Contents

1 Class Summaries **2**
1.1 Domain . . . 2
1.2 Metric . . . 4
1.3 RiemannSurface . . . 5

2 Additional Examples **6**
2.1 X-Ray Transform . . . 6
2.1.1 Forward Problem . . . 6
2.1.2 Representing Functions from Data . . . 6
2.1.3 Inversion . . . 6
2.1.4 Plotting . . . 6
2.2 Writing New Metric and Domain Subclasses . . . 6

3 Navigating Code **7**
3.1 Naming Convention . . . 7

4 Additional Notes on Numerical Implementations and Optimizations **8**
4.1 Interpolated Results . . . 8
4.2 IsInsideR2 Method . . . 8
4.3 Class Methods vs. Anonymous Functions . . . 8

1 Class Summaries

1.1 Domain

The Domain class handles information about a boundary defined in polar coordinates with respect to an origin and rotation. Additionally, this class handles various boundary-specific computations used in the RiemannSurface class.

1.1.1 Construction

The Domain class exists as a general superclass of its many specialized subclasses. While it is possible to construct any domain directly from this class, to maintain accuracy and efficiency of usage it is much better to instead construct any of its subclasses.

Note also that no construction of Domain takes arguments for the origin and rotation of the domain. These properties must be set after construction.

`Domain`	Default instance of Domain. It is recommended to instead use the circleDomain subclass.
`Domain(handle)`	Domain from a function handle. Derivatives and several methods are implemented numerically.

1.1.2 Properties

`instance.originX`	The X part of the origin vector.
`instance.originY`	The Y part of the origin vector.
`instance.theta`	The counterclockwise rotation of the domain.
`instance.exitInterpType`	The interpolation method used in the exitInterp method. Available methods are 'last', 'slinear' and 'slinearB'.

1.1.3 Method Summary

Boundary Functions

`instance.bdr(Th)`
: Evaluates the boundary function at the angles Th.

`instance.dbdr(Th)`
: Evaluates the first derivative of the boundary function at the angles Th.

`instance.ddbdr(Th)`
: Evaluates the second derivative of the boundary function at the angles Th.

Computers

`instance.getMinRadius`
: Returns the minimum radius of the boundary function.

`instance.getBoundingBox`
: Returns the lower left and upper right corners of the boundary's axis aligned bounding box relative to the origin.

Plotters

`instance.plot`
: Plots the boundary onto an available figure using instance.bdr.

Misc

`Domain.mustBeDomain(obj)`
: Errors if the passed argument is not an instance of Domain or its subclasses.

1.1.4 Subclasses

Note that all subclasses of Domain take Name-Values arguments.

`circleDomain(radius)`	Circular domain.
`ellipseDomain(radiusA, radiusB)`	Elliptical domain.
`cosineDomain(radius, cycles, amplitude)`	
`polygonDomain(radius, sides)`	Regular polygon domain.
`smoothDomain(radius, sides, bevelRadius)`	Regular polygon domain whose function is C^2.

1.1.5 Detailed Method Description

1.2 Metric

The Metric class stores the conformal factor of a conformally Euclidean metric and its derivatives for use in the RiemannSurface class.

1.2.1 Construction

The Metric class exists as a general superclass of its many specialized subclasses. While it is possible to construct any metric directly from this class, to maintain accuracy and efficiency of usage it is much better to instead construct any of its subclasses.

Note that all subclasses of Metric take Name-Values arguments.

`Metric`	Default instance of Metric. It is recommended to instead use the euclidMetric subclass.
`Metric(handle)`	
`euclidMetric`	Euclidean metric.
`sphereMetric(radius)`	Metric with constant positive curvature.
`hyperbolicMetric(radius)`	Metric with constant negative curvature.
`constcurveMetric(kappa)`	Metric with constant curvature. $4\kappa = R^{-2}$

1.2.2 Method Summary

Blah

`instance.lg(x,y)`

Evaluates the natural log of the g function at the points (x,y).

1.2.3 Detailed Method Description

1.3 RiemannSurface

The RiemannSurface class stores instances of the Domain and Metric classes and handles computations dealing with geodesics and the geodesic X-Ray transform.

1.3.1 Construction

`RiemannSurface`	Default instance of RiemannSurface using instances of euclidMetric and circleDomain.
`RiemannSurface(domain)`	RiemannSurface from a chosen domain and default parameters.
`RiemannSurface(domain, metric)`	RiemannSurface from a chosen domain and metric.
`RiemannSurface(__, stepType,stepSize,geoDur)`	RiemannSurface from a chosen domain and metric. The remaining properties are assigned on a name-value basis.

1.3.2 Properties

By default, stepSize and stepType are set for a balance of accuracy and speed. GeoDur is set to 100 units by default and will slow down the program if any geodesics end up trapped within the domain.

`instance.stepType`	The Runge-Kutta method used to solve differential equations. Available methods are 'EE', 'IE' and 'RK4'.
`instance.stepSize`	The distance a geodesic steps in relation to the metric.
`instance.geoDur`	The maximum length of a geodesic before the program assumes the geodesic is trapped.

1.3.3 Method Summary

Misc Computers

`[xO,yO,thO] = instance.geodesic(X,Y,Th)`

Evaluates the geodesics beginning at the points (X,Y) traveling in the directions Th until they exit the domain or are assumed trapped. Outputs are matrices indexed so that columns represent individual geodesics and rows represent the geodesics after some steps.

`[xO,yO,thO, aO,bO] = instance.geodesicJacobiAB(X,Y,Th)`

Evaluates geodesics in the fashion of instance.geodesic, but also computes the a and b functions described by the differential equations

$$(\ddot{b} + \kappa b)(t) \equiv 0, \quad b(0) \equiv 0, \ \dot{b} \equiv 1$$

where here κ denotes a function that describes the curvature at points along the geodesic.

`[xO,yO] = instance.findConjugates(X,Y,Th)`

Returns a list of points where conjugate points have been identified along geodesics. Similar results can be achieved by running instance.geodesicJacobiAB and searching the b function for zeros, but this method is faster and yields more accurate results using interpolation.

X-Ray Computers

Plotters

Misc Helpers

Geodesic Steppers

1.3.4 Detailed Method Description

2 Additional Examples

2.1 X-Ray Transform

2.1.1 Forward Problem

2.1.2 Representing Functions from Data

2.1.3 Inversion

2.1.4 Plotting

2.2 Writing New Metric and Domain Subclasses

3 Navigating Code

3.1 Naming Convention

3.1.1 Arguments and Outputs

3.1.2 Beta before Alpha

3.1.3 Plots and Figures

4 Additional Notes on Numerical Implementations and Optimizations

4.1 Interpolated Results

4.1.1 Boundary Exit Interpolation

4.1.2 Search Interpolation

4.2 IsInsideR2 Method

4.3 Class Methods vs. Anonymous Functions