Scilib-GEO

- Geometry
  - atan2 — Grid bearing t_0 used in geodetic coordinate systems, `atan()` for 4 quadrants in the intervall (0, 2π)
  - sphere — Creates a point cloud (mesh) of a spherical surface.
  - unitVec — Function returns the unit vector of a given vector or point in carthesian coordinates.
- GPS
  - alm_parse_yumafile — Reads almanac data from files in yuma-format (experimental).
  - sp3_parse_file — Reads orbit- and clockdata from a SP3-c formated file from the IGS.
  - sp3c_parse_header — Parses the headerdata of a SP3c-file (lines 1-22).
- Physical Geodesy
  - bl2gk — Transformation: Geodaetic ellipsidic coordinates to Gauß-Krüger-coordinates (convention of Germany)
  - eph2xyz — Function to calculate satellite coordinates (xyz) from the ephemerides of the satellite.
  - gk2bl — Transformation: Gauß-Krüger-coordinates (convention of Germany) to geodaetic ellipsoidic coordinates
  - gmf — Calculating the "global mapping function" (GMF).
  - kepler — Calculates the excentric anomaly E (Kepler element).
  - par_nutation — Calculates the parameter of the nutation.
  - par_praezession — Calculates the parameter of the Precession.
  - tropKor_sm — Calculates the tropospheric adjustment with assumed default values (Saastamoinen).
  - xyz2asz — Transformation: Cartesian (WGS84) to topozentric coordinates.
  - xyz2blh — Transformation: Cartesian to ellipsoidic coordinates.
- Utilities
  - compare_date — compares two given date vectors and returns the difference in seconds.
  - dat_cal2gpsweek — Returns the gpsweek for a given date.
  - dat_cal2j2k — Transforms a calendar date and time into julian centuries based on Epoch J2000.0.
  - dat_cal2jd — this Function converts a calender date to Julian Date (JD).
  - dat_cal2mjd — Transforms a calendar date and time in the Modified JD (MJD).
  - dat_jd2cal — Transforms a Julian Date (JD) in the calendar date and time.
  - dat_leapseconds — Returns a matrix with all leapseconds, or counts the leapseconds to the given date (includes all to Bulletin C Nr. 41, Feb. 2011).
  - deg2dms — Converts the given Value to degrees, minutes and seconds
  - format_linevec — Returns a formated String of a given linevector.
  - get_tkblattnr — Calculates the sheetnumber of TK25 maps from latitude and longitude.
  - pars_paramstr — Splits a parameterstring of type "`-param1=value1 -param2=value2 ...`" in a `List`.
  - plotWorldmap — Returns and/or plots a worldmap in geographic coordinates (`plot2d()`).

Geometry

- atan2 — Grid bearing t_0 used in geodetic coordinate systems, `atan()` for 4 quadrants in the intervall (0, 2π)
- sphere — Creates a point cloud (mesh) of a spherical surface.
- unitVec — Function returns the unit vector of a given vector or point in carthesian coordinates.

atan2

Grid bearing t_0 used in geodetic coordinate systems, `atan()` for 4 quadrants in the intervall (0, 2π)

Calling Sequence

```
[tangle] = atan2(yin, xin)
```

Parameters

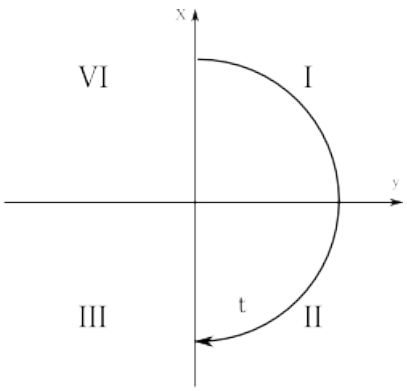
yin, xin
: coordinates (diferences) y and x, scalar or vectors of same size `[ ]`

tangle
: Grid bearing t_0, scalar or matrix `[rad]`

Description

Calculates the grid bearing t_0. In geodetic systems angels are clockwise defined contrary to the mathematical definition counter clockwise.

quadrant	x, y	grid bearing t
	x > 0, y = 0	t = 0
I	x, y > 0	0 < t < π/2
	x = 0, y > 0	t = π/2
II	x < 0, y > 0	π/2 < t < π
	x < 0, y = 0	t = π
III	x, y < 0	π < t < 1.5 · π
	x = 0, y < 0	t = 1.5 · π
IV	x > 0, y < 0	1.5 · π < t < 2 · π



- (matrix aware)
- No warranty, there are no input checks

Examples

```
x = [ 1 0 -1 ];
y = [ 1 -1 0 ];
atan2(y, x)
```

See Also

- atan

Authors

- cos - area7@web.de

sphere

Creates a point cloud (mesh) of a spherical surface.

Calling Sequence

```
[xx,yy,zz] = sphere(radius,[orig])
```

Parameters

radius
: radius of the sphere.

orig
: Origin of the sphere in [x,y,z], optional (default [0 0 0]).

xx
: Matrix 40x20 of the x-coordinates.

yy
: Matrix 40x20 of the y-coordinates.

zz
: Matrix 40x20 of the z-coordinates.

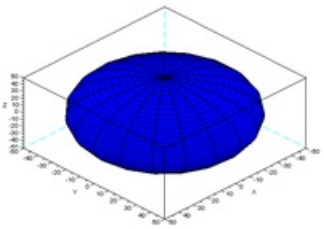
Description

Creates a point cloud (mesh) of a spherical surface. Can be used for `plot3d()`.

- (not matrix aware)
- No warranty, there are no inputchecks.

Examples

```
[xx,yy,zz] = sphere(5);
[xx,yy,zz] = sphere(5,[100,5,5]);
plot3d2(xx,yy,zz);
```



Authors

- cos - area7@web.de

unitVec

Function returns the unit vector of a given vector or point in carthesian coordinates.

Calling Sequence

```
E = unitVec(vector)
E = unitVec(listofvectors)
```

Parameters

vector

coordinates of a point or vector (row vector or matrix of row vectors).

listofvectors

list of vectors (column vectors) more than one!.

E

unit vector (as row vector or matrix of row vectors)

Description

This function calculates the unit vector to a point of a given vector or point in carthesian coordinates.

- (matrix aware)
- No warranty, there are no inputchecks.

Examples

```
unitVec([1, 1])
unitVec([1;1], [1;2])
unitVec([1,2,3;4,5,6])
```

Authors

- cos - area7@web.de

GPS

- alm_parse_yumafile — Reads almanac data from files in yuma-format (experimental).
- sp3_parse_file — Reads orbit- and clockdata from a SP3-c formated file from the IGS.
- sp3c_parse_header — Parses the headerdata of a SP3c-file (lines 1-22).

alm_parse_yumafile

Reads almanac data from files in yuma-format (experimental).

Calling Sequence

```
[satsdata] = alm_parse_yumafile(yfile)
```

Parameters

satsdata

Filepath as string (absolut or rel. to the akt. path), or the logical unit delivered from `mopen()`.

satsdata

matrix with all almanac data.

Description

Reads almanac data from files in yuma-format, and returns them as a matrix with linevectors `[PRN, health, es, toa, i0, omegadot, sqrta, w, m0, af0, af1, week]` for every satellit.

satsdata :

PRN	`[int]`, Identifier of the SV.
health	`[int]`,
es	`[float]`,
toa	`[float]`,
i0	`[float]`,
omegadot	`[float]`,
sqrta	`[float]`,
...	`[float]`,
...	`[float]`,
...	`[float]`,

- (not matrix aware)
- No warranty, no inputchecks.

See Also

- sp3_parse_file

Authors

- cos - area7@web.de

sp3_parse_file

Reads orbit- and clockdata from a SP3-c formated file from the IGS.

Calling Sequence

```
[satdata, header, parserror] = sp3_parse_file(yfile)
```

Parameters

satdata

list of SV-data.

header

structured data found in the header (lines 1-22) of the SP3-file: sp3c_parse_header().

parserror

Vector of errors occured, should be empty.

Description

Reads orbit- and clockdata from a SP3-c formated file of the IGS. Returns a list of structured data. Specification of the fileformat (SP3-c) can be found here: sp3c.txt (12.02.2007)

Data contained in the header has already been worked into this data.

Important: Modifications applied to the Format at 17. August 2010 are not yet implemented. Satellite System identifier J = QZSS und C = COMPASS and Time System identicator QSC for QZSS-time are not supported.

satdata : Each element consist is a dataset holding all data to a SV

.sv_name	`[String]`, Identifier of the SV.
.orbit_almpos_fieldnames	`[1x10]`, stringmatrix with collumnnames of the field `.orbit_almpos`.
.orbit_almpos_prec_fieldnames	`[1x3]`, stringmatrix with collumnames of the field `.orbit_almpos_prec` (headerdata)
.orbit_almvel_fieldnames	`[1xx]`, stringmatrix with collumnames of the field `.orbit_almvel`. (not implemented)
.orbit_almpos	`[Xx15]`, matrix with orbitdata sorted by date and datatype.
.orbit_almpos_prec	`[Xx8]`, matrix with accuracydata from the header.
.orbit_almvel	`[0x0]`, (not implemented)

- (not matrix aware)
- no warranty, there are not enough checks.

Examples

```
satdata =
   sv_name: "G01"
   orbit_almpos_fieldnames: ["time(1:6)","type","x","y","z","sigma_x","sigma_y","sigma_z
   orbit_almpos_prec_fieldnames: ["time","type","value(RMS)"]
   orbit_almvel_fieldnames: [0x0 constant]
   orbit_almpos: [96x15 constant]
   orbit_almpos_prec: [2007,12,14,0,0,0,4,8]
   orbit_almvel: [0x0 constant]
```

See Also

- alm_parse_yumafile
- sp3c_parse_header

Authors

- cos - area7@web.de

sp3c_parse_header

Parses the headerdata of a SP3c-file (lines 1-22).

Calling Sequence

```
[header, perr] = sp3c_parse_header(headerstrings)
```

Parameters

header

structured data found in the header (lines 1-22), struktured data.

perr

Vector of errors occured, should be empty.

headerstrings

Stringmatrix (22 lines) header data from an SP3-file.

Description

Parses the headerdata of a SP3c-file (lines 1-22) and returns a structured dataset (`struct()`)
Specification of the fileformat (SP3-c) can be found here: sp3c.txt (12.02.2007)

Important: Modifications applied to the format at 17. August 2010 are not yet implemented. Satellite System identifier J = QZSS und C = COMPASS and Time System identicator QSC for QZSS-time are not supported.

The header-objekt contains following data:

header

.date	`[1x6] [YYYY,MM,DD,hh,mm,ss.ss]`
.epochs	`[int]`, number of epochs
.orbit type	`[int]`, 1=FIT (fitted), 2=EXT (extrapolated or predicted), 3=BCT (broadcast), and 4=HLM (fitted after applying a Helmert transformation), 0= calculated or not defined.
.epoch_interval	`[int]`, interval between epochs [seconds]
.gpsweek	`[1x2], [gpsweek, secondsofweek].`
.svnumber	`[int]`, number of SVs
.satnames	`[31x1]` stringmatrix with Identifiers of SVs.
.accuracy	`[31x1]`, Linevector
.file_type	`[char]`, "M" - mixed, "G" - GPS only, "R" - GLONASS only, "L" - LEO only, "E" - Galileo (see below, SV-names)
	`[String(3)]`, "GPS" - GPS Time, "GLO" - GLONASS UTC time system, "GAL"

.time_sys	- Galileo system time, "TAI" - International Atomic Time, "UTC" - Coordinated Universal Time.
.base_values	`[float]`, base of position (or velocity) [mm or 10^-4 mm/sec].
.base_clkrate	`[char]`, base for vlock/rate [picosec or 10^-4 psec/sec].
.comment	`[4x1]`, stringmatrix, comment field (Zeile 19-22)..
.hash	not implemented

Some data is checked for plausibility and standard conformity, don't count on it.

Valid SV-names are only: "Gnn" - GPS, "Rnn" - GLONASS, "Lnn" - Low-Earth Orbiting (LEO) and "Enn" - Galileo.

- (not matrix aware)
- no warranty, there are not enough checks.

Examples

```
header  =
  date: [2007,12,14,0,0,0]
  epochs: 96
  orbit_type: 4
  epoch_interval: 900
  gpsweek: [1457,432000]
  svnumber: 31
  satnames: [31x1 string]
  accuracy: [31x1 constant]
  file_type: "G"
  time_sys: "GPS"
  base_values: 1.25
  base_clkrate: 1.025
  comment: [4x1 string]
  hash: 0
```

See Also

- alm_parse_yumafile
- sp3_parse_file

Authors

- cos - area7@web.de

Physical Geodesy

- bl2gk — Transformation: Geodaetic ellipsidic coordinates to Gauß-Krüger-coordinates (convention of Germany)
- eph2xyz — Function to calculate satellite coordinates (xyz) from the ephemerides of the satellite.
- gk2bl — Transformation: Gauß-Krüger-coordinates (convention of Germany) to geodaetic ellipsoidic coordinates
- gmf — Calculating the "global mapping function" (GMF).
- kepler — Calculates the excentric anomaly E (Kepler element).
- par_nutation — Calculates the parameter of the nutation.
- par_praezession — Calculates the parameter of the Precession.
- tropKor_sm — Calculates the tropospheric adjustment with assumed default values (Saastamoinen).
- xyz2asz — Transformation: Cartesian (WGS84) to topozentric coordinates.
- xyz2blh — Transformation: Cartesian to ellipsoidic coordinates.

bl2gk

Transformation: Geodaetic ellipsidic coordinates to Gauß-Krüger-coordinates (convention of Germany)

Calling Sequence

```
[R,H] = bl2gk(B, L, L0, ell)
```

Parameters

B
: latitute, `[decimaldegree]`

L
: longitute, `[decimaldegree]`

L0
: meridian of origin, `[degree]`

ell
: Ellipsoid, 'wgs84', 'grs80' or 'bessel' (default) [string]

R
: easting [m]

H
: northing [m]

Description

Transformation: Geodaetic ellipsidic coordinates to Gauß-Krüger-coordinates (convention of Germany).

- (matrix aware)
- No warranty, there are no inputchecks.

See Also

- gk2bl

Authors

- cos - area7@web.de

Bibliography

Geschlossene Formeln, Reihenentwicklung nur für Meridianbogenlänge aus Fußpunktbreite nach K. Schnädelbach, Skript Landesvermessung

gk2bl

Transformation: Gauß-Krüger-coordinates (convention of Germany) to geodaetic ellipsoidic coordinates

Calling Sequence

```
[B, L] = gk2bl(R, H [,ell])
```

Parameters

R
: easting [m]

H
: northing [m]

ell
: Ellipsoid, 'wgs84', 'grs80', oder 'bessel'(default) [string]

B
: latitute [degree.decimal]

L
: longitute [degree.decimal]

Description

Transformation: Gauß-Krüger-coordinates `R,H` (convention of Germany) to geodaetic ellipsoidic coordinates `B,L`.

- (matrix aware)
- No warranty, there are no inputchecks.

See Also

- bl2gk

Authors

- cos - area7@web.de

Bibliography

Geschlossene Formeln, Reihenentwicklung nur für Meridianbogenlänge aus Fußpunktbreite nach K. Schnädelbach, Skript Landesvermessung

eph2xyz

Function to calculate satellite coordinates (xyz) from the ephemerides of the satellite.

Calling Sequence

```
[xs,ys,zs,delta_Ts,delta_rho_rel] = eph2xyz(tc,Prs,a0,a1,a2,C_rs,delta_n,M_0,C_uc,e,C_us,
```

Parameters

tc

referencetime of clockparameters (time of clock), i.g. the same as t_e (quod vide).

Prs

Pseudorange `[ ]`

a0

SV-clock-offset at reference epoch t_c `[s]`

a1

SV-clock-drift at reference epoch t_c `[s/s]`

a2

SV-frequency-drift (ageing) at reference epoch t_c `[s/s²]`, i.a. =0

a

semi-mayor axis of the Keplerellipse `[m]` (Attention! also possible sqrt(a) then [sqrt(m)]).

t_e

time of ephemerides

e

numerical excentricity `[1]`

i_0

inclination at reference epoch t_e `[rad]`

Omega_0

rectaszension of ascending node at reference epoch t_e `[rad]`

omega_klein

argument of perigee `[rad]`

M_0

mean anomalie at reference epoch t_e `[rad]`

Omega_dot

alteration rate of rectaszension `[rad/s]`

i_dot

alteration rate of inclination `[rad/s]` (=di/dt).

delta_n

adjustment of mean motion `[rad/s]`

C_us

amplitude of the sinus-harmonic correction term for argument of latitude u `[1]`

C_uc

amplitude of the cosinus-harmonic correction term for argument of latitude u `[1]`

C_is

amplitude of the sinus-harmonic correction term for the inclination i `[1]`

C_ic

amplitude of the cosinus-harmonic correction term for inclination i `[1]`

C_rs

amplitude of the sinus-harmonic correction term for geocentric distance r `[1]`

C_rc

amplitude of the cosinus-harmonic correktion term for geocentric distance r `[1]`

Parameters

xs

SV-coordinate X [m]

ys

SV-coordinate Y [m]

zs

SV-coordinate Z [m]

delta_Ts

SV-clock diference [s]

delta_rho_rel

relativistic effect due to orbit excentricity [m]

Description

Function to calculate satellite coordinates (xyz) from the ephemerides of the satellite.

- (possibly matrix aware)
- No warranty, there are no inputchecks.

See Also

- kepler

Authors

- cos - area7@web.de

kepler

Calculates the excentric anomaly E (Kepler element).

Calling Sequence

```
E = kepler(M, e)
E = kepler(M, e, E0)
```

Parameters

M

mean anomaly `[rad]`

e

exzentricity `[ ]` (0<e<1)

E0

Startwert of the iteration, approximation of the ex. anomaly (optional; if not given approximated with `M`) `[rad]`

E

exzentric anomaly `[rad]`

Description

Kepler equation:

Function for the iterative calculation of the excentric anomaly E. (stop criterion: 1e-10).

-
- No warranty, there are no inputchecks.

Authors

- cos - area7@web.de

gmf

Calculating the "global mapping function" (GMF).

Calling Sequence

```
[gmfh,gmfw] = gmf(mjd,lat,lon,hgt,zd)
```

Parameters

mjd
: modified julian date (mjd).

lat
: latitute of station, [rad]

lon
: longitute of station, [rad]

hgt
: hight of station [m]

zd
: zenit distance of satellite over station [rad]

gmfh
: parameter h

gmfw
: parameter w

Description

Calculating the "global mapping function" (GMF).

Boehm, J., A.E. Niell, P. Tregoning, H. Schuh (2006), Global Mapping Functions (GMF): A new empirical mapping function based on numerical weather model data, Geophphysical Research Letters, Vol. 33, L07304, doi:10.1029/2005GL025545.

- No warranty, there are no inputchecks.
- (not matrix aware)

Authors

- cos - area7@web.de

Bibliography

Boehm, J., A.E. Niell, P. Tregoning, H. Schuh (2006), Global Mapping Functions (GMF): A new empirical mapping function based on numerical weather model data, Geophphysical Research Letters, Vol. 33, L07304, doi:10.1029/2005GL025545.

par_nutation

Calculates the parameter of the nutation.

Calling Sequence

```
[epsilon_0, delta_epsilon, delta_psi] = par_nutation(t_j2k)
```

Parameters

t_j2k

time in the epoch J2000.0

epsilon_0

mean obliquity of the ecliptic `[rad]`

delta_epsilon

nutation in longitude `[rad]`

delta_psi

nutation in obliquity `[rad]`

Description

Calculates the parameter of nutation, the mean obliquity of the ecliptic `epsilon_0`, the nutation in longitude `delta_psi` and in obliquity `delta_epsilon`.

- Precision: ca. 1'',
- (matrix aware)
- No warranty, there are no inputchecks.

See Also

- par_praezession

Authors

- cos - area7@web.de

par_praezession

Calculates the parameter of the Precession.

Calling Sequence

```
[z_A, theta_A, zeta_A] = par_praezession(t_j2k)
```

Parameters

t_j2k

time in the epoch J2000.0

z_A

z_A `[decimal degree]`

theta_A

theta_A `[decimal degree]`

zeta_A

zeta_A `[decimal degree]`

Description

Calculates the parameter of the Precession.

- Precision: ca. 1''
- (matrix aware).
- No warranty, there are no inputchecks.

See Also

- par_nutation

Authors

- cos - area7@web.de

tropKor_sm

Calculates the tropospheric adjustment with assumed default values (Saastamoinen).

Calling Sequence

```
tropKor = tropKor_sm(Zenit,h)
```

Parameters

Zenit

zenith, angle in [rad]

h

altitude in [m].

tropKor

tropospheric adjustment in [m]

Description

Calculates the tropospheric adjustment with assumed default values, formula according to Saastamoinen, 1972.

This modell's accuracy depends on the elevation, and is best for obervations near zenit. Should not be used for elevations lower than <15°.

- No warranty, there are no inputchecks.

Authors

- cos - area7@web.de

xyz2asz

Transformation: Cartesian (WGS84) to topozentric coordinates.

Calling Sequence

```
[A, Z, S] = xyz2asz(Xr, Yr, Zr, Xs, Ys, Zs)
```

Parameters

Xr
: x-coordinate of station [m]

Yr
: y-coordinate of station [m]

zr
: z-coordinate of station [m]

Xs
: x-coordinate of sv [m]

Ys
: y-coordinate of sv [m]

Zs
: z-coordinate of sv [m]

A
: azimuth under which the sv is observed [degree]

Z
: zenith angle under which the sv is observed [degree]

S
: distance between station and sv [m]

Description

Transformation: Cartesian (WGS84) to topozentric coordinate.

Formulas from Heck (1988)

- (matrix aware)
- No warranty, there are no inputchecks.

Authors

- cos - area7@web.de

xyz2blh

Transformation: Cartesian to ellipsoidic coordinates.

Calling Sequence

```
[bb,l,h] = xyz2blh(x,y,z)
[bb,l,h] = xyz2blh(x,y,z,ell)
```

Parameters

x
: x-coordinate `[m]`

y
: y-coordinate `[m]`

z
: z-coordinate, `[m]`

ell
: Identifier, Textstring: 'wgs84' (default), 'grs80' or 'bessel'

bb
: latitude `[decimaldegree]`

l
: longitude, `[decimaldegree]`

h
: elipsoidic altitude, `[m]`

Description

Transformation: Cartesian to ellipsoidic coordinates.

Formulas from Hofmann-Wellenhof u.a. (1994): "GPS in der Praxis"

- **No negative altitudes allowed!**
- No warranty, there are no inputchecks.

Examples

```
[bb,l,h]=xyz2blh(x,y,z)
[bb,l,h]=xyz2blh(x,y,z,'wgs84')
```

Authors

- cos - area7@web.de

Utilities

- compare_date — compares two given date vectors and returns the difference in seconds.
- dat_cal2gpsweek — Returns the gpsweek for a given date.
- dat_cal2j2k — Transforms a calendar date and time into julian centuries based on Epoch J2000.0.
- dat_cal2jd — this Function converts a calender date to Julian Date (JD).
- dat_cal2mjd — Transforms a calendar date and time in the Modified JD (MJD).
- dat_jd2cal — Transforms a Julian Date (JD) in the calendar date and time.
- dat_leapseconds — Returns a matrix with all leapseconds, or counts the leapseconds to the given date (includes all to Bulletin C Nr. 41, Feb. 2011).
- deg2dms — Converts the given Value to degrees, minutes and seconds
- format_linevec — Returns a formated String of a given linevector.
- get_tkblattnr — Calculates the sheetnumber of TK25 maps from latitude and longitude.
- pars_paramstr — Splits a parameterstring of type "`-param1=value1 -param2=value2 ...`" in a `List`.
- plotWorldmap — Returns and/or plots a worldmap in geographic coordinates (`plot2d()`).

compare_date

compares two given date vectors and returns the difference in seconds.

Calling Sequence

```
difference = compare_date(date_1, date_2)
```

Parameters

date_1

date vector [1x3] or [1x6]

date_2

date vector [1x3] or [1x6]

difference

difference in `[s]`, 0 if identical

Description

Compares two given date vecotrs [1x3] or [1x6], and returns the difference in seconds.

- (not matrix aware)
- No warranty, there are no inputchecks.

Examples

```
diff = compare_date([2010,8,24,12,0,0], [2010,8,24,12,0,15])
```

See Also

- datevec
- dat_jd2cal

Authors

- cos - area7@web.de

dat_cal2jd

this Function converts a calender date to Julian Date (JD).

Calling Sequence

```
jd = dat_cal2jd(cal)
jd = dat_cal2jd(year, month, day [, hour, minute, second])
```

Parameters

cal

date as vector (nx3 oder nx6) or matrix of nx3 or nx6 vectors.

year

year, integer `(1xn)`

month

month, integer `(1xn)`

day

day, integer `(1xn)`

hour, minute, second

time in hours (minutes can be contained as decimal), minues and seconds `(1xn). [year; month; day; hour; minutes; second]`

jd

Julian Date (JD) (1xn)

Description

Converts a calendar date in a Julian Date (JD).

Attention: dates before 4. Okt. 1582 24h are counted to the julian calendar and after to the gregorian calendar, which means they are after the 15. Okt 1582.

- (matrix aware)

Examples

```
JD = dat_cal2jd(y,m,d,ut1)
JD = dat_cal2jd([y;m;d])
JD = dat_cal2jd(datevec(now())')
```

See Also

- dat_jd2cal
- dat_cal2mjd

Bibliography

Meeus, J.: Astronomical Algorithms, Willmann-Bell, Richmond 2000 (2nd ed., 2nd printing).

Authors

- cos - area7@web.de

dat_jd2cal

Transforms a Julian Date (JD) in the calendar date and time.

Calling Sequence

```
[year, month, day, hour] = dat_jd2cal(jd)
[year, month, day, hour, minute, second] = dat_jd2cal(jd)
cal = dat_jd2cal(jd)
```

Parameters

jd

Julian Date (JD)

year

year, integer (1xn)

month

month, integer (1xn)

day

day, integer (1xn)

hour, minute, second

daytime, houres, minutes, seconds (1xn)

cal

date as vector (`[nx3]` or `[nx6]`) or matrix of vectors. `[year; month; day; [hour; minute; second]]`

Description

Converts a Julian Date (JD) in a calendar date.

Attention: Dates before 4. Okt. 1582 24h are counted to the julian calendar and after to the gregorian calendar, which means they are after the 15. Okt 1582.

- (matrix aware)

Examples

```
[y,m,d,h] = dat_jd2cal(2453750.1875)
```

See Also

- dat_cal2jd
- dat_cal2mjd

Bibliography

Meeus, J.: Astronomical Algorithms, Willmann-Bell, Richmond 2000 (2nd ed., 2nd printing).

Authors

- cos - area7@web.de

dat_cal2mjd

Transforms a calendar date and time in the Modified JD (MJD).

Calling Sequence

```
mjd = dat_cal2mjd(year, month, day [hour, minute, second])
mjd = dat_cal2mjd(cal)
```

Parameters

cal
: date vector `[1x6]`

year
: year, integer `[YYYY]`

month
: month, integer

day
: day, integer

hour, minute, second
: daytime UT1, `[hours, minutes, seconds]` optional

mjd
: Modified Julian Date (MJD)

Description

Calculates the Modified Julian Date (MJD) from a given calendar date and time.

Modified Julian Date: The number of days (with decimal fraction of the day) that have elapsed since midnight (UT1) at the beginning of Wednesday November 17, 1858.

```
MJD = JD - 2,400,000.5
```

- (matrix aware)
- No warranty, there are no input checks

Examples

```
mjd = dat_cal2mjd(2018, 11, 30, 18)
```

See Also

- dat_cal2jd
- dat_jd2cal
- dat_cal2j2k

Authors

- cos - area7@web.de

dat_cal2j2k

Transforms a calendar date and time into julian centuries based on Epoch J2000.0.

Calling Sequence

```
j2k = dat_cal2j2k(year, month, day [, hour, minute, second])
j2k = dat_cal2j2k(cal)
```

Parameters

cal
: datevector `[1x6]` or `[1x3]`

year
: year, integer `[YYYY]`

month
: month, integer

day
: day, integer

hour, minute second
: daytime UT1, `[hours, minutes, seconds]` optional

j2k
: date based on epoch J2000.0 in julian centuries

Description

Transforms a calendar date and time into julian centuries basd on Epoch J2000.0.

Date based on Epoch J2000.0

Timescale for the actual astronomical standard epoch J2000.0 (Bessel epoch). It is counted in julian centuries of 36525 days.

- (matrix aware)
- No warranty, there are no input checks.

Examples

```
j2k = dat_cal2j2k(2018, 11, 30, 18)
```

See Also

- dat_cal2jd
- dat_jd2cal

Authors

- cos - area7@web.de

dat_cal2gpsweek

Returns the gpsweek for a given date.

Calling Sequence

```
[weeks, wsecs] = dat_cal2gpsweek(year, month, day [,hour, min, sec])
[weeks, wsecs] = dat_cal2gpsweek(cal)
```

Parameters

cal

date as vector (`mx3`) or (`mx6`)

year
: year, integer (>0)

month
: month, integer (0<month<=12)

day
: day, Integer (0<day<=31)

hour, min, sec
: Daytime in UTC [hours, minutes, seconds] (optional and experimental)

weeks
: gps weeks, integer

wsecs
: seconds since beginn of week, integer (experimental)

Description

This function returns the gps week of a given date (year, month, day [, daytime in UTC]).

First day of the (GPS-)week is Sunday. Beginning with 0 at 6. Januar 1980.

At 22. August 1999 the gps week reached 1024. In some cases (broadcast directly from the GPS satelite) the GPS week rollover is to take into account, beginning again with 0.

- For details see: Zeitdefinitionen:GPS
- leapseconds are not taken into account
- No warranty, there are not enough input checks.

Authors

- cos - area7@web.de

dat_leapseconds

Returns a matrix with all leapseconds, or counts the leapseconds to the given date (includes all to Bulletin C Nr. 41, Feb. 2011).

Calling Sequence

```
lsecs = dat_leapseconds()
lsecs = dat_leapseconds(type)
lsecs = dat_leapseconds(dt, utc)
```

Parameters

type
: Format in which the dates should be returned, `'DT'`days as in `datenum()`, `'D'` date (year, month, day), string

dt
: Date in decimal days (see UTC), for now only days supported. Daytime ist highly experimental, float

utc
: Daytime (UTC), (has not been taken into account, highly experimental), float

lsecs
: Matrix with all leapseconds `[year, month, day, DT, leapsecond]`, `[year, month, day, leapsecond]` or `[DT, leapsecond]`

  Counter of the elapsed leapseconds till the given date, integer

Description

Returns a matrix with all leapseconds (including future onces), or counts the leapseconds to the given date.

All data is received from the actual Bulletin C (39 of 1 Jan 2010) published by the IERS.

- (not matrix aware)

See Also

- datenum

Authors

- cos - area7@web.de

deg2dms

Converts the given Value to degrees, minutes and seconds

Calling Sequence

```
[dms, dmsvec] = deg2dms(deg[, flag])
```

Parameters

deg

Scalar or vector `[decimal degree]`

flag

set to 's' to return a formated string [degree°minutes'seconds''] , default='', optional

dms

Scalar or vector of return values `[DD.MMSSss]`, or formated string or vector of formated strings following the form `[DD°MM'SS.ss'']`

dmgvec

Separated values `[DD, MM, SS.ss]`

Description

Formates the given skalar or vector of values in decimal degrees to the common form `[DD,MMSSss]`. `dms` can optionaly be returned as formated string (flag='s') `"GG°MM'SS.ss"`. The additional return value `dmsvec` containes the separated values as line vector `[Degree, Minutes, Seconds.decimalseconds]`.

- (matrix aware)
- No warranty, there are no input checks

Examples

```
deg2dms(270.45)
deg2dms(270.45, 's')
```

Authors

- cos - area7@web.de

format_linevec

Returns a formated String of a given linevector.

Calling Sequence

```
retstr = format_linevec(lvec [,style])
```

Parameters

lvec

Linevector or matrix.

style

seperator, default ', '.

retstr

Returnstring.

Description

Formats one or more Linevectors (matrix) with seperators and returns a string for each line.

The styleparameter describes the used seperators, it can containe aditional characters.

- (matrix aware)
- No warranty, there are no inputchecks.

Examples

```
format_linevec([15, 14, 13]);
format_linevec([15, 14, 13], ' : ');
```

Authors

- cos - area7@web.de

get_tkblattnr

Calculates the sheetnumber of TK25 maps from latitude and longitude.

Calling Sequence

```
tknr = get_tkblattnr(long, lat)
```

Parameters

long

geographic longitude `[decimaldegreed]`

lat

geographic latitude `[decimaldegree]`

tknr

sheetnumber of the TK25, integer

Description

This function calculates the relevant topographic map (Topographische Karte TK25) number from given geographic coordinates `[decimaldegree]`.

The size of a TK25 map is defined as 10 geografic minutes in width and 6 geografic minutes in height.

- (matrix aware)
- No warranty, there are no input checks.

Examples

```
get_tkblattnr(11.56, 48.14)
```

Authors

- cos - area7@web.de

pars_paramstr

Splits a parameterstring of type "`-param1=value1 -param2=value2 ...`" in a `List`.

Calling Sequence

```
[paraml, err] = pars_paramstr(parameterstrng [, seperator])
```

Parameters

parameterstring

Parameterstring "-param1=value1 -param2=value2 ..."

seperator

Characters which should be used to seperate the parameters [default=' ' (space)]

paraml

List of all parameter-value vectors.

err

Errorstring, or empty when no error occured.

Description

The given parameterstring is seperated in his parameter-value-vectors and returned as list. `paraml(1)(1)` ist the name of the first parameters, `paraml(1)(2)` the coresponding value.

Parameternames must not contain characters defined as seperator. Seperators are only allowed in values if the complete value is quoted (single or double).

Quotes are not allowed in the values, there ist no mechanic of escaping implemented.

- (not matrix aware)
- No warranty, there are no inputchecks.

Examples

```
pars_paramstr(" -path=~ -name=""""mein Name""""" ");
parameterstring="-para1=Test -para2=irgendwas -para3=45";
pars_paramstr(parameterstring, ' ');
```

Authors

- cos - area7@web.de

plotWorldmap

Returns and/or plots a worldmap in geographic coordinates (`plot2d()`).

Calling Sequence

```
worldmap = plotWorldmap()
worldmap = plotWorldmap(doplot)
worldmap = plotWorldmap(worldmapfile)
worldmap = plotWorldmap(doplot,worldmapfile)
```

Parameters

doplot

activates the ability to directly plot the worldmap, boolean.

worldmapfile

absolute or relative path to an alternative worldmap file, string. *Not supported*

worldmap

Matrix of points in geographic coordinates of the worldmap `[latitude, longitude]`, `[decimaldegrees]`.

Description

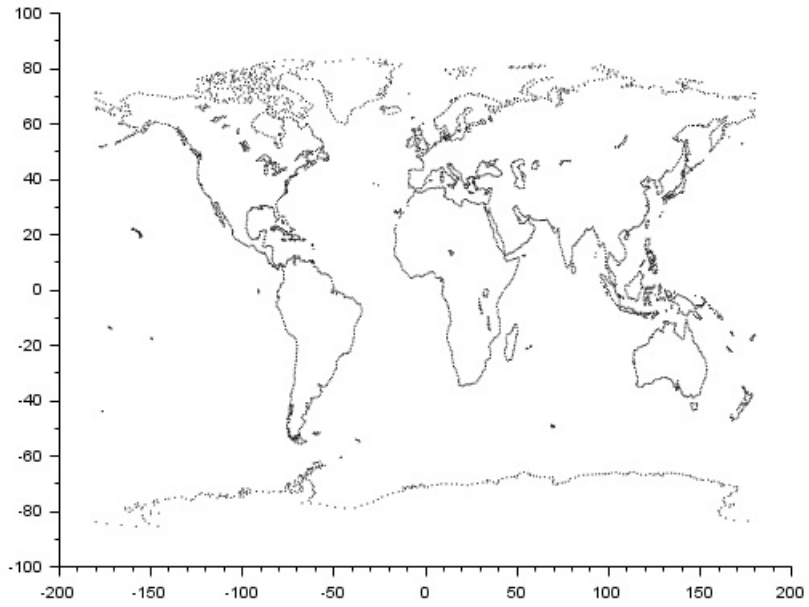
Returns the points of a worldmap (coastlines) in geographic coordinates.

With activated `doplot`-parameter it can directly plot in a active figure (`plot2d()`).

- (not matrix aware)

Examples

```
plotWorldmap(%T);
worldmap=plotWorldmap();
```



Authors

- cos - area7@web.de