Package ‘afpt’

March 19, 2020

Title Tools for Modelling of Animal Flight Performance

Version 1.1.0.1

Date 2020-03-18

Encoding UTF-8

Author Marco KleinHeerenbrink [aut, cre],
Anders Hedenström [fnd]

Description Allows estimation and modelling of flight costs in animal (vertebrate) flight, implementing the aerodynamic power model described in Klein Heerenbrink et al. (2015) <doi:10.1098/rspa.2014.0952>. Taking inspiration from the program 'Flight', developed by Colin Pennycuick (Pennycuick (2008) ``Modelling the flying bird''. Amsterdam: Elsevier. ISBN 0-19-857721-4), flight performance is estimated based on basic morphological measurements such as body mass, wingspan and wing area. 'afpt' can be used to make predictions on how animals should adjust their flight behaviour and wingbeat kinematics to varying flight conditions.

URL https://github.com/MarcoKlH/afpt-r/

BugReports https://github.com/MarcoKlH/afpt-r/issues

Depends R(>= 3.0.0)

Imports graphics, stats

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Maintainer Marco KleinHeerenbrink <Marco.KleinHeerenbrink@zoo.ox.ac.uk>

Repository CRAN

Date/Publication 2020-03-19 17:40:02 UTC

R topics documented:

`air2ground` *Compute groundspeed*

Description

Computes groundspeed from airspeed and wind.

Usage

```
air2ground(airSpeed, windSpeed = 0, windDir = 0, climbAngle = 0)
```

Arguments

`airSpeed`	airspeed
`windSpeed`	windspeed
`windDir`	wind direction relative to (intended) track direction in degrees
`climbAngle`	climb angle in degrees

Value

`driftAngle`	Angle between airspeed and groundspeed
`groundSpeed`	Speed over ground

Author(s)

Marco Klein Heerenbrink

`altitude2density` *Compute density in International Standard Atmopshere*

Description

This function computes the air density at a specified altitude in the Troposphere of the International Standard Atmosphere.

Usage

```
altitude2density(altitude = 0)
```

Arguments

`altitude`	(geopotential) altitude in meters above sealevel.

Details

$\rho = \rho_0 (1 + a\frac{h}{T_0})^{-\frac{g_0}{Ra}+1}$ with ρ_0 = 1.225 kg/m3, a = -0.0065 K/m, h geopotential altitude in meters, g_0 = 9.80665 m/s2, and R = 287.1 J/Kg/K.

Value

Numerical value or array for the density in kg/m3

Author(s)

M. Klein Heerenbrink

References

U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C.

Examples

```
altitude <- seq(0,3000,100) # meters above sealevel
density <- altitude2density(altitude)
```

amplitude	*Flapping flight optimal amplitude*

Description

This function returns the angular peak amplitude of the flapping motion, optimized for minimum induced power for prescribed reduced frequency (kf), strokeplane angle (phi), and thrust-to-lift ratio (TL).

Usage

```
amplitude(kf, phi, TL)
```

Arguments

kf	Using f for wingbeat frequency, b for wingspan, and U for air speed: reduced frequency ($k_f = \frac{2\pi f b}{U}$); valid range between 1 and 6
phi	strokeplane angle in radians; valid range between 0 and 0.87 rad (50 deg)
TL	thrust requirement or the trust-to-lift ratio; valid range between 0 and 0.3

Value

Angular peak amplitude of the flapping motion in degrees.

Author(s)

Marco Klein Heerenbrink

References

Klein Heerenbrink, M., Johansson, L. C. and Hedenström, A. 2015 Power of the wingbeat: modelling the effects of flapping wings in vertebrate flight. *Proc. R. Soc. A* **471**, 2177 doi: 10.1098/rspa.2014.0952

See Also

computeFlappingPower

Examples

```
## reduced frequency
kf <- 2*pi*4/10 # 4 Hz at 10 m/s for 1m wing span
## strokeplane angle
phi <- 20*pi/180 # 20 degrees
## thrust ratio
TL <- 0.2
## wingbeat amplitude
theta <- amplitude(kf,phi,TL)
```

```
print(theta)
#   [1] 49.17679
```

`Bird` *Bird description*

Description

This function creates a bird description object, which is basically just a list with predefined variable names. It is named a bird object, but could also contain a description of a bat or insect. Minimal input required to construct a bird are body mass, wing span and wing area (or wing aspect ratio). Other required variables will then be given default values, or they will be estimated from allometric relations from literature.

Usage

```
Bird(massTotal, wingSpan, wingArea, ...)
```

Arguments

`massTotal`	Total mass that needs to be lifted in flight in kg
`wingSpan`	The maximum distance between the wingtips in meters
`wingArea`	The area of the fully stretched wings including the root area (left wing, right wing and area in between the wing roots)
`...`	Any other properties of a valid bird object (see details)

Details

This function sets up a list of properties of a bird. This definition of the bird is then used by the other functions in the package to estimate flight performance. At least three properties need to be specified: `massTotal`, `wingSpan` and `wingArea`. Either `wingSpan` or `wingArea` could be replaced by `aspectRatio`; the missing variable will then be computed. If no other properties are specified, default values will be used. Wingspan and wingarea should be measured from the maximally stretched out wing as described in *Pennycuick (2008)*: wingspan as the maximum distance between the wingtips and wingarea as the area from a trace including the root area (where the body is).

To specify custom properties, these can simply be added as additional arguments to the function. Note that `massTotal` needs to be the sum of `massLoad`, `massFat` and `massEmpty`. The function will recompute the total mass if the specified masses are inconsistent. Allometric relations use the empty weight. Muscle mass is part of the empty mass, and as such it is represented by `muscleMass` as a fraction. It is used in the estimation of the mechanical power available for flight (together with the muscle properties `coef.activeStrain` and `coef.isometricStress`). The variable `type` is used for selected allometric relationships that are specific to that particular group. Currently, `bodyFrontalArea` distinguishes between `'passerine'` and anything else and `basalMetabolicRate` distinguishes between `'passerine'`, `'seabird'`, `'bat'` and anything else.

`name`	String	Common name

`name.scientific`	Sring	Scientific name
`source`	String	Source for information
`massLoad`	Numeric	Additional mass the bird is carrying (kg); **0**
`massFat`	Numeric	Fat mass, i.e. fuel (kg); **0**
`massEmpty`	Numeric	Empty mass, i.e. total mass - fat mass - load mass (kg)
`muscleFraction`	Numeric	Fraction [0,1] of empty mass that makes up flight muscle; **0.17***
`type`	String	Type of bird `'other'`*, `'passerine'`*, `'seabird'`, `'bat'`
`bodyFrontalArea`	Numeric	Reference body frontal area used for body drag (m2)
`wingbeatFrequency`	Numeric	Typical wingbeat frequency (Hz)
`coef.profileDragLiftFactor`	Numeric	Coefficient for lift dependent profile drag; **0.03** (*Klein Heerenbrinkn et al. 2015*
`coef.bodyDragCoefficient`	Numeric	Drag coefficient related to body frontal area; **0.2****
`coef.conversionEfficiency`	Numeric	Efficiency Chemical to Mechanical energy; **0.23***
`coef.respirationFactor`	Numeric	Multiplyer for metabolic overhead respiration; **1.1***
`coef.activeStrain`	Numeric	Muscle duty cycle factor; **0.26***
`coef.isometricStress`	Numeric	Maximum force produced per cross section muscle (Pa); **400000** (upper limit fr
`basalMetabolicRate`	Numeric	Minimum energy consumption required for sustain life functions (W) *.

* as in Flight 1.25 (*Pennycuick 2008*)

** Large body of data supporting higher body drag coefficients (>0.2) than in Flight 1.25 (0.1), e.g. *Pennycuick et al. (1988)*, *Hedenström & Liechti (2001)*, *Henningsson & Hedenström (2011)* and *KleinHeerenbrink et al. (2016)*

Value

bird object with variables required by the various power estimating functions (e.g. `computeFlappingPower`).

Author(s)

Marco Klein Heerenbrink

References

Hedenström, A. & Liechti, F. (2001) Field estimates of body drag coefficient on the basis of dives in passerine birds. *J. Exp. Biol.* **204**, 1167–75.

Henningsson, P. & Hedenström, A. (2011) Aerodynamics of gliding flight in common swifts. *J. Exp. Biol.* **214**, 382–93. doi: 10.1242/jeb.050609

Klein Heerenbrink, M., Johansson, L. C. & Hedenström, A. (2015) Power of the wingbeat: modelling the effects of flapping wings in vertebrate flight. *Proc. R. Soc. A* **471**. doi: 10.1098/rspa.2014.0952

KleinHeerenbrink, M., Warfvinge, K. & Hedenström, A. (2016) Wake analysis of aerodynamic components for the glide envelope of a jackdaw (*Corvus monedula*). *J. Exp. Biol.* **219**, 1572–1581. doi: 10.1242/jeb.132480

Pennycuick, C. J. & Rezende, M. A. (1984) The specific power output of aerobic muscle, related to the power density of mitochondria. *J. Exp. Biol.*, **108**, 377–392.

Pennycuick, C. J., Obrecht III, H. H. & Fuller, M. R. (1988) Empirical estimates of body drag of large waterfowl and raptors. *J. Exp. Biol.* **135**, 253–264.

Pennycuick, C. J. (2008). *Modelling the flying bird.* Amsterdam, The Netherlands: Elsevier.

See Also

computeAvailablePower, computeChemicalPower, computeFlappingPower, computeBodyFrontalArea, etc.

Examples

```
myBird = Bird(
  massTotal = 0.215,
  wingSpan = 0.67,
  wingArea = 0.0652,
  name = 'jackdaw',
  type =  'passerine'
)
print(myBird)
```

---

climbing_birds *Climbing birds*

Description

Data extracted from *Hedenström & Alerstam 1992.*

Usage

```
data("climbing_birds")
```

Format

A data frame with 15 observations on the following 11 variables.

`number` a numeric vector

`name` a character vector

`name.scientific` a character vector

`massEmpty` a numeric vector

`massFat` a numeric vector

`wingSpan` a numeric vector

`wingAspect` a numeric vector

`wingbeatFrequency` a numeric vector

`climbRate` a numeric vector

`climbSpeed` a numeric vector

`climbAlitude` a numeric vector

Source

Hedenström A., Alerstam, T. (1992) Climbing performance of migrating birds as a basis for estimating limits for fuel-carrying capacity and muscle work. *J. Exp. Biol* **164** 19-38 http://jeb.biologists.org/content/164/1/19

Examples

```
data(climbing_birds)
climbingBirds <- Bird(climbing_birds)
```

---

computeAvailablePower *Compute available power*

Description

Estimation of maximum available power available from the muscles.

Usage

```
computeAvailablePower(bird, maxPowerAero, ...)
```

Arguments

bird	bird description object (see Bird)
maxPowerAero	maximum continuous power
...	optional arguments (none yet)

Details

Available power is determined as a muscle property. It is assumed that part of the muscles tissue is chemically active (mitochondria), providing the required ATP energy to the mechanically active tissue (myofibrils). The fraction of mitochondria determines the maximum sustainable power output from the muscles. With a higher fraction of myofibrils, the muscles can produce more power, but only in a short burst, until all ATP runs out.

If only a Bird object is provided, the function will assume that maximum power equals maximum continuous power (maxPowerAero). Otherwise, it will compute the burst maximum power.

Value

numeric value of mechanical power

Note

Available power is determined as a constant for the muscles. In reality the muscle power output depends on strainrate and stress, which in vertebrates are directly linked to wingbeat kinematics and aerodynamic loads.

Flight 1.25, the model of *Pennycuick (2008)* uses an isometric stress of 560 kN/m2. This is much higher than any measured value (*Pennycuick & Rezende 1984*). A more reasonable yet still very optimistic value would be 400 kn/m2, which is the default value assigned by the `Bird` constructor.

Author(s)

Marco Klein Heerenbrink

References

Pennycuick, C. J. & Rezende, M. A. (1984) The specific power output of aerobic muscle, related to the power density of mitochondria. *J. Exp. Biol.*, **108**, 377–392.

Pennycuick, C. J. (2008). *Modelling the flying bird.* Amsterdam, The Netherlands: Elsevier.

See Also

`Bird`

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

## for maximum continuous power
power.max <- computeAvailablePower(myBird)
print(power.max)
#   [1] 5.233528

## for specified maximum continuous power:
power.max.continuous <- 0.8*power.max
power.max.burst <- computeAvailablePower(myBird,power.max.continuous)
print(power.max.burst)
#   [1] 5.466625
```

`computeBodyFrontalArea` *Body frontal area from scaling relation*

Description

Body frontal area is a parameter that relates to body drag. This function estimates body frontal area based on empirical scaling relations with mass.

Usage

```
computeBodyFrontalArea(massEmpty, type = "other")
```

Arguments

`massEmpty`	empty body mass (in kg)
`type`	type of bird; available options are: "passerine" and "other")

Details

Passerine (*Hedenström and Rosén 2003*): $S_b = 0.0129m^{0.614}$

Other (*Pennycuick* et al. *1988*): $S_b = 0.00813m^{0.666}$

Value

Numeric value for the body frontal area.

Note

Body frontal area is used for the computation of body drag. Only use this value if it matches the used definition of the body drag coefficient.

Author(s)

Marco Klein Heerenbrink

References

Pennycuick, C. J., Obrecht III, H. H. and Fuller, M. R. (1988) Empirical estimates of body drag of large waterfowl and raptors. *J. Exp. Biol.* **135**, 253–264.

Hedenström, A. and Rosén, M. (2003) Body frontal area in passerine birds. *J. Avian Biol.* **34**, 159–162.

See Also

`Bird`

Examples

```
massEmpty <- 0.215 # kg
Sb <- computeBodyFrontalArea(massEmpty)
print(Sb)
#   [1] 0.002920751 # m2

massEmpty <- 0.215 # kg
birdType <- "passerine" #
Sb <- computeBodyFrontalArea(massEmpty,birdType)
print(Sb)
#   [1] 0.005020037 # m2
```

---

computeChemicalPower *Convert mechanical power to chemical power*

Description

Redundant after chemical power is now computed in all functions by default.

Computes the chemical power, i.e. the rate at which chemical energy is consumed, during flight. It takes into account the basal metabolic rate, and the energy needed by the flight muscles to provide the mechanical power required for flight.

Usage

```
## S3 method for class 'power.mechanical'
computeChemicalPower(power.mech, bird, ...)
## S3 method for class 'numeric'
computeChemicalPower(power.mech, bird, ...)
```

Arguments

`power.mech`	mechanical power (either numeric (W) or as an mechanical power object (class power.mechanical)
`bird`	object describing the relevant morphological parameters of the bird (or bat); this object should be created using the `Bird` constructor.
`...`	optional arguments (none yet)

Details

Chemical power is computed as

$$P_{\text{chem}} = R(\frac{P_{\text{mech}}}{\eta} + \text{BMR})$$

as described by *Pennycuick (2008)*. Here R is the respiration factor, η is the muscle conversion efficiency and BMR the basal metabolic rate, see `Bird`.

Value

Chemical power of same type as inpute `power.chem`.

Author(s)

Marco Klein Heerenbrink

References

Pennycuick, C. J. (2008). *Modelling the flying bird.* Amsterdam, The Netherlands: Elsevier.

See Also

`Bird`, `computeFlappingPower`, `mech2chem`, `chem2mech`

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

## for maximum continuous power
power.max <- computeAvailablePower(myBird)
print(power.max)
#   [1] 5.233528

## convert to chemical power
power.max.chem <- computeChemicalPower(power.max,myBird)
print(power.max.chem)
#   [1] 27.28913
```

`computeFlappingPower` *Calculate aerodynamic power flapping flight*

Description

The function calculates the aerodynamic power required for the specified bird (or bat) at the specified flight speed.

Usage

```
computeFlappingPower(bird,speed,...,frequency,strokeplane)
```

Arguments

`bird` object describing the relevant morphological parameters of the bird (or bat); this object should be created using the `Bird` constructor.

`speed` a numeric vector of the airspeed.

`...` optional arguments (see details)

`frequency` wingbeat frequency as single numeric value, a numeric vector matching the speed vector, a closure object returning a numeric value as a function of speed, or the character string 'recompute'. The latter will recompute the default frequency for the current flight condition (density) and the current total mass of the bird (assuming the frequency in bird is the default wingbeat frequency). If not provided, the function will look for a default wingbeat frequency in the bird object.

`strokeplane` angle of the strokeplane in degrees, as a single numeric value, a numeric vector matching the speed vector, a closure object describing the strokeplane angle as a function of speed. Alternatively providing character string "opt" will tell the function to optimize the strokeplane angle for minimum aerodynamic power.

Details

This function estimates aerodynamic power for a animal in forward flight based on morphology and wingbeat kinematics (*Klein Heerenbrink, 2015*). The model takes into account span reduction during the upstroke, which is typical for vertebrate forward flight. ... The minimal input required for the function is a description of the animal (as provided by the Bird constructor) and the speed(range) for which to compute the aerodynamic power. Distinct from other models, this model also requires wingbeat frequency and strokeplane angle. Higher wingbeat frequency tends to lower the induced power, but it may increase profile power. If no wingbeat frequency is provided, the function will use the reference wingbeat frequency from the bird object. Otherwise the user can specify values (either as vectors or as closure object). The user can provide additional optional arguments:

`bodyDragCoefficient` single numeric value, a numeric vector matching the speed vector, or a closure object as a function of speed. If not provided, the function will look for a default value in the bird object.

`addedDrag` single numeric value or a numeric vector matching the speed vector. This represents additional "drag" (in Newtons) that must be overcome (e.g. during climb).

`flightcondition` object describing the atmospheric conditions (density, viscosity, gravity).

Aerodynamic model: `computeFlappingPower` first computes the drag components for non-flapping flight:

$$D_{\mathrm{ind}} = \frac{L^2}{q\pi b^2}$$

$$D_{\mathrm{pro,0}} = C_{D_{\mathrm{pro,0}}} q S$$

$$D_{\mathrm{pro,2}} = k_p \frac{L^2}{qS}$$

$$D_{\mathrm{par}} = C_{D_{\mathrm{b}}} q S_{\mathrm{b}} + D_{\mathrm{added}}$$

which combine to the non-flapping thrust requirement $T_0 = \sum D_{<>}$. Here $q = \frac{1}{2}\rho U^2$ is the dynamic pressure depending on density (ρ) and speed (U). To account for how flapping the wings affects the drag on the wings, `computeFlappingPower` computes factors $f_{D_{\text{ind}}}$, $f_{D_{\text{pro},0}}$ and $f_{D_{\text{pro},2}}$, which are functions of the strokeplane angle and the (reduced) wingbeat frequency. These factors relate to the returned drag factors `kD.ind`, `kD.pro0` and `kD.pro2` through

$$k_{D,<>} = 1 + f_{D,<>}\frac{T}{L}$$

The actual drag in flapping flight is found by multiplying each non-flapping drag component with its respective drag factor. This means that the actual thrust requirement (thrust ratio T/L) can be computed as

$$\frac{T}{L} = \frac{T_0}{L - f_{D\text{ind}}D_{\text{ind}} - f_{D\text{pro},0}D_{\text{pro},0} - f_{D\text{pro},2}D_{\text{pro},2}}$$

Finally, `computeFlappingPower` computes the power factors in a similar way to the drag factors (i.e. $k_{P,i} = 1 + f_{P,i}\frac{T}{L}$, with $f_{P,i}$ functions of strokeplane angle and wingbeat frequency). The total aerodynamic power is then computed as

$$P = k_{P\text{ind}}D_{\text{ind}}U + k_{P\text{pro},0}D_{\text{pro},0}U + k_{P\text{pro},2}D_{\text{pro},2}U + D_{\text{par}}U$$

Wingbeat optimization: The underlying numerical model that is represented by functions $f_{D,i}$ and $f_{P,i}$, has optimised the flapping amplitude for minimum induced power. This means `computeFlappingPower` implicitly optimizes flapping amplitude, which is the value `amplitude` returned in the output.

`computeFlappingPower` takes strokeplane angle as input. The underlying numerical model has only explored strokeplane angles over a range of 0 (vertical) to 50 degrees, the latter being defined as having the down-stroke moving forward. In many cases it will be possible to find a strokeplane angle for which the total aerodynamic power is minimal. At high speeds this optimum will be for a vertical strokeplane while at lower speeds it will be more horizontal. By passing `strokeplane="opt"` as an argument to `computeFlappingPower`, it will try to numerically find the optimal strokeplane angle, using the function `optimize`.

Value

A data.frame including elements

`speed`	specified speed for which power is computed.
`power`	total aerodynamic power.
`power.chem`	total chemical power.
`strokeplane`	used strokeplane angle (either specified or optimized).
`amplitude`	wingbeat amplitude (implicitly optimized for minimum induced power).
`frequency`	wingbeat frequency (specified).
`flags.redFreqLo`	TRUE if reduced frequency too low (<1; outside model range).
`flags.redFreqHi`	TRUE if reduced frequency too high (>6; outside model range).
`flags.thrustHi`	TRUE if thrust requirement too high (>0.3; outside model range).
`flags.speedLo`	TRUE if speed is too low (invalidating the forward flight assumption).
`kD.ind`	induced drag factor

kD.pro0	zero lift profile drag factor
kD.pro2	lift dependent profile drag factor
kP.ind	induced power factor
kP.pro0	zero lift profile power factor
kP.pro2	lift dependent profile power factor
CDpro0	used zero lift profile drag coefficient (laminar boundary layer friction)
ReynoldsNumber	mean chord Reynolds number
Dnf.ind	non-flapping induced drag (N)
Dnf.pro0	non-flapping zero lift profile drag (N)
Dnf.pro2	non-flapping lift dependent profile drag (N)
Dnf.par	non-flapping parasitic drag (including body drag and apparent drag due to climbing)
L	lift (N)

Note

This model aims to predict the optimal flight performance for a bird. Particularly, the induced drag and induced power assume an ideal load distribution over the wing equivalent to the elliptical lift distribution for non-flapping wings. This means that induced power will typically be underestimated.

Author(s)

Marco Klein Heerenbrink

References

Klein Heerenbrink, M., Johansson, L. C. and Hedenström, A. (2015) Power of the wingbeat: modelling the effects of flapping wings in vertebrate flight. *Proc. R. Soc. A* **471**, 2177 doi: 10.1098/rspa.2014.0952

See Also

Bird, amplitude, fD.ind, fD.pro0, fD.pro2, fP.ind, fP.pro0, fP.pro2

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

## define a speed range
speedrange <- seq(5,14,length.out=5)
```

```
## compute aerodynamic power for that speed range:
Paero <- computeFlappingPower(myBird,speedrange)
print(Paero[c("speed","power","frequency","strokeplane")])
#   speed    power frequency strokeplane
# 1  5.00 2.789751  5.948083    46.56887
# 2  7.25 2.129466  5.948083    31.89129
# 3  9.50 2.203773  5.948083    22.51896
# 4 11.75 2.740763  5.948083    16.49120
# 5 14.00 3.673714  5.948083    12.09174

## prescribe strokeplane angle:
Paero <- computeFlappingPower(myBird,speedrange,strokeplane=20)
print(Paero[c("speed","power","frequency","strokeplane")])
#   speed    power frequency strokeplane
# 1  5.00 2.950259  5.948083          20
# 2  7.25 2.141581  5.948083          20
# 3  9.50 2.204132  5.948083          20
# 4 11.75 2.741335  5.948083          20
# 5 14.00 3.676224  5.948083          20

## prescribe frequency as a function of speed:
funFrequency = function(U){19.8 - 4.7*U + 0.45*U^2 - 0.0138*U^3}
Paero <- computeFlappingPower(myBird,speedrange,frequency=funFrequency,strokeplane='opt')
print(Paero[c("speed","power","frequency","strokeplane")])
#   speed    power frequency strokeplane
# 1  5.00 2.810431  5.825000    46.16223
# 2  7.25 2.356278  4.119247    25.99702
# 3  9.50 2.390251  3.930725    17.94304
# 4 11.75 2.860463  4.316291    14.52910
# 5 14.00 3.794431  4.332800    11.70058

## examine effect of frequency for a single airspeed:
speedrange <- rep(10,5) #  repeated speed
freqrange <- seq(3,10,length.out=5) #  frequency range
Paero <- computeFlappingPower(myBird,speedrange,frequency=freqrange,strokeplane='opt')
print(Paero[c("speed","power","frequency","strokeplane")])
#   speed    power frequency strokeplane
# 1    10 2.681028      3.00    13.87797
# 2    10 2.367982      4.75    18.90949
# 3    10 2.263765      6.50    21.52433
# 4    10 2.219739      8.25    21.71519
# 5    10 2.200852     10.00    20.18503
```

`computeFlightPerformance` *Compute characteristics of a power curve*

Description

This function calculates the basic characteristic flight speeds for `bird`.

Usage

```
computeFlightPerformance(bird, ..., length.out=10)
```

Arguments

`bird`	description of the bird or bat, constructed using the `Bird` function
`...`	various optional arguments that are passed on to other functions; see details
`length.out`	length of calculated power curve; set `length.out=0` to not compute a power curve

Details

Optional arguments can be provided through `...`. These can be arguments of `computeFlappingPower`, e.g. `strokeplane`, `frequency`, etc., or arguments for `findMaximumRangeSpeed`, e.g. `windSpeed` and `windDir`. The latter will only affect the outcome of the maximum range speed, and should perhaps not be analysed through the current function...

Value

`birdWSName`	variable name in work-space of the bird object
`bird`	bird object
`table`	table with characteristic speeds
`maxClimb`	table with climb performance
`powercurve`	power curve from minimum to maximum speed of length `lenght.out`

Author(s)

Marco Klein Heerenbrink

References

Klein Heerenbrink, M., Johansson, L. C. and Hedenström, A. (2015) Power of the wingbeat: modelling the effects of flapping wings in vertebrate flight. *Proc. R. Soc. A* **471**, 2177 doi: 10.1098/rspa.2014.0952

See Also

`Bird`, `computeFlappingPower`

Examples

```
## Define a bird:
myBird = Bird(
    name = "Jackdaw",
    name.scientific = "Corvus monedula",
    massTotal = 0.215, #  (kg) total body mass
    wingSpan = 0.67, #  (m) maximum wing span
    wingArea = 0.0652, #  (m2) maximum wing area
```

```
    type = "passerine"
)

## simplest performance calculation
performance.myBird <- computeFlightPerformance(myBird)
performance.myBird
# Name: Jackdaw
# Sc. name: Corvus monedula
# Bird definitions: NA
#               speed power.aero power.chem strokeplane amplitude
# minimumSpeed  2.706      5.234      27.29        49.9      51.3
# minimumPower  8.031      2.093      12.27        28.1      34.5
# maximumRange 11.025      2.523      14.33        18.2      36.7
# maximumSpeed 16.590      5.235      27.29         6.8      50.2
# Maximum climb performance:
#                  speed power.aero power.chem strokeplane amplitude climbRate
# maximumClimbRate  8.89      5.234      27.29        24.5      53.9      1.18
# Minimized migration time:
#              speed speed.migration power.aero power.chem power.dep strokeplane amplitude
# minimumTimeSpeed 11.75         1.962     2.741     15.37    3.081      16.49    38.04

## Not run:  # computationally intensive
## optimize strokeplane angle and use speed dependent frequency
funFrequency = function(U){19.8 - 4.7*U + 0.45*U^2 - 0.0138*U^3}
performance.myBird <- computeFlightPerformance(myBird,strokeplane='opt',frequency=funFrequency)
performance.myBird
# Name: Jackdaw
# Sc. name: Corvus monedula
# Bird definitions: NA
#               speed power.aero power.chem strokeplane amplitude
# minimumSpeed  2.293      5.229      27.27        49.9      43.8
# minimumPower  8.192      2.319      13.35        21.6      42.8
# maximumRange 11.463      2.775      15.53        14.9      44.3
# maximumSpeed 16.088      5.233      27.29         8.3      64.5
# Maximum climb performance:
#                  speed power.aero power.chem strokeplane amplitude climbRate
# maximumClimbRate  8.89      5.234      27.29        24.5      53.9      1.18
# Minimized migration time:
#              speed speed.migration power.aero power.chem power.dep strokeplane amplitude
# minimumTimeSpeed 12.07         1.905     2.964     16.43    3.081      14.13    45.13

## plot variation of speed, power and flapping kinematics
plot(performance.myBird$powercurve[c('speed','power.aero','strokeplane','frequency','amplitude')])

## End(Not run) # end dontrun

## plot power factors
plot(performance.myBird$powercurve[c('speed','power.aero')])
plot(performance.myBird$powercurve[c('speed','kP.ind')])
plot(performance.myBird$powercurve[c('speed','kP.pro0')])
plot(performance.myBird$powercurve[c('speed','kP.pro2')])
```

`fDfPfunctions` *Coefficient for thrust dependency of drag and power factors*

Description

Computes the thrust requirement dependency factor for drag and power factors in flapping flight based on reduced frequency (`kf`) and strokeplane angle (`phi`).

Usage

```
fD.ind(kf, phi)
fD.pro0(kf, phi)
fD.pro2(kf, phi)
fP.ind(kf, phi)
fP.pro0(kf, phi)
fP.pro2(kf, phi)
```

Arguments

	Using f for wingbeat frequency, b for wingspan, and U for air speed:
`kf`	reduced frequency ($k_f = \frac{2\pi f b}{U}$); valid range between 1 and 6
`phi`	strokeplane angle in radians; valid range between 0 and 0.87 rad (50 deg)

Details

Flapping of the wings alters the drag components on the wing. A drag component in flapping flight can be related to the drag component in non-flapping flight as $D = k_D D'$. The factor k_D depends on reduced frequency k_f, strokeplane angle ϕ and the thrust-to-lift ratio T/L: $k_D = 1 + f_D(k_f, \phi)\frac{T}{L}$. Functions `fD.ind`,`fD.pro0` and `fD.pro2` compute $f_D(k_f, \phi)$ for induced drag, zero lift profile drag and lift dependent profile drag, respectively.

Similarly, the flapping power components can be computed as: $P = k_P D' U$, again with $k_P = 1 + f_P(k_f, \phi)\frac{T}{L}$. Functions `fP.ind`,`fP.pro0` and `fP.pro2` compute $f_P(k_f, \phi)$ for induced power, zero lift profile power and lift dependent profile power, respectively.

Value

Numeric value

Note

Thrust requirement is the sum of all drag components in flapping flight divided by the lift. This means the thrust requirement itself is a function of the values of f_D.

Author(s)

Marco Klein Heerenbrink

References

Klein Heerenbrink, M., Johansson, L. C. and Hedenström, A. 2015 Power of the wingbeat: modelling the effects of flapping wings in vertebrate flight. *Proc. R. Soc. A* **471**, 2177 doi: 10.1098/rspa.2014.0952

See Also

computeFlappingPower

Examples

```
## reduced frequency
kf <- 2*pi*4/10 # 4 Hz at 10 m/s

## strokeplane angle
phi <- 20*pi/180 # 20 degrees

## thrust ratio
TL <- 0.2

## induced drag factor:
fDind <- fD.ind(kf,phi)
kDind <- 1 + fDind*TL
print(kDind)
#   [1] 1.623659

## zero lift drag factor:
fDpro0 <- fD.pro0(kf,phi)
kDpro0 <- 1 + fDpro0*TL
print(kDpro0)
#   [1] 1.014899

## lift dependent profile drag factor:
fDpro2 <- fD.pro2(kf,phi)
kDpro2 <- 1 + fDpro2*TL
print(kDpro2)
#   [1] 1.511107

## induced power factor:
fPind <- fP.ind(kf,phi)
kPind <- 1 + fPind*TL
print(kPind)
#   [1] 1.996891

## zero lift power factor:
fPpro0 <- fP.pro0(kf,phi)
kPpro0 <- 1 + fPpro0*TL
print(kPpro0)
#   [1] 1.076046

## lift dependent profile power factor:
fPpro2 <- fP.pro2(kf,phi)
```

```
kPpro2 <- 1 + fPpro2*TL
print(kPpro2)
#   [1] 1.811983
```

---

findMaximumClimbRate *Find maximum climb rate*

Description

Numerically find the maximum attainable climb rate.

Usage

```
findMaximumClimbRate(bird, maximumPower, speed, ...)
```

Arguments

bird	bird description object (see Bird)
maximumPower	numeric value for maximum available mechanical power
speed	airspeed for which to compute the maximum climbrate
...	optional arguments for computeFlappingPower

Details

The function searches for a climb angle between -90 and 90 degrees that matches the specified maximum power available. If no speed provided, the function will also find the optimal airspeed for maximum climbrate.

Value

Data frame of class power.mechanical

speed	airspeed either prescribed or optimized for maximum climbrate
power	aerodynamic (mechanical) power matching maximum power
...	see computeFlappingPower for other variables
climbAngle	angle between flightpath and horizontal plane in degrees
climbRate	rate of vertical climb

Note

The function uses climb angle, rather than climb rate, in the search algorithm, to ensure that climb rate is always less than the airspeed (i.e. in a vertical climb the climb rate will simply equal airspeed). The actual climb rate is maximized by maximizing the product of climb angle and airspeed. However, in practice, the airspeed for best climb rate will be close to the minimum power airspeed, where the power margin is largest.

Author(s)

Marco Klein Heerenbrink

See Also

uniroot

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

## maximum power available:
Paero.available <- computeAvailablePower(myBird)

climbSpeed <- 8 #  airspeed during climb

## find maximum climbrate:
Paero.climb <- findMaximumClimbRate(myBird,Paero.available,climbSpeed)
print(Paero.climb[c('speed','amplitude','frequency','climbRate')])
#   speed amplitude frequency climbRate
# 1     8  54.84965  5.948083  1.162002
```

---

findMaximumPowerSpeed *Finds speed for which power required equals maximum available power*

Description

Numerically find the airspeed for which required power equals `maximumPower`.

Usage

```
findMaximumPowerSpeed(bird, maximumPower, lower, upper, ...)
```

Arguments

`bird`	bird description object (see `Bird`)
`maximumPower`	numeric value for maximum available mechanical power
`lower`	lower bound for search range airspeed (m/s)
`upper`	upper bound for search range airspeed (m/s)
`...`	optional arguments to `computeFlappingPower`

Details

Prepares arguments for a call to uniroot. The function searches for an airspeed between lower and upper that matches the specified maximum power available.

Value

Data frame

speed	airspeed for which power matches maximum power
power	aerodynamic (mechanical) power matching maximum power
power.chem	aerodynamic (mechanical) power matching maximum power
strokeplane	optimized or prescribed strokeplane angle in degrees (from vertical)
amplitude	optimized peak amplitude in degrees (see amplitude)
...	see computeFlappingPower for other variables

Note

Typically this function would be used to find the maximum speed, but may in some cases also be used for the minimum flight speed. However, note that the low speed limit is likely limited by other constraints as well (e.g. stall speed).

Author(s)

Marco Klein Heerenbrink

See Also

uniroot

Examples

```
## Define a bird:
myBird <- Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

Paero.available  <- computeAvailablePower(myBird)

## find maximum speed:
Vmin <- 5
Vmax <- 30
Paero.maxSpeed <- findMaximumPowerSpeed(myBird,Paero.available,Vmin,Vmax)
print(Paero.maxSpeed[c('speed','power','amplitude','strokeplane','frequency')])
#      speed    power amplitude strokeplane frequency
# 1 16.58797 5.233459  50.22762    6.812345  5.948083
```

findMaximumRangeSpeed *Find maximum range speed*

Description

This function performs a numerical optimization to find the airspeed for which $\frac{P}{U}$ is minimum. For this it uses the function `optimize`.

Usage

```
findMaximumRangeSpeed(bird,lower=NULL,upper=NULL,windSpeed=0,windDir=0,...)
```

Arguments

`bird`	bird description object (see `Bird`)
`lower`	lower speed limit (optional)
`upper`	upper speed limit (optional)
`windSpeed`	wind magnitude (in m/s; optional)
`windDir`	wind direction (in degrees; optional)
`...`	optional arguments: `climbAngle` (in degrees), and optional arguments for `computeFlappingPower`.

Details

This function performs a numerical optimization to find the airspeed for which $\frac{P}{U}$ is minimum. For this it uses the function `optimize`. This airspeed is searched for between `lower` and `upper` (if not provided, it will make a guess based on `bird`). Flying in wind changes the ground speed, and therefore the optimum flight speed for maximum range. This can be taken into account through the optional arguments for wind magnitude (`windSpeed` in m/s) and wind direction relative to the track direction (`windDir` in degrees; `windDir = 0` tail wind); see e.g. *Liechti et al. 1994*.

Value

Returns data.frame (power.chemical) of flight performance at maximum range speed for `bird`.

Author(s)

Marco Klein Heerenbrink

References

Liechti, F., Hedenström, A. and Alerstam, T. (1994). Effects of Sidewinds on Optimal Flight Speed of Birds. *J. Theor. Biol.* **170**, 219–225.

See Also

`computeChemicalPower`, `computeFlappingPower`

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

maximumRangeSpeed.chem <- findMaximumRangeSpeed(myBird)
maximumRangeSpeed.chem[c('speed','power','strokeplane','amplitude','frequency')]
#      speed    power strokeplane amplitude frequency
# 1 11.02543 14.32754    18.17729  36.69311  5.948083

maximumRangeSpeed.chem.wind <- findMaximumRangeSpeed(
  myBird,
  windSpeed = 5,
  windDir = 90
)
maximumRangeSpeed.chem.wind[c('speed','power','strokeplane','amplitude','frequency')]
#      speed    power strokeplane amplitude frequency
# 1 11.81974 15.47758    16.33727  38.17508  5.948083
```

`findMinimumPowerSpeed` *Find speed for minimum power*

Description

.

Usage

```
findMinimumPowerSpeed(bird, lower, upper, ...)
```

Arguments

`bird`	bird description object (see `Bird`)
`lower`	lower speed limit (optional)
`upper`	upper speed limit (optional)
`...`	optional arguments for `computeFlappingPower()`

Details

This is pretty much just a call to `optimize`.

Value

powercurve object (`funCalcPower` evaluated for the minimum speed)

Author(s)

Marco Klein Heerenbink

See Also

`optimize`

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

minimumPowerSpeed.aero <- findMinimumPowerSpeed(myBird)
minimumPowerSpeed.aero[c('speed','power','strokeplane','amplitude','frequency')]
#      speed    power strokeplane amplitude frequency
# 1 8.030022 2.092976    28.14514  34.52719  5.948083
```

---

findMinimumTimeSpeed *Find speed for migration time minimization*

Description

This function performs a numerical optimization to find the airspeed for which $\frac{P+P_{\mathrm{dep}}}{U}$ is minimum..

Usage

```
findMinimumTimeSpeed(bird,
  EnergyDepositionRate=1.5*bird$basalMetabolicRate,
  lower=NULL,upper=NULL,
  windSpeed=0,windDir=0,...)
```

Arguments

`bird`	bird description object (see `Bird`)
`EnergyDepositionRate`	The rate at which the bird accumulates energy at stopover sites
`lower`	lower speed limit (optional)
`upper`	upper speed limit (optional)
`windSpeed`	wind magnitude (in m/s; optional)
`windDir`	wind direction (in degrees; optional)
`...`	optional arguments: `climbAngle` (in degrees), and optional arguments for `computeFlappingPower`.

Details

This function performs a numerical optimization to find the airspeed that minimizes the combination of flight time and time required to (re)gain the energy reserves to cover the flight cost. If the bird would fly faster, it would need to spend more time refueling. If it flew slower, the reduced refueling time that comes with the lower cost of transport does not offset the longer flight time. Mathematically this problem works out as minimizing $\frac{P+P_{\mathrm{dep}}}{U}$ *Hedenström 1998*, which is technically the same optimization as for the maximum range speed (see details `findMaximumRangeSpeed`). The default energy deposition rate, the rate at which a bird accumulates energy during a stopover, is set to 1.5 times the basal metabolic rate (*Lindström 1991*).

Value

Returns data.frame (power.chemical) of flight performance at maximum range speed for `bird`.

Author(s)

Marco Klein Heerenbrink

References

Lindström, Å. (1991) Maximum fat deposition rates in migrating birds. *Ornis Scand.* **22**, 12-19 (doi:10.2307/3676616)

Hedenström, A. & Alerstam, T. (1997) Optimum fuel loads in migratory birds: distinguishing between time and energy minimization. *J. Theor. Biol.* **189**, 227–34. (doi:10.1006/jtbi.1997.0505)

Hedenström, A. & Alerstam, T. (1998) How fast can birds migrate? *J. Avian Biol.* **29**, 424-432. (doi:10.2307/3677161)

See Also

`computeChemicalPower`, `computeFlappingPower`

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

minimumTimeSpeed <- findMinimumTimeSpeed(myBird,1.5*myBird$basalMetabolicRate)
minimumTimeSpeed[c('speed','speed.migration',
    'power','power.chem','power.dep',
    'strokeplane','amplitude','frequency')]
#     speed speed.migration   power power.chem power.dep strokeplane amplitude frequency
#  11.74944        1.962213 2.74058   15.36634  3.080752    16.49244  38.03366  5.948083
```

PowerToFroMechChem *Convert between mechanical and chemical power*

Description

Functions convert between mechanical and chemical power

Usage

```
mech2chem(power.mech,bird,...)
chem2mech(power.chem,bird,...)
```

Arguments

power.mech	Numerical value for mechanical power
power.chem	Numerical value for chemical power
bird	object describing the relevant morphological parameters of the bird (or bat); this object should be created using the Bird constructor.
...	optional arguments (none yet)

Details

Chemical power is computed as

$$P_{\text{chem}} = R(\frac{P_{\text{mech}}}{\eta} + \text{BMR})$$

as described in *Pennycuick 2008*. Here R is the respiration factor, η is the muscle conversion efficiency and BMR the basal metabolic rate, see Bird.

Mechanical power is simply calculated inversely:

$$P_{\text{mech}} = \eta(\frac{P_{\text{chem}}}{R} - \text{BMR})$$

Value

Numerical value of either chemical power (`mech2chem()`) or mechanical power (`chem2mech()`).

Author(s)

Marco Klein Heerenbrink

References

Pennycuick, C. J. (2008). *Modelling the flying bird.* Amsterdam, The Netherlands: Elsevier.

See Also

computeChemicalPower

Examples

```
## Define a bird:
myBird = Bird(
  massTotal = 0.215, #  (kg) total body mass
  wingSpan = 0.67, #  (m) maximum wing span
  wingArea = 0.0652, #  (m2) maximum wing area
  type = "passerine"
)

## define a speed range
speedrange <- seq(5,14,length.out=5)

## compute aerodynamic power for that speed range:
Paero <- computeFlappingPower(myBird,speedrange)
Pchem <- Paero
Pchem$power <- mech2chem(Paero$power,myBird)
print(Pchem[c("speed","power","frequency","strokeplane")])
#   speed    power frequency strokeplane
# 1  5.00 15.60151  5.948083    46.56887
# 2  7.25 12.44362  5.948083    31.89129
# 3  9.50 12.79900  5.948083    22.51896
# 4 11.75 15.36721  5.948083    16.49120
# 5 14.00 19.82915  5.948083    12.09174

Pmech <- Pchem
Pmech$power <- chem2mech(Pchem$power,myBird)
print(Pmech[c("speed","power","frequency","strokeplane")])
#   speed    power frequency strokeplane
# 1  5.00 2.789751  5.948083    46.56887
# 2  7.25 2.129466  5.948083    31.89129
# 3  9.50 2.203773  5.948083    22.51896
# 4 11.75 2.740763  5.948083    16.49120
# 5 14.00 3.673714  5.948083    12.09174
```

`reducedFrequency` *Function to compute reduced frequency*

Description

This function computes the reduced frequency based on `wingSpan` (b), wingbeat `frequency` (f) and `speed` (U): $k_f = \frac{2\pi b f}{U}$.

Usage

```
reducedFrequency(wingSpan, frequency, speed)
```

Arguments

`wingSpan`	Tip-to-tip distance of the fully spread wing (m)

frequency	Wingbeat frequency (1/s)
speed	Airspeed (m/s)

Details

This parameter is the ratio of the wingspan to the wavelength of the convected wake. For very high reduced frequencies, the wake of one wingbeat is relatively short compared to the wingspan, meaning that previous wingbeats have a large influence on the aerodynamics of the current wingbeat. When the reduced frequency is low, there is relatively little interaction between the wingbeats.

This wingspan based reduced frequency should not be confused with the chord based (or half chord) based reduced frequency. That definition serves a similar function, however, it relates to the effect of unsteadyness on the aerofoil (i.e. it is somewhat like the 2D equivalent).

Another related parameter of unsteadyness, often mentioned in relation to animal flight, is the Strouhal number, representing the ratio of the amplitude of the wingbeat to the wavelength of the wake. This term is historically related to vortex shedding.

Value

Numeric value

Author(s)

Marco Klein Heerenbrink

References

Klein Heerenbrink, M., Johansson, L. C. and Hedenström, A. 2015 Power of the wingbeat: modelling the effects of flapping wings in vertebrate flight. *Proc. R. Soc. A* **471**, 2177 doi: 10.1098/rspa.2014.0952

See Also

computeFlappingPower

Examples

```
kf <- reducedFrequency(
  wingSpan = 0.67,
  frequency = 4,
  speed = 9
)
kf
# [1] 1.870993
```

Index