Documentation

Types

`Main.Branch` — *Type.*

a series of `Solution`s

`Main.Solution` — *Type.*

todo

`Main.ContinuationMethod` — *Type.*

`ContinuationMethod..`

continuation method implementations extend `ContinuationMethod` and the corresponding `show` and `step`. responsible for changing the project itself

`Main.PC` — *Type.*

a predictor-corrector-method with step-size adaption and GUI Controls

`Main.SystemCore` — *Type.*

todo

`Main.Galerkin` — *Type.*

todo

`Main.Session` — *Type.*

todo

Utility Functions

Differentiation

`mbNewton.centralDifference` — *Function*.

```
centralDifference(homotopy, v, epsilon)
```

numerical differentiation method

`mbNewton.forwardDifference` — *Function*.

```
forwardDifference(homotopy, v, epsilon)
```

numerical differentiation method

`mbNewton.broyden` — *Function*.

```
broyden(homotopy, jacobian)
```

`mbNewton.newton` — *Function*.

```
newton(homotopy, jacobian, v , ...)
```

Galerkin

`Main.findCycle` — *Function*.

```
findCycle(H, t0, y0, transientIterations, transientStepSize,
    steadyStateIterations, steadyStateStepSize)
```

`Main.findCyclePoincare` — *Function*.

```
findCyclePoincare(F, y[, plane, clusterRating, nIntersections,
    maxCycles, sampleSize, transientIterations, transientStepSize,
    steadyStateStepSize])
```

extracts a single cycle of the steady state of ode `F` using poincare cuts through the `plane`.

`Main.prepareCycle` — *Function*.

```
prepareCycle(data, h, P[, fac])
```

cut single cycle of length P*fac from data, resample, shift s.t. X(0) 0, Fourier transform.

`mbInterpolate.interpolateLanczos` — *Function*.

```
interpolateLanczos(V, a::Integer)
```

simple periodic (!) Lanczos interpolation

`mbInterpolate.interpolateTrigonometric` — *Function*.

```
interpolateTrigonometric(a , a, b)
```

returns trigonometric polynomial. use with 2a,-2b and divide by 2m+1 to use with rfft coefficients.

GUI

`Main.ctrl` — *Function*.

```
ctrl(D, x)
```

Tuple (name::String, ::Type, init, v…)

`Main.mkControlGrid` — *Function*.

```
mkControlGrid(D, C)
```

creates a grid of controls with labels, handlers and encapsulated storage `c` in `C` is Tuple (name::String, ::Type, init, v…)

Session Control

`Main.create` — *Function*.

```
create(homotopy, jacobian, projection)
```

`Main.save` — *Function*.

```
save(filename, session[, overwrite])
```

`Main.load` — *Function*.

```
load(filename, homotopy, jacobian, projection)
```

ODE

`mbRK.rk` — *Function*.

```
rk(butcherTableau)
```

returns a runge-kutta method using the respective tableau:

```
function(f, t0, y0, h, pred[, init, callback])
```

e.g. `rk1`, or `rk4`. Examines the ode `f` starting from `t0`, `y0` with fixed stepsize `h` until `pred` evalutes to `false`.

Examples

Roessler

Lorenz

Index