclosure, it only need to be the enclosures of value of solution in the past.

## 7.21.4.5 computeDDECoefficients() [1/2]

Implementation of virual func. See Interface docs. for k loop

### 7.21.4.6 computeDDECoefficients() [2/2]

computeDDECoefficients

see Interface docs.

#### 7.21.4.7 delaysCount()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::delaysCount ( ) const [inline]
```

number of delays

## 7.21.4.8 dimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

# 7.21.4.9 end() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::end ( ) [inline]
```

iterator over delays

# 7.21.4.10 end() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::end ( ) const [inline]
```

iterator over delays

#### 7.21.4.11 findRoughEnclosure()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
void capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >←
::findRoughEnclosure (

    RealType const & t0,
    RealType const & dt,
    ValueStorageType const & pastData,
    VectorType const & z,
    VectorType & out_Z,
    VectorType & out_Y ) const [inline], [protected]
```

Finds a roughEnclosure Z and Y such that:  $Y = z + h * f(t0+h, Z, enclosure_of_past_data(x)) \subset \interior Z h = [t0, t0+dt] and z \in Z, where set Z is chosen heuristically. The theorem guarantees that solution to DDE x(t) exists for initial data z (at t0) for any t \in h and x(t) \in Y. The enclosure_of_past_data(x) is the enclosure of the value of the solution in the past times over intervals of length h. See current paper for better explanation.$ 

Note: the code is direct reimplementation of CAPD code for ODEs

#### 7.21.4.12 imageDimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::imageDimension ( ) const [inline]
```

output dimension of the internal map

# 7.21.4.13 operator()()

Implementation of virual func. See Interface docs.

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

# 7.22 capd::ddes::DiscreteTimeGrid< RealSpec > Class Template Reference

#include <DDECommon.h>

#### **Classes**

class TimePointType

# **Public Types**

• typedef DiscreteTimeGrid ClassType

## **Public Member Functions**

- DiscreteTimeGrid (DiscreteTimeGrid const &orig)
- DiscreteTimeGrid & operator= (DiscreteTimeGrid const &orig)
- **DiscreteTimeGrid** (RealSpec h)
- TimePointType point (int i) const
- TimePointType operator() (int i) const
- void split (RealSpec t, TimePointType &ti, RealSpec &epsi) const

#### **Static Public Member Functions**

• static std::string badge ()

### **Static Public Attributes**

• static const RealSpec **zero** = RealSpec(0)

## **Protected Attributes**

RealSpec m\_h

## 7.22.1 Detailed Description

```
template<typename RealSpec> class capd::ddes::DiscreteTimeGrid< RealSpec >
```

represents time points on a grid i \* h, i - integer, h - real

TODO: (NOT URGENT, FAR FUTURE, RETHINK) rewrite so that TimePoint holds ref to Grid, then no problems will be presented! TODO: (NOT URGENT, FAR FUTURE, RETHINK) (one problem: when we want to have TimePoint(), i.e. to have 0.0 time point)

#### 7.22.2 Member Function Documentation

## 7.22.2.1 badge()

```
template<typename RealSpec >
static std::string capd::ddes::DiscreteTimeGrid< RealSpec >::badge ( ) [inline], [static]
```

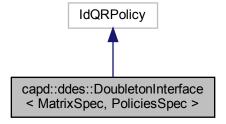
Badge must be a single word!

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

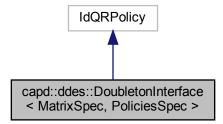
# 7.23 capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec > Class Template Reference

#include <DoubletonInterface.h>

Inheritance diagram for capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >:



Collaboration diagram for capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >:



## **Public Types**

- typedef MatrixSpec MatrixType
- typedef MatrixType::RowVectorType VectorType
- typedef MatrixType::ScalarType ScalarType
- typedef MatrixType::size type size type
- typedef DoubletonInterface Class
- typedef DoubletonInterface BaseClass
- typedef PoliciesSpec Policy
- typedef PoliciesSpec QRPolicy

#### **Public Member Functions**

- virtual void reinitialize (size\_type d, size\_type N0)=0
- virtual size\_type storageDimension () const =0
- virtual size type dimension () const
- virtual size\_type storageN0 () const =0
- virtual VectorType makeStorage x () const
- virtual MatrixType makeStorage C () const
- virtual VectorType makeStorage r0 () const
- virtual MatrixType makeStorage\_B () const
- virtual VectorType makeStorage r () const
- virtual VectorType midPoint () const
- virtual VectorType hull () const
- virtual operator VectorType () const
- virtual VectorType get\_x () const =0
- virtual MatrixType get C () const =0
- virtual MatrixType get B () const =0
- virtual MatrixType get Binv () const
- virtual VectorType get r () const =0
- virtual VectorType get\_r0 () const =0
- virtual Class & set x (VectorType const &x)=0
- virtual Class & set C (MatrixType const &C)=0
- virtual Class & set\_r0 (VectorType const &r0)=0
- virtual Class & set\_B (MatrixType const &B)=0
- virtual Class & set\_r (VectorType const &r)=0
- virtual Class & set\_Cr0 (MatrixType const &C, VectorType const &r0)=0
- virtual Class & set\_x (VectorType \*x, bool passOwnership=false)
- virtual Class & set C (MatrixType \*C, bool passOwnership=false)
- virtual Class & set r0 (VectorType \*r0, bool passOwnership=false)
- virtual Class & set B (MatrixType \*B, bool passOwnership=false)
- virtual Class & set\_r (VectorType \*r, bool passOwnership=false)
- virtual Class & set\_Cr0 (MatrixType \*C, VectorType \*r0, bool passOwnership1=false, bool pass
   Ownership2=false)
- virtual Class & set Binv (MatrixType \*Binv, bool passOwnership=false)
- virtual Class & set Binv (MatrixType const &Binv)
- virtual bool common\_x (VectorType const \*x) const
- virtual bool common C (MatrixType const \*C) const
- virtual bool common\_r0 (VectorType const \*r0) const
- virtual bool common B (MatrixType const \*B) const
- virtual bool common\_r (VectorType const \*r) const
- virtual ScalarType dot (VectorType const &v) const
- virtual Class & add (VectorType const &v)
- virtual Class & add (Class const &set)

- virtual Class & mul (ScalarType const &c)
- virtual Class & mulThenAdd (ScalarType const &c, Class const &set)
- virtual Class & affineTransform (MatrixType const &M, VectorType const &v)=0
- virtual Class & translate (VectorType const &v)=0
- virtual VectorType \* take x ()
- virtual MatrixType \* take\_C ()
- virtual VectorType \* take\_r0 ()
- virtual MatrixType \* take\_B ()
- virtual VectorType \* take\_r ()
- virtual MatrixType \* take Binv ()
- virtual ~DoubletonInterface ()
- DoubletonInterface & operator\*= (ScalarType const &c)

#### **Static Public Member Functions**

• static std::string badge ()

#### **Friends**

- std::ostream & operator<< (std::ostream &out, Class const &c)</li>
- std::istream & operator>> (std::istream &in, Class &c)

## 7.23.1 Detailed Description

template<typename MatrixSpec, typename PoliciesSpec = capd::dynset::ldQRPolicy> class capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >

interface I use in my code to store doubleton data of the form:

$$X = x + C * r0 + B * r$$

I use storage\* to name methods, as for example dimension() is usually related to the algebraic dimension of the space in CAPD. In my case I will have doubletons that represent data by a huge number of coefficients, but in small dimensional space, i.e. piecewise Taylor curves of order n in d -dimensional space. so the storageDimension will be (n+1)\*d;

NOTE: This is quite long and tedious code in C++ because of C++ This should be heavily tested for correctedness and memory leaks Reader (e.g. reviewer of the manuscript for publication) should not worry to check that code

NOTE: THIS IS JUST AN INTERFACE for a doubleton set. We provide two specific versions: Basic\* and Shared\* - first for testing, second for applications.

TODO: (NOT URGENT) move big implementation into .hpp part. TODO: (FAR FUTURE, RETHINK) make set 
\_\*(ptr\*) methods return bool if either they truly accepted to hold the ownership and the ptr and it is really shared. 
Then user would be able to take actions accordingly. Now, it is not allways so obvious.

## 7.23.2 Constructor & Destructor Documentation

#### 7.23.2.1 ∼ DoubletonInterface()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::~DoubletonInterface ( )
[inline], [virtual]
```

virtual destructor for strict warning compliance

#### 7.23.3 Member Function Documentation

### 7.23.3.1 add() [1/2]

add a Set to this representation. It takes the representation into consideration.

 $\label{lem:des::DDESolutionCurve} Reimplemented in capd::ddes::DDESolutionCurve < SetSpec >.$ 

## 7.23.3.2 add() [2/2]

add vector to this representation. It is distributed between x and B\*r part by default

Reimplemented in capd::ddes::DDESolutionCurve < SetSpec >, and capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >.

## 7.23.3.3 affineTransform()

applies in a smart way to this set X the affine transform f(y) = M \* (X - v). Needs to be implemented.

Implemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, PoliciesS

#### 7.23.3.4 common B()

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, Policies

## 7.23.3.5 common\_C()

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, Policies

#### 7.23.3.6 common r()

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, Policies

## 7.23.3.7 common\_r0()

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, Policie

#### 7.23.3.8 common\_x()

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, Policie

#### 7.23.3.9 dimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dimension ( )
const [inline], [virtual]
```

by default dimension is equal to storage dimension

#### 7.23.3.10 dot()

computes v . thisSet, it is virtual to allow optimization. Now uses get\_\* in computation (might be slow)

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >.

#### 7.23.3.11 get B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_B ( ) const
[pure virtual]
```

returns B value of the doubleton. Needs to be implemented.

# 7.23.3.12 get\_Binv()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_Binv ( )
const [inline], [virtual]
```

returns inverse of B value of the doubleton. Default implementation by capd inverse.

## 7.23.3.13 get\_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_C ( ) const
[pure virtual]
```

returns B value of the doubleton. Needs to be implemented.

#### 7.23.3.14 get\_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_r ( ) const
[pure virtual]
```

returns r value of the doubleton. Needs to be implemented.

## 7.23.3.15 get\_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_r0 ()
const [pure virtual]
```

returns r0 value of the doubleton. Needs to be implemented.

## 7.23.3.16 get\_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_x ( ) const
[pure virtual]
```

returns x value of the doubleton. Needs to be implemented.

#### 7.23.3.17 hull()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::hull ( ) const
[inline], [virtual]
```

returns mid point of the set, default: returns value obtained using get\_\*() methods. You might want to reimplement for better performance(?)

#### 7.23.3.18 makeStorage B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_B (
) const [inline], [virtual]
```

makes a vector that can store B part

## 7.23.3.19 makeStorage\_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_C (
) const [inline], [virtual]
```

makes a vector that can store C part

#### 7.23.3.20 makeStorage\_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_r (
) const [inline], [virtual]
```

makes a vector that can store r part

#### 7.23.3.21 makeStorage\_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_r0
( ) const [inline], [virtual]
```

makes a vector that can store r0 part

## 7.23.3.22 makeStorage x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_x (
) const [inline], [virtual]
```

makes a vector that can store x part

#### 7.23.3.23 midPoint()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::midPoint ( )
const [inline], [virtual]
```

returns mid point of the set, default: returns get\_X();

# 7.23.3.24 mul()

multiply set by a scalar. Should take set structure into consideration.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.25 mulThenAdd()

multiply set by a scalar, then add another set. Very simple basic implementation by first mul then add.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >.

#### 7.23.3.26 operator VectorType()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::operator VectorType ( )
const [inline], [virtual]
```

returns Vector representation of the set, default: returns hull()

## 7.23.3.27 operator\*=()

alternative to mul(); uses mul to gain polymorphism

# 7.23.3.28 reinitialize()

reinitializes the Doubleton to a gibven structure (all should be 0 or Id for B, Binv)

 $Implemented \ in \ capd:: ddes:: DDESolution Curve < SetSpec >, \ capd:: ddes:: Shared Doubleton < Matrix Spec, \ Policy Spec >, \ and \ capd:: ddes:: Basic Doubleton < Matrix Spec, \ Policies Spec >. \\$ 

#### 7.23.3.29 set\_B() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the preceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.30 set\_B() [2/2]

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Implemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, PoliciesS

#### 7.23.3.31 set\_Binv() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the preceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

 $\label{lem:lemented$ 

#### 7.23.3.32 set Binv() [2/2]

for future implementations. By default it does nothing - Binv is computed always by inverse matrix. NOTE: user of this function is responsible for providing TRUE inverse. Code does not check anything in this regard.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

## 7.23.3.33 set\_C() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the preceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.34 set\_C() [2/2]

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Implemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, PoliciesS

# 7.23.3.35 set\_Cr0() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the preceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

## 7.23.3.36 set\_Cr0() [2/2]

This is to allow to safely change the structure of the set. If changing r0 dimension, then C matrix must be updated to match the dimension, and this must be an atomic operation. Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Implemented in capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, PoliciesS

## 7.23.3.37 set\_r() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the preceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

## 7.23.3.38 set r() [2/2]

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

## 7.23.3.39 set\_r0() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

## 7.23.3.40 set\_r0() [2/2]

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Implemented in capd::ddes::DDESolutionCurve < SetSpec >, capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton < MatrixSpec, PoliciesSpec >.

## 7.23.3.41 set\_x() [1/2]

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set\_\* method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

 $Reimplemented \ in \ capd:: ddes:: DDESolution Curve < Set Spec >, \ and \ capd:: ddes:: Shared Doubleton < Matrix Spec, \ Policy Spec >.$ 

# 7.23.3.42 set\_x() [2/2]

```
\label{local_continuous_period} $$\operatorname{capd::dynset::IdQRPolicy} $$\operatorname{capd::des::DoubletonInterface} < \operatorname{MatrixSpec}, \operatorname{PoliciesSpec} > :: \operatorname{set\_x} ($$\operatorname{VectorType const \& } x ) [$\operatorname{pure virtual}]$
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return \*this for a cascade.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

#### 7.23.3.43 storageDimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::storageDimension
( ) const [pure virtual]
```

returns a dimension of the stored data. Needs to be implemented.

## 7.23.3.44 storageN0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::storageN0 ( )
const [pure virtual]
```

returns a dimension of the r0 vector. Needs to be implemented.

### 7.23.3.45 take\_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_B ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get\_\*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.46 take Binv()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_Binv ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get\_\*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.47 take C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_C ()
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get\_\*

Reimplemented in capd::ddes::DDESolutionCurve < SetSpec >, and capd::ddes::SharedDoubleton < MatrixSpec, PolicySpec >.

# 7.23.3.48 take\_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_r ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get\_\*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

## 7.23.3.49 take\_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_r0 ()
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get\_\*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.50 take\_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_x ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get \*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

#### 7.23.3.51 translate()

applies in a smart way to this set X the transform f(y) = X - v. Needs to be implemented.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

### 7.23.4 Friends And Related Function Documentation

## 7.23.4.1 operator <<

Should be good for any Doubleton that inherits

## 7.23.4.2 operator>>

Should be good for any Doubleton that inherits

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Doubleton
 — Interface.h

# 7.24 ElNino < ScalarSpec, ParamSpec > Class Template Reference

```
#include <setup-common.h>
```

# **Public Types**

- typedef ScalarSpec ScalarType
- · typedef unsigned int size\_type
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarType, 0 > VectorType

#### **Public Member Functions**

- **EINino** (ParamType kappa)
- **ElNino** (ParamType alpha, ParamType beta, ParamType gamma, ParamType kappa)
- ElNino (ElNino const &orig)
- **ElNino** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- ElNino & operator= (ElNino const &orig)
- template < typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

- static size\_type imageDimension ()
- static size\_type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

## **Protected Attributes**

- · ParamType alpha
- · ParamType beta
- ParamType gamma
- ParamType kappa

## 7.24.1 Detailed Description

```
template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class ElNino < ScalarSpec, ParamSpec >
```

A model map of what I expect from a Map  $R^{n} -> R^{n}$  to be good for use with DDE codes.

## 7.24.2 Member Function Documentation

#### 7.24.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type ElNino< ScalarSpec, ParamSpec >::dimension ( ) [inline], [static]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

## 7.24.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type ElNino< ScalarSpec, ParamSpec >::getParamsCount () [inline], [static]
```

number of parameters to fully configure equation

#### 7.24.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type ElNino< ScalarSpec, ParamSpec >::imageDimension ( ) [inline], [static]
```

output dimension of the internal map

## 7.24.2.4 operator()()

We require that solution class has a templated operator of this signature the program will pass in x the m values where m = dimension() in fx there will be reference to already initialized vector of dimension d = imageDimension(). The dimensions are as follows: if d = imageDimension() then usually m = d \* (number of delayed terms + 1). +1 is for the current term, which is always present, and always assumed to be stored in <math>x[0]. (so if your equation is not dependent on this, then you should not use it in your formulas). In this example (Mackey-Glass Eq.), we have: d = 1 (scalar), m = 2 (two terms: current value at 0 and delayed term at t = t - tau)

Note that tau is not explicitly present in the equation. The concrete value of the delay is defined when constructing Discrete Delays Functional Map from this template function. See docs there.

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

• /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/programs/examples/demo-elninio/setup-common.h

# 7.25 capd::ddeshelper::GeneratorRange< IType > Class Template Reference

# **Public Member Functions**

- GeneratorRange (IType lo=0, IType up=0, IType dt=1)
- int i ()
- IType **v** ()
- IType **lo** ()
- IType up ()
- void next ()
- bool isFinished ()
- · void reset ()
- bool isTrivial ()
- template < typename iterator > void fill (iterator from, iterator to)

## **Static Public Member Functions**

template < typename iterator >
 static bool increment (iterator from, iterator to)

#### **Public Attributes**

- IType start
- · IType end
- IType inc
- int index

# **Friends**

• std::ostream & operator<< (std::ostream &out, GeneratorRange const &range)

#### 7.25.1 Member Function Documentation

# 7.25.1.1 increment()

returns true if the incrementation was successful and the next element is available returns false if the incrementation reached last element and reseted all elements (we are back to all indices 0)

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

# 7.26 capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > Class Template Reference

```
#include <GenericJet.h>
```

#### **Classes**

struct rebind

# **Public Types**

- typedef TimePointSpec TimePointType
- typedef DataSpec DataType
- typedef VectorSpec VectorType
- typedef MatrixSpec MatrixType
- typedef GenericJet < TimePointType, DataType, VectorType, MatrixType > Class
- · typedef Class BaseClass
- typedef MatrixType::ScalarType
- typedef MatrixType::size\_type size\_type
- typedef ScalarType RealType
- typedef Class JetType
- typedef DataType const \* const\_iterator
- typedef DataType \* iterator

#### **Public Member Functions**

- virtual GenericJet & setupCoeffs (const iterator it, const iterator itEnd, size type overrideOrder=0)
- virtual GenericJet & setupCoeffs (std::vector < DataType > const &coeffs, size\_type overrideOrder=0)
- GenericJet & operator= (GenericJet const & orig)
- · GenericJet (GenericJet const &orig)
- GenericJet (TimePointType t0=TimePointType(), size type dimension=0, size type order=0)
- GenericJet (TimePointType t0, size\_type order, const DataType &v)
- GenericJet (TimePointType t0, const\_iterator it, const\_iterator itEnd)
- GenericJet (TimePointType t0, size type order, const iterator it, const iterator itEnd)
- GenericJet (TimePointType t0, std::vector< DataType > coeffs)
- size\_type dimension () const
- size\_type order () const
- size type storageDimension () const
- virtual VectorType evalAtDelta (const RealType &delta\_t) const
- virtual void evalAtDelta (const RealType &delta\_t, DataType &out) const
- virtual VectorType eval (const RealType &t) const
- virtual void eval (const RealType &t, DataType &out) const
- virtual VectorType evalCoeffAtDelta (size\_type n, const RealType &delta\_t) const
- virtual void evalCoeffAtDelta (size\_type n, const RealType &delta\_t, DataType &out) const
- virtual VectorType evalCoeff (size type n, const RealType &t) const
- virtual void evalCoeff (size\_type n, const RealType &t, DataType &out) const
- GenericJet dt (size type n=1)
- GenericJet coeff (size type n=1)
- · virtual std::string show () const
- operator VectorType () const
- virtual VectorType hull () const
- Class & set\_x (VectorType const &x)
- VectorType get\_x () const
- iterator begin ()
- iterator end ()
- iterator back ()
- iterator at (size type k)
- const\_iterator begin () const
- · const iterator end () const
- const\_iterator back () const
- const\_iterator at (size\_type k) const
- DataType & operator[] (size\_type k)
- DataType const & operator[] (size\_type k) const

- Class & setT0 (TimePointType const &t0)
- TimePointType getT0 () const
- TimePointType t0 () const
- DataType \* getCoeffs ()
- Class setCoeffs (DataType \*coeffs, size type order)
- Class setOrder (size type n)
- Class increasedOrder (size\_type n=1)
- Class decreasedOrder (size\_type n=1)
- virtual ∼GenericJet ()

#### Static Public Member Functions

• static std::string badge ()

# **Protected Member Functions**

- void allocateCoeffs ()
- void deallocateCoeffs ()
- void reallocateCoeffs ()

# **Protected Attributes**

- TimePointType m\_t0
- size\_type m\_dimension
- · size\_type m\_order
- DataType \* m coeffs

# **Friends**

- std::ostream & operator<< (std::ostream &out, GenericJet const &jet)
- std::istream & operator>> (std::istream &in, GenericJet &jet)

# 7.26.1 Detailed Description

template<typename TimePointSpec, typename DataSpec, typename VectorSpec = typename DataSpec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd::TypeTraits<typename MatrixSpec::ScalarType>::is <-- Interval>

class capd::ddes::GenericJet < TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >

A class to represent a generic jet for any data type

We assume that DataType has default constructor and is convertible form/to VectorType, i.e. DataType has:

- a constructor DataType(VectorType) or DataType(const VectorType&)
- a cast operator VectorType()
- · a copy operator and copy constructor.
- · default constructor

Examples: capd::Vector and Sets types are compatible. ddes/storage/doubleton types are compatible.

#### 7.26.2 Constructor & Destructor Documentation

#### 7.26.2.1 GenericJet() [1/6]

copy constructor

#### 7.26.2.2 GenericJet() [2/6]

makes a constant function of the vector value 0

# 7.26.2.3 GenericJet() [3/6]

makes a constant function of the vector value v

# 7.26.2.4 GenericJet() [4/6]

fills vector with data. Determine order from the size of data.

# 7.26.2.5 GenericJet() [5/6]

fills vector with data, but sets the order to n (fills higher orders with 0 vectors)

#### 7.26.2.6 GenericJet() [6/6]

fills vector with data. Determine order from the size of data.

# 7.26.2.7 ∼GenericJet()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual capd::ddes::GenericJet < TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::~GenericJet () [inline], [virtual]
```

standard thing

# 7.26.3 Member Function Documentation

# 7.26.3.1 allocateCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\hookrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\hookrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >\hookrightarrow::allocateCoeffs ( ) [inline], [protected]
```

TODO: docs

#### 7.26.3.2 at() [1/2]

iterator to the k-th element in coefficients. Standard thing in C++

### 7.26.3.3 at() [2/2]

iterator to the k-th element in coefficients. Standard thing in C++

# 7.26.3.4 back() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::back ( ) [inline]
```

iterator to the last element in coefficients. Standard thing in C++

# 7.26.3.5 back() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↔

Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↔

::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>

const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↔

Interval >::back ( ) const [inline]
```

iterator to the last element in coefficients. Standard thing in C++

# 7.26.3.6 badge()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
static std::string capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec,
isInterval >::badge ( ) [inline], [static]
```

Badge must be a single word!

# 7.26.3.7 begin() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::begin ( ) [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

# 7.26.3.8 begin() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↔

Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↔

::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>

const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↔

Interval >::begin () const [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

# 7.26.3.9 coeff()

makes a Jet that represents n-th coefficient (i.e. jet \(^{(n)}/n!\) of the jet as a function of t, by default n=1

### 7.26.3.10 deallocateCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data←
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd←
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >←
::deallocateCoeffs ( ) [inline], [protected]
```

TODO: docs

#### 7.26.3.11 decreasedOrder()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\leftrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\leftrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval> Class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > \leftrightarrow::decreasedOrder ( size_type n=1) [inline]
```

TODO: docs

#### 7.26.3.12 dimension()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data←
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd←
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
size_type capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::dimension ( ) const [inline]
```

dimension of the value space of the Jet (i.e.  $J : R \to R^{(dimension)}$ )

#### 7.26.3.13 dt()

makes a Jet that represents n-th derivative of the jet, by default n=1

#### 7.26.3.14 end() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::end () [inline]
```

iterator to the past the order place. Standard thing in C++

### 7.26.3.15 end() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data←
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd←
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is←
Interval >::end () const [inline]
```

iterator to the past the order place. Standard thing in C++

# 7.26.3.16 eval() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\leftrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\leftrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval> virtual VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::eval ( const RealType & t ) const [inline], [virtual]
```

evaluates the the jet at t

# 7.26.3.17 eval() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\leftrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\leftrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval> virtual void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is\leftrightarrow Interval >::eval ( const RealType & t, DataType & out ) const [inline], [virtual]
```

evaluates the jet at t

#### 7.26.3.18 evalAtDelta() [1/2]

```
\label{template} $$ \textbf{typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix} $$ \textbf{Spec , bool isInterval} $$ \textbf{VectorSpec capd::ddes::GenericJet} < \textbf{TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval} $$ \textbf{Spec , typename Matrix} $$ \textbf{Spec , DataSpec, VectorSpec, MatrixSpec, isInterval} $$ \textbf{Spec , Const RealType & delta_t ) const [virtual]} $$
```

evaluates the the jet at t0 + delta\_t

### 7.26.3.19 evalAtDelta() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix \leftrightarrow Spec , bool isInterval> void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > \leftrightarrow ::evalAtDelta ( const RealType & delta_t, DataType & out ) const [virtual]
```

evaluates the jet at t0 + delta t

#### 7.26.3.20 evalCoeff() [1/2]

evaluates the n-th time derivative of the jet at t (n-th coeff is  $1/n! * jet^{(n)}(t) = jet^{(n)}(t)$ )

# 7.26.3.21 evalCoeff() [2/2]

evaluates the n-th time derivative of jet at t (n-th coeff is  $1/n! * jet^{(n)}(t) = jet^{(n)}(t)$ )

#### 7.26.3.22 evalCoeffAtDelta() [1/2]

evaluates the n-th Coeff of jet at t0 + delta\_t ( n-th coeff is  $1/n! * jet^{(n)}(t) = jet^{[n]}(t)$  )

#### 7.26.3.23 evalCoeffAtDelta() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix \leftrightarrow Spec , bool isInterval> void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval> \leftrightarrow ::evalCoeffAtDelta ( size_type n, const RealType & delta_t, DataType & out) const [virtual]
```

evaluates the n-th time derivative of jet at t0 + delta\_t ( n-th coeff is  $1/n! * jet^{(n)}(t) = jet^{(n)}(t)$  )

#### 7.26.3.24 get x()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::get_x ( ) const [inline]
```

gets the coefficients as a single vector of dimension storageDimension()

### 7.26.3.25 getCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↔

Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↔

::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>

DataType* capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval

>::getCoeffs ( ) [inline]
```

TODO: docs

#### 7.26.3.26 getT0()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\hookrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\hookrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
TimePointType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is\hookrightarrow Interval >::getT0 () const [inline]
```

returns time at which this jet is located

#### 7.26.3.27 hull()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec,
isInterval >::hull ( ) const [inline], [virtual]
```

return vector representation of the jet. Other name (more explicit for VectorType), can be overwritten (virtual).

# 7.26.3.28 increasedOrder()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\leftrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\leftrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval> Class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > \leftrightarrow::increasedOrder ( size_type n=1) [inline]
```

TODO: docs

# 7.26.3.29 operator VectorType()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data←
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd←
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::operator
VectorType ( ) const [inline]
```

return VectorSpec representation of the jet. It is vector of dimension storageDimension and d-dim blocks of the natural order (0-th coefficient first).

# 7.26.3.30 operator=()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix←
Spec , bool isInterval>
GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > & capd::ddes::GenericJet<
TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::operator= (
GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > const &
orig )
```

copy operator

### 7.26.3.31 operator[]() [1/2]

returns n-th order Taylor coefficient.

# 7.26.3.32 operator[]() [2/2]

returns n-th order Taylor coefficient.

#### 7.26.3.33 order()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
size_type capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::order ( ) const [inline]
```

order of the Taylor part of the Jet

# 7.26.3.34 reallocateCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data←
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd←
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >←
::reallocateCoeffs ( ) [inline], [protected]
```

TODO: docs

### 7.26.3.35 set\_x()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data\leftrightarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd\leftrightarrow::TypeTraits<typename MatrixSpec::ScalarType>::isInterval> Class& capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > \leftrightarrow::set_x ( VectorType const & x ) [inline]
```

set the coeficients from a vector. get/set\_x() are compatible.

#### 7.26.3.36 setCoeffs()

TODO: docs

#### 7.26.3.37 setOrder()

TODO: docs

#### 7.26.3.38 setT0()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data \leftarrow Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd \leftarrow ::TypeTraits<typename MatrixSpec::ScalarType>::isInterval> Class& capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > \leftarrow ::setT0 ( TimePointType const & t0) [inline]
```

set the base time of the jet

#### 7.26.3.39 show()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix \leftarrow Spec , bool isInterval> std::string capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is \leftarrow Interval >::show [virtual]
```

more verbose output, more human-friendly. Not rigorous-friendly.

# 7.26.3.40 storageDimension()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
size_type capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::storageDimension ( ) const [inline]
```

how many data need to store everything

#### 7.26.3.41 t0()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
TimePointType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is
Interval >::t0 ( ) const [inline]
```

naming convention. See getT0()

#### 7.26.4 Friends And Related Function Documentation

#### 7.26.4.1 operator <<

TODO: docs

#### 7.26.4.2 operator>>

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data←

Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd←

::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>

std::istream& operator>> (

std::istream & in,

GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > & jet )

[friend]
```

TODO: docs

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

# 7.27 capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

# **Public Types**

- typedef ScalarSpec ScalarType
- · typedef unsigned int size\_type
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarType, 0 > VectorType

#### **Public Member Functions**

- LasotaWazewska (ParamType n)
- LasotaWazewska (ParamType beta, ParamType gamma, ParamType n)
- LasotaWazewska (LasotaWazewska const &orig)
- LasotaWazewska (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- LasotaWazewska & operator= (LasotaWazewska const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

- static size type imageDimension ()
- static size\_type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

# **Protected Attributes**

- · ParamType beta
- · ParamType gamma
- · ParamType n

# 7.27.1 Detailed Description

```
template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >
```

Lasota-Wazewska equation for chaos as mentioned in scholarpedia article on Mackey-Glass equation.

```
x'(t) = -gamma * x(t) + beta * x(t-tau)^n * e^(-x(t-tau))
```

# 7.27.2 Member Function Documentation

# 7.27.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::dimension ( ) [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

#### 7.27.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::getParamsCount ( )
[inline], [static]
```

number of parameters to fully configure equation

#### 7.27.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::imageDimension ( )
[inline], [static]
```

output dimension of the internal map

# 7.27.2.4 operator()()

Evaluate r.h.s of the equation.

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

# 7.28 capd::ddes::LinearMap< MatrixSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

# **Public Types**

- typedef MatrixSpec MatrixType
- typedef MatrixSpec::RowVectorType VectorType
- typedef VectorType::ScalarType ScalarType
- typedef VectorType::size type size type
- typedef ScalarType RealType
- typedef ParamSpec ParamType

#### **Public Member Functions**

- · void reinitialize ()
- LinearMap (size\_type d)
- LinearMap (MatrixType const &A, MatrixType const &B, VectorType const &b)
- LinearMap (LinearMap const &orig)
- LinearMap & operator= (LinearMap const &orig)
- size\_type imageDimension () const
- size type dimension () const
- template < typename RealSpec , typename InVectorSpec , typename OutVectorSpec > void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

# **Static Protected Member Functions**

• static std::string show ()

# **Protected Attributes**

- MatrixType A
- MatrixType B
- VectorType b
- MatrixType AB

# 7.28.1 Detailed Description

template<typename MatrixSpec, typename ParamSpec = typename MatrixSpec::ScalarType> class capd::ddes::LinearMap< MatrixSpec, ParamSpec >

Vector linear equation of the form:

```
x'(t) = A * x(t) + B * x(t-tau) + b
```

Use with 'DiscreteDelaysFunctionalMap' to define tau

#### 7.28.2 Member Function Documentation

# 7.28.2.1 dimension()

```
template<typename MatrixSpec , typename ParamSpec = typename MatrixSpec::ScalarType>
size_type capd::ddes::LinearMap< MatrixSpec, ParamSpec >::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one,  $x(t-tau_1)$  is another, etc.. (thus the formula)

#### 7.28.2.2 imageDimension()

```
template<typename MatrixSpec , typename ParamSpec = typename MatrixSpec::ScalarType>
size_type capd::ddes::LinearMap< MatrixSpec, ParamSpec >::imageDimension ( ) const [inline]
```

output dimension of the internal map

#### 7.28.2.3 operator()()

We require that solution class has a templated operator of this signature the program will pass in x the m values where m = dimension() in fx there will be reference to already initialized vector of dimension d = imageDimension(). The dimensions are as follows: if d = imageDimension() then usually m = d \* (number of delayed terms + 1). +1 is for the current term, which is always present, and always assumed to be stored in <math>x[0]. (so if your equation is not dependent on this, then you should not use it in your formulas). In this example (Mackey-Glass Eq.), we have: d = 1 (scalar), m = 2 (two terms: current value at 0 and delayed term at t = t - tau)

Note that tau is not explicitely present in the equation. The concrete value of the delay is defined when constructing DiscreteDelaysFunctionalMap from this template function. See docs there.

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

# 7.29 capd::ddes::MackeyGlass< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

# **Public Types**

- typedef ScalarSpec ScalarType
- typedef unsigned int size\_type
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarType, 0 > VectorType

#### **Public Member Functions**

- MackeyGlass (ParamType n)
- MackeyGlass (ParamType beta, ParamType gamma, ParamType n)
- MackeyGlass (MackeyGlass const &orig)
- MackeyGlass (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- MackeyGlass & operator= (MackeyGlass const & orig)
- template < typename RealSpec , typename InVectorSpec , typename OutVectorSpec > void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

# **Static Public Member Functions**

- static size\_type imageDimension ()
- static size type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

# **Protected Attributes**

- ParamType beta
- · ParamType gamma
- ParamType n

# 7.29.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >

For now, you need to use class similar to those below to define f(x, y, ...) in your equation. Then you define delays in 'DiscreteDelaysFunctionalMap' or some other capd::ddes::FunctionalMap derived classes. Please remember that usually 'DiscreteDelaysFunctionalMap' must specify the Solution template parameter to be able to determine how to handle solution curves.

The setup is much complicated than in the case of ODEs, as the equations must take into account the problem with the grid over which the solution is defined (as of now). See the problem with the continuity of solution at grid points and long enough integration time described in the papers. Because of those problems, the grid, the solution and the equation (DiscreteDelaysFunctionalMap) must be aware to some extent of each other. TODO: (FAR FUTURE) try to reduce dependency only to Grid (i.e. DiscreteDelaysFunctionalMap and Solution depend on the Grid, TODO: (FAR FUTURE) but not on each other). It could be done, if DiscreteDelaysFunctionalMap accepts all solutions that can TODO: (FAR FUTURE) produce concrete Interface of a Jet at Grid::TimePoints)

Example: typedef BasicDoubleton<IMatrix> SetType; typedef DDESolutionCurve<SetType> SolutionType; typedef MackeyGlass<typename SolutionType::ScalarType> F; typedef DiscreteDelaysFunctionalMap<F> RHS; typedef typename SolutionType::GridType GridType; int p = 128; GridType grid(2.0 / p); F f(2.0, 1.0, 8.0); RHS rhs(f, grid(2 \* p));

TODO: (FAR FUTURE) make a class that can use integral differential equation to computeDDECoefficients()

TODO: (FAR FUTURE) Unfortunatelly, as of today, capd::map::Map is not good for my use. TODO: (FAR FUTURE) Need to consult Daniel Wilczak on the matter. TODO: (FAR FUTURE) Need to understand the Node and the Graph of execution used by Daniel Wilczak TODO: (FAR FUTURE) to implement my version of computeDDECoefficients with capd::map::Map (seems doable) A model map of what I expect from a Map  $R^m -> R^n$  to be good for use with DDE codes.

#### 7.29.2 Member Function Documentation

#### 7.29.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::dimension ( ) [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

#### 7.29.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::getParamsCount ( ) [inline],
[static]
```

number of parameters to fully configure equation

# 7.29.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::imageDimension ( ) [inline],
[static]
```

output dimension of the internal map

# 7.29.2.4 operator()()

We require that solution class has a templated operator of this signature the program will pass in x the m values where m = dimension() in fx there will be reference to already initialized vector of dimension d = imageDimension(). The dimensions are as follows: if d = imageDimension() then usually m = d \* (number of delayed terms + 1). +1 is for the current term, which is always present, and always assumed to be stored in <math>x[0]. (so if your equation is not dependent on this, then you should not use it in your formulas). In this example (Mackey-Glass Eq.), we have: d = 1 (scalar), m = 2 (two terms: current value at 0 and delayed term at t = t - tau)

Note that tau is not explicitly present in the equation. The concrete value of the delay is defined when constructing Discrete Delays Functional Map from this template function. See docs there.

#### 7.29.3 Member Data Documentation

#### 7.29.3.1 gamma

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::gamma [protected]
classical parameter, default: 2
```

#### 7.29.3.2 n

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::n [protected]
```

classical parameter, default: 1

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

# 7.30 capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec > Class Template Reference

#include <DDEHelperNonrigorous.h>

# **Public Types**

- · typedef EqSpec Eq
- · typedef Eq::ScalarType Scalar
- typedef Eq::RealType Real
- typedef Eq::ParamType ParamType
- typedef capd::vectalg::Vector < Scalar, 0 > Vector
- typedef capd::vectalg::Matrix < Scalar, 0, 0 > Matrix
- typedef capd::vectalg::Vector< ParamType, 0 > ParamsVector
- typedef capd::ddes::DiscreteTimeGrid < Real > Grid
- typedef Grid::TimePointType TimePoint
- typedef capd::ddes::GenericJet < TimePoint, Vector, Vector, Matrix > Jet
- typedef capd::ddes::GenericJet < TimePoint, capd::ddes::VectorWithJacData < Vector, Matrix >, Vector, Matrix > JetWithJacobian
- typedef capd::ddes::DDEPiecewisePolynomialCurve< Grid, Jet > Solution
- typedef capd::ddes::DDEPiecewisePolynomialCurve< Grid, JetWithJacobian > JacSolution
- typedef Jet CurvePiece
- typedef capd::ddes::BasicDiscreteDelaysFunctionalMap< Eq, JacSolution > JacDDEq
- typedef capd::ddes::BasicDiscreteDelaysFunctionalMap< Eq, Solution > DDEq
- $\bullet \ \ type def \ capd:: ddes:: DDEN on rigorous Taylor Solver < Jac DDEq > \textbf{Jac Solver} \\$
- typedef capd::ddes::DDENonrigorousTaylorSolver< DDEq > Solver
- typedef JacSolver::VariableStorageType Variables
- typedef JacSolver::JacobianStorageType Jacobians
- typedef JacSolver::ValueStorageType Values
- typedef JacSolver::size\_type size\_type
- · typedef int step\_type
- typedef capd::ddes::DDEJetSection < JacSolution > JacJetSection
- typedef JacJetSection::JetType JacSecJet
- typedef capd::ddes::DDEBasicPoincareMap< JacSolver, JacJetSection > JacPoincareMap
- typedef capd::ddes::DDEJetSection < Solution > JetSection
- typedef capd::ddes::DDEBasicPoincareMap< Solver, JetSection > PoincareMap
- typedef capd::ddeshelper::CoordinateFrame< Matrix, Vector, Scalar > CoordinateFrame

#### **Public Member Functions**

NonrigorousHelper (ParamsVector params, size\_type p, size\_type n, step\_type reqSteps=0, step\_type max
 —
 Steps=0, size type maxOrder=10)

- NonrigorousHelper (std::string filepath, step\_type reqSteps=0, step\_type maxSteps=0, size\_type max

   Order=10)
- NonrigorousHelper (std::istream &input, step\_type reqSteps=0, step\_type maxSteps=0, size\_type max

   Order=10)
- Solver makeSolver ()
- Solution integrate (int iters, const Vector &initial, Vector &result, bool use\_extension=false)
- Solution integrate (int iters, const Vector &initial)
- Solution integrate (int iters, const Vector &initial, Vector &result, PathConfig const &outconfig)
- Real iteratePoincare (std::ostream &info, int iters, JacJetSection &section, Vector &x, Matrix &V, Matrix &DP)
- template < typename SecionType > void makeSection (Vector const &s, Scalar const &c, SecionType &out section)
- Real refinePeriodic (std::ostream &info, JacJetSection &section, Vector &x, Matrix &V, Matrix &DP)
- capd::poincare::CrossingDirection detectCorssingDirection (JacJetSection section, Vector const &x)
- JacSolution poincare (JacJetSection section, Vector const &x, Matrix &initV, double &reachTime, int &steps, Vector &Px, Vector &fPx, Matrix &V, Matrix &DP)
- JacSolution poincare (JacJetSection section, Vector const &x0, double &reachTime, Vector &Px, Vector &fPx, Matrix &V, Matrix &DP)
- Solution poincare (JetSection section, Vector const &x, double &reachTime, Vector &Px)
- Solution vectorToSolution (Vector const &x) const
- JacSolution vectorToJacSolution (Vector const &x) const
- template<typename SolutionT >
  - std::string drawSolution (std::string dirpath, std::string filename, SolutionT const &X, double tshift=0.) const
- std::string drawSolution (std::string dirpath, std::string filename, Vector const &x, double tshift=0.) const
- template<typename SolutionT >
  - void drawSolution (PathConfig const &paths, SolutionT const &X, double tshift=0.) const
- template<typename SolutionT >
  - void drawDelayMap (std::string dirpath, std::string filename, SolutionT const &X) const
- $\bullet \quad {\sf template}{<} {\sf typename SolutionT} >$
- void drawDelayMap (PathConfig const &paths, SolutionT const &X) const
- template<typename VectorIteratorSpec >
  - void **saveData** (std::string filepath, VectorIteratorSpec start, VectorIteratorSpec end, std::string extra ← Comment="")
- void loadSetup (std::string filepath)
- std::vector < Vector > loadData (std::string filepath)
- void **setRequiredSteps** (step\_type reqSteps, bool control\_steps=true)
- void setMaximumSteps (step\_type maxSteps, bool control\_steps=true)
- step type getRequiredSteps () const
- · step\_type getMaximumSteps () const
- step\_type getMaximumOrder () const
- ParamsVector params () const
- size\_type M () const
- size\_type p () const
- size\_type n () const
- size\_type d () const
- Real h () const
- Real getBasicIntervalLength () const
- void setCrossingDirection (capd::poincare::CrossingDirection const &d)
- · const Grid & grid () const
- NonrigorousHelper & setExperimentalRenormalizeVariational (bool v)
- ParamsVector setParams (ParamsVector const &new\_params)
- ParamType setParam (size\_type index, ParamType const &new\_param)
- TimePoint t (int i) const

#### **Static Public Attributes**

- static const size\_type PARAMS\_COUNT = EqSpec::getParamsCount() + delaysSpec
- static const size type **DIMENSION** = EqSpec::imageDimension()

# 7.30.1 Detailed Description

```
template<typename EqSpec, int delaysSpec = 1> class capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >
```

A helper class that made all the fuzzy boilerplate for you, so you do not need to know various things by yourself.

It provides functions like integrate and poincare to compute images of the initial functions under your own equations. You only need to supply the equation and the number of delays in the system

It only server discrete delay differential equations in the form of

```
x'(t) = f(x(t), x(t-\lambda u_1), ..., x(t-tau_m))
```

TODO: (IMPORTANT) Currently it only support one delay!!!!!! TODO: make sure all const qualifiers are put in the right places!

#### 7.30.2 Constructor & Destructor Documentation

# 7.30.2.1 NonrigorousHelper() [1/3]

you need to call at least one of the constructors and then use the setup variable

This constructor takes parameters vector:

```
(q_1, ..., q_k, tau_1, ... tau_m)
```

where q\_i's are the parameters of the equation (r.h.s f :  $\R^{d*(m+1)}\$  to  $\R^d$  of the DDE) and tau\_i's are the delays, so that the equation is

```
x'(t) = f(x(t), x(t-tau 1), ...)
```

p, n - are parameters of the (p,n)-representation of the functions. p is the number of grid points in the basic interval [-tau, 0],  $tau = max(tau_i)$  n is the order of the jet at each point. See papers for more details.

# 7.30.2.2 NonrigorousHelper() [2/3]

```
template<typename EqSpec , int delaysSpec = 1>
capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::NonrigorousHelper (
    std::string filepath,
    step_type reqSteps = 0,
    step_type maxSteps = 0,
    size_type maxOrder = 10 ) [inline]
```

similar to the first one, but reads the parameters and p and n from a given file.

#### 7.30.2.3 NonrigorousHelper() [3/3]

similar to the second one, but user gives istream instead of filepath

# 7.30.3 Member Function Documentation

#### 7.30.3.1 integrate() [1/3]

just integrate the solution and return the solution

initial can be either of (1) d-dimension ( $\A^\circ$ d) or (2) the M(d,p,n)-dimension (see papers) In the (1) case it integrates initial function  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  In the (2) case it constructs initial function from data stored in  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  In the (2) case it constructs initial function from data stored in  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  In the (2) case it constructs initial function from data stored in  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  and  $s \in [-tau, 0]$  had is extremely cumbersome. It is best to use output of other functions o get initials in this form.

# 7.30.3.2 integrate() [2/3]

just integrate the solution and return the solution

initial can be either of (1) d-dimension ( $\R^{\circ}$ d) or (2) the M(d,p,n)-dimension (see papers) In the (1) case it integrates initial function  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  In the (2) case it constructs initial function from data stored in  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  In the (2) case it constructs initial function from data stored in  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  In the (2) case it constructs initial function from data stored in  $x_0(s) == initial$  for all  $s \in [-tau, 0]$  and  $s \in [-tau, 0]$  by the initial function from data stored in  $s \in [-tau, 0]$  for all  $s \in [-tau, 0]$  for a

Integrates for time  $T = iters * tau_max / p = iters * h$ . The value h is the step size of the method / grid

In the result the last segment x\_T from the solution in the vector format. Can be used as an input to the next integrate() procedure, to create an initial solution curve, or in other functions from the helper (e.g. poincare).

TODO: use\_extension was used for backward compatibility of some programs it should work with use\_\circ} extension=true in the default version and user should control it with setting m\_maxOrder with getter/setter

#### 7.30.3.3 integrate() [3/3]

integrate, save output to files and draw the solution. See other versions for more information on input and output.

The variable outconfig holds a pair of strings that define paths to where store the results of the computations. See documentation there.

# 7.30.3.4 loadData()

load data only if compatible with setup

# 7.30.3.5 makeSection()

makes a section and stores it in out\_section

if dim(s) == d, then we setup section in the coordinate x(0). s = c otherwise we set full-space section s.  $x_0 = c$ 

#### 7.30.3.6 makeSolver()

```
template<typename EqSpec , int delaysSpec = 1>
Solver capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::makeSolver ( ) [inline]
```

This creates a raw solver, if you want to do nonstandard tasks If you want just to integrate initial values, consider using iterate() or poincare() instead. TODO: add functions to create more elements

# 7.30.3.7 poincare() [1/3]

if you do not know what initV is, then you should probably stick to using the other JacSolution poincare() method (without initV).

#### 7.30.3.8 poincare() [2/3]

compuytes poincare map and the (approximate) solution to the variational equation on the coefficients in V it returns the monodromy matrix, that is  $D_x \vee p(t_P(x), x)$  in DP you get  $D_x \otimes p(t_P(x), x)$  (sic! the difference in  $t_p \otimes p(t_P(x))$  argument) in fPx you get the value of the "vector field" at P(x) this is not straightforward as in ODE that is for ODE you have  $t_p(t_P(x))$ , where f is r.h.s. of the ODE. But for DDE you do not have r.h.s for all of the points in [-tau, 0] But it can be shown, that if  $t_p \otimes p(t_P(x))$  (note  $t_P(x)$ ) (note  $t_P(x)$ ) (note  $t_P(x)$ ) (sic! the difference in  $t_P(x)$ ) But it can be shown, that if  $t_P(x)$  (note  $t_P(x)$ ) (note  $t_P(x)$ ) (note  $t_P(x)$ ) (sic! the difference in  $t_P(x)$ ) (sic! the diff

# 7.30.3.9 poincare() [3/3]

the simples computation of poincare map, without extra data

#### 7.30.3.10 refinePeriodic()

helper function to refine a candidate periodic orbit with a Newton method.

First parameter is to get some text info back during the process, you can pass std::cout there.

Section you need to set-up, it will be s \* x = 0 = x,  $s \in \mathbb{R}^{n}$  {M(d,p,n),  $c \in \mathbb{R}$ }

# 7.30.3.11 setParam()

returns old param value; TODO: add index checking

# 7.30.3.12 setParams()

returns old params

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

# 7.31 capd::ddes::ODEPendulum Class Reference

```
#include <SampleEqns.h>
```

# **Public Types**

typedef long size\_type

#### **Public Member Functions**

- ODEPendulum (ODEPendulum const &orig)
- ODEPendulum & operator= (ODEPendulum const &orig)
- size\_type imageDimension () const
- size\_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

# **Static Public Member Functions**

• static std::string show ()

# 7.31.1 Detailed Description

Mainly for testing how DDE code compares to ODE in CAPD

# 7.31.2 Member Function Documentation

# 7.31.2.1 dimension()

```
size_type capd::ddes::ODEPendulum::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau 1) is another, etc.. (thus the formula)

### 7.31.2.2 imageDimension()

```
size_type capd::ddes::ODEPendulum::imageDimension ( ) const [inline]
```

output dimension of the internal map

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

# 7.32 capd::ddeshelper::PathConfig Class Reference

#include <DDEHelperCommon.h>

#### **Public Member Functions**

- PathConfig (std::string dirPath=".", std::string prefix="")
- PathConfig suffix (std::string suffix) const
- std::string fullpath () const
- std::string filepath (std::string name) const
- std::string filename (std::string name) const
- void mkdir\_p () const
- std::string dirpath () const

# **Public Attributes**

- · std::string dirPath
- · std::string prefix

# 7.32.1 Detailed Description

used in some helpers to define where the output goes. It does not check the correctness of those paths! You need to assure they are correct!

dirPath - a base directory to where the output save prefix - as name says - prefix of all generated files.

# 7.32.2 Constructor & Destructor Documentation

# 7.32.2.1 PathConfig()

make a config path, default is ./ with no prefix

#### 7.32.3 Member Function Documentation

#### 7.32.3.1 filename()

makes just a prefixed filename (no path attached)

# 7.32.3.2 filepath()

makes a new prefixed filepath for a given filename

# 7.32.3.3 fullpath()

```
std::string capd::ddeshelper::PathConfig::fullpath ( ) const [inline]
```

dirpath with prefix

#### 7.32.3.4 mkdir\_p()

```
void capd::ddeshelper::PathConfig::mkdir_p ( ) const [inline]
```

assure there is really this folder - not very safe!!!

# 7.32.3.5 suffix()

make a new PathConfix with a longer prefix : "prefix-suffix"

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

• /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D 

DEHelperCommon.h

# 7.33 capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

# **Public Types**

- typedef ScalarSpec ScalarType
- typedef unsigned int size\_type
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarType, 0 > VectorType

#### **Public Member Functions**

- PerezMaltaCoutinho (ParamType b1)
- PerezMaltaCoutinho (ParamType gamma, ParamType b0, ParamType b1)
- PerezMaltaCoutinho (PerezMaltaCoutinho const &orig)
- PerezMaltaCoutinho (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- PerezMaltaCoutinho & operator= (PerezMaltaCoutinho const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

- static size type imageDimension ()
- static size\_type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

# **Protected Attributes**

- · ParamType gamma
- ParamType b0
- · ParamType b1

# 7.33.1 Detailed Description

```
template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >
```

Perez-Malta-Coutinho equation for chaos as mentioned in scholarpedia article on Mackey-Glass equation.

```
x'(t) = -gamma * x(t) + (b0 - b1 * x(t-tau)) * x(t-tau)
```

# 7.33.2 Member Function Documentation

# 7.33.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::dimension ( ) [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

#### 7.33.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::getParamsCount ( )
[inline], [static]
```

number of parameters to fully configure equation

#### 7.33.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::imageDimension ( )
[inline], [static]
```

output dimension of the internal map

# 7.33.2.4 operator()()

Evaluate r.h.s of the equation.

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

# 7.34 capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind< OtherSolutionSpec, OtherJetSpec > Struct Template Reference

# **Public Types**

• typedef DiscreteDelaysFunctionalMap< FinDimMapSpec, OtherSolutionSpec, OtherJetSpec > other

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

7.35 capd::ddes::BasicDiscreteDelaysFunctionalMap < FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind < OtherSolutionSpec, OtherJetSpec > Struct Template Reference

# **Public Types**

• typedef BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, OtherSolutionSpec, OtherJetSpec > other

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Basic
   DiscreteDelaysFunctionalMap.h
- 7.36 capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec,
  MatrixSpec, isInterval >::rebind< OtherTimePointSpec,
  OtherDataSpec, OtherVectorSpec, OtherMatrixSpec, OtherIsInterval
  > Struct Template Reference

# **Public Types**

 typedef GenericJet< OtherTimePointSpec, OtherDataSpec, OtherVectorSpec, OtherMatrixSpec, OtherIs⇔ Interval > other

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

- 7.37 capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rebind< OtherJetTypeSpec > Struct Template Reference

# **Public Types**

typedef DDEPiecewisePolynomialCurve< GridSpec, OtherJetTypeSpec > other

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

- 7.38 capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec > Class Template Reference

#include <DDEHelperRigorous.h>

#### Classes

class CoordinateSystem

# **Public Types**

- typedef EqSpec Eq
- · typedef Eq::ScalarType Scalar
- typedef Eq::RealType Real
- typedef Eg::ParamType ParamType
- typedef capd::vectalg::Vector < Scalar, 0 > Vector
- typedef capd::vectalg::Matrix< Scalar, 0, 0 > Matrix
- typedef capd::vectalg::Vector< ParamType, 0 > ParamsVector
- typedef PoliciesSpec Policies
- typedef capd::ddes::SharedDoubleton< Matrix, Policies > SetType
- typedef capd::ddes::DDESolutionCurve< SetType > Solution
- typedef Solution::GridType Grid
- typedef Solution::TimePointType TimePoint
- typedef Solution::CurvePieceType CurvePiece
- typedef capd::ddes::DiscreteDelaysFunctionalMap< Eq, Solution > DDEq
- typedef capd::ddes::DDETaylorSolver< DDEq > Solver
- typedef Solver::VariableStorageType Variables
- typedef Solver::JacobianStorageType Jacobians
- typedef Solver::ValueStorageType Values
- typedef Solver::size\_type size\_type
- typedef capd::ddes::DDEJetSection< Solution > Section
- typedef Section::JetType SectionJet
- typedef capd::ddes::DDEPoincareMap< Solver, Section > PoincareMap
- typedef DDECompareHelper< Vector > Comparator

# **Public Member Functions**

- RigorousHelper (std::string filepath, int reqSteps=0, int maxSteps=0, int maxOrder=10)
- RigorousHelper (int p, int n, ParamsVector const &params, int reqSteps=0, int maxSteps=0, int max
   —
   Order=10)
- Solution dataToSolution (CoordinateSystem const &coords, Vector dx, Vector r0, Vector Xi) const
- Solution dataToSolution (Vector dx, Vector r0, Vector Xi) const
- Solution constantInitialSolution (Vector const &vx) const
- Solution constantInitialSolution (Vector const &vx, size\_type order) const
- template<typename AnyMatrixSpec >
- Solution functionToSolution (capd::map::Map < AnyMatrixSpec > f) const
- $\bullet \;\; {\sf template}{<} {\sf typename AnyMatrixSpec} >$ 
  - ${\bf Solution\ function To Solution\ (capd::map::Map{<}\ Any Matrix Spec>f,\ int\ starti)\ const}$
- Solution r0ToSolution (CoordinateSystem const &coords, Vector r0, Vector Xi) const
- Solution r0ToSolution (Vector r0, Vector Xi) const
- capd::poincare::CrossingDirection detectCrossingDirection ()
- Solver makeSolver ()
- Solution timemap (Solution &X, const size\_type &steps, const Real &epsilon=0.)
- Solution timemap (CoordinateSystem const &in\_coords, Vector const &dx, Vector const &r0, Vector const &Xi, const size\_type &steps, const Real &epsilon, CoordinateSystem const &out\_coords, Vector &Tdx, Vector &Tr0, Vector &TXi)
- Solution timemap (Vector const &dx, Vector const &r0, Vector const &Xi, const size\_type &steps, const Real &epsilon, Vector &Tdx, Vector &Tr0, Vector &TXi)

- Solution poincare (Section section, capd::poincare::CrossingDirection crossing\_direction, Solution const &X, Real &reachTime, size\_type &steps, Real &epsilon, Solution &PX)
- Solution **poincare** (Section section, capd::poincare::CrossingDirection crossing\_direction, Solution const &X, Real &reachTime, Solution &PX)
- Solution **poincare** (CoordinateSystem const &in\_coords, Vector const &dx, Vector const &r0, Vector const &Xi, CoordinateSystem const &out\_coords, Real &reachTime, size\_type &steps, Real &epsilon, Vector &Pdx, Vector &Pr0, Vector &PXi)
- void toCoords (Solution const &PX, CoordinateSystem const &coords, Vector &Pdx, Vector &Pr0, Vector &PXi)
- Solution poincare (CoordinateSystem const &in\_coords, Vector const &dx, Vector const &r0, Vector const &Xi, Real &reachTime, Vector &Pdx, Vector &Pr0, Vector &PXi)
- Solution poincare (Vector const &dx, Vector const &r0, Vector const &Xi, Real &reachTime, Vector &Pdx, Vector &Pr0, Vector &PXi)
- void loadSetup (std::string filepath)
- void setRequiredSteps (int regSteps, bool ensure long enough=true)
- int getRequiredSteps () const
- void setMaximumSteps (int maxSteps, bool ensure\_long\_enough=true)
- int getMaximumSteps () const
- void setMaximumOrder (int maxOrder)
- int getMaximumOrder () const
- ParamsVector params () const
- · int M () const
- int p () const
- int n () const
- int d () const
- TimePoint h () const
- · Real H () const
- Real getBasicIntervalLength () const
- TimePoint tau () const
- Grid & grid ()
- · const Grid & grid () const
- capd::poincare::CrossingDirection getCrossingDirection ()
- void **setCrossingDirection** (capd::poincare::CrossingDirection d)
- · CoordinateSystem coords () const
- void setCoords (CoordinateSystem const &c)
- · Vector reference ()
- Vector x0 ()
- Matrix C ()
- Matrix invC ()
- Vector sectionVector ()
- Vector sectionValue ()
- Section section ()
- void dumpData (std::ostream &out, CoordinateSystem const &coords, Vector const &dx, Vector const &r0,
   Vector const &Xi)
- void saveData (PathConfig const &paths, CoordinateSystem const &coords, Vector const &dx, Vector const &r0, Vector const &Xi, std::string extraMsg="")
- CoordinateSystem loadData (std::istream &input, Vector &dx, Vector &r0, Vector &Xi)
- CoordinateSystem loadData (std::string filepath, Vector &dx, Vector &r0, Vector &Xi)
- template<typename SolutionT >
   void drawSolution (std::string dirpath, std::string filename, SolutionT const &X)
- ParamsVector setParams (ParamsVector const &new params)
- ParamType setParam (size\_type index, ParamType const &new\_param)

#### **Static Public Attributes**

- static const int **PARAMS\_COUNT** = EqSpec::getParamsCount() + delaysSpec
- static const int **DIMENSION** = EqSpec::imageDimension()

# 7.38.1 Detailed Description

template < typename EqSpec, int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11Rect2Policies > class capd::ddeshelper::RigorousHelper < EqSpec, delaysSpec, PoliciesSpec >

allows to supply good coordinates and compute in those coordinates (also, to give data in those coordinates) So for example if you have coordinates C such that the are given by eigenvectors of Jacobian of poincare map then you can slice your data by 1 along any vector j by specifying  $dx = \{0,...,0,1,0,...\}$ , where 1 is on j-th place.

The coordinate change P to good coordinates is: B(dx) = reference + C \* dx (so that B(0) = reference) So the backward change is  $B^{-1}(y) = C^{-1} * (y - reference)$  Backward change is used in poincare to produce result in 'good' coordinates That is we compute  $B^{-1}(P(B(.)))$  applied on a set (dx, r0, Xi) And we get (Pdx, Pr0, PXi) for an easy comparison (coord by coord)

#### 7.38.2 Member Function Documentation

# 7.38.2.1 constantInitialSolution() [1/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11 \leftarrow Rect2Policies> Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::constant \leftarrow InitialSolution ( Vector const & vx) const [inline]
```

makes a constant solution over basic delay interval [-tau, 0]

#### 7.38.2.2 constantInitialSolution() [2/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11 \leftarrow Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::constant \leftarrow InitialSolution (

Vector const & vx,

size_type order) const [inline]
```

makes a constant solution over basic delay interval [-tau, 0]

#### 7.38.2.3 dataToSolution() [1/2]

Makes a set of the form (reference + C \* dx) + C \* r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

#### 7.38.2.4 dataToSolution() [2/2]

Makes a set of the form (reference + C \* dx) + C \* r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

## 7.38.2.5 dumpData()

this is to dump data as text format (vectors as text representation) Not the best to use in the proofs, but can be of use in preparation of data.

## 7.38.2.6 functionToSolution() [1/2]

makes a solution based on the function values over basic delay interval [-tau, 0] The function must be capd::map $\leftrightarrow$ ::Map type, IMAP works great. Function must be R -> R^dimension, otherwise - exception!

#### 7.38.2.7 functionToSolution() [2/2]

makes a solution based on the function values over basic delay interval [-tau, 0] The function must be capd::map  $\leftrightarrow$  ::Map type, IMAP works great. Function must be R -> R^dimension, otherwise - exception!

#### 7.38.2.8 poincare() [1/3]

out\_coords are given by the coords of this Helper

## 7.38.2.9 poincare() [2/3]

TODO: (IMPORTANT!!! IN THIS CASE!) docs,

## 7.38.2.10 poincare() [3/3]

in\_ and out\_ coords are given by the coords of this Helper

#### 7.38.2.11 r0ToSolution() [1/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11\leftarrow Rect2Policies> Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::r0ToSolution ( CoordinateSystem const & coords, Vector r0, Vector Xi ) const [inline]
```

Makes a set of the form reference + C \* r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

## 7.38.2.12 r0ToSolution() [2/2]

Makes a set of the form reference + C \* r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

## 7.38.2.13 setParam()

returns old param value; TODO: add index checking

#### 7.38.2.14 setParams()

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11←
Rect2Policies>
ParamsVector capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::setParams (
ParamsVector const & new_params) [inline]
```

returns old params

#### 7.38.2.15 timemap()

computes timemap using the solution as initial data WARNING: it will extend the solution given in X by a given number of steps! returns: it returns the last segment of the solution in a shape of the same representation as the initial segment of X. if the segment cannot be produced (due to continuity class issues) the exception will be thrown. if epsilon is given as != 0. then the procedure will try to do the epsilonStep procedure.

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

## 7.39 capd::ddes::RosslerDelay< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

## **Public Types**

- typedef ScalarSpec ScalarType
- · typedef unsigned int size type
- typedef ScalarType RealType
- typedef ParamSpec ParamType

## **Public Member Functions**

- RosslerDelay (ParamType a, ParamType b, ParamType c)
- RosslerDelay (RosslerDelay const &orig)
- template<typename VecSpec >

RosslerDelay (VecSpec const &params)

- RosslerDelay & operator= (RosslerDelay const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

- static size\_type imageDimension ()
- static size\_type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

#### **Protected Attributes**

- ParamType a
- ParamType b
- ParamType c
- ParamType epsi

## 7.39.1 Detailed Description

```
template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >
```

A Toy Example for proof of chaotic behaviour in DDE (chaotic ODE perturbed by a small DDE term - delay might be big, but perturbation should be small) The equation is:

```
v'(t) = f(v(t)) + epsilon f(v(t-tau))
```

where  $v = (x, y, z) \ln R^3$  and f is the r.h.s. of original R"ossler ODE:

```
f(x, y, z) = (-(y+z), (x + ay), (b + z * (x - c)))
```

The parameters are: a, b, c (as in R"ossler) and \epsilon.

## 7.39.2 Member Function Documentation

#### 7.39.2.1 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::getParamsCount ( ) [inline],
[static]
```

number of parameters to fully configure equation

## 7.39.2.2 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::imageDimension ( ) [inline],
[static]
```

output dimension of the internal map

## 7.39.3 Member Data Documentation

#### 7.39.3.1 a

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::a [protected]
```

classic rossler parameters, see scholarpedia.

#### 7.39.3.2 b

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::b [protected]
```

classic rossler parameters, see scholarpedia.

#### 7.39.3.3 c

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::c [protected]
```

classic rossler parameters, see scholarpedia.

## 7.39.3.4 epsi

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::epsi [protected]
```

extra parameter for a size of the delayed term

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Egns.h

## 7.40 capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

## **Public Types**

- typedef ScalarSpec ScalarType
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef int size\_type

#### **Public Member Functions**

- ScalarLinearMap (ScalarType const &a, ScalarType const &b, ScalarType const &c)
- ScalarLinearMap (ScalarLinearMap const &orig)
- ScalarLinearMap & operator= (ScalarLinearMap const & orig)
- size\_type imageDimension () const
- size\_type dimension () const
- template < typename RealSpec , typename InVectorSpec , typename OutVectorSpec > void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

• static std::string show ()

#### **Protected Attributes**

- · ParamType a
- · ParamType b
- · ParamType c

## 7.40.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec> class capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >

Scalar linear equation of the form:

```
x'(t) = a * x(t) + b * x(t-tau) + c
```

Use with 'DiscreteDelaysFunctionalMap' to define tau

#### 7.40.2 Member Function Documentation

## 7.40.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
size_type capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

## 7.40.2.2 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
size_type capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >::imageDimension ( ) const
[inline]
```

output dimension of the internal map

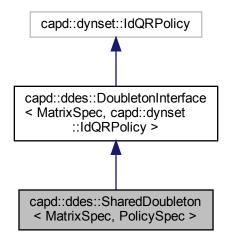
The documentation for this class was generated from the following file:

• /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

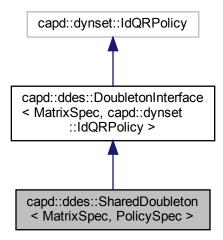
# 7.41 capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec > Class Template Reference

#include <SharedDoubleton.h>

Inheritance diagram for capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >:



Collaboration diagram for capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >:



## **Public Types**

- typedef MatrixSpec MatrixType
- typedef MatrixType::RowVectorType VectorType
- typedef MatrixType::ScalarType
- typedef MatrixType::size\_type size\_type
- typedef SharedDoubleton Class
- typedef DoubletonInterface< MatrixSpec, PolicySpec > BaseClass
- typedef VectorType \* VectorTypePtr
- typedef MatrixType \* MatrixTypePtr
- typedef std::bitset< 6 > OwnershipType
- · typedef PolicySpec Policy
- typedef PolicySpec QRPolicy

## **Public Member Functions**

- Class & operator= (Class const &orig)
- SharedDoubleton ()
- SharedDoubleton (VectorType const &x, VectorType \*set external r0=0, bool passOwnership=false)
- SharedDoubleton (Class const &orig)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, VectorType const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, MatrixType const &Binv, VectorType const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType \*r0, MatrixType const &B, Vector
   —
   Type const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType \*r0, MatrixType const &B, Matrix
   —
   Type const &Binv, VectorType const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType \*r0)

- SharedDoubleton (VectorType \*x, MatrixType \*C, VectorType \*r0, MatrixType \*B, VectorType \*r)
- SharedDoubleton (size type d, size type N0=-1)
- virtual ∼SharedDoubleton ()
- size\_type storageN0 () const
  - < import dimension from interface
- size\_type storageDimension () const
- VectorType \* take x ()
- MatrixType \* take\_C ()
- VectorType \* take r0 ()
- MatrixType \* take B ()
- VectorType \* take\_r ()
- MatrixType \* take\_Binv ()
- VectorType get\_x () const
- MatrixType get\_C () const
- VectorType get r0 () const
- MatrixType get\_B () const
- VectorType get\_r () const
- MatrixType get\_Binv () const
- BaseClass & set\_x (VectorType const &x)
- BaseClass & set C (MatrixType const &C)
- BaseClass & set\_r0 (VectorType const &r0)
- BaseClass & set\_B (MatrixType const &B)
- BaseClass & set\_r (VectorType const &r)
- BaseClass & set Cr0 (MatrixType const &C, VectorType const &r0)
- BaseClass & set Binv (MatrixType const &Binv)
- VectorType midPoint () const
- VectorType hull () const
- BaseClass & set\_x (VectorType \*x, bool passOwnership=false)
- BaseClass & set C (MatrixType \*C, bool passOwnership=false)
- BaseClass & set r0 (VectorType \*r0, bool passOwnership=false)
- BaseClass & set\_Cr0 (MatrixType \*C, VectorType \*r0, bool passCOwnership=false, bool passR0

   Ownership=false)
- BaseClass & set\_B (MatrixType \*B, bool passOwnership=false)
- BaseClass & set\_r (VectorType \*r, bool passOwnership=false)
- BaseClass & set\_Binv (MatrixType \*Binv, bool passOwnership=false)
- bool common\_x (VectorType const \*x) const
- bool common\_C (MatrixType const \*C) const
- bool common\_r0 (VectorType const \*r0) const
- bool common\_B (MatrixType const \*B) const
- bool common\_r (VectorType const \*r) const
- bool common\_Binv (MatrixType const \*Binv) const
- BaseClass & add (VectorType const &v)
- BaseClass & add (BaseClass const &set)
- BaseClass & mul (ScalarType const &c)
- BaseClass & mulThenAdd (ScalarType const &c, BaseClass const &set)
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- std::string show () const
- virtual void reinitialize (size\_type d, size\_type N0)
- void reinit (const VectorType &x, const MatrixType &C, const VectorType &r0, const MatrixType &B, const VectorType &r)
- void **reinit** (const VectorType \*x, const MatrixType \*C, const VectorType \*r0, const MatrixType \*B, const VectorType \*r)
- virtual size\_type dimension () const

## **Protected Types**

```
    enum {
    OWNERBIT_x = 5, OWNERBIT_C = 4, OWNERBIT_r0 = 3, OWNERBIT_B = 2,
    OWNERBIT r = 1, OWNERBIT Binv = 0 }
```

## **Protected Member Functions**

- void updateBinv ()
  - bitset "I own nothing" used for binary operations
- void rawSetup (size\_type dim, size\_type N0, const VectorType \*x, const MatrixType \*C, const VectorType \*r0, const MatrixType \*B, const VectorType \*r, const MatrixType \*Binv, OwnershipType ownership)
- void assureOwner (OwnershipType const &what=OWN\_ALL)
- void sanityCheck (std::string const &what="") const
- void deallocate (OwnershipType const &what=OWN\_ALL)
- void allocate (size\_type d, size\_type N0=0, OwnershipType const &what=OWN\_ALL)
- void reallocate (size\_type d, size\_type N0=0, OwnershipType const &what=OWN\_ALL)

## **Protected Attributes**

- VectorTypePtr m\_x
- MatrixTypePtr m C
- VectorTypePtr m r0
- MatrixTypePtr m B
- VectorTypePtr m\_r
- MatrixTypePtr m Binv
- OwnershipType m\_owner

#### **Static Protected Attributes**

```
    static OwnershipType OWN_x
```

static OwnershipType OWN\_C

bitset "I own only x" - used for binary operations

• static OwnershipType OWN r0

similarly to above - used for binary operations

static OwnershipType OWN B

similarly to above - used for binary operations

• static OwnershipType OWN\_r

similarly to above - used for binary operations

• static OwnershipType OWN\_Binv

similarly to above - used for binary operations

static OwnershipType OWN\_ALL

similarly to above - used for binary operations

static OwnershipType OWN\_NONE

bitset "I own everything" - used for binary operations

#### **Additional Inherited Members**

## 7.41.1 Detailed Description

```
template<typename MatrixSpec, typename PolicySpec = capd::dynset::ldQRPolicy>class capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >
```

This class stores a set  $X \in \mathbb{R}^d$  of the form:

```
X = x + C * r0 + B * r
```

with  $x \in \mathbb{R}^d$ , \$r0, r\$ - closed intervals centered at 0, \$r0 \subset \R^{\N\_0}\$, \$C \in Lin(\R^{\N\_0}, \R^d)\$, \$B \in Lin(\R^d, \R^d)\$, \$r \in \R^d\$, \$B\$ being a matrix easy to invert, e.g. B = ID (Interval form of the remainder) or \$B\$ orthogonal (Doubleton set with QR decomposition).

NOTE: compare to the classic Lohner Doubleton Set in CAPD, where authors assume  $N_0 = d$  NOTE: the set allows for its components to be set outside of the object and controlled there. We use pointers for this purpose. However, we always work with the set to assure that no pointer is NULL. If user does not supply us with a pointer, then we are creating a default one instead. We use the fact that CAPD can have Vectors of dimension 0 and Matrices of dimension (d, 0) and (0, d) and algebra is well defined for them.

NOTE: This is quite long and tedious code in C++ because of C++ This should be heavily tested for correctedness and memory leaks Reader (e.g. reviewer of the manuscript for publication) should not worry to check that code

TODO: (NOT URGENT) move big implementation into .hpp part.

## 7.41.2 Constructor & Destructor Documentation

#### 7.41.2.1 SharedDoubleton() [1/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton ( ) [inline]
```

default constructor makes a 1D point-set at 0

## 7.41.2.2 SharedDoubleton() [2/11]

setup set with a given vector, if it is an interval vector, then it be split into midPoint and the error part, which goes to B\*r part (error part). If you want some structure (i.e. Lohner part C\*r0) then use appropriate constructor.

#### 7.41.2.3 SharedDoubleton() [3/11]

copies the original set (ownership is preserved)

#### 7.41.2.4 SharedDoubleton() [4/11]

setup this set with a given data, set owns everything

#### 7.41.2.5 SharedDoubleton() [5/11]

setup this set with a given data, set owns everything

#### 7.41.2.6 SharedDoubleton() [6/11]

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

## 7.41.2.7 SharedDoubleton() [7/11]

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

#### 7.41.2.8 SharedDoubleton() [8/11]

setup this set with a given data, set is owner of everything

## 7.41.2.9 SharedDoubleton() [9/11]

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

#### 7.41.2.10 SharedDoubleton() [10/11]

setup this set with a given data but the set does not own the data (user is responsible for deleting)

#### 7.41.2.11 SharedDoubleton() [11/11]

setup this set as zero vector, but the structure of given dimensions. If second arg is < 0 then d is used instead.

#### 7.41.2.12 ~SharedDoubleton()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::~SharedDoubleton ( ) [inline],
[virtual]
```

standard thing

#### 7.41.3 Member Function Documentation

#### 7.41.3.1 add() [1/2]

reimplemented for better perfoprmance

## 7.41.3.2 add() [2/2]

reimplemented for better perfoprmance

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dynset::IdQRPolicy >.

## 7.41.3.3 affineTransform()

applies in a smart way to this set X the affine transform f(y) = M \* (X - v)

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.4 allocate()

allocate specified elements of the object (default: all) so it is d-dimensional, and has N0 dimensional r0 part.

#### 7.41.3.5 assureOwner()

does nothing if is owner of others, otherwise it reasigns the memory and copies values form the external data

#### 7.41.3.6 common B()

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.7 common\_Binv()

checked by pointer equality

## 7.41.3.8 common\_C()

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.41.3.9 common r()

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.10 common\_r0()

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.11 common\_x()

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dqnset::IdQRPolicy >.

## 7.41.3.12 deallocate()

safely removes all specified data (default: all) from the object, checking the ownership if neccesary.

#### 7.41.3.13 dimension()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dimension [inline]
```

by default dimension is equal to storage dimension

#### 7.41.3.14 get B()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_B ( ) const [inline]
```

see interface docs

## 7.41.3.15 get\_Binv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_Binv ( ) const [inline]
```

see interface docs

#### 7.41.3.16 get\_C()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_C ( ) const [inline]
```

see interface docs

## 7.41.3.17 get\_r()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_r ( ) const [inline]
```

see interface docs

#### 7.41.3.18 get r0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_r0 ( ) const [inline]
```

see interface docs

#### 7.41.3.19 get x()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_x ( ) const [inline]
```

see interface docs

#### 7.41.3.20 mul()

reimplemented for better performance

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.21 mulThenAdd()

reimplemented for better perfoprmance

#### 7.41.3.22 operator=()

assign operator - it needs to deallocate memory if necessary, then setup set anew

## 7.41.3.23 rawSetup()

Low level setup function to be called in constructors for DRY. This does not deallocate memory! For safer version see public setup().

## 7.41.3.24 reallocate()

dealocates and allocates new memory for specified elements (default: all)

#### 7.41.3.25 reinitialize()

see base class

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.26 sanityCheck()

helper function to check if the object represents sane data (all dimensions compatible and all pointers set if neccesary) should be called after constructing or manipulating object

#### 7.41.3.27 set B() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dynset::IdQRPolicy >.

## 7.41.3.28 set\_B() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.41.3.29 set Binv() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.30 set\_Binv() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.31 set C() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.41.3.32 set\_C() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

 $Implements\ capd:: ddes:: Doubleton Interface < Matrix Spec,\ capd:: dynset:: ldQRPolicy >.$ 

#### 7.41.3.33 set\_Cr0() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dqnset::IdQRPolicy >.

## 7.41.3.34 set\_Cr0() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.35 set\_r() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.41.3.36 set\_r() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.37 set\_r0() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

## 7.41.3.38 set\_r0() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.41.3.39 set\_x() [1/2]

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.40 set\_x() [2/2]

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.41 storageDimension()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::storageDimension ( ) const
[inline]
```

see interface description

#### 7.41.3.42 storageN0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::storageN0 () const [inline]
```

< import dimension from interface

see interface description

#### 7.41.3.43 take B()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_B ( ) [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dqnset::IdQRPolicy >.

#### 7.41.3.44 take Binv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_Binv ( ) [inline],
[virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

#### 7.41.3.45 take C()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_C ( ) [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

 $Reimplemented \ from \ capd:: does:: Doubleton Interface < \ Matrix Spec, \ capd:: dynset:: IdQRPolicy >.$ 

## 7.41.3.46 take\_r()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_r ( ) [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.47 take\_r0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_r0 ( ) [inline],
[virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface < MatrixSpec, capd::dqnset::IdQRPolicy >.

#### 7.41.3.48 take x()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_x ( ) [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

#### 7.41.3.49 translate()

applies in a smart way to this set X the transform f(y) = X - v

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::ldQRPolicy >.

## 7.41.3.50 updateBinv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::updateBinv ( ) [inline], [protected]
```

bitset "I own nothing" - used for binary operations

assures m\_Binv holds real inverse of B

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

• /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Shared 

Doubleton.h

## 7.42 capd::ddes::DiscreteTimeGrid< RealSpec >::TimePointType Class Reference

## **Public Types**

typedef RealSpec RealType

#### **Public Member Functions**

- operator int () const
- · int tolnt () const
- · operator RealType () const
- · bool isZero () const
- std::string show ()
- TimePointType operator+= (TimePointType const &b)
- TimePointType operator-= (TimePointType const &b)
- TimePointType (TimePointType const &orig)
- TimePointType & operator= (TimePointType const &orig)

#### **Static Public Member Functions**

static std::string badge ()

#### **Protected Member Functions**

- void checkGridCompatible (TimePointType const &other, std::string extraInfo="") const
- TimePointType (RealType const &h, int i)

## **Protected Attributes**

- const RealSpec & m\_h
- int **m\_i**

## **Friends**

- class DiscreteTimeGrid < RealType >
- TimePointType operator+ (TimePointType const &a, TimePointType const &b)
- TimePointType operator- (TimePointType const &a, TimePointType const &b)
- TimePointType operator- (const TimePointType &a)
- TimePointType operator\* (RealType const &a, TimePointType const &b)
- TimePointType operator\* (TimePointType const &a, RealType const &b)
- bool operator== (TimePointType const &a, TimePointType const &b)
- bool operator< (TimePointType const &a, TimePointType const &b)</li>
- bool operator> (TimePointType const &a, TimePointType const &b)
- bool operator<= (TimePointType const &a, TimePointType const &b)</li>
- bool **operator**>= (TimePointType const &a, TimePointType const &b)
- std::ostream & operator<< (std::ostream &out, TimePointType const &t)
- std::istream & operator>> (std::istream &in, TimePointType &t)

## 7.42.1 Member Function Documentation

## 7.42.1.1 badge()

```
template<typename RealSpec >
static std::string capd::ddes::DiscreteTimeGrid< RealSpec >::TimePointType::badge ( ) [inline],
[static]
```

Badge must be a single word!

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

## 7.43 capd::ddes::ToyModel Class Reference

```
#include <SampleEqns.h>
```

## **Public Types**

typedef long size\_type

## **Public Member Functions**

- ToyModel (ToyModel const &orig)
- ToyModel & operator= (ToyModel const &orig)
- size\_type imageDimension () const
- size\_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec > void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

• static std::string show ()

## 7.43.1 Detailed Description

```
Scalar eq. x(t) * x(t-tau).
```

Use with 'DiscreteDelaysFunctionalMap' to define tau

#### 7.43.2 Member Function Documentation

#### 7.43.2.1 dimension()

```
size_type capd::ddes::ToyModel::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

#### 7.43.2.2 imageDimension()

```
size_type capd::ddes::ToyModel::imageDimension ( ) const [inline]
```

output dimension of the internal map

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Egns.h

## 7.44 capd::ddes::ToyModelSq Class Reference

```
#include <SampleEqns.h>
```

## **Public Types**

• typedef long size\_type

## **Public Member Functions**

- ToyModelSq (ToyModelSq const &orig)
- ToyModelSq & operator= (ToyModelSq const &orig)
- size\_type imageDimension () const
- size\_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec > void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

## **Static Public Member Functions**

• static std::string show ()

## 7.44.1 Detailed Description

Scalar eq. Depends only on the past,  $f(x(t), x(t-1)) = 2*x(t-1) + (x(t-1))^2$ . Used for tests.

Use with 'DiscreteDelaysFunctionalMap' to define tau

#### 7.44.2 Member Function Documentation

## 7.44.2.1 dimension()

```
size_type capd::ddes::ToyModelSq::dimension ( ) const [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

#### 7.44.2.2 imageDimension()

```
size_type capd::ddes::ToyModelSq::imageDimension ( ) const [inline]
```

output dimension of the internal map

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

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Eqns.h

## 7.45 capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

## **Public Types**

- · typedef unsigned int size\_type
- typedef ParamSpec ParamType
- typedef ScalarSpec ScalarType
- typedef ScalarType RealType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarSpec, 0 > VectorType

#### **Public Member Functions**

- ToyModelSqA (ParamType a)
- ToyModelSqA (ToyModelSqA const &orig)
- ToyModelSqA (ParamsVectorType const &params)
- ToyModelSqA & operator= (ToyModelSqA const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

- static size\_type imageDimension ()
- static size\_type dimension ()
- static size\_type getParamsCount ()
- static std::string show ()

## **Public Attributes**

- · ParamSpec a
- · ParamSpec c

## 7.45.1 Detailed Description

```
template<typename ScalarSpec, typename ParamSpec> class capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >
```

Scalar eq. Depends only on the past,  $f(x(t), x(t-1)) = -a*x(t-1) + c * x(t-1) + (x(t-1))^2$ . For c = 0 it has Hopf biffurcation at  $a^* * = (5*pi) / (3*sqrt(3))$  It has 2 fixed points: 0 and a, a is repelling with one unstable to infty and to 0, 0 is stable Param c is used to move the fixed point a to 0, set c = 2a Use with 'DiscreteDelaysFunctionalMap' to define tau. Tau should be 1

#### 7.45.2 Member Function Documentation

## 7.45.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec >
static size_type capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >::dimension ( ) [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau\_1) is another, etc.. (thus the formula)

## 7.45.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec >
static size_type capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >::getParamsCount ( ) [inline],
[static]
```

number of parameters to fully configure equation

## 7.45.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec >
static size_type capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >::imageDimension ( ) [inline],
[static]
```

output dimension of the internal map

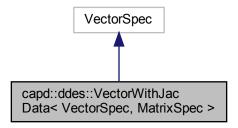
The documentation for this class was generated from the following file:

/home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample ← Egns.h

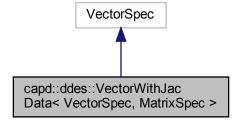
## 7.46 capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec > Class Template Reference

#include <DDENonrigorousTaylorSolver.h>

Inheritance diagram for capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec >:



Collaboration diagram for capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec >:



## **Public Types**

- typedef MatrixSpec MatrixType
- typedef MatrixSpec::ScalarType ScalarType
- typedef VectorWithJacData< VectorSpec, MatrixSpec > Class
- typedef VectorSpec BaseClass
- typedef BaseClass::size\_type size\_type
- typedef std::vector< MatrixSpec > MatrixStorageType

#### **Public Member Functions**

- VectorWithJacData (size type d=0)
- VectorWithJacData (const VectorWithJacData &orig)
- VectorWithJacData (const VectorSpec &orig)
- VectorWithJacData (const VectorSpec &v, const std::vector< MatrixSpec > &D)
- VectorWithJacData (const VectorSpec &v, const MatrixSpec &D)
- VectorWithJacData & setMatrix (MatrixType const &D)
- VectorWithJacData & operator= (const VectorWithJacData & orig)
- VectorWithJacData & operator= (const BaseClass &orig)
- VectorWithJacData & operator\*= (ScalarType const &c)
- VectorWithJacData & operator+= (VectorWithJacData const & other)
- operator VectorSpec ()
- operator MatrixSpec ()
- const MatrixStorageType & getMatrixData () const
- MatrixStorageType & getMatrixData ()

## **Protected Attributes**

std::vector< MatrixSpec > m\_Jac

## 7.46.1 Detailed Description

template<typename VectorSpec, typename MatrixSpec>class capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec>

a class to hold additional data to compute Jacobian of the (nonrigorous / approximate) flow w.r.t. initial data.

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

## 7.47 capd::ddes::Wischert< ScalarSpec, ParamSpec > Class Template Reference

#include <SampleEqns.h>

## **Public Types**

- typedef ScalarSpec ScalarType
- typedef unsigned int size\_type
- typedef ScalarType RealType
- typedef ParamSpec ParamType
- typedef capd::vectalg::Vector< ParamSpec, 0 > ParamsVectorType
- typedef capd::vectalg::Vector< ScalarType, 0 > VectorType

## **Public Member Functions**

- Wischert (ParamType a=2.)
- Wischert (Wischert const &orig)
- Wischert (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- Wischert & operator= (Wischert const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
   void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

#### **Static Public Member Functions**

- static size\_type imageDimension ()
- static size\_type dimension ()
- static size type getParamsCount ()
- static std::string show ()

## **Protected Attributes**

ParamType a

## 7.47.1 Detailed Description

```
\label{eq:continuous} $$ template < typename ScalarSpec, typename ParamSpec = ScalarSpec > $$ class capd::ddes::Wischert < ScalarSpec, ParamSpec > $$ x'(t) = a*sin(x(t-1))$
```

## 7.47.2 Member Function Documentation

#### 7.47.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::Wischert< ScalarSpec, ParamSpec >::dimension ( ) [inline], [static]
```

input dimension of the internal map, x(t) is one,  $x(t-tau_1)$  is another, etc.. (thus the formula)

## 7.47.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::Wischert < ScalarSpec, ParamSpec >::getParamsCount ( ) [inline],
[static]
```

number of parameters to fully configure equation

## 7.47.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::Wischert< ScalarSpec, ParamSpec >::imageDimension ( ) [inline],
[static]
```

output dimension of the internal map

## 7.47.2.4 operator()()

evaluation of rhs

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

 $\bullet \ / home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample \leftarrow Eqns.h \\$ 

## Index

~BasicDiscreteDelaysFunctionalMap	capd::ddes::DDESolutionCurve< SetSpec >, 117	
capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet-	addPiece capd::ddes::DDEPiecewisePolynomialCurve<	
Spec >, 17	GridSpec, JetSpec >, 89	
$\sim$ BasicDoubleton	affineTransform	
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >, 28	capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >, 29	
~DDEBasicFunctionalMap	capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >, 63	
capd::ddes::DDEBasicFunctionalMap< Solution-		
CurveSpec, JetSpec >, 45	capd::ddes::DDEPiecewisePolynomialCurve<	
~DDEForwardTaylorCurvePiece	GridSpec, JetSpec >, 89	
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DDESolutionCurve< SetSpec >, 117	
TimePointSpec, SetSpec, isInterval >, 63	capd::ddes::DoubletonInterface< MatrixSpec, Poli-	
$\sim$ DDEPiecewisePolynomialCurve	ciesSpec >, 144	
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::SharedDoubleton< MatrixSpec, Poli-	
GridSpec, JetSpec >, 88	cySpec >, 209	
$\sim$ DDERigorousFunctionalMap	allocate	
capd::ddes::DDERigorousFunctionalMap< Solu-	capd::ddes::DDEForwardTaylorCurvePiece<	
tionCurveSpec, JetSpec >, 105	TimePointSpec, SetSpec, isInterval >, 63	
$\sim$ DiscreteDelaysFunctionalMap	capd::ddes::SharedDoubleton< MatrixSpec, Poli-	
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	cySpec >, 209	
imMapSpec, SolutionCurveSpec, JetSpec >,	allocateCoeffs	
136	capd::ddes::GenericJet< TimePointSpec, DataSpec	
$\sim$ DoubletonInterface	VectorSpec, MatrixSpec, isInterval >, 163	
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	allocateJet	
ciesSpec >, 143	capd::ddes::DDEForwardTaylorCurvePiece<	
~GenericJet	TimePointSpec, SetSpec, isInterval >, 63	
capd::ddes::GenericJet< TimePointSpec, DataSpec,		
VectorSpec, MatrixSpec, isInterval >, 163	capd::ddes::DDEForwardTaylorCurvePiece<	
~SharedDoubleton	TimePointSpec, SetSpec, isInterval >, 64	
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	allocateXi	
cySpec >, 208	capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >, 64	
a	assureOwner	
capd::ddes::RosslerDelay< ScalarSpec, Param-	capd::ddes::SharedDoubleton< MatrixSpec, Poli-	
Spec >, 199	cySpec >, 209	
add	at	
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::DDEPiecewisePolynomialCurve<	
ciesSpec >, 29	GridSpec, JetSpec >, 89	
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::GenericJet< TimePointSpec, DataSpec,	
GridSpec, JetSpec >, 88	VectorSpec, MatrixSpec, isInterval >, 163,	
capd::ddes::DDESolutionCurve< SetSpec >, 116,	164	
117		
1 /	b	
ciesSpec >, 144	capd::ddes::RosslerDelay< ScalarSpec, Param-	
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	Spec >, 200	
cySpec >, 209	back	
addPastPiece	capd::ddes::GenericJet< TimePointSpec, DataSpec,	
capd::ddes::DDEPiecewisePolynomialCurve<	VectorSpec, MatrixSpec, isInterval >, 164	
GridSpec, JetSpec >, 88, 89	backJet	

230 INDEX

capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >, 64	BasicDoubleton, 26–28 common_B, 29	
badge	common_C, 29	
capd::ddes::DDEForwardTaylorCurvePiece<	common_r, 30	
TimePointSpec, SetSpec, isInterval >, 64	common_r0, 30	
capd::ddes::DDEPiecewisePolynomialCurve<	common_x, 30	
GridSpec, JetSpec >, 89	dimension, 30	
capd::ddes::DiscreteTimeGrid< RealSpec >, 140	get_B, 31	
capd:: ddes:: Discrete Time Grid < Real Spec > :: Time Poul Foundation (Control of the Control	intTy <b>ge</b> t_C, <mark>31</mark>	
220	get_r, <mark>31</mark>	
capd::ddes::GenericJet< TimePointSpec, DataSpec,	get_r0, <mark>31</mark>	
VectorSpec, MatrixSpec, isInterval >, 164	get_x, <mark>31</mark>	
BasicDoubleton	hull, 31	
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	midPoint, 31	
ciesSpec >, 26–28	mul, 32	
begin	mulThenAdd, 32	
capd::ddes::BasicDiscreteDelaysFunctionalMap<	operator=, 32	
FinDimMapSpec, SolutionCurveSpec, Jet-	reinitialize, 32	
Spec >, 18	sanityCheck, 32	
capd::ddes::DDEPiecewisePolynomialCurve<	set_B, 32	
GridSpec, JetSpec >, 90	set_C, 33	
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	set_Cr0, 33	
imMapSpec, SolutionCurveSpec, JetSpec >,	set_r, 33	
136	set_r0, 33	
capd::ddes::GenericJet< TimePointSpec, DataSpec,	set_x, 34	
VectorSpec, MatrixSpec, isInterval >, 164,	setupDimension, 34	
165	setupFromData, 34	
beginJet	setupFromOther, 35	
capd::ddes::DDEForwardTaylorCurvePiece<	show, 35	
TimePointSpec, SetSpec, isInterval >, 65	storageDimension, 35	
	storageN0, 35	
C	translate, 35	
capd::ddes::RosslerDelay< ScalarSpec, Param-	capd::ddes::CubicIkeda< ScalarSpec,	ParamSpec >.
Opec >, 200	41	. a.aepec > ,
capd, 11	dimension, 42	
capd::ddes::BasicDiscreteDelaysFunctionalMap< FinD-	getParamsCount, 42	
imMapSpec, SolutionCurveSpec, JetSpec >,	imageDimension, 42	
14	capd::ddes::DDEBasicFunctionalMap<	Solution-
Basis Biolisto Bolayer another landap, 17	CurveSpec, JetSpec >, 42	Coldion
begin, 18	~DDEBasicFunctionalMap, 45	
collectComputationData, 18	checkCurveDimension, 45	
computeDDECoefficients, 18–20	collectComputationData, 45	
const_iterator, 17	computeDDECoefficients, 46–48	
delaysCount, 21	convert, 48	
DelayStorageType, 17	deconvert, 48	
dimension, 21		
end, 21	imageDimension, 49	
imageDimension, 22	JacobianStorageType, 44	
iterator, 17	operator(), 49	
operator(), 22	ValueStorageType, 44	
capd::ddes::BasicDiscreteDelaysFunctionalMap< FinD-	VariableStorageType, 44	D. va C. va C. a. a.
	capd::ddes::DDEBasicPoincareMap<	DynSysSpec,
>::rebind< OtherSolutionSpec, OtherJetSpec	SectionSpec >, 50	
>, 191	detectCrossingDirection, 51	
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec	findCrossingTime, 51	
>, 22	integrateUntilSectionCrossing, 52	
~BasicDoubleton, 28	operator(), 52, 53	
add, 29	setCurrentV, 53	
affineTransform, 29	setInitiaIV, 54	

capd::ddes::DDEForwardTaylorCurvePiece<	Time-	show, 77	
PointSpec, SetSpec, isInterval >, 56		storageDimension, 77	
$\sim$ DDEForwardTaylorCurvePiece, 63		storageN0, 77	
affineTransform, 63		summa, 77	
allocate, 63		summaAtDelta, 78	
allocateJet, 63		t0, 78	
allocateR0, 64		take_r0, 78	
allocateXi, 64		take_Xi, 78	
backJet, 64		taylor, 78, 79	
badge, 64		taylorAtDelta, 79	
beginJet, 65		translate, 79	
DDEForwardTaylorCurvePiece, 60–62		updateCommonR0, 79	
deallocate, 65		capd::ddes::DDEJetSection< CurveSpec, isInte	erval >,
deallocateJet, 65		80	
deallocateR0, 65		DDEJetSection, 81	
deallocateXi, 65		getGradient, 81	
dimCheck, 66		capd::ddes::DDENonrigorousTaylorSolver<	Func-
dimCheckB, 66		tionalMapSpec >, 82	
dimCheckC, 66		oneStep, 83	
dimension, 66		operator(), 83, 84	
dot, 66		capd::ddes::DDEPiecewisePolynomialCurve<	Grid-
dt, 67		Spec, JetSpec >, 84	
endJet, 67		$\sim$ DDEPiecewisePolynomialCurve, 88	
eval, 67		add, 88	
evalAtDelta, 68		addPastPiece, 88, 89	
evalCoeff, 68		addPiece, 89	
evalCoeffAtDelta, 68, 69		affineTransform, 89	
get_B, 69		at, 89	
get_C, 69		badge, 89	
get_r, 69		begin, 90	
get_r0, 69		clear, 90	
get_x, 70		copyPieces, 90	
get_Xi, 70		currentTime, 90	
getT0, 70		DDEPiecewisePolynomialCurve, 87, 88	
isMidCurve, 70		deallocatePieces, 90	
jetAt, 70		dimension, 91	
makeStorage_x, 71		dot, 91	
midCurve, 71		end, 91	
mul, 71		epsilonShift, 91, 92	
operator=, 71		eval, 92	
operator[], 71, 72		extend, 92	
order, 72		get_x, 92	
reallocate, 72		getCurrentTime, 93	
reallocateJet, 72		getGrid, 93	
reallocateR0, 72		getPiece, 93	
reallocateXi, 73		getStep, 93	
reinitialize, 73		getT0, 93	
set_B, 73		getValueAtCurrent, 93, 94	
set_Binv, 73		grid, 94	
set_C, 73		j, 94	
set_Cr0, 74		jet, 95	
set_r, 74		jetCommonMaximalOrder, 95	
set_r0, 74		jetOrderAt, 95	
set_x, 74		jetPtr, 95	
set_Xi, 75		length, 96	
setAsConstant, 75, 76		mul, 96	
setT0, 76		mulThenAdd, 96	
setupFromData, 76		operator<<, 100	
•		•	

operator>>, 100	length, 121
operator*=, 96	m_r0, 130
operator=, 96	makeStorage_x, 122
pastTime, 96	move, 122
pointToIndex, 97	mul, 122
rbegin, 97	mulThenAdd, 122
readFrom, 97	operator<<, 130
rend, 97	operator=, 122
set_x, 98	pointToIndex, 123
setCurrentTime, 98	reduce, 123
setT0, 98	reinitialize, 123
setValueAtCurrent, 98	set_B, 123
show, 98	set_Binv, 124
storageDimension, 98	set_C, 124
subcurve, 99	set_Cr0, 124
t0, 99	set r, 125
translate, 99	set_r0, 125
writeTo, 100	set_x, 126
capd::ddes::DDEPiecewisePolynomialCurve< Grid-	set_Xi, 126
Spec, JetSpec >::rebind< OtherJetTypeSpec	setValueAtCurrent, 126
>, 191	show, 127
capd::ddes::DDEPoincareMap< DynSysSpec, Section-	
Spec >, 100	storageDimension, 127
detectCrossingDirection, 102	storageN0, 127
findCrossingTime, 102	subcurve, 127, 128
integrateUntilSectionCrossing, 102	take_B, 128
operator(), 102, 103	take_Binv, 128
capd::ddes::DDERigorousFunctionalMap< Solution-	take_C, 128
CurveSpec, JetSpec >, 103	take_r, 128
∼DDERigorousFunctionalMap, 105	take_r0, 129
checkCurveDimension, 105	take_x, 129
collectComputationData, 105, 106	translate, 129
computeDDECoefficients, 107, 108	writeTo, 129
convert, 108	capd::ddes::DDETaylorSolver< FunctionalMapSpec >,
imageDimension, 109	131
capd::ddes::DDESolutionCurve< SetSpec >, 109	encloseSolution, 131, 132
add, 116, 117	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
addPastPiece, 117	imMapSpec, SolutionCurveSpec, JetSpec
affineTransform, 117	>, 133
copyPieces, 117	$\sim$ DiscreteDelaysFunctionalMap, 136
DDESolutionCurve, 114–116	begin, 136
dot, 118	collectComputationData, 136, 137
dt, 118	computeDDECoefficients, 137
epsilonShift, 118	const_iterator, 135
eval, 119	delaysCount, 138
extend, 119	DelayStorageType, 135
get_B, 119	dimension, 138
get_Binv, 120	end, 138
get_C, 120	findRoughEnclosure, 138
get_r, 120	imageDimension, 139
get_r0, 120	iterator, 135
get_x, 120	operator(), 139
get_Xi, 120	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
getLastEnclosure, 120	imMapSpec, SolutionCurveSpec, JetSpec
getPiece, 121	>::rebind< OtherSolutionSpec, OtherJetSpec
getValueAtCurrent, 121	>, 190
jetOrderAt, 121	capd::ddes::DiscreteTimeGrid< RealSpec >, 139
jetPtr, 121	badge, 140
յցս ս, ւշ ւ	bauge, 140

capd::ddes::DiscreteTimeGrid< RealSpec >::TimePoint	Гуре, begin, 164, 165
219	coeff, 165
badge, 220	deallocateCoeffs, 165
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	decreasedOrder, 165
ciesSpec >, 141	dimension, 165
~DoubletonInterface, 143	dt, 166
add, 144	end, 166
affineTransform, 144	eval, 166
common_B, 144	eval, 160 evalAtDelta, 167
common_C, 145	evalCoeff, 167
common_r, 145	evalCoeffAtDelta, 167, 168
common_r0, 145	GenericJet, 162, 163
common_x, 145	get_x, 168
dimension, 146	getCoeffs, 168
dot, 146	getT0, 168
get_B, 146	hull, 168
get_Binv, 146	increasedOrder, 169
get_C, 146	operator VectorType, 169
get_r, 147	operator<<, 172
get_r0, 147	operator>>, 172
get_x, 147	operator=, 169
hull, 147	operator[], 169
makeStorage_B, 147	order, 170
makeStorage_C, 147	reallocateCoeffs, 170
makeStorage_r, 148	
	set_x, 170
makeStorage_r0, 148	setCoeffs, 170
makeStorage_x, 148	setOrder, 170
midPoint, 148	setT0, 171
mul, 148	show, 171
mulThenAdd, 148	storageDimension, 171
operator VectorType, 149	t0, 171
operator<<, 156	capd::ddes::GenericJet< TimePointSpec, DataSpec,
operator>>, 156	VectorSpec, MatrixSpec, isInterval >::rebind<
operator*=, 149	OtherTimePointSpec, OtherDataSpec, Oth-
reinitialize, 149	erVectorSpec, OtherMatrixSpec, OtherIsInter-
set_B, 149, 150	val >, 191
set_Binv, 150	capd::ddes::LasotaWazewska< ScalarSpec, Param-
set_C, 150, 151	Spec >, 172
set_Cr0, 151	dimension, 173
set_r, 152	getParamsCount, 173
set_r0, 152, 153	imageDimension, 174
set_x, 153	operator(), 174
storageDimension, 154	capd::ddes::LinearMap< MatrixSpec, ParamSpec >,
•	174
storageN0, 154	
take_B, 154	dimension, 175
take_Binv, 154	imageDimension, 175
take_C, 154	operator(), 176
take_r, 155	capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >,
take_r0, 155	176
take_x, 155	dimension, 178
translate, 155	gamma, 178
capd::ddes::GenericJet< TimePointSpec, DataSpec,	getParamsCount, 178
VectorSpec, MatrixSpec, isInterval >, 159	imageDimension, 178
∼GenericJet, 163	n, 179
allocateCoeffs, 163	operator(), 178
at, 163, 164	capd::ddes::ODEPendulum, 185
back, 164	dimension, 186
badge, 164	imageDimension, 186
Jaago, IVI	inagobinionolon, 100

capd::ddes::PerezMaltaCoutinho< ScalarSpec, Param-	take_C, 218
Spec >, 188	take_r, 218
dimension, 189	take_r0, 218
getParamsCount, 189	take_x, 218
imageDimension, 190	translate, 219
operator(), 190	updateBinv, 219
capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >,	capd::ddes::ToyModel, 221
198	dimension, 221
a, 199	imageDimension, 222
b, 200	capd::ddes::ToyModelSq, 222
c, 200	dimension, 223
epsi, 200	imageDimension, 223
getParamsCount, 199	capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec
imageDimension, 199	>, 223
capd::ddes::ScalarLinearMap< ScalarSpec, Param-	dimension, 224
Spec >, 200	getParamsCount, 224
dimension, 201	imageDimension, 224
imageDimension, 201	capd::ddes::VectorWithJacData< VectorSpec, Ma-
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec	trixSpec >, 225
>, 202	capd::ddes::Wischert< ScalarSpec, ParamSpec >, 226
$\sim$ SharedDoubleton, 208	dimension, 227
add, 209	getParamsCount, 227
affineTransform, 209	imageDimension, 228
allocate, 209	operator(), 228
assureOwner, 209	capd::ddeshelper::ArgumentParser, 13
common_B, 210	operator<<, 14
common_Binv, 210	capd::ddeshelper::BinaryData< T >, 36
common_C, 210	capd::ddeshelper::CoordinateFrame< MatrixSpec, Vec-
common_r, 210	torSpec, ScalarSpec >, 36
common_r0, 210	capd::ddeshelper::CubeSet< VectorSpec >, 38
common_x, 211	insert_cover, 39
deallocate, 211	operator<<, 40
dimension, 211	operator>>, 40
get_B, 211	subcubeset, 39
get_Binv, 211	capd::ddeshelper::CubeSet< VectorSpec >::CubeSetIterator,
get_C, 212	40
get_r, 212	capd::ddeshelper::DDECompareHelper< VectorSpec
get r0, 212	>, 54
<del>-</del> -	getUnknownsCount, 55
get_x, 212 mul, 212	capd::ddeshelper::GeneratorRange< IType >, 158
mulThenAdd, 212	increment, 159
operator=, 213	
•	capd::ddeshelper::NonrigorousHelper< EqSpec, de- laysSpec >, 179
rawSetup, 213 reallocate, 213	integrate, 182, 183
reinitialize, 213	loadData, 183
sanityCheck, 213	makeSection, 183
set_B, 214	makeSolver, 183
set_Binv, 214	NonrigorousHelper, 181, 182
set_C, 215	poincare, 184
set_Cr0, 215	refinePeriodic, 184
set_r, 216	setParam, 185
set_r0, 216	setParams, 185
set_x, 217	capd::ddeshelper::PathConfig, 186
SharedDoubleton, 206–208	filename, 187
storageDimension, 217	filepath, 187
storageN0, 217	fullpath, 188
take_B, 217	mkdir_p, 188
take_Binv, 218	PathConfig, 187

suffix, 188	ciesSpec >, 145
capd::ddeshelper::RigorousHelper< EqSpec, de-	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
laysSpec, PoliciesSpec >, 191	cySpec >, 210
constantInitialSolution, 194	common_r0
dataToSolution, 194, 195	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
dumpData, 195	ciesSpec >, 30
functionToSolution, 195	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
poincare, 196	ciesSpec >, 145
r0ToSolution, 196, 197	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
setParam, 197	cySpec >, 210
setParams, 197	common_x
timemap, 197	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddeshelper::RigorousHelper< EqSpec, de-	ciesSpec >, 30
laysSpec, PoliciesSpec >::CoordinateSystem,	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
	ciesSpec >, 145
37	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
checkCurveDimension	cySpec >, 211
capd::ddes::DDEBasicFunctionalMap< Solution-	- •
CurveSpec, JetSpec >, 45	computeDDECoefficients
capd::ddes::DDERigorousFunctionalMap< Solu-	capd::ddes::BasicDiscreteDelaysFunctionalMap<
tionCurveSpec, JetSpec >, 105	FinDimMapSpec, SolutionCurveSpec, Jet-
clear	Spec >, 18–20
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::DDEBasicFunctionalMap< Solution-
GridSpec, JetSpec >, 90	CurveSpec, JetSpec >, 46–48
coeff	capd::ddes::DDERigorousFunctionalMap< Solu-
capd::ddes::GenericJet< TimePointSpec, DataSpec,	tionCurveSpec, JetSpec >, 107, 108
VectorSpec, MatrixSpec, isInterval >, 165	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
· · · · · · · · · · · · · · · · · · ·	imMapSpec, SolutionCurveSpec, JetSpec >,
collectComputationData	137
capd::ddes::BasicDiscreteDelaysFunctionalMap<	const iterator
FinDimMapSpec, SolutionCurveSpec, Jet-	capd::ddes::BasicDiscreteDelaysFunctionalMap<
Spec >, 18	FinDimMapSpec, SolutionCurveSpec, Jet-
capd::ddes::DDEBasicFunctionalMap< Solution-	Spec >, 17
CurveSpec, JetSpec >, 45	•
capd::ddes::DDERigorousFunctionalMap< Solu-	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
tionCurveSpec, JetSpec >, 105, 106	imMapSpec, SolutionCurveSpec, JetSpec >,
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	135
imMapSpec, SolutionCurveSpec, JetSpec >,	constantInitialSolution
136, 137	capd::ddeshelper::RigorousHelper< EqSpec, de-
common_B	laysSpec, PoliciesSpec >, 194
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	convert
1 ,	capd::ddes::DDEBasicFunctionalMap< Solution-
ciesSpec >, 29	CurveSpec, JetSpec >, 48
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddes::DDERigorousFunctionalMap< Solu-
ciesSpec >, 144	tionCurveSpec, JetSpec >, 108
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	copyPieces
cySpec >, 210	capd::ddes::DDEPiecewisePolynomialCurve<
common_Binv	GridSpec, JetSpec >, 90
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	
cySpec >, 210	capd::ddes::DDESolutionCurve< SetSpec >, 117
common_C	currentTime
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::DDEPiecewisePolynomialCurve<
ciesSpec >, 29	GridSpec, JetSpec >, 90
	dete Te Celution
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	dataToSolution
ciesSpec >, 145	capd::ddeshelper::RigorousHelper< EqSpec, de-
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	laysSpec, PoliciesSpec >, 194, 195
cySpec >, 210	DDEForwardTaylorCurvePiece
common_r	capd::ddes::DDEForwardTaylorCurvePiece<
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	TimePointSpec, SetSpec, isInterval >, 60-
ciesSpec >, 30	62
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	DDEJetSection

capd::ddes::DDEJetSection< CurveSpec, isInterval >, 81	TimePointSpec, SetSpec, isInterval >, 66 dimCheckC
DDEPiecewisePolynomialCurve	capd::ddes::DDEForwardTaylorCurvePiece<
capd::ddes::DDEPiecewisePolynomialCurve<	TimePointSpec, SetSpec, isInterval >, 66
GridSpec, JetSpec >, 87, 88	dimension
DDESolutionCurve	capd::ddes::BasicDiscreteDelaysFunctionalMap<
capd::ddes::DDESolutionCurve< SetSpec >, 114-	FinDimMapSpec, SolutionCurveSpec, Jet- Spec >, 21
deallocate	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddes::DDEForwardTaylorCurvePiece<	ciesSpec >, 30
TimePointSpec, SetSpec, isInterval >, 65	capd::ddes::CubicIkeda< ScalarSpec, ParamSpec
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	>, 42
cySpec >, 211	capd::ddes::DDEForwardTaylorCurvePiece<
deallocateCoeffs	TimePointSpec, SetSpec, isInterval >, 66
capd::ddes::GenericJet< TimePointSpec, DataSpec,	•
VectorSpec, MatrixSpec, isInterval >, 165	GridSpec, JetSpec >, 91
deallocateJet	capd::ddes::DiscreteDelaysFunctionalMap FinD- imMonSpace SolutionCurveSpace JotSpace
capd::ddes::DDEForwardTaylorCurvePiece < TimePointSpec, SetSpec, isInterval >, 65	imMapSpec, SolutionCurveSpec, JetSpec >, 138
deallocatePieces	capd::does::DoubletonInterface< MatrixSpec, Poli-
capd::ddes::DDEPiecewisePolynomialCurve<	ciesSpec >, 146
GridSpec, JetSpec >, 90	capd::ddes::GenericJet< TimePointSpec, DataSpec
deallocateR0	VectorSpec, MatrixSpec, isInterval >, 165
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::LasotaWazewska< ScalarSpec,
TimePointSpec, SetSpec, isInterval >, 65	ParamSpec >, 173
deallocateXi	capd::ddes::LinearMap< MatrixSpec, ParamSpec
capd::ddes::DDEForwardTaylorCurvePiece<	>, 175
TimePointSpec, SetSpec, isInterval >, 65	capd::ddes::MackeyGlass< ScalarSpec, Param-
deconvert capd::ddes::DDEBasicFunctionalMap< Solution-	Spec >, 178 capd::ddes::ODEPendulum, 186
capd::ddes::DDEBasicFunctionalMap< Solution- CurveSpec, JetSpec >, 48	capd::ddes::DEF endulum, 100  capd::ddes::PerezMaltaCoutinho< ScalarSpec,
decreasedOrder	ParamSpec >, 189
capd::ddes::GenericJet< TimePointSpec, DataSpec,	
VectorSpec, MatrixSpec, isInterval >, 165	ParamSpec >, 201
delaysCount	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
capd::ddes::BasicDiscreteDelaysFunctionalMap<	cySpec >, 211
FinDimMapSpec, SolutionCurveSpec, Jet-	capd::ddes::ToyModel, 221
Spec >, 21	capd::ddes::ToyModelSq, 223
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	capd::ddes::ToyModelSqA< ScalarSpec, Param- Spec >, 224
imMapSpec, SolutionCurveSpec, JetSpec >, 138	capd::ddes::Wischert< ScalarSpec, ParamSpec
DelayStorageType	>, 227
capd::ddes::BasicDiscreteDelaysFunctionalMap<	ElNino < ScalarSpec, ParamSpec >, 157
FinDimMapSpec, SolutionCurveSpec, Jet-	dot
Spec >, 17	capd::ddes::DDEForwardTaylorCurvePiece<
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	TimePointSpec, SetSpec, isInterval >, 66
imMapSpec, SolutionCurveSpec, JetSpec >, 135	capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 91
detectCrossingDirection	capd::ddes::DDESolutionCurve< SetSpec >, 118
capd::ddes::DDEBasicPoincareMap< Dyn-	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
SysSpec, SectionSpec >, 51	ciesSpec >, 146
capd::ddes::DDEPoincareMap<	dt
SectionSpec >, 102	capd::ddes::DDEForwardTaylorCurvePiece<
dimCheck	TimePointSpec, SetSpec, isInterval >, 67
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >, 66	capd::ddes::DDESolutionCurve< SetSpec >, 118 capd::ddes::GenericJet< TimePointSpec, DataSpec
dimCheckB	VectorSpec, MatrixSpec, isInterval >, 166
capd::ddes::DDEForwardTaylorCurvePiece<	dumpData

capd::ddeshelper::RigorousHelper< EqSpec, de- laysSpec, PoliciesSpec >, 195	capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 92 capd::ddes::DDESolutionCurve< SetSpec >, 119
ElNino < ScalarSpec, ParamSpec >, 156	
dimension, 157	filename
getParamsCount, 157	capd::ddeshelper::PathConfig, 187
imageDimension, 157	filepath
operator(), 158	capd::ddeshelper::PathConfig, 187
encloseSolution	findCrossingTime
capd::ddes::DDETaylorSolver< FunctionalMap-	capd::ddes::DDEBasicPoincareMap< Dyn-
Spec >, 131, 132	SysSpec, SectionSpec >, 51
end	capd::ddes::DDEPoincareMap< DynSysSpec,
capd::ddes::BasicDiscreteDelaysFunctionalMap<	SectionSpec >, 102
FinDimMapSpec, SolutionCurveSpec, Jet-	findRoughEnclosure
Spec >, 21	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
capd::ddes::DDEPiecewisePolynomialCurve<	imMapSpec, SolutionCurveSpec, JetSpec >,
GridSpec, JetSpec >, 91	138
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	fullpath
imMapSpec, SolutionCurveSpec, JetSpec >,	capd::ddeshelper::PathConfig, 188
138	functionToSolution
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddeshelper::RigorousHelper< EqSpec, de-
VectorSpec, MatrixSpec, isInterval >, 166	laysSpec, PoliciesSpec >, 195
• • • • • • • • • • • • • • • • • • • •	
endJet capd::ddes::DDEForwardTaylorCurvePiece<	gamma
	capd::ddes::MackeyGlass< ScalarSpec, Param-
TimePointSpec, SetSpec, isInterval >, 67	Spec >, 178
epsi	GenericJet
capd::ddes::RosslerDelay< ScalarSpec, Param-	capd::ddes::GenericJet< TimePointSpec, DataSpec,
Spec >, 200	VectorSpec, MatrixSpec, isInterval >, 162,
epsilonShift	163
capd::ddes::DDEPiecewisePolynomialCurve<	get_B
GridSpec, JetSpec >, 91, 92	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddes::DDESolutionCurve< SetSpec >, 118	ciesSpec >, 31
eval	capd::ddes::DDEForwardTaylorCurvePiece<
capd::ddes::DDEForwardTaylorCurvePiece<	TimePointSpec, SetSpec, isInterval >, 69
TimePointSpec, SetSpec, isInterval >, 67	capd::ddes::DDESolutionCurve < SetSpec >, 119
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
GridSpec, JetSpec >, 92	ciesSpec >, 146
capd::ddes::DDESolutionCurve< SetSpec >, 119	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
capd::ddes::GenericJet< TimePointSpec, DataSpec,	cySpec >, 211
VectorSpec, MatrixSpec, isInterval >, 166	get_Binv
evalAtDelta	capd::ddes::DDESolutionCurve < SetSpec >, 120
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
TimePointSpec, SetSpec, isInterval >, 68	ciesSpec >, 146
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
VectorSpec, MatrixSpec, isInterval >, 167	cySpec >, 211
evalCoeff	get_C
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
TimePointSpec, SetSpec, isInterval >, 68	ciesSpec >, 31
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddes::DDEForwardTaylorCurvePiece<
VectorSpec, MatrixSpec, isInterval >, 167	TimePointSpec, SetSpec, isInterval >, 69
evalCoeffAtDelta	capd::ddes::DDESolutionCurve< SetSpec >, 120
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
TimePointSpec, SetSpec, isInterval >, 68,	ciesSpec >, 146
69	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
capd::ddes::GenericJet < TimePointSpec, DataSpec,	cySpec >, 212
VectorSpec, MatrixSpec, isInterval >, 167,	get_r
168	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
extend	ciesSpec >, 31

capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >, 69	capd::ddes::RosslerDelay< ScalarSpec, Param- Spec >, 199
capd::ddes::DDESolutionCurve< SetSpec >, 120 capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddes::ToyModelSqA< ScalarSpec, Param- Spec >, 224
ciesSpec >, 147 capd::ddes::SharedDoubleton< MatrixSpec, Poli-	capd::ddes::Wischert< ScalarSpec, ParamSpec >, 227
cySpec >, 212	ElNino < ScalarSpec, ParamSpec >, 157
get_r0	getPiece
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::DDEPiecewisePolynomialCurve<
ciesSpec >, 31	GridSpec, JetSpec >, 93
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DDESolutionCurve< SetSpec >, 121
TimePointSpec, SetSpec, isInterval >, 69	getStep
capd::ddes::DDESolutionCurve< SetSpec >, 120	capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 93
capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >, 147	getT0
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	capd::ddes::DDEForwardTaylorCurvePiece<
cySpec >, 212	TimePointSpec, SetSpec, isInterval >, 70 capd::ddes::DDEPiecewisePolynomialCurve<
get_x	GridSpec, JetSpec >, 93
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::GenericJet< TimePointSpec, DataSpec,
<pre>ciesSpec &gt;, 31 capd::ddes::DDEForwardTaylorCurvePiece&lt;</pre>	VectorSpec, MatrixSpec, isInterval >, 168
TimePointSpec, SetSpec, isInterval >, 70	getUnknownsCount
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddeshelper::DDECompareHelper< Vector-
GridSpec, JetSpec >, 92	Spec >, 55
capd::ddes::DDESolutionCurve< SetSpec >, 120	getValueAtCurrent
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddes::DDEPiecewisePolynomialCurve<
ciesSpec >, 147	GridSpec, JetSpec >, 93, 94 capd::ddes::DDESolutionCurve< SetSpec >, 121
capd::ddes::GenericJet < TimePointSpec, DataSpec,	grid
VectorSpec, MatrixSpec, isInterval >, 168 capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, 212	capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 94
get_Xi	
capd::ddes::DDEForwardTaylorCurvePiece<	hull
TimePointSpec, SetSpec, isInterval >, 70 capd::ddes::DDESolutionCurve< SetSpec >, 120	capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >, 31
getCoeffs	capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >, 147
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >, 168	capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >, 168
getCurrentTime	
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 93	<pre>imageDimension     capd::ddes::BasicDiscreteDelaysFunctionalMap&lt;</pre>
getGradient	FinDimMapSpec, SolutionCurveSpec, Jet-
capd::ddes::DDEJetSection< CurveSpec, isInter-	Spec >, 22
val >, 81	capd::ddes::CubicIkeda < ScalarSpec, ParamSpec
getGrid capd::ddes::DDEPiecewisePolynomialCurve<	>, 42
GridSpec, JetSpec >, 93	capd::ddes::DDEBasicFunctionalMap< Solution- CurveSpec, JetSpec >, 49
getLastEnclosure	capd::ddes::DDERigorousFunctionalMap< Solu-
capd::ddes::DDESolutionCurve< SetSpec >, 120	tionCurveSpec, JetSpec >, 109
getParamsCount	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
capd::ddes::Cubiclkeda< ScalarSpec, ParamSpec >, 42	imMapSpec, SolutionCurveSpec, JetSpec >, 139
capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >, 173	capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >, 174
capd::ddes::MackeyGlass< ScalarSpec, Param- Spec >, 178	capd::ddes::LinearMap< MatrixSpec, ParamSpec >, 175
capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >, 189	capd::ddes::MackeyGlass< ScalarSpec, Param- Spec >, 178

capd::ddes::ODEPendulum, 186	jetPtr
capd::ddes::PerezMaltaCoutinho< ScalarSpec,	capd::ddes::DDEPiecewisePolynomialCurve<
ParamSpec >, 190	GridSpec, JetSpec >, 95
capd::ddes::RosslerDelay< ScalarSpec, Param-	capd::ddes::DDESolutionCurve< SetSpec >, 121
Spec >, 199	length
capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >, 201	capd::ddes::DDEPiecewisePolynomialCurve<
capd::ddes::ToyModel, 222	GridSpec, JetSpec >, 96
capd::ddes::ToyModelSq, 223	capd::ddes::DESolutionCurve< SetSpec >, 121
capd::ddes::ToyModelSqA< ScalarSpec, Param-	loadData
Spec >, 224	capd::ddeshelper::NonrigorousHelper< EqSpec,
capd::ddes::Wischert< ScalarSpec, ParamSpec >, 228	delaysSpec >, 183
ElNino < ScalarSpec, ParamSpec >, 157	m_r0
increasedOrder	capd::ddes::DDESolutionCurve< SetSpec >, 130
capd::ddes::GenericJet< TimePointSpec, DataSpec,	makeSection
VectorSpec, MatrixSpec, isInterval >, 169	capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >, 183
increment	makeSolver
capd::ddeshelper::GeneratorRange< IType >, 159	capd::ddeshelper::NonrigorousHelper< EqSpec,
insert_cover capd::ddeshelper::CubeSet< VectorSpec >, 39	delaysSpec >, 183
integrate	makeStorage_B
capd::ddeshelper::NonrigorousHelper< EqSpec,	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
delaysSpec >, 182, 183	ciesSpec >, 147
integrateUntilSectionCrossing	makeStorage_C
capd::ddes::DDEBasicPoincareMap< Dyn-	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
SysSpec, SectionSpec >, 52	ciesSpec >, 147
capd::ddes::DDEPoincareMap< DynSysSpec,	makeStorage_r
SectionSpec >, 102	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
isMidCurve	ciesSpec >, 148 makeStorage_r0
capd::ddes::DDEForwardTaylorCurvePiece<	capd::does::DoubletonInterface< MatrixSpec, Poli-
TimePointSpec, SetSpec, isInterval >, 70	ciesSpec >, 148
iterator capd::ddes::BasicDiscreteDelaysFunctionalMap<	makeStorage_x
FinDimMapSpec, SolutionCurveSpec, Jet-	capd::ddes::DDEForwardTaylorCurvePiece<
Spec >, 17	TimePointSpec, SetSpec, isInterval >, 71
capd::ddes::DiscreteDelaysFunctionalMap< FinD-	capd::ddes::DDESolutionCurve< SetSpec >, 122
imMapSpec, SolutionCurveSpec, JetSpec >,	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
135	ciesSpec >, 148
	midCurve
j	capd::ddes::DDEForwardTaylorCurvePiece<
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 94	TimePointSpec, SetSpec, isInterval >, 71 midPoint
JacobianStorageType	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddes::DDEBasicFunctionalMap< Solution-	ciesSpec >, 31
CurveSpec, JetSpec >, 44	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
jet	ciesSpec >, 148
capd::ddes::DDEPiecewisePolynomialCurve<	mkdir_p
GridSpec, JetSpec >, 95	capd::ddeshelper::PathConfig, 188
jetAt	move
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DDESolutionCurve< SetSpec >, 122
TimePointSpec, SetSpec, isInterval >, 70	mul
jetCommonMaximalOrder	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddes::DDEPiecewisePolynomialCurve<	ciesSpec >, 32
GridSpec, JetSpec >, 95	capd::ddes::DDEForwardTaylorCurvePiece<
jetOrderAt	TimePointSpec, SetSpec, isInterval >, 71
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 95	capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 96
capd::ddes::DDESolutionCurve< SetSpec >, 121	capd::ddes::DDESolutionCurve< SetSpec >, 122

capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >, 148	capd::ddes::DDEBasicPoincareMap< Dyn-
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	SysSpec, SectionSpec >, 52, 53 capd::ddes::DDENonrigorousTaylorSolver< Func-
cySpec >, 212 mulThenAdd	tionalMapSpec >, 83, 84 capd::ddes::DDEPoincareMap< DynSysSpec,
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	SectionSpec >, 102, 103
ciesSpec >, 32	capd::ddes::DiscreteDelaysFunctionalMap< FinD-
capd::ddes::DDEPiecewisePolynomialCurve<	imMapSpec, SolutionCurveSpec, JetSpec >,
GridSpec, JetSpec >, 96	139
capd::ddes::DDESolutionCurve< SetSpec >, 122	capd::ddes::LasotaWazewska< ScalarSpec,
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	ParamSpec >, 174
ciesSpec >, 148 capd::ddes::SharedDoubleton< MatrixSpec, Poli-	capd::ddes::LinearMap< MatrixSpec, ParamSpec
cySpec >, 212	>, 176 capd::ddes::MackeyGlass< ScalarSpec, Param-
0,00000,1.1	Spec >, 178
n	capd::ddes::PerezMaltaCoutinho< ScalarSpec,
capd::ddes::MackeyGlass< ScalarSpec, Param-	ParamSpec >, 190
Spec >, 179	capd::ddes::Wischert< ScalarSpec, ParamSpec
NonrigorousHelper capd::ddeshelper::NonrigorousHelper< EqSpec,	>, 228
delaysSpec >, 181, 182	ElNino < ScalarSpec, ParamSpec >, 158
doia,000p00 > , 10 : , 10 I	operator=
oneStep	capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >, 32
capd::ddes::DDENonrigorousTaylorSolver< Func-	capd::ddes::DDEForwardTaylorCurvePiece<
tionalMapSpec >, 83	TimePointSpec, SetSpec, isInterval >, 71
operator VectorType capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddes::DDEPiecewisePolynomialCurve<
ciesSpec >, 149	GridSpec, JetSpec >, 96
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddes::DDESolutionCurve< SetSpec >, 122
VectorSpec, MatrixSpec, isInterval >, 169	capd::ddes::GenericJet< TimePointSpec, DataSpec,
operator<<	VectorSpec, MatrixSpec, isInterval >, 169
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, 213
GridSpec, JetSpec >, 100	operator[]
capd::ddes::DDESolutionCurve< SetSpec >, 130 capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddes::DDEForwardTaylorCurvePiece<
ciesSpec >, 156	TimePointSpec, SetSpec, isInterval >, 71,
capd::ddes::GenericJet< TimePointSpec, DataSpec,	72
VectorSpec, MatrixSpec, isInterval >, 172	capd::ddes::GenericJet< TimePointSpec, DataSpec,
capd::ddeshelper::ArgumentParser, 14	VectorSpec, MatrixSpec, isInterval >, 169
capd::ddeshelper::CubeSet< VectorSpec >, 40	order capd::ddes::DDEForwardTaylorCurvePiece<
operator>>	TimePointSpec, SetSpec, isInterval >, 72
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 100	capd::ddes::GenericJet< TimePointSpec, DataSpec,
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	VectorSpec, MatrixSpec, isInterval >, 170
ciesSpec >, 156	
capd::ddes::GenericJet < TimePointSpec, DataSpec,	pastTime
VectorSpec, MatrixSpec, isInterval >, 172	capd::ddes::DDEPiecewisePolynomialCurve<
capd::ddeshelper::CubeSet< VectorSpec >, 40	GridSpec, JetSpec >, 96
operator*=	PathConfig capd::ddeshelper::PathConfig, 187
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >, 96	poincare
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddeshelper::NonrigorousHelper< EqSpec,
ciesSpec >, 149	delaysSpec >, 184
operator()	capd::ddeshelper::RigorousHelper< EqSpec, de-
capd::ddes::BasicDiscreteDelaysFunctionalMap<	laysSpec, PoliciesSpec >, 196
FinDimMapSpec, SolutionCurveSpec, Jet-	pointToIndex
Spec >, 22	capd::ddes::DDEPiecewisePolynomialCurve<
capd::ddes::DDEBasicFunctionalMap< Solution- CurveSpec, JetSpec >, 49	GridSpec, JetSpec >, 97 capd::ddes::DDESolutionCurve< SetSpec >, 123
	capaaccobbbcolationouive Coetopec /, 120

r0ToSolution	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
capd::ddeshelper::RigorousHelper< EqSpec, de- laysSpec, PoliciesSpec >, 196, 197	ciesSpec >, 149, 150 capd::ddes::SharedDoubleton< MatrixSpec, Poli-
rawSetup	cySpec >, 214
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	set Binv
cySpec >, 213	capd::ddes::DDEForwardTaylorCurvePiece<
rbegin	TimePointSpec, SetSpec, isInterval >, 73
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::DDESolutionCurve< SetSpec >, 124
GridSpec, JetSpec >, 97	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
readFrom	ciesSpec >, 150
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
GridSpec, JetSpec >, 97	cySpec >, 214
reallocate	set_C
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
TimePointSpec, SetSpec, isInterval >, 72	ciesSpec >, 33
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	capd::ddes::DDEForwardTaylorCurvePiece<
cySpec >, 213	TimePointSpec, SetSpec, isInterval >, 73
reallocateCoeffs	capd::ddes::DDESolutionCurve< SetSpec >, 124
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
VectorSpec, MatrixSpec, isInterval >, 170	ciesSpec >, 150, 151
reallocateJet	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
capd::ddes::DDEForwardTaylorCurvePiece<	cySpec >, 215
TimePointSpec, SetSpec, isInterval >, 72	set_Cr0
reallocateR0	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddes::DDEForwardTaylorCurvePiece<	ciesSpec >, 33
TimePointSpec, SetSpec, isInterval >, 72	capd::ddes::DDEForwardTaylorCurvePiece<
reallocateXi	TimePointSpec, SetSpec, isInterval >, 74
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DDESolutionCurve< SetSpec >, 124
TimePointSpec, SetSpec, isInterval >, 73	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
reduce	ciesSpec >, 151
capd::ddes::DDESolutionCurve< SetSpec >, 123	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
refinePeriodic	cySpec >, 215
capd::ddeshelper::NonrigorousHelper< EqSpec,	set r
delaysSpec >, 184	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
reinitialize	ciesSpec >, 33
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::DDEForwardTaylorCurvePiece<
ciesSpec >, 32	
capd::ddes::DDEForwardTaylorCurvePiece<	TimePointSpec, SetSpec, isInterval >, 74 capd::ddes::DDESolutionCurve< SetSpec >, 125
TimePointSpec, SetSpec, isInterval >, 73	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
capd::ddes::DDESolutionCurve< SetSpec >, 123	ciesSpec >, 152
capd::ddes::DoubletonInterface< MatrixSpec, Poli-	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
ciesSpec >, 149	cySpec >, 216
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	
cySpec >, 213	set_r0 capd::ddes::BasicDoubleton< MatrixSpec, Poli-
rend	ciesSpec >, 33
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::DDEForwardTaylorCurvePiece<
GridSpec, JetSpec >, 97	TimePointSpec, SetSpec, isInterval >, 74
	capd::ddes::DDESolutionCurve< SetSpec >, 125
sanityCheck	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	ciesSpec >, 152, 153
ciesSpec >, 32	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	cySpec >, 216
cySpec >, 213	
set_B cand::ddas::RasicDoubleton / MatrixSpac Poli-	set_x capd::ddes::BasicDoubleton< MatrixSpec. Poli-
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >, 34
<pre>ciesSpec &gt;, 32 capd::ddes::DDEForwardTaylorCurvePiece&lt;</pre>	capd::ddes::DDEForwardTaylorCurvePiece<
TimePointSpec, SetSpec, isInterval >, 73	TimePointSpec, SetSpec, isInterval >, 74
capd::ddes::DDESolutionCurve< SetSpec >, 123	capd::ddes::DDEPiecewisePolynomialCurve<
Japanadon De Lociation du Vo V Octopolo /, 120	Sapandaconder icocwisci ciyriciilalcai ve

0.110	
GridSpec, JetSpec >, 98	ciesSpec >, 35
capd::ddes::DDESolutionCurve< SetSpec >, 126 capd::ddes::DoubletonInterface< MatrixSpec, Poli-	setValueAtCurrent capd::ddes::DDEPiecewisePolynomialCurve<
ciesSpec >, 153	GridSpec, JetSpec >, 98
capd::ddes::GenericJet< TimePointSpec, DataSpec,	·
VectorSpec, MatrixSpec, isInterval >, 170	SharedDoubleton
capd::ddes::SharedDoubleton< MatrixSpec, Poli-	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
cySpec >, 217	cySpec >, 206-208
set_Xi	show
capd::ddes::DDEForwardTaylorCurvePiece<	capd::des::BasicDoubleton< MatrixSpec, Poli-
TimePointSpec, SetSpec, isInterval >, 75	<pre>ciesSpec &gt;, 35 capd::ddes::DDEForwardTaylorCurvePiece&lt;</pre>
capd::ddes::DDESolutionCurve< SetSpec >, 126 setAsConstant	TimePointSpec, SetSpec, isInterval >, 77
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DDEPiecewisePolynomialCurve<
TimePointSpec, SetSpec, isInterval >, 75,	GridSpec, JetSpec >, 98
76	capd::ddes::DDESolutionCurve < SetSpec >, 127
setCoeffs	capd::ddes::GenericJet< TimePointSpec, DataSpec
capd::ddes::GenericJet< TimePointSpec, DataSpec,	VectorSpec, MatrixSpec, isInterval >, 171
VectorSpec, MatrixSpec, isInterval >, 170	storageDimension
setCurrentTime	capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >, 35
capd::ddes::DDEPiecewisePolynomialCurve<	capd::ddes::DDEForwardTaylorCurvePiece<
GridSpec, JetSpec >, 98 setCurrentV	TimePointSpec, SetSpec, isInterval >, 77
capd::ddes::DDEBasicPoincareMap< Dyn-	capd::ddes::DDEPiecewisePolynomialCurve<
SysSpec, SectionSpec >, 53	GridSpec, JetSpec >, 98
setInitialV	capd::ddes::DDESolutionCurve< SetSpec >, 127
capd::ddes::DDEBasicPoincareMap< Dyn-	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
SysSpec, SectionSpec >, 54	ciesSpec >, 154
setOrder	capd::ddes::GenericJet< TimePointSpec, DataSpec VectorSpec, MatrixSpec, isInterval >, 171
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
VectorSpec, MatrixSpec, isInterval >, 170	cySpec >, 217
setParam capd::ddeshelper::NonrigorousHelper< EqSpec,	storageN0
delaysSpec >, 185	capd::ddes::BasicDoubleton< MatrixSpec, Poli-
capd::ddeshelper::RigorousHelper< EqSpec, de-	ciesSpec >, 35
laysSpec, PoliciesSpec >, 197	capd::ddes::DDEForwardTaylorCurvePiece<
setParams	TimePointSpec, SetSpec, isInterval >, 77 capd::ddes::DDESolutionCurve< SetSpec >, 127
capd::ddeshelper::NonrigorousHelper< EqSpec,	capd::ddes::DoubletonInterface< MatrixSpec, Poli-
delaysSpec >, 185	ciesSpec >, 154
capd::ddeshelper::RigorousHelper< EqSpec, de-	capd::ddes::SharedDoubleton< MatrixSpec, Poli-
laysSpec, PoliciesSpec >, 197 setT0	cySpec >, 217
capd::ddes::DDEForwardTaylorCurvePiece<	subcubeset
TimePointSpec, SetSpec, isInterval >, 76	capd::ddeshelper::CubeSet< VectorSpec >, 39
capd::ddes::DDEPiecewisePolynomialCurve<	subcurve capd::ddes::DDEPiecewisePolynomialCurve<
GridSpec, JetSpec >, 98	GridSpec, JetSpec >, 99
capd::ddes::GenericJet< TimePointSpec, DataSpec,	capd::ddes::DDESolutionCurve< SetSpec >, 127,
VectorSpec, MatrixSpec, isInterval >, 171	128
setupDimension	suffix
<pre>capd::ddes::BasicDoubleton&lt; MatrixSpec, Poli- ciesSpec &gt;, 34</pre>	capd::ddeshelper::PathConfig, 188
setupFromData	summa
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	capd::ddes::DDEForwardTaylorCurvePiece<
ciesSpec >, 34	TimePointSpec, SetSpec, isInterval >, 77 summaAtDelta
capd::ddes::DDEForwardTaylorCurvePiece<	capd::ddes::DDEForwardTaylorCurvePiece<
TimePointSpec, SetSpec, isInterval >, 76	TimePointSpec, SetSpec, isInterval >, 78
setupFromOther	
capd::ddes::BasicDoubleton< MatrixSpec, Poli-	t0

```
Poli-
    capd::ddes::DDEForwardTaylorCurvePiece<
                                                          capd::ddes::BasicDoubleton< MatrixSpec,
         TimePointSpec, SetSpec, isInterval >, 78
                                                               ciesSpec >, 35
    capd::ddes::DDEPiecewisePolynomialCurve<
                                                          capd::ddes::DDEForwardTaylorCurvePiece<
                                                               TimePointSpec, SetSpec, isInterval >, 79
         GridSpec, JetSpec >, 99
                                                          capd::ddes::DDEPiecewisePolynomialCurve<
    capd::ddes::GenericJet< TimePointSpec, DataSpec,
                                                               GridSpec, JetSpec >, 99
         VectorSpec, MatrixSpec, isInterval >, 171
                                                          capd::ddes::DDESolutionCurve < SetSpec >, 129
take B
                                                          capd::ddes::DoubletonInterface< MatrixSpec, Poli-
    capd::ddes::DDESolutionCurve< SetSpec >, 128
                                                               ciesSpec >, 155
    capd::ddes::DoubletonInterface< MatrixSpec, Poli-
                                                          capd::ddes::SharedDoubleton< MatrixSpec, Poli-
         ciesSpec >, 154
                                                               cySpec >, 219
    capd::ddes::SharedDoubleton< MatrixSpec, Poli-
         cySpec >, 217
                                                      updateBinv
take Binv
                                                          capd::ddes::SharedDoubleton< MatrixSpec, Poli-
    capd::ddes::DDESolutionCurve < SetSpec >, 128
                                                               cySpec >, 219
    capd::ddes::DoubletonInterface< MatrixSpec, Poli-
                                                      updateCommonR0
         ciesSpec >, 154
                                                          capd::ddes::DDEForwardTaylorCurvePiece<
    capd::ddes::SharedDoubleton< MatrixSpec, Poli-
                                                               TimePointSpec, SetSpec, isInterval >, 79
         cySpec >, 218
take C
                                                      ValueStorageType
    capd::ddes::DDESolutionCurve< SetSpec >, 128
                                                          capd::ddes::DDEBasicFunctionalMap<
                                                                                                 Solution-
    capd::ddes::DoubletonInterface< MatrixSpec, Poli-
                                                               CurveSpec, JetSpec >, 44
         ciesSpec >, 154
                                                      VariableStorageType
    capd::ddes::SharedDoubleton< MatrixSpec, Poli-
                                                          capd::ddes::DDEBasicFunctionalMap<
         cySpec >, 218
                                                               CurveSpec, JetSpec >, 44
    capd::ddes::DDESolutionCurve< SetSpec >, 128
                                                      writeTo
    capd::ddes::DoubletonInterface< MatrixSpec, Poli-
                                                          capd::ddes::DDEPiecewisePolynomialCurve<
         ciesSpec >, 155
                                                               GridSpec, JetSpec >, 100
    capd::ddes::SharedDoubleton< MatrixSpec, Poli-
                                                          capd::ddes::DDESolutionCurve < SetSpec >, 129
         cySpec >, 218
take r0
    capd::ddes::DDEForwardTaylorCurvePiece<
         TimePointSpec, SetSpec, isInterval >, 78
    capd::ddes::DDESolutionCurve< SetSpec >, 129
    capd::ddes::DoubletonInterface < MatrixSpec, Poli-
         ciesSpec >, 155
    capd::ddes::SharedDoubleton< MatrixSpec, Poli-
         cySpec >, 218
take x
    capd::ddes::DDESolutionCurve< SetSpec >, 129
    capd::ddes::DoubletonInterface< MatrixSpec, Poli-
         ciesSpec >, 155
    capd::ddes::SharedDoubleton< MatrixSpec, Poli-
         cySpec >, 218
take Xi
    capd::ddes::DDEForwardTaylorCurvePiece<
         TimePointSpec, SetSpec, isInterval >, 78
taylor
    capd::ddes::DDEForwardTaylorCurvePiece<
         TimePointSpec, SetSpec, isInterval >, 78,
         79
taylorAtDelta
    capd::ddes::DDEForwardTaylorCurvePiece<
         TimePointSpec, SetSpec, isInterval >, 79
timemap
    capd::ddeshelper::RigorousHelper< EqSpec, de-
         laysSpec, PoliciesSpec >, 197
translate
```