The TCV Psi–Toolbox reference guide

Contents

1. \texttt{psitbxdcdd}
2. \texttt{psitbxdcdd/intersect}
3. \texttt{psitbxfun}
4. \texttt{psitbxfun/cumsum}
5. \texttt{psitbxfun/mean}
6. \texttt{psitbxfun/psitbxf2f}
7. \texttt{psitbxfun/sum}
8. \texttt{psitbxfun Mathematical operators and functions}
9. \texttt{psitbxgrid}
10. \texttt{psitbxgrid/metric}
11. \texttt{psitbxgrid/psitbxg2g}
12. \texttt{psitbxpsi}
13. \texttt{psitbxpsi/psitbxfsg}
14. \texttt{psitbxpsi/psitbxp2p}
15. \texttt{psitbxtcv}

\section*{\texttt{psitbxdcdd}}

\subsection*{Purpose}

Creates a \texttt{psitbxdcdd} object containing the description of the geometry of the chords of a diagnostic.

\subsection*{Syntax}

\begin{verbatim}
dcd = psitbxdcdd(rd, zd, phid, pvd, tvd, nd, t)
\end{verbatim}

\subsection*{Description}

The function \texttt{psitbxdcdd} is used to create a \texttt{psitbxdcdd} object that contains the description of the geometry of the chords of a diagnostic. The fields of a \texttt{psitbxdcdd} object and the corresponding input arguments have the following meaning:

- \texttt{rd}:
  The radial position in cylindrical coordinate of a reference point along the chord, very often the pin–hole.

- \texttt{zd}:
  The vertical position of the same point.

- \texttt{phid}:
  The toroidal angle of the same point.

- \texttt{pvd}:
  The poloidal viewing angle, measured between the horizontal plane and the chord, positive if the chord is pointing above the plane.
The toroidal viewing angle, measured between the vertical plane containing the machine axis and the chord reference point and the vertical plane containing the chord itself, positive if the chord is looking in the direction of increasing toroidal angle.

The number of points used to discretise the chord when building a psitbxgrid object of type 'Diagnostic-Chords'.

The time base for moving chords.

The size of $rd$, $zd$, $phid$, $pvd$, $tvd$ follows the following rules:

- The size can retain the organisation of the diagnostic, for example $20 \times 10$ for 10 cameras with 20 detectors each, or $1028 \times 1028$ for a CCD.
- All these arguments must have the same size unless
  - It is the same for all chords and can be summarised in a scalar
  - It is empty and will be replaced by a default value
- For chords moving in time, the last dimension must correspond to the length of $t$.

Example

Example 1

Here is an example with two cameras with 20 detectors each, the first one with a poloidal coverage, the second one with a toroidal coverage:

```matlab
rd = 1.2;
zd = 0;
phid = pi/8;
pvd = [linspace(-pi/3,pi/3,20);zeros(1,20)];
tvd = [zeros(1,20);linspace(-pi/3,pi/3,20)];
dcd = psitbxxdc(drd,zd,phid,pvd,tvd);
dcd is a PsiTbx-Diagnostic-Chord-Description object =
  rd: [2x20 double]
  zd: [2x20 double]
  phid: [2x20 double]
  pvd: [2x20 double]
  tvd: [2x20 double]
  nd: 81
  t: []
plot(dcd)
```
Example 2: moving chord

The possibility to have real–time scanning diagnostic is included. In that case the last dimension of the fields of the `psitbxcdcd` object represents the time base. Given as an example *Le cauchemar du gyrotroniste*:

```matlab
td = linspace(0,1,73);
rd = 1.2;
zd = 0;
phid = pi/2;
pvd = pi/6*cos(pi*td);
tvd = pi/6*sin(2*pi*td);
LeCauchemarDuGyrotroniste = psitbxcdcd(rd,zd,phid,pvd,tvd,[]); plot(LeCauchemarDuGyrotroniste)
```
Purpose

Computes the intersection between diagnostic chords described by a `psitbxdcld` object and a cylindrical coordinate grid contained in a `psitbxgrid` object.

Syntax

`s = intersect(dcd,g)`

Description

The method `intersect` applies to a `psitbxdcld` object and computes the intersection between the chords and a grid in cylindrical coordinates. It returns an array of linear coordinates along the chords of size `[dcd.nd, size(dcd)]`. The main purpose of `intersect` is to provide the coordinate of the points for building a `psitbxgrid` object of type 'Diagnostic-Chords'.

For those chords who do not cross the domain of the grid, NaN's are returned. For those chords who cross the domain twice, the first segment is considered.

Example

```matlab
dcd = psitbxdcld('Demo');
s = intersect(dcd,psitbxgrid('Cylindrical','Grid','Default'));
gdc = psitbxgrid('Diagnostic-Chords','Points',{s},dcd)
gdc is a PsiTbx-Grid object =
    type: 'Diagnostic-Chords'
    label: {1x1 cell  }
    storage: 'Points'
        x: {1x1 cell  }
        t: []
    par: [1x1 struct]
    csp: [1x1 struct]
```

Algorithm

Only awfull calculation of intersections between lines, planes and cylinders using analytical geometry.

Purpose

Creates a `psitbxfun` object containing a function defined on a spatial grid.

Syntax

```matlab
f = psitbxfun(x,g)
```
The function `psitbxfun` is used to create a `psitbxfun` object that contains a function defined on a spatial grid. If this function does not depend on time, `psitbxfun(x,g)` is used with `x` containing the function value(s) defined on the `psitbxgrid` `g`. If the function varies in time, use the form `psitbxfun(x,g,t)`.

The size of the matrix `x` must be `[size(g),[n1,...],[nt]]`. The optional dimension `n1,...` can be used for multi-value functions (for example the wave length for spectroscopic measurements). If the time argument is given, then the last dimension of `x`, `nt` must correspond to `length(t)`.

For degenerated coordinates, that have a size of one, the following rules apply:

- if this coordinate is a scalar, the function is defined only on the corresponding subspace and extrapolation outside this subspace is not possible
- if this coordinate is a `nAN`, the function is assumed to be independant of this spatial coordinate

Singleton dimension in `x` arising from degenerated coordinates can be omitted.

**Example**

**Example 1**

A fairly simple function defined on a cylindrical coordinate grid:

```matlab
rzphi = [0.62:.02:0.7,-.05:.02:.05,pi/180*(-3:2:3)];
g = psitbxgrid('Orthogonal','Grid',rzphi);
[rzphi{:}] = ndgrid(rzphi{:});
x = exp(-300*(rzphi{1} - g.x{1}(1)).^2) .* exp(-3000*rzphi{2}.^2);
f = psitbxfun(x,g)
f is a PsiTbx-Function object =
   x: [5x6x4 double]
   grid: [5x6x4 psitbxgrid]
   t: []
plot(f)
```
Example 2

A very common concept in Tokamak physics is the profile, that is a quantity constant on a flux surface and therefore depending only on the flux coordinate. Such a profile is trivially represented by a function on a 1–D grid, here the canonical parabolic profile:

```matlab
% Example 2: Parabolic Profile in Tokamak Physics

% Canvas grid coordinates
linspace(0,1,41)

% Function representation
f = psitbxfun(1-linspace(0,1,41).^2,g)

% Display grid properties
x: [41x1 double]
grid: [41x1x1 psitbxgrid]
t: []

% Display profile
plot(f)
```

```
% Canvas grid coordinates
linspace(0,1,41)

% Function representation
f = psitbxfun(1-linspace(0,1,41).^2,g)

% Display grid properties
x: [41x1 double]
grid: [41x1x1 psitbxgrid]
t: []

% Display profile
plot(f)
```
<< psitbxfun/cumsum

Purpose

Method to compute the indefinite integral of a psitbxfun object over spatial coordinates

Syntax

\[ f = \text{cumsum}(f, \text{dim}) \]

\[ f = \text{cumsum}(f, \text{dim}, '\text{metric}') \]

Description

The method \text{cumsum} applies on a psitbxfun object defined on grid with 'Grid' storage and returns a psitbxfun containing its indefinite integral along the specified dimensions \text{dim}. For example if \text{dim} = 1, the integral is performed along the first coordinate, if \text{dim} = [2, 3], the surface integral along coordinates 2 and 3 is returned, if \text{dim} = [1, 2, 3], the volume integral is returned.

By specifying the option 'metric', the appropriate metric will be taken into account; simply the function \( f.\text{metric}(f.\text{grid}, \text{dim}) \) is integrated.

Algorithm

Uses a trapeze integration.

<< psitbxfun/mean

Purpose

Method to average a psitbxfun object over spatial coordinates

Syntax

\[ f = \text{mean}(f, \text{dim}) \]

\[ f = \text{mean}(f, \text{dim}, w) \]

\[ f = \text{mean}(f, \text{dim}, '\text{metric}') \]

\[ f = \text{mean}(f, \text{dim}, w, '\text{metric}') \]
**Description**

The method `mean` applies on a `psitbxfun` object defined on grid with 'Grid' storage and returns a `psitbxfun` containing the function averaged along the specified dimensions `dim`. For example if `dim = 1`, the average is performed along the first coordinate, if `dim = [2, 3]`, the surface average along coordinates 2 and 3 is returned, if `dim = [1, 2, 3]`, the volume average is returned.

Note that the grid on which the returned function is defined will become degenerated in the dimension along which average is carried out.

In the call `mean(f, dim, w)`, `w` is a `psitbxfun` object playing the role of a weight in the average.

By specifying the option 'metric', the appropriate metric will be taken into account; simply the function `f.*metric(f.grid, dim)` is average.

**Example**

Average plasma current density

```matlab
jphi = psitbxtcv(10000,'JPHI');
jphiavg = mean(jphi,[1,2],'metric');
plot(jphiavg)
```

**Algorithm**

```matlab
"mean(f,dim,w)
    sum(f.*w,dim)./sum(w,dim)

mean(f,dim,w,'metric')
    sum(f.*w.*metric(f.grid,dim),dim)./sum(w.*metric(f.grid,dim),dim)
```
Purpose

Method to interpolate a `psitbxfun` object on a specified grid.

Syntax

\[ f_2 = \text{psitbxf2f}(f, g_2) \]

\[ f_2 = \text{psitbxf2f}(f, g_2, \text{der}) \]

\[ f_2 = \text{psitbxf2f}(\text{der}) \]

Description

The method `psitbxf2f` applies on a `psitbxfun` object to interpolate the embedded function on new space points. In the call `psitbxf2f(f, g_2)` the `psitbxfun` object represents a function defined on a `psitbxgrid` which must have 'Grid' storage. \( g_2 \) is a `psitbxgrid` object specifying the new space points on which the function \( f \) must be interpolated. Those can be given in any coordinate systems. Points outside the definition domain of the function will be assigned a NaN. If any coordinate of \( g_2 \) is a NaN, which means that any function defined on \( g_2 \) is independent of this coordinate, then the function \( f \) must also be independent along the corresponding coordinate.

The optional argument `der` is a three element vector that specifies the spatial derivative order to apply along each spatial coordinate. Note that since functions are interpolated with cubic spline, any order bigger than 3 will results in a zero function.

Example

Example 1

How to compute the normalised flux coordinate along diagnostic chords:

```matlab
dcd = psitbxcdcd('Demo');
psi = psitbxtcv(10000,.5,'01');
gdc = psitbxgrid('Diagnostic-Chords','Points',{intersect(dcd,psi.grid)},dcd);
rho = sqrt(psitbxf2f(psi,gdc));
plot(rho)
```
Example 2

How to compute the radial derivative of the poloidal flux which is related to the vertical magnetic field:

```matlab
psi = psitbxtcv(10000);
dpsidz = psitbxf2f(psi,[1,0,0]);
plot(dpsidz)
```

Algorithm

In a first step the points of \( g2 \) where the function \( f \) must be interpolated are transformed to compute their coordinates in the coordinate system on which the function \( f \) is defined, using the method \( \text{psitbxg2g} \). Then the function \( f \) is interpolated in each spatial direction with cubic splines. The edge conditions are \( \text{Not–A–Knot} \) conditions except along
angular coordinates which span 2π, in which case *periodic* edge conditions are used. Finally this cubic spline interpolant is evaluated at the new positions, to the appropriate derivative order.

Note that interpolation requires the function to be smooth. For example interpolating the square–root of the poloidal flux defined in cylindrical coordinates yields problem on the magnetic axis where its first derivative is not continuous.

**<< psitbxfun/sum**

**Purpose**

Method to integrate a psitbxfun object over spatial coordinates

**Syntax**

```plaintext
f = sum(f,dim)
```

```plaintext
f = sum(f,dim,'metric')
```

**Description**

The method *sum* applies on a psitbxfun object defined on grid with 'Grid' storage and returns a psitbxfun containing the function integrated along the specified dimensions dim. For example if dim = 1, the integral is performed along the first coordinate, if dim = [2,3], the surface integral along coordinates 2 and 3 is returned, if dim = [1,2,3], the volume integral is returned.

Note that the grid on which the returned function is defined will become degenerated in the dimension along which integration is carried out.

By specifying the option 'metric', the appropriate metric will be taken into account; simply the function 

```plaintext
f.*metric(f.grid,dim)
```

is integrated.

**Example**

A non trivial way to compute the plasma current

```plaintext
jphi = psitbxxtcv(10000,'JPHI');
ip = sum(jphi,[1,2]);
plot(ip)
```
Algorithm

Uses a trapeze integration.

<< psitbxfun Mathematical operators and functions

Purpose

Methods for mathematical operations on psitbxfun objects

Syntax

\[ f = f + a \]
\[ f = f + g \]
\[ f = f \cdot a \]
\[ f = \sqrt{f} \]
\[ f = f > 0 \]

Description

Most of the mathematical operators and functions have been overloaded for psitbxfun objects. For multi operand functions, the following rules apply:

- Some of them can be of class double, either scalar or with the appropriate size
• If all operands of class psitbxfun are not defined on the same grid, they will be interpolated on the definition grid of the first one using the psitbxf2f method.

Example

Express the current density in MA/m²

\[
\text{jphi} = \text{psitbxtcv(10000,'JPHI')}/1e6;
\]

\[\text{psitbxgrid}\]

Purpose

Creates a psitbxgrid object containing the definition of a spatial grid

Syntax

\[
g = \text{psitbxgrid(type,storage,x[,p1,...])}
\]

Description

The function psitbxgrid is used to create a psitbxgrid object that contains the description of a spatial grid. The usual call is psitbxgrid(type,storage,x).

type

The character argument type specifies which coordinate system is used. Possible choices are:

- **'Orthogonal'**
  This is the usual orthogonal cartesian coordinate system:
  \[
  \begin{align*}
  X & \quad \text{This axis lies on the midplane at toroidal position zero between sector #16 and #1, zero on machine axis} \\
  Y & \quad \text{This axis lies on the midplane at toroidal position } \pi/2, \text{ zero on machine axis} \\
  Z & \quad \text{This axis coincides with machine axis pointing upward, zero on the midplane}
  \end{align*}
  \]

- **'Cylindrical'**
  The cylindrical coordinate system whose axis coincides with the main axis of the machine:
  \[
  \begin{align*}
  R & \quad \text{The radial coordinate} \\
  Z & \quad \text{The vertical coordinate, positive upward, zero on the midplane} \\
  \Phi & \quad \text{The azimuthal coordinate, positive ccw seen from the top, zero between sector #16 and #1}
  \end{align*}
  \]

- **'Toroidal'**
The toroidal coordinate system whose main axis coincides with the main axis of the machine:

A

The minor radius measured from the axis whose cylindrical coordinates are \((R_0, Z_0)\)

\(\Theta\)
The poloidal angle, zero on the HFS midplane, positive cw (This strange definition to get a right handed system; this also corresponds to the magnetic probe ordering)

\(\Phi\)
The toroidal angle, positive ccw seen from the top, zero between sector #16 and #1

'Flux-coordinate'

\(\rho\)
The flux coordinate, usually the square-root of the poloidal flux normalised such that it is zero on magnetic axis and one on the last close flux surface. But can also be for example the toroidal flux

\(\Theta\)
The poloidal angle, zero on the HFS midplane, positive cw (strange definition to get a right handed system; this also corresponds to the magnetic probe ordering)

\(\Phi\)
The toroidal angle, positive ccw seen from the top, zero between sector #16 and #1

'Diagnostic-Chords'

This one dimensional coordinate system allows to scan a chord of a diagnostic:

\(S\)
The linear distance along the chord measured from the chord reference point (see \texttt{psitbxcdcd})

\textbf{storage}

The character argument \texttt{storage} specifies the way the grid points are stored in the \(x\) field of the \texttt{psitbxgrid} object. This field and argument is a cell array containing three double arrays. Possible values for \texttt{storage} are:

'Grid'

In this case the grid is made of the points of a mesh specified by the three vectors of \(x\). Note that this storage method can not be used with \texttt{Diagnostic-Chords}

'Points'

This storage method is used for points which does not align on a grid. In that case all the arrays of \(x\) must have the same size and the grid is made of the points whose coordinates by each triplet. Note that the shape of these arrays is free and can reflect the organisation of the problem

'Time-Points'

This form is used for points which move in time. All the arrays of \(x\) must also be of the same size but in addition their last dimension represents time samples

For 1-D or 2-D grids, the corresponding array in \(x\) can be replaced either by a constant, for example for things that lie entirely on a plane, or by a single NaN which means that the problem does not depend on this particular coordinate. A grid can also be extended to other dimension that the basic space and time coordinates. For example a fourth dimension can be added that would represents a frequency or a wave length if necessary.

There are some combinations of \texttt{type} and \texttt{storage} for which the \(x\) argument can be given as the string 'Default', which will returns a predefined grid, in particular \texttt{psitbxgrid('Flux-Coordinate','Grid','Default')}
Optional arguments

For its points to be absolutely positioned in space, some coordinate systems requires additional information. Such an information is given in a list of optional arguments to the function `psitbxgrid`. Here is a list of these pieces of information:

**t**

The time base for 'Time-Points' storage. It can be found in:
- A double array
- The t field of a `psitbxgrid` object
- The t field of a `psitbxfun` object
- The t field of a `psitbxpsi` object
- The t field of a `psitbxdcd` object

**r0, z0**

The position of the axis for a 'Toroidal' coordinate system or of the magnetic axis for a 'Flux-Coordinate' system. Note that this axis can move in time, in which case its size must be coherent with the time base. It can be found in:
- A cell array made of two double array
- The r0 and z0 fields of a structure
- The r0 and z0 fields of the par field of a `psitbxgrid` object
- The rmag and zmag fields of a `psitbxpsi` object
- The r0 and z0 fields of the grid field of a `psitbxfun` object
- The r0 and z0 fields of the grid field of a `psitbxpsi` object

**psi**

A `psitbxpsi` object containing the distribution in cylindrical coordinate of the normalised poloidal flux. This information can be optionaly given for a 'Flux-Coordinate' grid. It can be found in:
- A `psitbxpsi` object of format '01'
- In the psi field of a structure

**fsd**

A `psitbxpsi` object containing the description of the constant poloidal flux contour in toroidal coordinates. This information can be optionaly given for a 'Flux-Coordinate' grid. It can be found in:
- A `psitbxpsi` object of format 'FS'
- In the fsd field of a structure

**dcd**

The description of diagnostic chords. This information is required for grid of type 'Diagnostic-Chords' and can be found in:
- A `psitbxdcd` object
- In the dcd field of a structure

Example

Example 1

A 'Toroidal' grid with 'Grid' storage

```matlab
x = (linspace(0,.2,6),linspace(-pi,pi,7),linspace(0,2*pi,25));
r0z0 = [1,0];
```
Example 2

Two probes moving along the Z direction

t = linspace(0, .1, 11);
x = (repmat([.6; .7], 1, 11), [t; t] − .75, pi/3);
g = psitbxgrid('Cylindrical', 'Time-Points', x, t)
g is a PsiTbx-Grid object =
    type: 'Cylindrical''
    label: {1x3 cell }
    storage: 'Time-Points'
    x: {1x3 cell }
    t: [1x11 double]
    par: [1x1 struct]
    csp: [1x1 struct]
plot(g)
**psitbxgrid/metric**

**Purpose**

Method to compute the metric of a psitbxgrid object

**Syntax**

```
f = metric(g,dim)
```

**Description**

The method `metric` applies on a `psitbxgrid` object with 'Grid' storage and returns a `psitbxfun` containing the metric along the specified dimensions `dim`. For example if `dim = 1`, the metric along the first coordinate is returned, if `dim = [2,3]`, the metric of the surface element along coordinates 2 and 3 is returned, if `dim = [1,2,3]`, the volume element is returned. In addition if `dim` is negative, the norm of the gradient of the specified coordinate is returned.

**Example**

The very important gradient of rho entering the expression of the SEF (Shape Enhancement Factor)

```
fsd = psitbxtcv(10000,'FS');
grho = metric(fsd,-1);
plot(grho)
```
Algorithm

For the flux-coordinate system, the metric matrix is not diagonal; for the others it is trivial. Here are the formulae:
Orthogonal coordinate system metric

\[
\hat{h} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

Cylindrical coordinate system metric

\[
\hat{h} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R^2 \end{bmatrix}
\]

Toroidal coordinate system metric

\[
\hat{h} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a^2 & 0 \\ 0 & 0 & (R_0 - a \cos \theta)^2 \end{bmatrix}
\]

Flux-coordinate system metric

\[
\hat{h} = \begin{bmatrix} \left(\frac{\partial \alpha}{\partial \rho}\right)^2 & \frac{\partial \alpha}{\partial \rho} \frac{\partial \alpha}{\partial \theta} & 0 \\ \frac{\partial \alpha}{\partial \rho} \frac{\partial \alpha}{\partial \theta} & a^2 + \left(\frac{\partial \alpha}{\partial \theta}\right)^2 & 0 \\ 0 & 0 & (R_0 - a \cos \theta)^2 \end{bmatrix}
\]

\[d l_\rho = \frac{\partial \alpha}{\partial \rho} d \rho\]

\[d l_\theta = \left(a^2 + \left(\frac{\partial \alpha}{\partial \theta}\right)^2\right)^{1/2} d \theta\]

\[d l_\phi = (R_0 - a \cos \theta) d \phi\]

\[d s_{\rho \theta} = a \frac{\partial \alpha}{\partial \rho} d \rho d \theta\]

\[d s_{\rho \phi} = \frac{\partial \alpha}{\partial \rho} (R_0 - a \cos \theta) d \rho d \phi\]

\[d s_{\theta \phi} = \left(a^2 + \left(\frac{\partial \alpha}{\partial \theta}\right)^2\right)^{2/3} (R_0 - a \cos \theta) d \theta d \phi\]

\[d V = a \frac{\partial \alpha}{\partial \rho} (R_0 - a \cos \theta) d \rho d \theta d \phi\]

\[\| \nabla_\rho \| = \frac{\left(a^2 + \left(\frac{\partial \alpha}{\partial \theta}\right)^2\right)^{1/2}}{\frac{\partial \alpha}{\partial \rho}} \rightarrow \left(\frac{\partial \alpha}{\partial \rho}\right)^{-1} \bigg|_{\rho = 0}\]

\[\| \nabla_\theta \| = 1/\alpha\]

\[\| \nabla_\phi \| = (R_0 - a \cos \theta)^{-1}\]
**Purpose**

Method to transform a psitbxgrid object on a different coordinate system.

**Syntax**

```matlab
g2 = psitbxg2g(g,type)
g2 = psitbxg2g(g,type,p1,...)
```

**Description**

The method `psitbxg2g` applies on a `psitbxgrid` object to transform the coordinates of its points in a new coordinate system. In the call `psitbxg2g(g,type), g` is a `psitbxgrid` object and `type` a character string that specifies the new coordinate system; it can take the value of the `type` argument of the `psitbxgrid` function. `psitbxg2g` returns a `psitbxgrid` object with the same spatial points but with coordinates expressed in the new system.

Note that the storage may be altered; in particular 'Grid' storage may be transformed in 'Points' or 'Time-Points' storage because points in the new system do not align on a grid anymore.

It is impossible to map points on a 'Diagnostic-Chords' system, since this system is discrete in space.

Some transformations require additional information that will be sought in the `par` field of the `g` argument or in additional optional arguments, following the same rules as for the `psitbxgrid` function. These transformations are:

- **Cylindrical to toroidal and toroidal to cylindrical**
  - Requires `r0` and `z0`

- **Cylindrical to Flux coordinate**
  - Requires `r0`, `z0` and a `psitbxpsi` of form '01'

- **Flux coordinate to toroidal**
  - Requires a `psitbxpsi` of form 'FS'

**Example**

Position of the chord points in toroidal coordinates:

```matlab
dcd = psitbxdcld('Demo');
s = intersect(dcd,psitbxgrid('Cylindrical','Grid','Default'));
gdc = psitbxgrid('Diagnostic-Chords','Points',{s},dcd);
g = psitbxg2g(gdc,'Toroidal',{.89,0});
plot(g)
```
Algorithm

There are a set of basic transformations defined. All the others are composed from the basic set, which is:
Orthogonal to cylindrical
\[ R^2 = X^2 + Y^2 \]
\[ Z = Z \]
\[ \tan \phi = Y / X \]

Cylindrical to orthogonal
\[ X = R \cos \phi \]
\[ Y = R \sin \phi \]
\[ Z = Z \]

Cylindrical to toroidal
\[ \alpha^2 = (R_0 - R)^2 + (Z - Z_0)^2 \]
\[ \tan \phi = (Z - Z_0) / (R_0 - R) \]
\[ \phi = \phi \]

Toroidal to cylindrical
\[ R = R_0 - \alpha \cos \theta \]
\[ Z = Z_0 + \alpha \sin \theta \]
\[ \phi = \phi \]

Cylindrical to flux-coordinate
\[ \rho^2 = \psi^*(R, Z) \]
\[ \tan \phi = (Z - Z_0) / (R_0 - R) \]
\[ \phi = \phi \]

Flux-coordinate to toroidal
\[ \alpha = \alpha^*(\rho, \theta) \]
\[ \theta = \theta \]
\[ \phi = \phi \]

Diagnostic-Chord to cylindrical
\[ R^2 = (R_d - s \cos \theta_d \cos \phi_d)^2 + (s \cos \theta_d \sin \phi_d)^2 \]
\[ Z = Z_d + s \sin \theta_d \]
\[ \tan(\phi - \phi_d) = (s \cos \theta_d \sin \phi_d) / (R_d - s \cos \theta_d \cos \phi_d) \]

```
<< psitbxpsi
```

**Purpose**

Creates a psitbxpsi object containing the poloidal flux description or the flux surface description.

**Syntax**

```matlab
psi = psitbxpsi(x,grid,t,form)
```

```matlab
psi = psitbxpsi(x,grid,t,form,rmag,zmag)
```
Description

The function \texttt{psitbxpsi} is used to create a \texttt{psitbxpsi} object, which is a child of the \texttt{psitbxfun} class that contains either the poloidal flux in cylindrical coordinates or the flux surface contours in toroidal coordinates. In the call \texttt{psitbxpsi(x,grid,t,form)} depending on the \texttt{form} argument:

'FS'
\begin{itemize}
  \item \texttt{x} is the minor radius in toroidal geometry of the surfaces of constant poloidal flux. \texttt{x} must be defined on the \texttt{psitbxgrid} object \texttt{grid} of type 'Flux-Coordinate' with storage 'Grid'. \texttt{t} is the time base.
\end{itemize}

'10','+0','-0','01'
\begin{itemize}
  \item \texttt{x} is the poloidal flux defined on the \texttt{psitbxgrid} object \texttt{grid} of type 'Cylindrical' with storage 'Grid'. The first character of \texttt{form} represents the value or the sign of the poloidal flux on the magnetic axis, the second on the last close flux surface. '+0' is the LIUQE representation, '01' the normalised flux used for flux coordinate.
\end{itemize}

Optional arguments allows to specify the position of the magnetic axis \texttt{rmag} and \texttt{zmag} and convergence parameters for the method \texttt{psitbxp2p}, \texttt{iter} and \texttt{tol}.

<< \texttt{psitbxpsi/psitbxfsg} 

Purpose

Method to compute geometrical quantities from a \texttt{psitbxpsi} object

Syntax

\begin{verbatim}
fsg = psitbxfsg(fsd)
\end{verbatim}

Description

The method \texttt{psitbxfsg} applies on a \texttt{psitbxpsi} object with format 'FS' and returns a structure with fields of class \texttt{psitbxfun}:

\begin{itemize}
  \item \texttt{grho} flux surface average of the gradient of rho
  \item \texttt{surf} flux surface area
  \item \texttt{area} flux surface cross-section
  \item \texttt{vol} volume inside a flux surface
  \item \texttt{darea} cross-section between two flux surfaces divided by the flux coordinate increment
  \item \texttt{dvol} 
\end{itemize}
volume between two flux surfaces divided by the flux coordinate increment

Example

```matlab
fad = psitbxtcv(10000,'FS');
fsg = psitbxfsg(fsd);
plot(fsg.grho)
```

Algorithm

Uses the following definitions:

- **grho**: mean(metric(g,−1),[2,3])
- **surf**: sum(metric(g,[2,3]),[2,3])
- **area**: cumsum(sum(metric(g,[1,2]),2),1)
- **vol**: cumsum(sum(metric(g,[1,2,3]),[2,3]),1)
- **darea**: sum(metric(g,[1,2]),2)
- **dvol**: sum(metric(g,[1,2,3]),[2,3])

<< psitbxpsi/psitbxp2p
### Purpose

Method to change the format of a `psitbxpsi` object

### Syntax

```matlab
psi = psitbxp2p(psi,'01')
fsd = psitbxp2p(psi,'FS')
fsd = psitbxp2p(psi,'FS',g)
```

### Description

The method `psitbxp2p` applies on a `psitbxpsi` object and returns a `psitbxpsi` object with the specified form:

- `'01'`  
  The poloidal flux contained in the `psi` argument will be searched for its maximum, normalised and the location of the maximum stored in the `rmag` and `zmag` fields

- `'FS'`  
  The contours of constant normalised poloidal flux will be searched. A third argument allows to specify the flux-coordinate grid on which these contours are to be sought.

### Example

```matlab
psi = psitbxtcv(10000,.5);
fsd = psitbxp2p(psi,'FS');
plot(fsd)
```
Algorithm

Uses Gauss–Newton iterations to locate either the maximum or the contour level. Iteration convergence is controlled by the \textit{iter} and \textit{tol} fields of the \texttt{psitbxpsi} object.

\texttt{psitbxtcv}

Purpose

Easy access to quantities relevant for the Psi–Toolbox directly from the TCV shot files

Syntax

\begin{verbatim}
psitbxtcv(shot,form)
psitbxtcv(shot,time,form)
\end{verbatim}

Description

The function \texttt{psitbxtcv} builds object compatible with the Psi–Toolbox from the TCV shot files. Depending on the \textit{form} argument it returns:

\begin{itemize}
  \item \texttt{'+0'}
    \begin{itemize}
      \item A \texttt{psitbxpsi} object of form '\+0' containing the poloidal flux from the equilibrium reconstruction by LIUQE
    \end{itemize}
  \item \texttt{'01'}
    \begin{itemize}
      \item A \texttt{psitbxpsi} object of form '01' containing the poloidal flux from the equilibrium reconstruction by LIUQE normalised using the \texttt{psitbxp2p} method
    \end{itemize}
  \item \texttt{'FS'}
    \begin{itemize}
      \item A \texttt{psitbxpsi} object of form 'FS' containing the flux surface description as precalculated by the Psi–Toolbox and stored in the shot file
    \end{itemize}
  \item \texttt{'JPHI'}
    \begin{itemize}
      \item A \texttt{psitbxfun} object containing the toroidal current density from the equilibrium reconstruction by LIUQE
    \end{itemize}
\end{itemize}

An optional time argument can be specified. Only the times for which a sample in the shot file at most 1e–6 second away exists are returned.

Example

\begin{verbatim}
psi = psitbxtcv(10000,'+0')
psi is a PsiTbx-Poloidal-Flux object =
  format: '+0'
  rmag: []
  zmag: []
  psitbxfun: [28x65x46 psitbxfun]
  iter: 10
  tol: 1.0000e-04
plot(psi)
\end{verbatim}