igloo v.01

Generated by Doxygen 1.8.6

Fri Aug 24 2018 13:35:22

Online web application: http://igloo.uni-goettingen.de

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 demo Namespace Reference

Variables

- tuple modObj

4.1.1 Variable Documentation

4.1.1.1 tuple demo.modObj

Initial value:

```
= mod.IGLOO(gradientExt=(10,35),walkDur=200.0,startPos=62.5,rearingT=25.0,
                 simulationType='interpolate',gradientDist=75.0)
```

4.2 IGLOO Namespace Reference

Classes

- class IGLOO

4.3 locomotionInterpolation Namespace Reference

Functions

- def gauss
- def ratio
- def linear
- def poly2
- def poly3
- def poly4
- def hillUp
- def hillDown
- def hillDouble
- def velFunc
- def durFunc

- def velFuncLarva
- def durFuncLarva
- def calcParameters

4.3.1 Function Documentation

4.3.1.1 def locomotionInterpolation.calcParameters (*T*)

```
This function returns the rearing temperature depending parameters to
adjust the velocity and step duration functions. Most parameters can be fit
with a 2nd degree polynom. As only 3 different rearing temperatures were
used the fits might not be to reliable and it is saver to stay with the
three measured rearing temperatures: 18,25,30 °C

@param T float rearing temperature
@return lists two lists 1) v velocity parameters 2) d duration parameters
```

4.3.1.2 def locomotionInterpolation.durFunc (*x_p, x_t, d*)

```
This function calculates the fit of the fit duration in dependence of the
temperature and rearing temperature of the ADULT animal.  The function
consitits of two gaussian fits for the temperature domain and a 2nd
degree polynom for the p Values. In this case both the temperature domain
and the p-values are dependent on the rearing temperature.

@param x_p float random value between  0 and 1
@param x_t float Drosophila body temperature
@param d   list of floats containing the rearing temperature depending
           parameters
@return list of floats with new step duration values
```

4.3.1.3 def locomotionInterpolation.durFuncLarva (*x_p, x_t*)

```
This function calculates the fit of the step duration in dependens of the
temperature of the LARVAL animal (reared at 18°C). The function
consitits of two Gaussian fits for the temperature domain and two Gaussian
fits for the p Values.

@param x_p float random value between  0 and 1
@param x_t float Drosophila body temperature
@return list of floats with new step duration values
```

4.3.1.4 def locomotionInterpolation.gauss (*x, a, x0, sigma*)

```
Implementation of a Gaussian distribution

                       2
             (x - x0)
          - ------------
                       2
            (2.0*sigma)
f(x) = a*e

@param x     float x - value(s as list)
@param a     float factor before the gaussian
@param x0    float mu of the distribution
@param sigma float sigma of the distribution
@return list of floats with the results corresponding to the x-values
```

4.3.1.5 def locomotionInterpolation.hillDouble (*x*, *o*, *n*, *s*, *s2*, *b*)

```
Implementation of a Hill function

       /     n       \
       |    x        | /    b        \
f(x) = |b ------- + s| |-------- + s2|
       |   n    n    | |       n     |
       \  o  + x     / |    /x\      |
                       |1 + |-|      |
                       \    \o/      /

@param x     float x - value(s as list)
@param o     float see above
@param n     float see above
@param s     float see above
@param s2    float see above
@param b     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.6 def locomotionInterpolation.hillDown (*x*, *o*, *n*, *s*, *b*)

```
Implementation of the falling  slope of a hill equation

           b
f(x) = -------- + s2
              n
           /x\
       1 + |-|
           \o/


@param x     float x - value(s as list)
@param o     float see above
@param n     float see above
@param s     float see above
@param b     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.7 def locomotionInterpolation.hillUp (*x*, *o*, *n*, *s*, *b*)

```
Implementation of the rising slope of a hill equation

            n
           x
f(x) = b ------- + s
          n    n
         o  + x

@param x     float x - value(s as list)
@param o     float see above
@param n     float see above
@param s     float see above
@param b     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.8 def locomotionInterpolation.linear (*x*, *a*, *b*)

```
Implementation of a simple linear function

f(x) = a x + b

@param x     float x - value(s as list)
@param a     float see above -4.71*x^4+167.91*x^3-2568.55*x^2+9292.99*x+0.05
@param b     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.9 def locomotionInterpolation.poly2 (*x*, *a*, *b*, *c*)

```
Implementation of a 2nd degree polynom

          2
f(x) = a x  + b x + c

@param x     float x - value(s as list)
@param a     float see above
@param b     float see above
@param c     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.10 def locomotionInterpolation.poly3 (*x*, *a*, *b*, *c*, *d*)

```
Implementation of a 3rd degree polynom

          3      2
f(x) = a x  + b x  + c x + d

@param x     float x - value(s as list)
@param a     float see above
@param b     float see above
@param c     float see above
@param d     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.11 def locomotionInterpolation.poly4 (*x*, *a*, *b*, *c*, *d*, *e*)

```
Implementation of a 4th degree polynom

          4      3      2
f(x) = a x  + b x  + c x  + d x + e

@param x     float x - value(s as list)
@param a     float see above
@param b     float see above
@param c     float see above
@param d     float see above
@param e     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.12 def locomotionInterpolation.ratio (*x*, *a*, *b*)

```
Implementation of a simple ratio function

         a
f(x) = -----
       x + b

@param x     float x - value(s as list)
@param a     float see above
@param b     float see above
@return list of floats with the results corresponding to the x-values
```

4.3.1.13 def locomotionInterpolation.velFunc (*x_p*, *x_t*, *v*)

```
This function calculates the fit of the velocity in dependens of the
temperature and rearing temperature of the ADULT animal. The function
consitits of two gaussian fits for the temperature domain and a 2nd
degree polynom for the p Values. This 2nd degree polyom changes with the
rearing temperature.
```

```
@param x_p float random value between  0 and 1
@param x_t float Drosophila body temperature
@param v   list of floats containing the rearing temperature depending
           parameters
@return list of floats with new velocity values
```

4.3.1.14 def locomotionInterpolation.velFuncLarva (*x_p*, *x_t*)

```
This function calculates the fit of the velocity in dependens of the
temperature of the LARVAL animal (reared at 18°C). The function
consitits of a 4th degree polynom for the temperature domain and a 3rd
degree polynom for the p Values.

@param x_p float random value between  0 and 1
@param x_t float Drosophila body temperature
@return list of floats with new velocity values

                4           3            2
f(x) = (-4.71) x  + 167.91 x  - 2568.55 x  + 9292.99 x + 0.05
```

4.4 locomotionOnRawData Namespace Reference

Functions

- def pickDataSet
- def findValue

4.4.1 Function Documentation

4.4.1.1 def locomotionOnRawData.findValue (*t*, *p*, *dataSet*)

```
This is the central function that finds the closest combination of
original data and the momentary values in the simulation.

@param t       float Drosophila body temperature
@param p       float random number between 0 and 1
@param dataset list with either the original velocity or duration values
```

4.4.1.2 def locomotionOnRawData.pickDataSet (*rearTemp*, *probVec*)

```
This function picks the correct dataset for the rearing temperature.

@param rearTemp mixed can be int 18,25,30 or string 'larval'
@param probVec  list of list containing the data
@return velocity and durationlists with the data, will be empty if rearTemp
        is not as expected.
```

Chapter 5

Class Documentation

5.1 IGLOO.IGLOO Class Reference

Public Member Functions

- def __init__
- def resetTempTrace
- def stepFunc
- def move
- def updatePosition
- def updateAmbientTemp
- def drosoTbyConduction
- def simulateSingleFly
- def simulateFlyPopulation
- def calcHistogram
- def plotSingleTrace

  _______ ___ _______ _______ _______ ___ ___ ___ _______ # /\ == \ /\ \ /\ ___ \ /___ _\ /___ _\ /\ \ /\ "-. \ /\ ____\ # \ \ _-/ \ \ _____ \
  \ \/\ \ \/_/\ \/ \/_/\ \/ \ \ \ \ \ \ -.
- def plotHistogram
- def save4MatlabSingleTra

  _______ ___ ___ _______ ___ _______ # /\ ____\ /\ \ /\ \ /\ ____\ /\ \ /\ ___ \ # \ \ ___\ \ \ \ \ \ \ _____ \ \ ___\ \ \ \ \ \ \ \/\ \ # \ _\ \ \ _\ \
  ______\ \ \ ______\ \ \ _\ \ \ ______\ # \/_/ \/_/ \/______/ \/______/ \/_/ \/______/ #
- def save4TXTSingleTra
- def save4TXTPopulation
- def save4MatlabPopulation
- def save4TXTHistogram
- def save4MatlabHistogram
- def save4MatlabAll
- def make_txt_header

Public Attributes

- startPosition
- position
- gradientExt
- gradientDist
- sps
- walkDur
- rearingT

- simulationType
- ambientT
- drosoT
- degPerMM
- step
- histData
- selfTestData
- flyPop
- allData
- durData
- velData
- time
- tempTrace
- direction
- dParams
- bins

5.1.1 Constructor & Destructor Documentation

5.1.1.1 def IGLOO.IGLOO.__init__ (*self*, *startPos* = `25.`, *gradientExt* = `(12.,32.`, *gradientDist* = `50.`, *walkDur* = `300.`, *rearingT* = `25.`, *sps* = `50`, *simulationType* = `'interpolate'`)

```
This function intialises the monte carlo random walk class. Here most
of the simulation task will be done. Each fly is represented by 3 values
its position [mm], the environment temperature at this position [°C] and
its body temperature [°C]. The gradient is a one dimensional strip of
x mm length that has a linear temperature gradient with gradientExt as
the gradient extreme temperatures. The lower extreme temperature is
situated at 0 mm the hotter extreme temperature at the far end of the
gradient. The fly is further described by ots rearing temperature
(default: 25°C) and its preferred temperature (default: 21°C). The
original null model is a random walk.

@param startPos         float  default: 25
    start position of the fly in mm on the gradient
@param gradientExt      tuple  default: (12.,32.)
    temperature extremes of the linear gradient in °C
@param gradiantDist     float  default: 50.0
    total length of the gradient in mm
@param walkDur          float  default: 300.0
    total duration of the simulation in seconds
@param rearingT         float  default: 25.0
    rearing temperature of the fly in °C if you simulate on the
    original data only 18.0, 25.0 and 30.0 can be used. If instead of
    float rearingT is set as 'larval' larvae reared at 25°C are
    simulated.
@param sps              int    default: 50
    Samples per second. After the walking duration is reached. The whole
    trajectory will be resampled with this framerate.
@param simulationType   string default: 'interpolate'
    String that defines if the simulation is run on the original data
    set or on the interpolation functions. The later is much faster and
    allows to set the rearing temperature to any value between 18 and
    30. To run on the original data set this string to "onData"
```

5.1.2 Member Function Documentation

5.1.2.1 def IGLOO.IGLOO.calcHistogram (*self*, *tempFlag*)

```
This function calculates a  position histogram for every fly in the
population and normalises it to its surface. Afterwards it calculates
mean +/- SEM of all histograms and normalises again.
```

```
The result is saved in self.histData a tupel consisting of the
number of samples normalised to their total and the bins of the
histogram. The number of samples variable has two rows 1st is mean and
the 2nd is the SEM of each bin

@param tempFlag int if set to 3 it calculates ambient temperature if
            set to 4 it calculates body temperature
```

5.1.2.2 def IGLOO.IGLOO.drosoTbyConduction (*self*)

```
This function calculates the temperature change for animals in our TLM
model. It uses Newtons law for convection and models the Drosophila as an
3 by 0.5 mm cylinder walking in a 3  mm wide cylinder. Internally the
values given will be transferred to the correct dimensions. The
following values are set conductance:
air 0.0262 W/(m²*K)  David R. Lide (Hrsg.): CRC Handbook of Chemistry and Physics. 90. Auflage. (Internet-Vers
water 0.6 W/(m²*K)   https://de.wikipedia.org/wiki/Eigenschaften_des_Wassers#W.C3.A4rmeleitf.C3.A4higkeit

The formula for rate of heat flow is:      dQ               T1-T2
                              Q  = -- = lambda * A* -----
                                   dt                 D

where D is the wallt to wall thickness of the object lambda is its
conductance A its surface T1, T2 the temperature of the object and its
surroundings.

Drosophila surface'  as a cylinder would be for the cylinders hull:
2*pi*r*l and the two disks: (2*pi*r**2)*2. If the cylinder has an
r = 0.5mm and a length of l =2mm the resulting surface is 7.85 mm²

To calculate the temperature change we have to know the conductance of
our fly which we approximate as a cylinder of water lambda 0.6. A
cylinder of that size would have a mass of  1.57 mg

Model limits: Drosophila is a 1 times 2 mm cylinder consisting of
1.58 mg of water suspended in air.

The result is saved in self.drosoT
```

5.1.2.3 def IGLOO.IGLOO.make_txt_header (*self*)

```
All our txt output files have to record the environmental data of the simulation
as shown below. These are put in a single string variable and saved as the header
of the text file.

startPos
start position of the fly in mm on the gradient
gradientExt
temperature extremes of the linear gradient in °C
gradiantDist
total length of the gradient in mm
sps
Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
walkDur
total duration of the simulation in seconds
rearingT
rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
25°C are simulated.
simulationType
String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"
```

```
@return headStr a string carrying the above information and written into all
text files.
```

5.1.2.4 def IGLOO.IGLOO.move (*self*)

```
This is a subroutine of the step function. This function updates the
fly position. This can be done using the original data (simulation:
'onData') or interpolation functions (simulation: 'interpolated').

The results are saved in self.step and self.time
```

5.1.2.5 def IGLOO.IGLOO.plotHistogram (*self*, *fHandle* = -1)

```
This function calculates a temperature histogram and plots it as bar
plots to the figure handle provided by the user

@param fHandle  int default: -1 figure handle if set to default a new
        figure will be opend. Otherwise the figure with this
        handle will be cleared and the data plotted there.
```

5.1.2.6 def IGLOO.IGLOO.plotSingleTrace (*self*, *fHandle* = -1, *showNow* = False)

_______ __ _______ _______ _______ __ __ __ _______ # /\ == \ /\ \ /\ __ \ /__ _\ /__ _\ /\ \ /\ "-. \ /\ ___\ # \ \ _-/ \ \ _____ \ \ \/\ \ \ V_/\ V V_/\ V \ \ \ \ \ -.
\ \ \ __ \ # \ _\ \ ______\ \ ______\ \ _\ \ _\ \ _\ \ _\"_\ \ ______\ # V_/ V______/ V______/ V_/ V_/ V_/ V_/ V_/ V______/ #

```
This plots the last calculated trace as line plots. There are 3
subplots 1) Time vs Position 2) Time vs Ambient Temperature 3) Time
vs Body Temperature

@param fHandle  int default: -1 figure handle if set to default a new
        figure will be opend. Otherwise the figure with this
        handle will be cleared and the data plotted there.
```

5.1.2.7 def IGLOO.IGLOO.resetTempTrace (*self*)

```
This function resets the fly and gradient so that a new simulation trial
can begin, e.g. fly returns to original position and default temperature
time is reset to zero.

The results are saved in self.position .drosoT .time and .tempTrace

self.tempTrace is a nx4 vector, where column 1) time [s], 2) position
[mm], 3) ambient temperature [°C], 4) body temperature [°C]
```

5.1.2.8 def IGLOO.IGLOO.save4MatlabAll (*self*, *fPos*)

```
This function saves the following variables to a Matlab file:

@return lastTra is a nx4 vector, where column 1) time [s], 2) position
[mm], 3) ambient temperature [°C], 4) body temperature [°C] and
n is self.walkDur*self.sps
@return histData a tupel consisting of the number of samples
normalised to their total and the bins of the histogram. The
number of samples variable has two rows 1st is mean and the 2nd
is the SEM of each bin
@return time a vetor of self.walkDur*self.sps length holding the time
in seconds
```

```
@return flyPop a list of vectors nx3 vector, where column 1) position
[mm], 2) ambient temperature [°C], 3) body temperature [°C] and
n is self.walkDur*self.sps. Each fly is saved in a single entry
of the list
@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  fPos string the absolute position of the file you want to save
data to.
```

5.1.2.9 def IGLOO.IGLOO.save4MatlabHistogram (*self*, *fPos*)

```
This function saves the following variables to a Matlab file:

@return histData a tupel consisting of the number of samples
normalised to their total and the bins of the histogram. The
number of samples variable has two rows 1st is mean and the 2nd
is the SEM of each bin
@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  fPos string the absolute position of the file you want to save
data to.
```

5.1.2.10 def IGLOO.IGLOO.save4MatlabPopulation (*self*, *fPos*)

```
This function saves the following variables to a Matlab file:

@return time a vetor of self.walkDur*self.sps length holding the time
in seconds
@return flyPop a list of vectors nx4 vector, where column
1) position [mm], 2) ambient temperature [°C], 3) body temperature
[°C] and n is self.walkDur*self.sps. Each fly is saved in a single
entry of the list
@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
```

```
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  fPos string the absolute position of the file you want to save
data to.
```

5.1.2.11 def IGLOO.IGLOO.save4MatlabSingleTra (*self*, *fPos*)

______ __ __ ______ __ ______ # /\ ___\ /\ \ /\ \ /\ ___\ /\ \ /\ __ \ # \ \ __\ \ \ \ \ \ ____ \ \ __\ \ \ \ \ \ \/\ \ # \ _\ \ _\ \
_____\ \ _____\ \ _\ \ _____\ # \/_/ \/_/ \/_____/ \/_____/ \/_/ \/_____/ #

```
This function saves the following variables to a Matlab file:

@return lastTra is a nx4 vector, where column 1) time [s],
2) position [mm], 3) ambient temperature [°C], 4) body temperature
[°C] and n is self.walkDur*self.sps
@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  fPos string the absolute position of the file you want to save
data to.
```

5.1.2.12 def IGLOO.IGLOO.save4TXTHistogram (*self*, *fPos*)

```
This function saves the histogram data to disk. The resulting file has three
columns:
1) The middle temperature of each bin
2) The mean of the histograms of all flies / normalised to an integral of one
3) The standard error of the mean of the histograms of all flies

@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  fPos string the absolute position of the file you want to save
data to.
```

5.1.2.13 def IGLOO.IGLOO.save4TXTPopulation (*self*, *dirName*, *prefix* = ′ IGLOO′)

```
This function saves the following variables for each fly into a single
ASCII text file with 4 digits before and 5 after the point. :

@return lastTra is a nx4 vector, where column 1) time [s],
2) position [mm], 3) ambient temperature [°C], 4) body temperature
[°C] and n is self.walkDur*self.sps.   in a 4.5 float format
@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  dirName string the absolute position of the director you want to
save your single fly trace to.
@param  prefix string with the prefix for all fly trajectory files. Example:
if set to 'cantonS' and you simulated one hundred flies the files
will be called cantonS_0000.txt to cantonS_0099.txt default: 'IGLOO'
```

5.1.2.14 def IGLOO.IGLOO.save4TXTSingleTra (*self*, *fpos*, *trace* = ″)

```
This function saves the following variables to a txt file:

@return lastTra is a nx4 vector, where column 1) time [s],
2) position [mm], 3) ambient temperature [°C], 4) body temperature
[°C] and n is self.walkDur*self.sps.   in a 4.5 float format
@return startPos start position of the fly in mm on the gradient
@return gradientExt temperature extremes of the linear gradient in °C
@return gradiantDist total length of the gradient in mm
@return sps Samples per second. After the walking duration is reached.
The wholetrajectory will be resampled with this framerate.
@return walkDur total duration of the simulation in seconds
@return rearingT  rearing temperature of the fly in °C if you simulate
on the original data only 18.0, 25.0 and 30.0 can be used. If
instead of float rearingT is set as 'larval' larvae reared at
 25°C are simulated.
@return simulationType String that defines if the simulation is run on
the original data set or on the interpolation functions. The
later is much faster and allows to set the rearing temperature
to any value between 18 and 30. To run on the original data set
this string to "onData"

@param  fPos string the absolute position of the file you want to save
data to.
```

5.1.2.15 def IGLOO.IGLOO.simulateFlyPopulation (*self*, *flyN*, *plotFlag* = `False`)

```
This function wraps the simualteSingleFly function and itereates it for
the number of flies given by flyN.

@param flyN     int  number of flies to be simulated
@param plotFlag bool default: 0     if set to 1 all trajectories are
       plotted

The result is saved in self.flypop
```

5.1.2.16 def IGLOO.IGLOO.simulateSingleFly (*self*)

```
This function simulates the random walk of a single fly. For the
duration given by self.walkDur in [s] on the gradient defined
by self.gradientExt and self.gradientDist
To simulate more than one fly, please use simulateFlyPopulation

The result is saved in self.tempTrace

self.tempTrace is a nx4 vector, where column 1) time [s], 2) position
[mm], 3) ambient temperature [°C], 4) body temperature [°C] and n is
self.walkDur*self.sps
```

5.1.2.17 def IGLOO.IGLOO.stepFunc (*self*)

```
This function creates a step during random walk. The direction is
randomly determined. Velocity and step duration are determined by the
subroutine self.move()
```

5.1.2.18 def IGLOO.IGLOO.updateAmbientTemp (*self*)

```
This subroutine updates the ambient temperature to the spot where the
fly arrived after the position was updated. This is trivial.


The result is saved in self.ambientT
```

5.1.2.19 def IGLOO.IGLOO.updatePosition (*self*)

```
This function updates the position of the animal. This is important, as
the ends of the gradient are reflective, e.g.: The gradient is 50 mm
long the fly is at 47 mm and makes a 10 mm step towards the near end.
In this case the animal walk 3 mm to the near end and gets reflected
for 7 mm. The new fly position would be 43 mm.

The result is saved in self.position
```

5.1.3 Member Data Documentation

5.1.3.1 IGLOO.IGLOO.allData

5.1.3.2 IGLOO.IGLOO.ambientT

5.1.3.3 IGLOO.IGLOO.bins

5.1.3.4 IGLOO.IGLOO.degPerMM

5.1.3.5 IGLOO.IGLOO.direction

5.1.3.6 IGLOO.IGLOO.dParams

5.1.3.7 IGLOO.IGLOO.drosoT

5.1.3.8 IGLOO.IGLOO.durData

5.1.3.9 IGLOO.IGLOO.flyPop

5.1.3.10 IGLOO.IGLOO.gradientDist

5.1.3.11 IGLOO.IGLOO.gradientExt

5.1.3.12 IGLOO.IGLOO.histData

5.1.3.13 IGLOO.IGLOO.position

5.1.3.14 IGLOO.IGLOO.rearingT

5.1.3.15 IGLOO.IGLOO.selfTestData

5.1.3.16 IGLOO.IGLOO.simulationType

5.1.3.17 IGLOO.IGLOO.sps

5.1.3.18 IGLOO.IGLOO.startPosition

5.1.3.19 IGLOO.IGLOO.step

5.1.3.20 IGLOO.IGLOO.tempTrace

5.1.3.21 IGLOO.IGLOO.time

5.1.3.22 IGLOO.IGLOO.velData

5.1.3.23 IGLOO.IGLOO.walkDur

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

- /home/bgeurten/PyProjects/Igloo/IGLOO.py

Chapter 6

File Documentation

6.1 /home/bgeurten/PyProjects/Igloo/demo.py File Reference

Namespaces

- demo

Variables

- tuple demo.modObj

6.2 /home/bgeurten/PyProjects/Igloo/IGLOO.py File Reference

Classes

- class IGLOO.IGLOO

Namespaces

- IGLOO

6.3 /home/bgeurten/PyProjects/Igloo/locomotionInterpolation.py File Reference

Namespaces

- locomotionInterpolation

Functions

- def locomotionInterpolation.gauss
- def locomotionInterpolation.ratio
- def locomotionInterpolation.linear
- def locomotionInterpolation.poly2
- def locomotionInterpolation.poly3
- def locomotionInterpolation.poly4
- def locomotionInterpolation.hillUp

- def locomotionInterpolation.hillDown
- def locomotionInterpolation.hillDouble
- def locomotionInterpolation.velFunc
- def locomotionInterpolation.durFunc
- def locomotionInterpolation.velFuncLarva
- def locomotionInterpolation.durFuncLarva
- def locomotionInterpolation.calcParameters

6.4 /home/bgeurten/PyProjects/Igloo/locomotionOnRawData.py File Reference

Namespaces

- locomotionOnRawData

Functions

- def locomotionOnRawData.pickDataSet
- def locomotionOnRawData.findValue

Index