

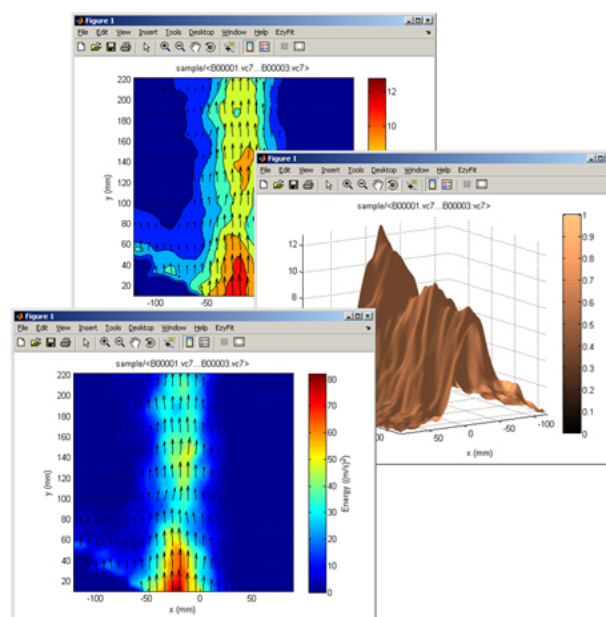


# pivmat

A PIV post-processing and  
data analysis toolbox for Matlab

## PIVMat 1.60 Reference Manual

April 2007, 17



Frédéric Moisy

FAST - Univ. Paris Sud & CNRS UMR 7608

[moisy@fast.u-psud.fr](mailto:moisy@fast.u-psud.fr)

# PIVMat Getting Started

---

The PIVMat Toolbox contains a set of command-line functions to import, post-process and analyse 2D fluid flows velocity fields from [LaVision](#)'s DaVis PIV (particle image velocimetry) software. It enables to handle and perform operations over large amount of velocity fields, and to produce high-quality vector/scalar outputs. The PIVMat Toolbox in itself does not perform any PIV computations.

## What is DaVis?

DaVis (Data Acquisition and Visualization Software) is a general purpose commercial software developed by [LaVision GmbH](#). In its PIV (particle image velocimetry) use, DaVis computes velocity fields from images of particles. These velocity fields can be saved in a specific DaVis file format (VEC or VC7 files), which can be imported in MATLAB with the ReadIMX Loader package provided by LaVision. Based on this package, the PIVMat Toolbox offers a number of command-line functions to further post-process and analyse the data from DaVis.

## Examples

The first step of the PIVMat Toolbox is to import DaVis VEC or VC7 files into a MATLAB structure (or structure array) using the function [loadvec](#) (or [loadset](#)). This structure contains the velocity fields, axes, units etc., and can be processed by several functions of the toolbox.

Once imported, the velocity fields can be displayed using [showvec](#), or converted into scalar fields using [vec2scal](#).

A sample directory, named `sample`, with 3 VC7 fields, is provided with the toolbox to test the following examples (note that all the commands work with both VEC and VC7 files).

### Example 1

```
v = loadvec('B00001.VC7');           % loads the file B00001.VC7
showvec(v);                          % displays it
curl = vec2scal(filterf(v,2),'rot'); % computes the filtered vorticity
figure, showscal(curl);              % displays it
```

### Example 2

Various statistics may be computed from vector and scalar fields:

```
v = loadvec('sample/*.vc7');         % loads all the directory sample
curl = vec2scal(filterf(v,2),'rot'); % computes the filtered vorticity
showscal(curl);                      % displays it as a movie
showscal(averf(curl));               % displays its ensemble-average
histscal_disp(curl);                 % displays its histogram
statf(v)                             % computes some statistics
```

Go to the [Frequently Asked Questions](#) section or to the [Function by category](#) section to learn more about this toolbox.

2005-2007 [PIVMat Toolbox](#)

# PIVMat Installation

---

## Compatibility

The PIVMat Toolbox works with MATLAB 7 or higher, 32-bits version only. To process Davis files, it is necessary to install first the MATLABIMX package (version 1.4 or higher) provided by LaVision -- except the March 2006 release, which does not work.

The PIVMat Toolbox reads all the files from DaVis 6 and 7 (VEC, VC7, IMG, IMX, IM7, SET and EXP) and files from MatPIV (MAT).

Note that the MATLABIMX package works only on the 32-bits version of Matlab (default version). If you have a 64-bits Windows XP edition, you will need to install the 32-bits version of Matlab to run the PIVMat toolbox.

## Installation procedure

1. Download the latest version of MATLABIMX ('ReadIMX Loader package for MATLAB') from the 'Download' area of the LaVision web site [www.lavision.de](http://www.lavision.de) (you need to sign up first and login to access to the Download area). Extract the ZIP file in an installation folder on your computer (eg, / toolbox/readimx).
2. Download the PIVMat Toolbox and extract the ZIP file in another installation folder (eg, / toolbox/pivmat) (make sure the subdirectories `html`, `sample` and `private` are correctly unzipped as well). If you upgrade from an older version, first empty the previous directory.
3. From the menu 'File > Set Path', click on 'Add Folder' and select the two directories `readimx` and `pivmat`. Click on 'Save' and 'Close'.
4. Restart MATLAB. To get started, type `doc pivmat`, or select Toolboxes > PIVMat from the Start button.
5. If you upgrade from an earlier version, type `rehash toolbox` to make sure that MATLAB refreshes the function cache.

2005-2007 [PIVMat Toolbox](#)

## PIVMat Function Reference

# Functions -- By Category

---

### Basic VEC/VC7 and SET files operations

- [loadvec](#) - Load vector/image field(s)
- [loadset](#) - Load set(s) of vector/image fields
- [showvec](#) - Display vector field(s)
- [showscal](#) - Display scalar field(s)
- [showf](#) - Shortcut for SHOWVEC/SHOWSCAL
- [vec2scal](#) - Convert vector into scalar field(s)

### Field processing (both for vector and scalar fields)

- [filterf](#) - Spatial filter of a field
- [bwfilterf](#) - Spatial Butterworth filter of a field
- [averf](#) - Average of several fields
- [spaverf](#) - Spatial average over X and/or Y of a field
- [azaverf](#) - Azimuthally averaged field
- [subaverf](#) - Subtract the average of field
- [truncf](#) - Truncate a field
- [extractf](#) - Extract a rectangle from a field
- [rotatef](#) - Rotate a field about the center
- [shiftof](#) - Shift the axis of a field
- [operf](#) - Other operations on fields
- [addnoise](#) - Add noise to a field
- [gradientf](#) - Gradient of a scalar field

### Histograms & statistics

- [statf](#) - Mean, standard deviation and rms of a field.
- [histf](#) - Histogram of a field.
- [corr](#) - Correlation function of a scalar field.
- [histvec\\_disp](#) - Display histograms from vector fields.
- [histscal\\_disp](#) - Display histograms from scalar fields.
- [statvec\\_disp](#) - Display statistics from vector fields.
- [vsf](#) - Structure functions.
- [vsf\\_disp](#) - Display structure functions.
- [jpdfscal](#) - Joint probability density function.
- [jpdfscal\\_disp](#) - Display joint PDF

### Advanced VEC/VC7 and SET files operations

- [loadpivtxt](#) - Load a DaVis vector field exported in TXT mode
- [loadarrayvec](#) - Load a 2D array of vector fields
- [batchvec](#) - Execute some functions for a series of files

[vec2mat](#) - Convert DaVis files into MAT-Files

### **VEC/VC7 and SET Attribute handling**

[getattribute](#) - Get attributes from an IMX or VEC/VC7 file

[readsetfile](#) - Get attributes from the definition .SET file of a set

[getpivtime](#) - Returns the time of IMX or VEC/VC7 files

### **Miscellaneous files handling (from Fileseries v1.20)**

[rdir](#) - Recursive list directory.

[rdelete](#) - Delete files recursively.

[rrmdir](#) - Delete directories recursively.

[renamefile](#) - Rename a series of files.

[renumberfile](#) - Re-number the indices of a series of files

[getfilenum](#) - Get the index of a series of files.

[expandstr](#) - Expand indexed strings.

### **Miscellaneous**

[getsetname](#) - Get the name of the current set

[randvec](#) - Random vector field.

[vortex](#) - Vector field with a vortex.

[surfheight](#) - Surface height reconstruction.

2005-2007 [PIVMat Toolbox](#)

## PIVMat Function Reference

# Functions -- Alphabetical List

---

[about\\_pivmat](#)

[addnoise](#)

[averf](#)

[azaverf](#)

[batchvec](#)

[bwfilterf](#)

[checkupdate\\_pivmat](#)

[corr](#)

[expandstr](#)

[extractf](#)

[filterf](#)

[getattribute](#)

[getfileenum](#)

[getimx](#)

[getpivtime](#)

[getsetname](#)

[gradientf](#)

[histf](#)

[histscal\\_disp](#)

[histvec\\_disp](#)

[jpdfscal](#)

[jpdfscal\\_disp](#)

[loadarrayvec](#)

[loadpivtxt](#)

[loadset](#)

[loadvec](#)

[operf](#)

[randvec](#)

[rdelete](#)

[rdir](#)

[readsetfile](#)

[renamefile](#)

[renumberfile](#)

[rotatef](#)

[rrmdir](#)



[shiftf](#)

[showf](#)

[showscal](#)

[showvec](#)

[spaverf](#)

[statf](#)

[statvec\\_disp](#)

[subaverf](#)

[surfheight](#)

[truncf](#)

[vec2mat](#)

[vec2scal](#)

[vortex](#)



2005-2007 [PIVMat Toolbox](#)

## PIVMat Frequently Asked Questions

# PIVMat Frequently Asked Questions

---

If you have a question which is not answered here, or a suggestion on how you would improve this section, please feel free to send an e-mail to the author, [moisy@fast.u-psud.fr](mailto:moisy@fast.u-psud.fr).

## Basic questions

1. [How to import and display velocity fields from Davis?](#)
2. [How to have a quick help on a function from the PIVMat toolbox?](#)
3. [What Matlab knowledge should I have to use the PIVMat toolbox?](#)
4. [My velocity fields are noisy, how to smooth?](#)
5. [How can I display the vorticity from a velocity field?](#)
6. [I just want to display the first 10 fields of my set!](#)
7. [Can I import an IMX or IM7 image file?](#)
8. [Can I import a vector field saved in TXT mode?](#)
9. [Why all the vectors are not displayed with showvec?](#)
10. [What is the difference between the ReadIMX package provided by LaVision and the PIVMat toolbox?](#)

## Advanced questions

1. [How to plot lines of iso-velocity?](#)
2. [How to navigate into a set of fields?](#)
3. [How are stored the vector fields in Matlab?](#)
4. [How can I plot a velocity profile from a vector field?](#)
5. [How can I plot a vorticity profile?](#)
6. [How can I create an AVI movie file from my vector fields?](#)
7. [What is the mean kinetic energy of my field?](#)
8. [How to save fields processed with Matlab?](#)
9. [Where is stored the acquisition time of a vector field?](#)
10. [How can I compute the rms \(root-mean-square\) of a series of vector fields?](#)
11. [How can I plot the velocity at some given point as a function of time?](#)
12. [How can I plot a spatio-temporal diagram from a vector field?](#)
13. [What are all the successive operations performed on a given field?](#)
14. [Should I always use `loadvec` to display/process my set of files?](#)
15. [Can I display the current time in my movie?](#)

## Expert questions

1. [I have to process 1000 vector fields but I haven't enough memory!](#)
  2. [I want to load n sets of p vector fields!](#)
  3. [How to compute ensemble averages from my n sets of p fields?](#)
  4. [How to compute scalar fields other than the default ones proposed in vec2scal?](#)
  5. [How to load multiframe fields?](#)
  6. [What is the variable ysign?](#)
- 

## Basic questions

### 1. How to import and display velocity fields from Davis?

Go to the directory where your VEC or VC7 files are stored (a sample directory `sample` with 3 velocity fields is present in the `pivmat` directory). Type

```
v = loadvec('B00001.vec');
```

to load a single file (here `'B00001.vec'`). To load all the VEC files in the current directory, type

```
v = loadvec('*.vec');
```

To display the fields, type

```
showvec(v);
```

There are several shortcuts: For instance, this will display a movie of all the files in the current directory:

```
showvec('*.vec');
```

### 2. How to have a quick help on a function from the PIVMat toolbox?

To get information about the function `loadvec`, type

```
doc loadvec
```

(open the help browser with the help for this function), or

```
help loadvec
```

(display the text of the help in the command window). If you don't know what you look for, have a look to the [Function by category](#) section from the PIVMat main page:

```
doc pivmat
```

### 3. What Matlab knowledge should I have to use the PIVMat toolbox?

You should be familiar with the basic Matlab syntax, especially the use of colons (:) to index arrays, and the use of structures and structure arrays.

### 4. My velocity fields are noisy, how to smooth?

Use [filterf](#) or [bwfilterf](#), eg:

```
showvec(filterf(v,2));
```

### 5. How can I display the vorticity from a velocity field?

First load your vector field with

```
v = loadvec('B00001.vec');
```

Then display it with the background option 'rot'

```
showvec(v, 'rot');
```

Another way to do this is to create a scalar field curl,

```
curl = vec2scal(v, 'rot');
```

and then to display it:

```
showscal(curl);
```

Before computing the vorticity field, you will probably want to filter your vector field:

```
showvec(filterf(v,1), 'rot');
```

Many other vector-to-scalar conversions (divergence, kinetic energy, velocity derivatives...) are available from [vec2scal](#).

## 6. I just want to display the first 10 fields of my set!

You may load all the vector fields and display only the ones you want:

```
v = loadvec( '*' .vec' );
```

Just refer to the fields you want with `v(n)`, where `n` is a number (eg, `showvec(v(4))` displays the field #4).

If you want to refer to a range of fields, `n` may be an array, like `[1 3 8]`, or `1:10` (type `doc colon` to learn more about Matlab's colon (:)). So, to display only the first 10 fields, type

```
showvec(v(1:10));
```

If you want to skip one field every 2, type

```
showvec(v(1:2:10));
```

You may also only load the fields you want, by using wildcards (\*) and/or brackets []. In order to load only the files `B00001.vec` to `B00010.vec`, type

```
v = loadvec('B[1:10].vec');
```

See [expandstr](#) for details about the usage of the brackets []. There is also a shortcut to do so,

```
v = loadvec(1:10);
```

(in this case, the 10 first files are loaded in the alphabetically-ordered set of files of the current directory).

## 7. Can I import an IMX or IM7 image file?

Yes, [loadvec](#) accepts IMX/IM7 files; they will be considered as normal scalar fields, ie: use [showscal](#) to display them. You may use `colormap gray` to use a black-and-white color map, which is usually more suitable for images.

## 8. Can I import a vector field saved in TXT mode?

Yes. Simply use [loadvec](#) (see also [loadpivtxt](#)). However, it is better to use directly `VEC/VC7` files: it is faster, and these files contain additional informations.

## 9. Why all the vectors are not displayed with showvec?

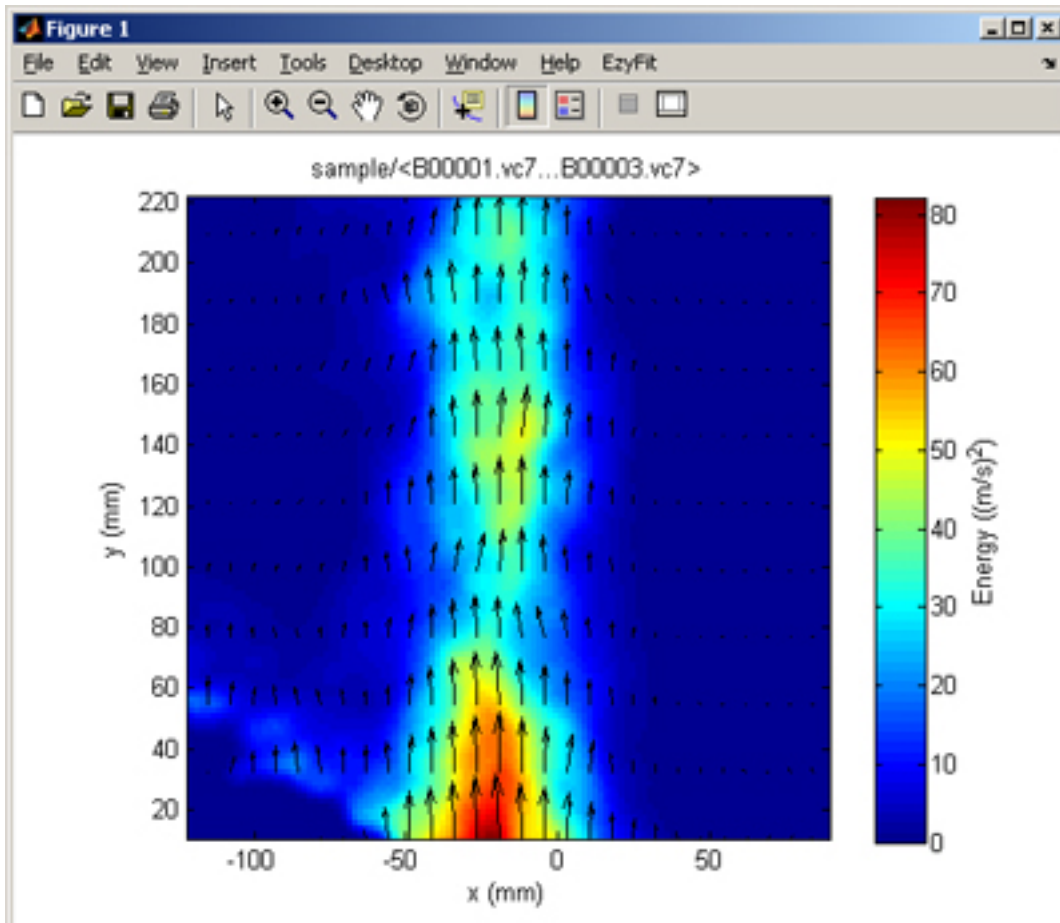
A vector field is a bit confusing when too many vectors are present in a field. By default, [showvec](#) displays at most 32 vectors in each direction, whatever the vector field resolution. This

can be changed using the 'Spacing' option of [showvec](#), eg:

```
showvec(v,'norm','spacing',2);
```

You may also display specify different spacing in the x- and y-direction:

```
showvec(filterf(v,1.5),'ken','spacing',[4 12]);
```



## 10. What is the difference between the MatlabIMX package provided by LaVision and the PIVMat toolbox?

The MatlabIMX package ('ReadIMX Loader package for Matlab') provided by LaVision allows to import a (single) VEC/VC7/IMX/IM7 file from Davis into Matlab, and to display it. That's all.

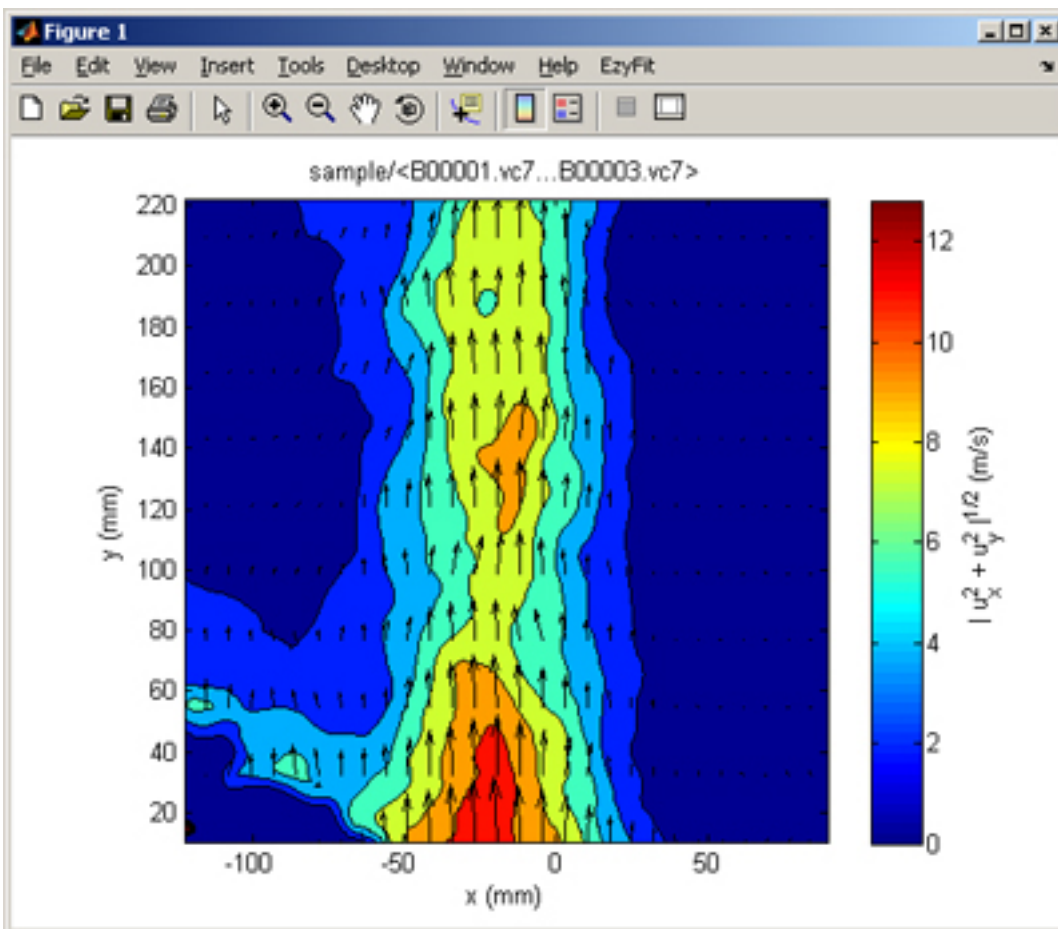
The PIVMat toolbox is based on the function `readimx` from the MatlabIMX package, but offers much more. Note that the MatlabIMX package has to be installed separately for the PIVMat toolbox to work correctly (see the [Installation](#) page).

## Advanced questions

### 1. How to plot lines of iso-velocity?

Use [showvec](#) or [showscal](#) with the option 'contour' or 'contourf', eg

```
showvec(v,'contourf',8,'spacing',[4 12]);
```



## 2. How to navigate into a set of fields?

First load your vector fields with

```
v = loadvec('*.vec');
```

Display a movie of your fields,

```
showvec(v);
```

You can enter into the 'pause' mode by pressing the space bar. Then navigate using the left or right arrow. Press the space bar to run the movie again. Press ESC to escape. Use option 'loop' to run the movie in a loop. You may also directly enter into the 'pause' mode by specifying the option 'pause'. See [showvec](#) or [showscal](#) for more movie options.

## 3. How are stored the vector fields in Matlab?

A vectors field loaded with

```
v = loadvec( 'B00001.vec' );
```

is stored into a Matlab "structure" `v`. A Matlab structure is a variable that contains several variables of different types (see 'Data Types > Structures' in the Matlab help). The structure `v` contains the axis, the velocity components, the field names, and many attributes saved by Davis (acquisition time, etc). For instance, the x-component of the velocity field is stored in the matrix `vx` of the structure `v`. In order to refer (eg, plot, display, modify...) to the x-component of the velocity at the point  $(i, j)$ , you should type

```
v.vx(i, j)
```

If you have loaded several fields in a single structure `v`, eg with

```
v = loadvec( '* .vec' );
```

then `v` is now a "structure array". You can refer to the  $(i, j)$  point of the x-component of the  $n$ -th vector field with

```
v(n).vx(i, j)
```

See [loadvec](#) for a full description of what is in the structure `v`.

#### 4. How can I plot a velocity profile from a vector field?

The velocity components are stored in the two arrays `vx` and `vy` of the structure `v` loaded by `loadvec` (see question above). So, to plot a velocity profile along the horizontal line at height  $y=12$  (in mesh units), type

```
plot(v.x, v.vx(:, 12))
```

(type `doc colon` to learn more about Matlab's colon (:)).

You may also plot the velocity profile averaged over several horizontal lines, for instance between  $y=12$  and  $y=14$  (in mesh units):

```
plot(v.x, mean(v.vx(:, 10:14), 2))
```

#### 5. How can I plot a vorticity profile?

First compute the vorticity from a velocity field,

```
vort = vec2scal(v, 'rot');
```

Then follow the same procedure as above. The vorticity is contained into the variable `wof` of the structure `vort`:



```
plot(vort.x, vort.w(:,12))
```

## 6. How can I create an AVI movie file from my vector fields?

First load your vector fields with

```
v = loadvec('*.vec');
```

Then create the movie in a variable mov:

```
mov = showvec(v, 'norm');
```

(here the norm of the field is used for the background color). Then save your movie into an AVI file with

```
movie2avi(mov, 'mymovie.avi');
```

See [showvec](#) or [showscal](#) to learn more about movies.

## 7. What is the mean kinetic energy of my field?

First compute the kinetic energy of your field v

```
ke = vec2scal(v, 'ken');
```

Then call the function statf

```
st = statf(ke);
```

What you want is the mean of ke:

```
st.mean
```

## 8. How to save fields processed with Matlab?

Suppose you have computed complex operations from a series of vector fields,

```
v=loadvec('*.vc7');
v=shiftof(extractf(v,[10 10 100 100],'phys'),'center');
fv=bwfilterf(v, 2, 8);
curl=vec2scal(fv, 'rot');
```

Then you can save your final set of scalar fields `curl` in a Matlab file,

```
save('mycurl.mat','curl');
```

You can retrieve your fields using

```
rot = loadvec('mycurl.mat');
```

Note that you can also load your fields using the standard Matlab's `load` function,

```
load mycurl
```

In this case, the field name (here 'curl') will be the same as the one specified to the `save` command.

## 9. Where is stored the acquisition time of a vector field?

The acquisition time, and other parameters of Davis, are stored in the variable `Attribute` of a field `v`. Use the function [getattribute](#) to get the value of a given attribute. For instance, for VC7 files (from Davis 7), the time acquisition is stored in the attribute `AcqTimeSeries0`:

```
t = getattribute(v,'AcqTimeSeries0');
```

returns the time `t` of `v` (or the array of times, if `v` is an array of fields). You may also use the function [getpivtime](#), which is a shortcut for this attribute.

For files from Davis 6, please note that the acquisition time is *\*not\** saved by default in the VECfile, but in the image IMXfile! See [getpivtime](#).

## 10. How can I compute the rms (root-mean-square) of a series of vector fields?

First load your vector fields with

```
v = loadvec('*.vc7');
```

Then use [averf](#) with the following output arguments and plot the result:

```
[af rms] = averf(v);  
showscal(rms);
```

## 11. How can I plot the velocity at some given point as a function of time?

First load your vector fields with

```
v = loadvec('*.vc7');
```

Then create an array where the time is stored,

```
t = getpivtime(v, '0');
```

(the option '0' ensures that the first field has time  $t = 0$ ). Then store the x-component of your point of interest, say (12,20) (in mesh units), into the array 'vel',

```
for i=1:length(v),
    vel(i)=v(i).vx(12,20);
end;
```

then plot the result

```
plot(t, vel);
```

If this does not work, then the time is not stored into your files. See [getpivtime](#) and [getattribute](#) to solve this.

## 12. How can I plot a spatio-temporal diagram from a vector field?

You should build a matrix, say  $d(i, j)$ , where the quantity of interest is a vector (index  $i$ ) and the time is along  $j$ . Once the vector fields  $v$  and the time  $t$  are obtained (see the above question), type

```
for j=1:length(v)
    d(:,j) = v(j).vx(:,12);
end;
imagesc(v(1).x, t, d);
xlabel('x'); ylabel('t');
```

## 13. What are all the successive operations performed on a given field?

Among the variables stored in the structure  $v$ , one is called "history". This variable is a cell array of strings, and each operation adds a new string to this cell array, which describes this operation. For instance, suppose you do

```
v=loadvec('B00001.vc7');
v=shiftof(extractf(v,[10 10 100 100],'phys'),'center');
fv=bwfilterf(v, 2, 8);
curl=vec2scal(fv, 'rot');
```

If you type

```
curl.history
```

you obtain the following cell array of strings:

```
'loadvec('B00001.vc7')'
'extractf(ans, [10 10 100 100], 'phys')'
'shiftf(ans, 'center')'
'bwfilterf(ans, 2, 8, 'low')'
'vec2scal(ans, 'rot', 'nonzero')'
```

In this list, `ans` refers to the result of the previous operation. See `Programming > Data Types > Cell arrays` in the Matlab help to learn more about cell arrays.

#### 14. Should I always use `loadvec` to display/process my set of files?

Most of the time you have to first load your files, e.g. `v=loadvec('*.vc7')`, and then you perform your computations from `v`, e.g. `showvec(filterf(v,2))`. However, a number of operations can be performed directly from the files themselves. For example, try

```
filterf('*.vc7',2);
```

When a string is passed as the first argument of `filterf`, the file is first loaded by `loadvec`. When no output argument is specified, then `showvec` or `showscal` is automatically called. Numeric arguments may also be used:

```
averf(1:10);
```

displays the average of the 10 first files on the current directory.

#### 15. Can I display the current time in my movie?

Yes, you may use the option `'title'` in `showvec` or `showscal`, e.g.

```
showvec(v,'loop','title','\s, t = \t s');
```

In this example, the string `\s` is replaced by the setname and the string `\t` is replaced by the current time, as obtained by `getpivtime`.

## Expert questions...

#### 1. I have to process 1000 vector fields but I haven't enough memory!

Loading a large amount of fields into a single big structure with `v=loadvec('*.vc7')` is not always possible, so you should process your files one by one without storing the whole set. For simple operations, you may use the function [batchvec](#)

```
batchvec( '*.vc7', 'showvec', 'norm' );
```

This calls sequentially the function `showvec` for each file matching `'*.vc7'` in the current directory, passing the additional parameter `'norm'`.

More complex operations should be performed using a `for` loop:

```
file = rdir( '*.vc7' );
for i=1:length(file)
    v = loadvec(file{i});
    st = statf(vec2scal(v, 'ken'));
    k(i) = st.mean;
end;
plot(k);
```

(with the use of curly braces `{}` for cell arrays of strings - see [rdir](#) for details).

## 2. I want to load $n$ sets of $p$ vector fields!

If you want to concatenate  $n$  sets of  $p$  vector fields into a single big structure  $v$  of size  $n \times p$ , assuming the fields are stored into directories named `myset1`, `myset2`..., use

```
v = loadvec( 'myset*/*.vc7' );
```

or

```
v = loadset( 'myset*' );
```

If now you prefer to store your fields into a 2D array of fields, use [loadarrayvec](#)

```
v = loadarrayvec( 'myset*', '*.vc7' );
```

Use something like `showvec(v(1,:))` to display all the fields from the first directory, or `showvec(v(:,1))` for all the first fields from each directory.

## 3. How to compute ensemble averages from my $n$ sets of $p$ fields?

First load your fields with (see previous question):

```
v = loadarrayvec( 'myset*', '*.vc7' );
```

Then you can average all the first fields from each directory, all the second fields from each directory, etc:

```
for i=1:size(v,1),
```

```

        af(i) = averf(v(i,:));
    end;
    showvec(af);

```

#### 4. How to compute scalar fields other than the default ones proposed in `vec2scal`?

[vec2scal](#) allows to convert vector fields into most classical scalar fields of interest (vorticity, divergence etc.) For more exotic operations, you may use [operf](#). You may also write your own `.m` file, starting from a basic scalar field. For instance, suppose you want to compute a scalar field given by  $2*v_x + v_y^2$ :

```

function res = strangeField(v)
res = vec2scal(v,'norm');
res.w = 2*v.vx + v.vy.^2;
res.namew = 'strange field';

```

Then simply call your function with:

```

showscal(strangeField(loadvec('B00001.vec')));

```

#### 5. How to load multiframe fields?

Davis' multiframe fields are not supported by PIVMat. `loadvec` will load only the first frame of a field. You should first use Davis to expand your multiframe buffer into several single-frame buffers, and then load the resulting buffers as usual with `loadvec`.

#### 6. What is the variable `ysign`?

The calibration of the Y-axis from Davis may be upward or downward. In Matlab, the natural way to plot a matrix using the function `image` is using a downward Y-axis (default mode `axis ij`, for matrix representation). When using upward an Y-axis (mode `axis xy`, for function representation), Matlab re-orders the Y vector, so the Y-axis is wrong.

The variable `ysign` in the structure `v` is a string, that may be 'Y axis downward' or 'Y axis upward', which is determined by `loadvec` when loading the field.

# PIVMat Release Notes

---

See the [PIVMat known software problems](#) section.

V1.60 2007/04/17 **Major change:** Scalar fields built from derivatives (e.g., curl, divergence, strain rate etc.) are now computed from 2nd order centered differences, using Matlab's built-in numerical derivative functions, whereas they were computed from a 1st order scheme in the previous versions (see [vec2scal](#)). The resulting scalar field has now the same size as the original vector field. Specify 'rot1','div1' etc. instead of 'rot','div' to use the old 1st order scheme (for which the resulting scalar field is smaller than the original vector field).

**Movie player improved** (both [showvec](#) and [showscal](#)):

- Movies may now be displayed within docked windows. If the window is not docked, it will stay in front of all other windows ('modal' WindowStyle).
- Option 'Jump' (or 'Go') by pressing the key 'j' or 'g'.
- Enter into the pause mode by pressing the arrow keys
- Escape the movie by pressing ESC, 'q', 'x' or Ctrl+C (works only with selected windows if they are docked) or simply by closing the window if it is undocked.

**New functions:**

- [showf](#): shortcut for showvec or showscal.
- [vsf](#) (vector structure functions) and [vsf\\_disp](#) (displays the result of vsf).
- [jpdfscal](#) (joint probability density function of two scalar fields) and [jpdfscal\\_disp](#) (displays the result of jpdfscal).
- [surfheight](#): Computes the surface height from a displacement vector field.

**New file format supported:** loadvec now accepts Mat-files saved from MatPIV (see <http://www.math.uio.no/~jks/matpiv/>).

**Minor changes:**

- [showvec](#): New option 'colorvec' (specifies the color of the vector arrows).
- [showvec](#) and [showscal](#): New options 'cmap' (specifies the color map, 'gray','jet'...) and 'title' (e.g., display the time of the field in the title).
- [loadvec](#): The structure V contains a new field choice, where the number of 1st, 2nd etc. choice vectors are stored.
- [renumberfile](#), [getfilenum](#) and [renamefile](#): New options 'fileonly', 'dironly'.
- [getfilenum](#) now works with both integer and real indices.
- [batchvec](#): New option 'nodisp'.

**Bugs fixed:** vortex (the undefined field setname caused problems with showvec), operf (minor bug for scalar outputs).

V1.51 2006/09/08 **Minor change:** [vec2mat](#) now works with all the file formats supported by loadvec (VEC/VC7/IMX/IM7/IMG/TXT/SET).

**Minor change:** [azaverf](#) now does not return the leading zeros when the center is outside the field, and does not include zero (erroneous) elements in the computation of the azimuthal average by default (see option 'keepzero').

**Bugs fixed:** renumberfile, getfilenum, renamefile (FileSeries 1.20).

V1.50 2006/07/21 **New** [Frequently Asked Questions](#) section.

**Major change:** New syntax for [showvec](#) and [showscal](#): now works on the basis of PropertyName/PropertyValue pairs.

New options ('surfl', 'mesh', 'delay', 'backward', 'loop'...).

**Movie viewer improved** ([showvec](#) and [showscal](#)): it is now possible to enter/exit the 'pause' mode during movies.

[showscal](#) now does not refresh the camera settings for 3D views (see the option 'KeepCameraSettings').

New functions [azaverf](#) (azimuthal average) and [gradientf](#).

[rotatef](#) can now rotate a field about a point different from the center.



New options for [shiftf](#) (bottomright, topleft...).

Coordinates in [extractf](#) and [truncf](#) may now be specified in physical or mesh units.

[loadvec](#) now accepts old IMG files (uncompressed DaVis 6 images).

V1.41 2006/05/22 New function [corrxf](#) (correlation function).

[vec2scal](#) now considers vectors with a zero component as erroneous (or masked), and does not use them for the computation of derivative fields (rot, div, duxdx...)

New option for [truncf](#): truncates to the smallest rectangular excluding zero (erroneous or masked) elements.

Bugs fixed with [randvec](#) (history field) and [showvec](#) (lines of other figures were turned to black).

V1.40 2006/04/28 [loadvec](#) and [loadset](#) now accept both vector fields (VEC/VC7) and images (IMX/IM7).

New functions [getattribute](#), [readsetfile](#) and [getpivtime](#). Function `getimxtime` obsolete.

[loadvec](#) now checks that the ReadIMX package (LaVision) is installed.

Field 'history' is now a cumulative cell array of strings.

Bugs fixed for [bwfilterf](#) and [rotatef](#).

Several functions moved to a subdirectory 'private'.

Better compatibility of the documentation with the MATLAB help browser.

**V1.30 2005/12/20 The toolbox is now called PIVMat.**

Documentation improved. New 'Known bugs' section.

Standard items for the toolbox in the MATLAB 'Start' menu.

New about and checkupdate. New home page ([www.fast.u-psud.fr/pivmat](http://www.fast.u-psud.fr/pivmat)).

File 'readme.txt' retired.

Bugs fixed for loadvec, due to changes in ReadIMX 1.4 (changes in the VY sign for Y-upward fields).

Multiple file selection allowed.

Bug fixed for showvec (bounds).

**V1.20 2005/11/18 LaVision's MATLABIMX package not included any more; it has to be installed separately.**

showvec improved (accept file numbers).

vec2scal improved (option 'angle').

**V1.13 2005/10/30 New functions vec2mat, operf, batchvec, addnoise.**

Bug fixed for filterf.

statf improved (history).

V1.12 2005/10/21 The bugs with up/downward Y axis are fixed in loadvec, loadpivtxt, vec2scal, showvec and showscal.

New functions shiftf and loadarrayvec.

showvec and showscal improved (pause mode).

vec2scal improved (option minus).

spaverf now computes average of non zero elements.

vortex improved (burgers vortex with divergence).

Online help files from m2html.

V1.11 2005/10/14 truncf, extractf, bwfilterf, filterf, rotatef, averf, subaverf and ensaverf work now for both vector and scalar fields (truncvec, truncscal etc. retired).

readpivtxt and dav2mat retired, replaced by loadpivtxt.

loadvec now accepts TXT, SET and MAT files.

New functions from Fileseries toolbox 1.01 (rrmdir, rdelete).

V1.10 2005/10/07 New functions getfilenum, rdir, renamefile.

renumberfile improved.

completefilename and pickfile retired (see rdir).

loadvec and loadset improved (uses rdir, txt files allowed).

showvec(-scal) improved (colormap label, titles and arrow scaling).

submeanvec improved, now called subavervec(scal).

New functions avervec(scal) and ensavervec(scal).

- V1.05 2005/09/29 loadset and loadvec improved ('nosave' option added, filename expansion).  
filtervec(-scal) improved.  
New functions expandstr and pickfile.  
buildfilename retired.
- v1.04 2005/09/23 Upward and Downward Y-axis recognized: Files that have a downward Y-axis have the v\_y component inversed.
- v1.03 2005/09/09 Sample directory and readme.txt file (this one) added.  
loadset and showvec improved.
- v1.02 2005/09/05 loadvec and loadset improved: multiple files/sets and wildcards allowed.
- v1.01 2005/08/31 First online release.  
Compatible with DaVis 7.  
New functions included (filters, stats...)
- v1.00 2005/02/23 First release (original name: davis toolbox)

## Acknowledgements

This toolbox has been extensively tested by C. Morize during his PhD.

LaVision and J. Heers are acknowledged for the ReadIMX Loader package.

J. Kristian Sveen is acknowledged for allowing the connection with his PIV toolbox "MatPIV".

J. D'Errico for his function `intgrad2.m` used in `surfheight`.

Acknowledgements for bug reports and suggestions to S. Kiesgen, L. Messio, J. Casoli, M. D'Olce, J. Seiwert, Z. Nagel and A. Muller.

2005-2007 [PIVMat Toolbox](#)

## PIVMat Known software problems

# PIVMat Known software problems

---

- **When I use `readimx` or `loadvec`, I obtain the errors '??? Out of memory. Type HELP MEMORY for your options', or '??? ReadIMX error #4: Invalid data! '.**

This problem originates from a buggy release of LaVision's ReadIMX1.4. You should check the date of your file '`readimx.dll`' in the `readimx` directory. November 2005 and May 2006 releases work well, but the March 2006 release was buggy. Upgrade your ReadIMX package directly from the LaVision homepage [www.lavision.de](http://www.lavision.de) (menu 'Download > DaVis Tools', see the step 1 of the [installation procedure](#)).

- **PIVMat does not work on Windows XP 64-bits edition.**

The MATLABIMX package from LaVision works only on the 32-bits version of Matlab (default version). If you have a 64-bits Windows XP edition, you will need to install the 32-bits version of Matlab to run the PIVMat toolbox.

- When using file selection from a dialog box in [loadvec](#), the titles from [showvec](#) are buggy.

---

2005-2007 [PIVMat Toolbox](#)

# about\_pivmat

About PIVMat

## Description

**about\_pivmat** displays the 'about' informations in the command window.

**about\_pivmat**('dialog') displays the 'about' info in a dialog box.

## See Also

[checkupdate\\_pivmat](#)

[Previous: Contents](#)

[Next: addnoise](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# addnoisef

Add noise to vector/scalar fields

## Description

`FF = addnoisef(F)` adds 10% white noise to the vector/scalar field(s) `F`.

`FF = addnoisef(F, EPS)` specifies that the standard deviation of the white noise is `EPS` times that of the field(s) `F` (`EPS=0.1` by default).

`FF = addnoisef(F, EPS, OPT)` specifies whether the noise is additive (`OPT='add'`, by default) or multiplicative (`OPT='mul'`).

`FF = addnoisef(F, EPS, OPT, N)` first filters the noise at scale `N` (in mesh units) with `GAUSSMAT` before adding/multiplying it to the field(s) `F`.

## Example

```
showvec(addnoisef(loadset));
```

## See Also

[operf](#), [randvec](#), [GAUSSMAT](#).



# averf

Average of vector/scalar fields

## Description

$AF = \mathbf{averf}(F)$  returns the average of the vector/scalar fields  $F$ .  $AF$  is a field of the same type as  $F$ , whose elements are the average of the elements of the fields  $F$ .

Depending on the nature of the fields  $F$ , the field  $\mathbf{averf}(F)$  is usually called "Ensemble average", "Phase average", or "Time average".

By default, **averf** considers that the zero elements of  $F$  are erroneous, and does not include them in the computations. If however you want to force the zero elements to be included in the computations, specify **averf**( $F$ , '0').

$[AF \text{ RMS } FR] = \mathbf{averf}(F, \dots)$  also returns the rms field and the fluctuation rate field of the vector/scalar fields  $F$ . RMS is a scalar field, whose elements are the rms of the elements of the fields  $F$ . FR is a scalar field, whose elements are the local RMS normalized by the local norm of  $AF$ .

If no output argument is specified, display the average field  $AF$  with `showvec` or `showscal`.

## Examples

```
showvec(averf(loadset));
```

```
[af rms fr]=averf(vec2scal(loadset,'uy')); showscal(fr,[0 1]);
```

```
averf *.vc7           % shows the average of the fields matching *.vc7
```

## See Also

[spaverf](#), [subaverf](#), [azaverf](#).

**[Previous: addnoise](#)**

**[Next: azaverf](#)**

2005-2007 [PIVMat Toolbox 1.60](#)

# azaverf

Azimuthal average a vector/scalar field.

## Description

$[R, P] = \mathbf{azaverf}(S, X0, Y0)$ , where  $S$  is a scalar field, returns the azimuthally-averaged profile  $P(R)$  of  $S$  with respect to the center  $X0, Y0$  (given in physical units). The output vector  $R$  is the radius (in physical units), and the vector  $P$  is the profile. If  $S$  is an array of  $N$  scalar fields,  $R$  and  $P$  are  $M \times N$  matrix. Use  $\text{PLOT}(R, P)$  to plot the resulting profiles.

$[R, UR, UT] = \mathbf{azaverf}(V, X0, Y0)$ , where  $V$  is a vector field, returns the radial  $UR$  and azimuthal  $UT$  components of the azimuthally-averaged profiles of  $V$ .

The center  $(X0, Y0)$  does not need to be inside the field. If  $X0$  and  $Y0$  are not specified, the point  $(0,0)$  is taken (in physical units).

$AF = \mathbf{azaverf}(F, X0, Y0)$ , where  $F$  is a scalar or a vector field, returns the azimuthally averaged field  $AF$ . If  $F$  is an array of fields,  $AF$  is also an array of fields of same dimension. If no output argument specified, the result is displayed with [showscal](#) or [showvec](#).

$\dots = \mathbf{azaverf}(\dots, I0, J0, 'mesh')$  specifies the center  $(I0, J0)$  in mesh units instead of physical units ( $I0$  and  $J0$  do not need to be integer, and do not need to be inside the field).

By default, zero elements are considered as erroneous, and are not used for the computation of the azimuthal average. If however you want to keep them in the computation, specify  $\mathbf{azaverf}(\dots, 'keepzero')$ .

## Examples

```
v=loadvec('* .vec');
showvec(azaverf(v,10,10));

k=vec2scal(v,'ken');
[r,e] = azaverf(k,40,32,'mesh');
plot(r,e,'o-');
```

```
xlabel('r (mm)'); ylabel('energy');  
  
v=addnoise(vortex,0.2);  
showvec(v);  
showvec(azaverf(v,64,64));
```

## See Also

[showvec](#), [showscal](#), [averf](#), [spaverf](#), [subaverf](#), [filterf](#),  
[vec2scal](#), [rotatef](#).

[Previous: averf](#)

[Next: batchvec](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# batchvec

Execute a function for a series of files

## Description

`RES = batchvec(FILENAME,FUN)` executes the function `FUN` for each file matching `FILENAME`. Wildcards (\*) and brackets ([]) may be used (see [expandstr](#)). This is formally equivalent to calling the function `FUN` with the argument [loadvec](#)(`FILENAME`), except that the fields are not stored in a structure array, but sent one by one to the function `FUN`. This is useful for handling a large number of files which cannot be stored into a structure array for lack of memory. All the file formats supported by [loadvec](#) are also supported by **batchvec**. The result `RES` is an array whose elements are the output of the function `FUN` applied to each file matching `FILENAME`.

`RES = batchvec(FILENAME,FUN,ARG)` also passes the argument(s) `ARG` to the function `FUN`. This is formally equivalent as `FUN(loadvec(...), ARG)` (this works only for simple arguments, as strings and numbers, not for arrays and cell arrays).

`RES = batchvec(..., 'nodisp')` does not display the function call for each file matching `FILENAME` in the command window.

## Examples

`st=batchvec('set*/B00001.VEC','statf')` calls the function [statf](#) for each file `B00001.VEC` in each directory matching `'set*'`.

`batchvec('set*/B*.VEC','showvec','rot',8)` is equivalent to [showvec](#)([loadvec](#)('set\*/B\*.VEC'),'rot',8), except that the files are not stored in a structure array.

If the function `FUN` to be executed is more complex, you may create a M-File called `'myfunction.m'`, that contains the complex computation to

be executed for each field  $V$ , and to execute  $RES =$   
**batchvec**(FILENAME,'myfunction').

## Example

Create the following M-File called 'enstrophy.m':

```
function z = enstrophy(v),  
stat = statf(vec2scal(filterf(v,1),'rot'));  
z = stat.rms^2;
```

Then call:

```
z = batchvec('set*/*.vec','enstrophy');
```

## See Also

[loadvec](#), [loadarrayvec](#), [loadset](#).

[Previous: azaverf](#)

[Next: bwfilterf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# bwfilterf

Butterworth filter for a vector/scalar field

## Description

`FF = bwfilterf(F,FSIZE,ORDER)` applies a lowpass Butterworth filter to the vector/scalar field `F` with a cutoff size `FSIZE` (in grid units) and order `ORDER`. `F` must be a square vector or scalar field; if it is not square, **bwfilterf** first extracts the central square of `F` (see [extractf](#)). Typical values for `FSIZE` are around 1, and typical values for `ORDER` are in the range 2-10.

`FF = bwfilterf(F,FSIZE,ORDER,OPT)`, where `OPT` is a string that may contain one or several substrings:

'low', 'high': specifies a lowpass (by default) or highpass filter  
't': truncates the borders of width `FSIZE`, which are affected by the filtering.

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

**bwfilterf** first Fourier transforms the field(s), low/high-pass filters and inverse Fourier transforms `F`.

Note: A highpass filter of order `ORDER` is equivalent to a lowpass filter of order `-ORDER`.

## Example

```
showvec(bwfilterf(loadset,3,8));
```

## See Also

[filterf](#), [addnoise](#), [truncf](#), [extractf](#).

# checkupdate\_pivmat

Check for update for PIVMat

## Description

**checkupdate\_pivmat** connects on the PIVMat webpage and checks for a new version.

**checkupdate\_pivmat**('dialog') does the same, but outputs the result in a dialog box.

## See Also

[about\\_pivmat](#).

[Previous: bwfilterf](#)

[Next: corrf](#)

2005-2007 [PIVMat Toolbox 1.60](#)



# corr

Correlation function of a scalar field

## Description

`COR = corr(F, DIM)` returns the correlation function of the scalar field `F` along the dimension `DIM` (with `DIM=1, 2`, or `'x','y'`, for correlation along the `'X'` or `'Y'` direction). If `F` is an array of scalar fields, `COR` returns the average of the correlation functions of each field.

`COR` is a structure which contains the following fields:

- `r`: separation length
- `f`: correlation function
- `unitr`: unit of separation length
- `unitf`: unit of correlation function
- `namef`: name of correlation function
- `isinf`: integral scale, computed as the integral of `f` up to `inf`
- `r5`: scale at which `f(r5)=0` (linearly interpolated)
- `is5`: integral scale, computed as the integral of `f` up to `r5`
- `r2, is2, r1, is1`: idem as `r5, is5`, for crossovers at 0.2 and 0.1

`COR = corr(F, DIM, 'norm')` normalizes the correlation function.

Note that `corr(vec2scal(V,'ux'),'x')` and `corr(vec2scal(V,'uy'),'y')` are the longitudinal velocity correlation functions, and `corr(vec2scal(V,'ux'),'y')` and `corr(vec2scal(V,'uy'),'x')` are the transverse ones.

If no output argument, the correlation function is plotted.

## Example

```
v = loadvec('B00001.VEC');
cor = corr(vec2scal(v,'vx'),'x');
plot(cor.r, cor.f, 'o-');
```

## See Also

[vec2scal](#), [statf](#), [histf](#), [vsf](#).

[Previous: checkupdate\\_pivmat](#)

[Next: expandstr](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# expandstr

Expand indexed strings.

## Description

`F = expandstr('PP[RANGE]SS')` returns a cell array of strings in the form

'PP0000nSS', where 'PP' and 'SS' are prefix and suffix substrings, n is an index lying in the range RANGE, padded with 5 zeros. RANGE is a vector, that can be in the form [N1 N2 N3..], or START:END, or START:STEP:END, or all other MATLAB valid syntax.

`F = expandstr('PP[RANGE,NZ]SS')` also specifies the number of zeros to pad the index string (NZ=5 by default). For example, `'B[1:4,2].v*'` gives {'B01.v\*', 'B02.v\*', 'B03.v\*', 'B04.v\*'}

If the input string has more than one bracket pair [], **expandstr** is called recursively for each pair. For example, `'B[1:4,2]_[1 2,1]'` gives {'B01\_1', 'B01\_2', 'B02\_1', 'B02\_2', 'B03\_1', 'B03\_2', 'B04\_1', 'B04\_2'}

**expandstr** is useful when applied to file names, e.g. with [rdir](#). In particular, wildcards (\*) may be present in PP or SS (but they are kept as wildcards, i.e. they are not interpreted). For example, `expandstr('B[1 2 3,5]*.*')` returns {'B00001\*.\*', 'B00002\*.\*', ...}. Note that **expandstr** is automatically called from [rdir](#).

## Examples

`expandstr('DSC[2:2:8,4].JPG')` returns  
{'DSC0002.JPG', 'DSC0004.JPG', 'DSC0006.JPG', 'DSC0008.JPG'}

`rdir(expandstr('B[1:10,5]*.*'))` is equivalent to `rdir('B[1:10,5]*.*')`

## See Also

[rdir](#).



# extractf

Extract a rectangular area from a vector/scalar field.

## Description

$EF = \mathbf{extractf}(F, [X1 \ Y1 \ X2 \ Y2])$  extracts a rectangular area of coordinates  $[X1 \ Y1 \ X2 \ Y2]$  from the original vector/scalar field(s)  $F$ . By default, the coordinates are given in mesh units. Specify  $\mathbf{extractf}(F, [X1 \ Y1 \ X2 \ Y2], 'phys')$  to give them in physical units.

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

## See Also

[truncf](#), [rotatef](#), [shiftf](#).

[Previous: expandstr](#)

[Next: filterf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# filterf

Apply a lowpass filter to a vector/scalar field.

## Description

`FF = filterf(F,FSIZE)` applies a Gaussian filter of size `FSIZE` to the vector/scalar field(s) `F`. (`FSIZE=1` is taken if not specified).

`FF = filterf(F,FSIZE,METHOD)` specifies the filter:

'flat':	flat (or top-hat) matrix, <code>ones(FSIZE,FSIZE)</code> ( <code>FSIZE</code> must be an even integer)
'gauss':	gaussian (by default)
'igauss':	derivative of the integrated gaussian (minimized discretisation effects for small <code>FSIZE</code> ).

If the zeros of the fields correspond to erroneous values, the filtering may affect the neighbouring elements. It is therefore better to interpolate the 0s, by specifying `filterf(...,'zero')`.

The size of the filtered field is smaller than the original field, to avoid boundary effects (the convolution is done by `CONV2` with the option 'valid'). If you prefer to keep the whole field, use `filterf(...,'same')`

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

## Examples

```
showvec(filterf(loadvec('B00001.vec'),1));
showvec(filterf(loadset,2,'f'));
showscal(filterf(vec2scal(loadset,'rot'),2));
```

## See Also

[showvec](#), [showscal](#), [bwfilterf](#), [addnoise](#), `GAUSSMAT`, `CONV2`.



# getattribute

Get attribute of a field

## Description

A = **getattribute**(F) returns a structure containing the attributes of the vector or scalar field F. If F is an array, then A is a cell array.

F may be vector or scalar fields, or filenames of any file format supported by [loadvec](#). Wildcards (\*) and brackets ([]) may be used (see [expandstr](#) for details).

The field names of the structure A are given by the attribute names (leading '\_' are removed from the attribute names). The attribute values are converted to numerical values whenever a numerical conversion is possible.

VAL = **getattribute**(F, NAME) returns only the value of the specified attribute NAME. NAME is not case sensitive, and '\_' are ignored.

To obtain the image acquisition time of F, see [getpivtime](#).

## Examples

Returns the times of all the vector fields of the current set:

```
time = getattribute('*.vec', 'time');
```

Returns a cell array of attribute structure for all the current set:

```
attr = getattribute(loadvec(1));  
attr.DATE
```

## See Also

[getpivtime](#).



# getfilenum

Get the index of a series of files.

## Description

NUM = **getfilenum** (NAME, P) returns an array of numbers indexing the filenames (or directory names) matching NAME from the current directory. The indices are searched in the strings following the substring P. Wildcards (\*) and brackets ([], see [expandstr](#)) may be used in NAME, including in intermediate pathnames.

NUM = **getfilenum** (NAME, P, OPT), where OPT is 'dironly', 'fileonly' or 'filedir', gets the numbers only from the directory names, file names, or both (by default), respectively.

## Examples

If the files 'B01\_t12.vec', 'B02\_t18.vec', 'B03\_t24.vec' are present in the current directory,

```
getfilenum ('*.vec', '_t')    returns [12 18 24],  
getfilenum ('*.vec', 'B')     returns [1 2 3].
```

n=**getfilenum**('\*/\*.JPG','DSC') returns the indices of all the files matching 'DSCxxx.JPG' in all directories.

## See Also

[expandstr](#), [rdir](#), [renumberfile](#).

# getimx

```
[lhs1,lhs2,lhs3,lhs4,lhs5,lhs6,lhs7]=getimx(A, frame);
```

## Description

This command is a clone of LaVision's SHOWIMX command, without graphic output, for the PIVMat toolbox by F. Moisy.

FUNCTION:   Displaying data of LaVision's IMX structure  
             (one vector field, all image frames or only single image  
frame)

ARGUMENTS: A   = IMX-structure created by READIMX/READIM7 function

RETURN:       in case of images (image type=0):  
              lhs1 = scaled x-coordinates  
              lhs2 = scaled y-coordinates  
              lhs3 = scaled image intensities  
              in case of 2D vector fields (A.IType = 1,2 or 3):  
              lhs1 = scaled x-coordinates  
              lhs2 = scaled y-coordinates  
              lhs3 = scaled vx-components of vectors  
              lhs4 = scaled vy-components of vectors  
              lhs5 = vector choice field  
              in case of 3D vector fields (A.IType = 4 or 5):  
              lhs1 = scaled x-coordinates  
              lhs2 = scaled y-coordinates  
              lhs3 = scaled z-coordinates  
              lhs4 = scaled vx-components of vectors  
              lhs5 = scaled vy-components of vectors  
              lhs6 = scaled vz-components of vectors  
              lhs7 = vector choice field

[Previous: getfilenum](#)[Next: getpivtime](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# getpivtime

Get the time from the Attribute string.

## Description

`T = getpivtime(F)` returns the time(s), in seconds, of the field(s) `F`. `F` may be vector or scalar fields, or filenames of any file format supported by [loadvec](#). Wildcards (\*) and brackets ([]) may be used (see [expandstr](#) for details).

`T = getpivtime(F,'0')` does the same, starting at time 0 for the first field.

Note that, for VC7 vector fields (DaVis 7), **getpivtime** returns the image acquisition time, as stored in the attribute 'AcqTimeSeries0'. However, for VEC vector fields (DaVis 6), **getpivtime** returns the PIV computation time, as stored in the attributed 'TIME', which is usually of no use. To get the true image acquisition time instead, use **getpivtime** with IMX files.

## Examples

```
t = getpivtime(loadset);  
  
t = getpivtime('B[1:10].vc7','0');
```

## See Also

[getattribute](#).

[Previous: getimx](#)

[Next: getsetname](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# getsetname

Get the name of the current SET

## Description

`S = getsetname` returns the name of the current directory. While `PWD` returns the full path 'dir1/dir2/../../dirn', **getsetname** returns only the last element 'dirn'.

This command is useful to obtain the name of the SET for VEC files.

## See Also

`PWD`, [loadset](#).

# gradientf

Gradient of scalar field

## Description

$V = \text{gradientf}(S)$  returns the gradient of the scalar field(s)  $S$ .  $S$  is a structure (or a structure array) that can be obtained by [vec2scal](#).

## Example

```
showvec(gradientf(vec2scal(v, 'norm')));
```

## See Also

[vec2scal](#).

[Previous: getsetname](#)

[Next: histf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# histf

Histogram of a vector/scalar field

## Description

$H = \text{histf}(F)$  computes the histogram  $H$  of the field(s)  $F$  into 100 equally spaced bin. The appropriate range of bins is estimated from the rms of the first field.

$H = \text{histf}(F, \text{BIN})$  computes the histograms among the bins specified by the vector  $\text{BIN}$ .  $\text{BIN}$  can be in the form  $\text{Linspace}(-X, X, N)$ .

If  $F$  is a vector field, use  $[H_X, H_Y] = \text{histf}(F, \dots)$  to compute the histogram for each component of the vector field.

$[\dots, \text{BIN}] = \text{histf}(\dots)$  also returns the bins vector  $\text{BIN}$  used for the computation of the histogram. This is useful if  $\text{BIN}$  is estimated by **histf** itself.

By default, **histf** considers the values 0 as erroneous, and does not include them in the histogram. If however the values 0 are to be included, specify **histf**(..., '0');

## Example

```
semilogy(histf(vec2scal(loadset,'rot')));
```

## See Also

[histscal\\_disp](#), [statf](#), [corrff](#), [vsf](#).

[Previous: gradientf](#)

[Next: histscal\\_disp](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# histscal\_disp

Displays distributions of scalar fields.

## Description

**histscal\_disp**(S) displays the distributions of the scalar field(s) S.

**histscal\_disp**(S,SMOOTH) computes the velocity distributions averaged over SMOOTH consecutive fields. Use SMOOTH=1 for no smoothing, or SMOOTH=0 to smooth over all the fields (by default).

**histscal\_disp**(S,SMOOTH,BIN) computes the histograms among the bins specified by the vector BIN. If this argument is not present, the appropriate range of bins is estimated from the rms of the first field.

**histscal\_disp**(...,OPT), where OPT is a string that may contain one or

more of the following letters (all by default):

- 'n': normalize the histogram (pdf)
- 'g': also displays a gaussian fit
- 'l': log-coordinate for the y axis

## Example

```
histscal_disp(vec2scal(loadset,'rot'));
```

## See Also

[histf](#), [histvec\\_disp](#).

[Previous: histf](#)

[Next: histvec\\_disp](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# histvec\_disp

Displays distributions of vector fields.

## Description

**histvec\_disp**(V) displays the distributions of the components of the vector field(s) V.

**histvec\_disp**, without input argument, uses the current set.

**histvec\_disp**(V,SMOOTH) computes the velocity distributions averaged over SMOOTH consecutive fields. Use SMOOTH=1 for no smoothing, or SMOOTH=0 to smooth over all the fields (by default).

**histvec\_disp**(V,SMOOTH,BIN) computes the histograms among the bins specified by the vector BIN. If this argument is not present, the appropriate range of bins is estimated from the rms of the first field.

**histvec\_disp**(...,OPT), where OPT is a string that may contain one or

more of the following letters (all by default):

- 'n': normalize the histogram (pdf)
- 'g': also displays a gaussian fit
- 'l': log-coordinate for the y axis

## Example

```
histvec_disp(loadset,5,linspace(-0.5,0.5,50),'ngl');
```

## See Also

[histf](#), [statf](#), [histscal\\_disp](#).

[Previous: histscal\\_disp](#)

[Next: jpdfscal](#)

2005-2007 [PIVMat Toolbox 1.60](#)



# jpdfscal

Joint PDF of two scalar fields

## Description

`jpdf = jpdfscal(S1,S2)` computes the joint PDF of the two scalar fields

`S1` and `S2`.

`jpdf = jpdfscal(S1,S2,BIN)` specifies the number of bins over each direction (101 by default).

If no output argument specified, call [jpdfscal\\_disp](#).

## Example

```
jpdf = jpdfscal(vec2scal(v,'rot'),vec2scal(v,'div'));
```

## See Also

[jpdfscal\\_disp](#).

[Previous: histvec\\_disp](#)

[Next: jpdfscal\\_disp](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# jpdfscal\_disp

Displays the joint PDF computed by JPDFSCAL

## Description

`jpdfscal_disp(jpdf)` displays the joint PDF computed by [jpdfscal](#).

## Example

```
jpdfscal_disp(jpdfscal(vec2scal(v,'rot'),vec2scal(v,'div')));
```

## See Also

[jpdfscal](#).

[Previous: jpdfscal](#)

[Next: loadarrayvec](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# loadarrayvec

Load a 2D array of vector fields

## Description

`V = loadarrayvec(DIR,FILE)` loads the vector fields matching `FILE` from the directories matching `DIR` into a structure array `V(I,J)` of vector fields, where the index `I` scans the directories and the index `J` scans the files. Each element `V(I,J)` is a single vector field (see [loadvec](#) for the vector field description). Wildcards (\*) and brackets ([]) may be used in `DIR` and `FILE` (see [expandstr](#)). All the file formats supported by [loadvec](#) are also supported by `loadarrayvec`.

## Examples

```
v = loadarrayvec('dir*', '*.vec');  
showvec(v(1,:)); % displays the files from the first directory.  
showvec(v(:,1)); % displays the first file from each directory.  
  
v = loadarrayvec('dir*', 'set.mat');
```

## See Also

[loadvec](#), [loadset](#), [batchvec](#).

[Previous: jpdfscal\\_disp](#)

[Next: loadpivtxt](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# loadpivtxt

Load a vector field exported in text format from DaVis

## Description

`V = loadpivtxt(FILENAME)` loads a vector field `FILENAME` saved in text format from DaVis into the structure `V`.

The fields of the structure `V` are the same as for [loadvec](#), except the field `V.Attributes`, which contains the first line of the text file `FILENAME`.

Using [loadvec](#) to load vector fields in TXT mode allows for wildcards and brackets (filename expansion).

Note: Using VEC/VC7 files instead of TXT files is strongly encouraged.

If you prefer platform-independant file formats, you may convert your original DaVis files into Mat-files using [vec2mat](#).

## Example

```
v=loadpivtxt('B00001.txt');
```

## See Also

[loadvec](#), [vec2mat](#).

# loadset

Loads set(s) of vector/image fields.

## Description

`F = loadset` loads all the vector/image fields of the current directory into the structure array `F`. This is essentially equivalent to `F = loadvec({'*.vec','*.vc7','*.imx','*.im7'})`, except that **loadset** makes use of a MAT-file called 'set.mat' to save time (see below). See [loadvec](#) for the content of the structure array `F`.

`F = loadset(DIR)` loads all the vector/image fields from the specified directory `DIR` in the structure array `F`. If `DIR` is a cell array (e.g. `['dir1','dir2']`), then loads all the files from each directory and concatenates them in a single structure array `F` (see [loadarrayvec](#) to load the files in a 2D structure array). Wildcards (\*) may be used: `F = loadset('set*')` loads all the files in all the directories matching 'set\*'. Brackets ([]) are also accepted (see [expandstr](#) for details). For example, `F = loadset('set[1:3,1]')` loads all the vector/scalar fields from the directories 'set1', 'set2', 'set3'. `DIR` can be a cell array with a combination of wildcards and brackets.

`F = loadset(DIR, FILE)` loads only the file(s) `FILE` in the directory(s) `DIR`. This is useful when `DIR` and/or `FILE` are cell arrays and/or contain wildcards (\*) and/or brackets (see [expandstr](#)). This is equivalent to `F = loadvec([DIR '/' FILE])`.

`F = loadset(DIR, NUM)` loads only the file number `NUM` in the directory (s) `DIR` (works in alphanumeric order, only for VEC/VC7 and IMX/IM7 files). `NUM` may be a simple number or any valid MATLAB vector (e.g., 1:10, [1 10], 4:-1:1, etc.)

`loadset(...)` without output argument is a shortcut for [showvec\(loadset\(...\)\)](#) or [showscal\(loadset\(...\)\)](#)

## Examples

`V = loadset('myset')` loads all the VEC/VC7 files in the directory 'myset' (loads the 'set.mat' file if it exists; see below).

[showvec\(loadset\)](#) displays all the files from the current directory.

`V = loadset('set*', 'B00001.vec')` loads the files 'B00001.vec' contained in each directory 'set\*'.

`V = loadset('set*', '*.vec')` loads all the VEC files in each directory 'set\*'.

`V = loadset({'set1', 'set2'}, {'file1.vec', 'B*.vec'})` loads 'file1.vec' and all the VEC files 'B\*' in the two directories 'set1' and 'set2'.

`V = loadset('set*', 'B[5:10]*')` loads 'B00005.vec' to 'B00010.vec' in each directory 'set\*' (see [loadvec](#) and [expandstr](#) for details).

`V = loadset('set[200:5:400,2]*', 'B01.vec')` loads the files 'B01.vec' contained in each directory 'set200\*', 'set205\*', 'set210\*' etc.

`V = loadset('set*', 1:10)` loads the 10 first files from each directory 'set\*'.

The first time **loadset** is used to load all the vector/scalar fields in a directory (i.e. without the argument `FILE`), a MAT-file called 'set.mat' containing the structure array `F` for all the files is created. The next time **loadset** is used, this file 'set.mat' is loaded instead of each original file, to save time (this may be 2 to 5 times faster). Once this file 'set.mat' has been created, the original files are not necessary any more.

`F = loadset(..., 'overwrite')` loads the original vector/scalar fields even if a 'set.mat' file already exists, and saves (overwrites) a new 'set.mat'.

`F = loadset(..., 'nosave')` loads the original vector/scalar files even if a 'set.mat' file already exists, and does not save the 'set.mat' file.

## See Also

[loadvec](#), [vec2mat](#), [loadarrayvec](#), [batchvec](#), [readsetfile](#), [showvec](#), [expandstr](#), [rdir](#).

[Previous: loadpivtxt](#)

[Next: loadvec](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# loadvec

Loads vector/image fields

## Description

`F = loadvec(FILENAME)` loads the vector/image fields matching `FILENAME` into the structure array `F`. Supported file formats are:

<code>.VEC</code>	Vector field from DaVis 6.
<code>.VC7</code>	Vector field from DaVis 7.
<code>.IMX,IMG</code>	Image from DaVis 6.
<code>.IM7</code>	Image from DaVis 7.
<code>.TXT</code>	Vector field exported in TXT mode (see <a href="#">loadpivtxt</a> )
<code>.SET</code>	Set of vector fields (see <a href="#">loadset</a> for details)
<code>.MAT</code>	Any MATLAB file containing a valid structure array of vector/scalar fields (e.g., files converted by <a href="#">vec2mat</a> , or the file 'set.mat' generated by <a href="#">loadset</a> ).

The resulting structure array `F` can be displayed using [showvec](#) or [showscal](#). For multi-frame images, only the first frame is loaded. Use [vec2scal](#) to compute scalar fields from vector fields.

If `FILENAME` is a cell array, (e.g. `{'B1.vec','B2.vec'}`), then **loadvec** loads all the files in the structure array `F`. Wildcards (\*) may be used (e.g., **loadvec**('B\*.vec')). Brackets ([]) may also be used (see [expandstr](#) for details). For example, `F = loadvec('B[2:2:6].vec')` loads the files 'B00002.vec', 'B00004.vec', 'B00006.vec'.

To load directly all the vector/image fields of a directory, see [loadset](#).

`F = loadvec(NUM)` loads the file number `NUM` from the current directory (works in alphanumeric order, only for `VEC/ VC7` and `IMX/IM7` files). `NUM` may be a simple number or any valid MATLAB vector (e.g., `1:10`, `[1 10]`, `2:2:8`, `10:-1:1`, etc.)

**loadvec** without input argument first opens a dialog box for file selection.

**loadvec** `file_name` or **loadvec**('file\_name') without output argument is a shortcut for [showvec](#)(**loadvec**('file\_name')).

For vector fields, the structure `F` contains the following fields:

<code>x,y:</code>	vectors containing the X and Y coordinates
<code>vx,vy:</code>	matrices of the x and y components of the velocity
<code>ysign:</code>	string, upward or downward Y axis
<code>namevx, unitvx, namex, unitx...:</code>	strings
<code>name:</code>	name of the VEC/VC7 file from which originates V

**setname:** name of the parent directory (called 'SET' in DaVis)  
**Attributes:** Additional informations from DaVis (see [getattribute](#))  
**choice:** An array of 6 integers, giving the 1st, 2nd, 3rd, 4th choice vectors, the number of filled/processed vectors and the number of missing vectors.  
**history:** Remind from which command V has been obtained

For scalar fields, vx and vy are replaced by w (idem for namevx,...)

## Examples

```

loadvec B00001.vc7
showvec(loadvec('*.vec'));
showscal(vec2scal(loadvec('B00001.vc7'),'rot'));
v=loadvec('B[2:2:6]*.v*');
v=loadvec({'data1.set','data2.set'});
v=loadvec(1:10);

```

## See Also

[showvec](#), [loadset](#), [loadarrayvec](#), [vec2mat](#), [filterf](#), [vec2scal](#),  
[batchvec](#), [expandstr](#), [rdir](#).

[Previous: loadset](#)

[Next: operf](#)

2005-2007 [PIVMat Toolbox 1.60](#)



# operf

Perform operation on vector/scalar fields

## Description

$FF = \text{operf}(OP, F1, F2)$  performs operation specified by  $OP$  on the fields  $F1$  and  $F2$ . Valid operations are '+', '-', '.\*' and './'.  $F1$  and  $F2$  must be of the same type (vector or scalar fields). If  $\text{LENGTH}(F1) = \text{LENGTH}(F2)$ , the operation is performed for each field. If  $\text{LENGTH}(F2) = 1$ , the operation is performed using the single field  $F2$  for each field  $F1$ .

$FF = \text{operf}(OP, F, N)$ , where  $N$  is a number, performs operation specified by  $OP$  on the field  $F$ . Valid operations are '+', '-', '\*', '/' and '^'.

$FF = \text{operf}(OP, F)$  performs the unary operation  $OP$  on field(s)  $F$ . Valid unary operations are:

- '+' (does nothing), '-' (inverts the field),
- 'log', 'exp', 'abs', 'logabs'
- any MODE operation from [vec2scal](#) (e.g.,  $\text{operf}('rot', V)$  is

equivalent

to [vec2scal](#)( $V, 'rot'$ )).

$\text{operf}(\dots)$  without output argument shows the result with [showvec](#) or [showscal](#).

## Examples

```
v=loadset;
showvec(operf('-',v));
showscal(operf('/',vec2scal(v,'rot'),2));
```

## See Also

[vec2scal](#), [showvec](#), [showscal](#), [averf](#), [spaverf](#), [subaverf](#).

# randvec

Synthetic vector field.

## Description

$V = \text{randvec}$  returns a synthetic vector field with a  $k^{-5/3}$  spectrum, random phase and zero 2D divergence. The size of the vector field is 256x256 by default.  $V$  has the same structure as vector fields loaded by [loadvec](#), with arbitrary units.

$V = \text{randvec}(N)$  returns a field of size  $N \times N$ .

$V = \text{randvec}(N, NF)$  returns an array of  $NF$  fields.

$V = \text{randvec}(N, NF, SLOPE)$  imposes a spectrum  $k^{-SLOPE}$ . ( $-5/3$  by default).

$V = \text{randvec}(N, NF, SLOPE, NC, NL)$  does the same, but specifies the small scale  $NC$  and large-scale  $NL$  cut-offs (in units of vector mesh). By default,  $NC=3$  and  $NL=N/3$ . For  $k < NL$ , the spectrum is  $k^2$ . For  $k > NC$ , the spectrum falls gaussianly.

## Example

```
showvec(randvec);
```

## See Also

[showvec](#), [addnoise](#), [vortex](#).

[Previous: operf](#)

[Next: rdelete](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# rdelete

Delete a series of files.

## Description

**rdelete** file\_name deletes the named files. Wildcards may be used in the filename, as with DELETE, but also in the pathnames, which allows for deleting a series a files belonging to different directories. Brackets ([]) may also be used (see [expandstr](#)).

Use the functional form of **rdelete**, such as **rdelete**('file'), when the file names are stored in a string or a cell array of strings.

To delete directories, see [rrmdir](#).

## Examples

**rdelete**('DSC\*.JPG') is equivalent to DELETE('DSC\*.JPG')

**rdelete**('\*/\*.JPG') deletes all the JPG-files in all the directories.

**rdelete**('mydir\*/DSC[10:30].\*') deletes the files DSC00010.\* to DSC00030.\* in each directory 'mydir\*'.

## See Also

DELETE, [expandstr](#), [rdir](#), RMDIR, [rrmdir](#).

[Previous: randvec](#)

[Next: rdir](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# rdir

Recursive list directory.

## Description

`F = rdir(NAME)` returns a cell array of file names matching `NAME`. `NAME` may be a cell array of strings, and may contain cascading pathnames separated by `'/'` followed by a filename. Wildcards may be used both in the final filename, as in `DIR`, but also in the intermediate pathnames. For example, `F=rdir('mydir*/*.m')` returns all the M-Files contained in each directory that begins with 'mydir'. In addition, brackets `[]` may also be used (see [expandstr](#)), both in the pathnames and in the filename. If wildcards or brackets are present in the pathnames, **rdir** lists recursively all the directories matching the pathnames.

**rdir** `file_name` or **rdir**('file\_name') displays the result.

`F = rdir(NAME,'fileonly')` only returns file names.

`F = rdir(NAME,'dironly')` only returns directory names, but does not list their content.

`F = rdir(NAME,'filedir')` returns both files and directories (as `DIR`). (Note that, when options 'dironly' or 'filedir' are used, the fictitious directories `'.'` (current) and `'..'` (parent) are not returned, contrarily to `DIR`). (by default)

## Examples

`F = rdir('set*/B[1:8,2].v*')` returns the file names matching 'B01.v\*' to 'B08.\*' in each directory matching 'set\*'.

`F = rdir('set[1 2 3]*/*/B01.vec')` returns the file name 'B01.vec', if present, in all the subdirectories of each directory matching 'set1\*', 'set2\*', 'set3\*'.

## See Also

[expandstr](#) , DIR .

[Previous: rdelete](#)

[Next: readsetfile](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# readsetfile

Read the variables of a .SET or .EXP file

## Description

`A = readsetfile(FILENAME)` returns a structure containing the attributes of the .SET or .EXP file FILENAME.

`VAL = readsetfile(FILENAME, ATTRNAME)` returns only the value of the specified attribute ATTRNAME. ATTRNAME is not case sensitive, and '\_' are ignored.

## Example

```
bufstr = readsetfile('myexp.set','SetBuffers');
```

## See Also

[getattribute](#), [loadset](#).

[Previous: rdir](#)

[Next: renamefile](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# renamefile

Rename a series of files.

## Description

**renamefile**(NAME, P1, P2) renames the files matching NAME, replacing the substring P1 by P2. NAME may be a cell array of strings, and may contain wildcards (\*) and brackets (see [expandstr](#)).

**renamefile**(NAME, P1, P2, OPT), where OPT is 'dironly', 'fileonly' or 'filedir', renames only the directory names, the file names, or both (by default), respectively.

## Examples

```
renamefile('DSC*.JPG','DSC','myphoto')  
renames the files 'DSC00001.JPG','DSC00002.JPG',... as  
'myphoto00001.JPG','myphoto00002.JPG',...
```

```
renamefile('*/DSC*.JPG','DSC','myphoto')  
does the same in all the directories containing JPG files.
```

```
renamefile('B[1:100,3]*.VEC','B','PIV') renames the files  
'B001*.VEC' to 'B100*.VEC' as 'PIV001*.VEC' to 'PIV100*.VEC'
```

```
renamefile('set*','set','newset','dironly') renames the  
directories  
'set*' as 'newset*'.  

```

## See Also

[renumberfile](#), `MOVEFILE`, [expandstr](#), [rdir](#), [getfilenum](#).

[Previous: readsetfile](#)

[Next: renumberfile](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# renumberfile

Re-number the indices of a series of files.

## Description

**renumberfile**(NAME, P) renumbers the files matching NAME having an index following the substring P, using consecutive indices starting from 1. Wildcards (\*) and brackets (see [expandstr](#)) may be used in NAME. If P is present several times in a filename, uses only the last occurrence. **renumberfile** works only in the current directory (NAME cannot contain pathnames).

**renumberfile**(NAME, P, NEWNUM) does the same, using NEWNUM to renumber the files. If NEWNUM is a number, uses consecutive indices starting from NEWNUM. If NEWNUM is an array, uses it to renumber the files. By default, NEWNUM=1.

**renumberfile**(NAME, P, NEWNUM, NZ) specifies the number of 0 padding the index (e.g., '302' padded with 5 zeros is '00302'). By default, NZ=5.

**renumberfile**(NAME, P, NEWNUM, NZ, OPT), where OPT is 'dironly', 'fileonly' or 'filedir', renumbers only the directory names, the file names, or both (by default), respectively.

## Examples

If **renumberfile**('DSC\*.JPG','DSC') is used in a directory that contains 100 JPG-files with arbitrary file numbers, the files are renumbered as 'DSC00001.JPG'...'DSC00100.JPG'.

**renumberfile**('DSC\*.JPG','DSC',401) does the same, renumbering the files as 'DSC00401.JPG'...'DSC00500.JPG'.

**renumberfile**('DSC\*.JPG','DSC',1,3) does the same, renumbering the files as 'DSC001.JPG'...'DSC100.JPG'.

## See Also

[getfilenum](#), [expandstr](#), [rdir](#), [renamefile](#).



[Previous: renamefile](#)

[Next: rotatef](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# rotatef

Rotates a vector/scalar field.

## Description

`RF = rotatef(F,THETA)` rotates the vector/scalar field(s) `F` about its center through the angle `THETA` in the trigonometric direction (`THETA` is given in radians). Points which have antecedent from outside the original field are filled with 0. The new field `RF` is obtained from interpolation of the original field `F` on the new grid.

`RF = rotatef(F, THETA, X0, Y0)` rotates about the center specified by `(X0, Y0)`, in physical units. The center does not need to be inside the field. Specify `rotatef(F, THETA, X0, Y0, 'mesh')` to give the center in mesh units instead of physical units.

`RF = rotatef(...,'trunc')` keeps only the largest rectangular area (with the aspect ratio kept constant) containing only valid (i.e. nonzero) elements. LIMITATION: This option works only for rotation about the center - do not use it for any other center!

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

**rotatef** is useful for correcting vector/scalar fields obtained from skewed images (e.g. misaligned camera).

## Example

```
showvec(rotatef(loadset,0.1));
```

## See Also

[loadvec](#), [vec2scal](#), [showvec](#), [filterf](#), [truncf](#), [extractf](#), [azaverf](#).

# rmdir

Delete a series of directories.

## Description

**rmdir** dir\_name deletes the named directories. Wildcards may be used in the directory name and in the intermediate pathnames, contrarily to RMDIR. Brackets ([]) may also be used (see [expandstr](#)).

Use the functional form of **rmdir**, such as **rmdir**('dir'), when the directory names are stored in a string or a cell array of strings.

The syntax is the same as for MATLAB's RMDIR. In particular, the additional input and output arguments of RMDIR can be used, i.e. [SUCCESS,MESSAGE,MESSAGEID] = **rmdir**(DIRECTORY,MODE). See RMDIR for details.

To delete files, see [rdelete](#).

## Examples

```
rmdir('mydir*') deletes all the directories 'mydir*'
```

```
rmdir('mydir*/sub[1:10,2]') deletes the subdirectories 'sub01',  
'sub02',... in each directory mydir*.
```

## See Also

RMDIR, [expandstr](#), [rdir](#), [rdelete](#), DELETE.

[Previous: rotatef](#)[Next: shiftf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# shiftf

Shift the axis of a vector/scalar field.

## Description

SF = **shiftf**(F) returns the same vector/scalar field(s) with the axis X and Y shifted, so that (X,Y)=(0,0) is in the left corner of the field (bottom-left corner when Y is upward, and top-left corner when Y is downward).

SF = **shiftf**(F, OPT) does the same, specifying the new origin (0,0):

- 'center', 'c', 'middle'
- 'bottomleft', 'bl' (by default)
- 'bottomright', 'br'
- 'topleft', 'tl'
- 'topright', 'tr'

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

## See Also

[truncf](#), [extractf](#), [rotatef](#), [bwfilterf](#).

# showf

Shortcut for SHOWVEC or SHOWSCAL

## Description

**showf**(...) or MOV=**showf**(...) calls [showvec](#) for vector fields, and [showscal](#) for scalar fields.

## Examples

```
showf *.vec
```

```
showf(vec2scal(v,'norm'),'cmap','gray');
```

## See Also

[showvec](#), [showscal](#).

[Previous: shiftf](#)

[Next: showscal](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# showscal

Display scalar field(s)

## Description

**showscal**(S) displays the scalar field(s) S, as a still image or as a movie. S is a structure (or a structure array) that can be obtained by [vec2scal](#).

During the movie, the following commands are available:

- Space: Enter/exit the 'pause' mode
- Right arrow, or 'n' : next frame.
- Left arrow, or 'b' : previous frame.
- 'g' or 'j' : jump to a specified frame
- Esc or 'x' or 'q' or Ctrl+C : exit **showscal**.

**showscal**(S, 'PropertyName', PropertyValue, ...) specifies the property name / property value pairs (only the first letters of the PropertyName are required):

'CMap':	argcolormap	:	Specifies the colormap, where 'argcolormap' may be a string ('gray', 'jet' etc) or a m-by-3 matrix of real numbers between 0.0 and 1.0. See the reference page for COLORMAP.
'CLim':	'all', 0	:	normalize the bounds of the color map using the min and max for all the scalar fields (by default).
	'each', -1	:	normalize the bounds of the color map using the min and max of each scalar field.
	[CMIN CMAX]	:	specifies the min and max of the colormap.
	CMAX	:	specifies the max of the colormap (0 or -CMAX is taken by default for the min)
'Contour':	n	:	Displays n contour lines (n=8 by default)
'Contourf':	n	:	Displays n filled contour lines (n=8 by def.)
'Contour3':	n	:	Displays n 3D contour lines (n=8 by def.)
'Surf':	-	:	3-D shaded surface (see SURF)
'Surfc':	-	:	3-D shaded surface with a contour plot beneath the surface (see SURFC)
'Surfl':	-	:	3-D shaded surface with colormap-based lighting (see SURFL)
'Mesh':		:	3-D wireframe parametric surface (see MESH)
'Meshc':		:	3-D wireframe parametric surface with a contour plot beneath the mesh(see MESHc)
'Pause':	-	:	directly enter in the 'pause' mode (press 'space' to exit the pause mode)
'Loop':	-	:	loop the movie (press 'esc' to exit)
'Backward':	-	:	plays backward

'Delay'            n            : waits n seconds between each frame  
 'KeepCameraSettings' : do not refresh the camera angle, view etc.  
 'Title'            TIT           : string of the title. The following  
 replacements

are performed (default = '%s [%i]'):  
 '%n' : name of the field  
 '%s' : name of the set  
 '%i' : field number in the set  
 '%t' : time (see [getpivtime](#))

**showscal**(S, C) is equivalent to **showscal**(S, 'CLim', C), where C can be 'all', 'each', CMAX, [CMIN CMAX] (see below).

MOV = **showscal**(S,...) returns a movie, that can be re-displayed using MOVIE, or saved as an AVI file using MOVIE2AVI.

If S is a vector field, **showscal** displays [vec2scal](#)(S,'norm').

If S is a string (or a cell array of strings), first [loadvec](#) the field(s) and displays [vec2scal](#)(S,'norm').

## Examples

```
showscal(vec2scal(loadset,'norm'));
```

```
showscal(vec2scal(loadset,'norm'),'cmap','gray');
```

```
m=showscal(s,'CLim',[0 14]); movie(m);  
movie2avi(m, 'mymovie');
```

```
rot=vec2scal(filterf(loadset,1),'rot');  
showscal(rot,'cont',12,'pause','CLim','each');
```

```
showscal('b[1:5].v*');        % loads the files 'b00001.vec' etc, and  
                              % displays the norm.
```

```
showscal *.vc7 pause
```

```
showscal('*.vec','loop','back');
```

## See Also

[vec2scal](#), [loadvec](#), [loadset](#), [showvec](#), [operf](#), MOVIE, IMAGE, SURF, SURFC, SURFL, CONTOUR, MESH, MESH, COLORMAP, [showf](#).

[Previous: showf](#)

[Next: showvec](#)

2005-2007 [PIVMat Toolbox 1.60](#)



# showvec

Display vector field(s)

## Description

**showvec**(V) displays the vector field(s) V, as a still image or as a movie. V is a structure (or a structure array) that can be obtained from [loadvec](#) (or [loadset](#)). By default, **showvec** uses the norm of the vector field for the background.

During the movie, the following commands are available:

```

Space: Enter/exit the 'pause' mode
Right arrow, or 'n' : next frame.
Left arrow, or 'b' : previous frame.
'g' or 'j' : jump to a specified frame
Esc or 'x' or 'q' or Ctrl+C : exit showvec.

```

**showvec**(V, BGMODE) uses the BGMODE background ('rot','norm' etc.) See [vec2scal](#) for the list of available background modes. If no BGMODE specified, 'norm' is used by default. Specify BGMODE='off' or '' for no background (white).

**showvec**(V, 'PropertyName', PropertyValue, ...) specifies the property name / property value pairs (only the first letters of the PropertyName are required):

```

'Background' BGMODE : uses the BGMODE background ('rot','norm' etc.)
'Spacing'    [NX NY] : displays 1 vector every NX (resp. NY) in the
                        X (resp. Y) direction.
                N      : idem, with NX=NY=N.
                        (by default, displays 32 vectors in each
                        direction).
'Scale'      SA       : stretches the arrows by SA. Use SA=1 for the
                        arrows to fit in the grid mesh (by default).
'Colorvec'   COL       : Color of the vector arrows. COL may be a
                        string ('r', 'k' etc) or an array [R G B].
'Pause'      -         : directly enter in the 'pause' mode (press
                        'space' to exit the pause mode)
'Loop':      -         : loop the movie (press 'esc' to exit)
'Backward':  -         : plays backward
'Delay'      n         : waits n seconds between each frame
'Title'      TIT       : string of the title. The following

```

replacements

```

are performed (default = '%s [%#i]'):
'%n' : name of the field

```

```

'%s' : name of the set
'%i' : field number in the set
'%t' : time in sec. (see getpivtime)
'%t0' : time, with origin t(1)=0.

```

Some additional PropertyName/Value pairs required for the display of the background scalar field may also be provided; they will be passed to [showscal](#):

```

'CMap': argcolormap : Specifies the colormap, where 'argcolormap'
                    may be a string ('gray', 'jet' etc) or a
                    m-by-3 matrix of real numbers between 0.0 and
                    1.0. See the reference page for COLORMAP.
'CLim':  'all', 0    : normalize the colormap using the min and
                    max for all the scalar fields (by default).
        'each', -1 : normalize the colormap using the min and
                    max of each scalar field.
        [CMIN CMAX]: specifies the min and max of the colormap.
        CMAX       : specifies the max of the colormap (0 or
                    -CMAX is taken by default for the min)
'Contour'   n      : Displays n contour lines
'Contourf'  n      : Displays n filled contour lines

```

MOV = **showvec**(...) returns a movie, that can be re-displayed using MOVIE, or saved as an AVI file using MOVIE2AVI.

**showvec** file\_name is a shortcut for **showvec**([loadvec](#)('file\_name')).  
**showvec**(FILENAME,...) is a shortcut for **showvec**([loadvec](#)(FILENAME),...)

**showvec** without input argument opens a dialog box for file selection.

## Examples

```

v=loadvec('B00001.vec');
showvec(v,'norm','scale',2);

showvec *.vec

showvec('set.mat','loop','title','t = %t s');

showvec(filterf(v,1),'rot','clim','all','spacing',8);

movie2avi(showvec(loadset,'rot','contour'),'curl.avi');

```

```
showvec(v(1:10), 'norm', 'cmap', 'gray', 'colorvec', 'y');
```

## See Also

[loadvec](#), [loadset](#), [vec2scal](#), [showscal](#), [operf](#), MOVIE, IMAGE, QUIVER, COLORMAP, [showf](#).

[Previous: showscal](#)

[Next: spaverf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# spaverf

Spatial average over X and/or Y of a vector/scalar field

## Description

$FF = \mathbf{spaverf}(F)$  computes the spatial average of the vector/scalar field  $F$ .  $FF$  is a uniform vector/scalar field. If  $F$  is an array of fields,  $\mathbf{spaverf}(F)$  computes the individual spatial average of each field.

$FF = \mathbf{spaverf}(F, \text{axis})$  averages over the  $\text{axis}$  direction, where  $\text{axis}$  is 'X', 'Y' or 'XY'. Default is 'XY'.  $FF$  is a vector/scalar field uniform in the  $\text{axis}$  direction.

By default, **spaverf** considers that the zero elements of  $F$  are erroneous, and does not include them in the computations. If however you want to force the zero elements to be included in the computations, specify  $\mathbf{spaverf}(F, '0')$ .

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

## See Also

[averf](#), [subaverf](#), [azaverf](#).

[Previous: showvec](#)

[Next: statf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# statf

Statistics of a vector/scalar field

## Description

STAT = **statf**(F) returns a structure that contains statistics of the vector/scalar field F. F is a structure (or structure array) of fields

as obtained by [loadvec](#) (vector field) or [vec2scal](#) (scalar field).

If F is a scalar field, STAT contains the following fields:

- mean, std, rms: mean, standard deviation and rms.
- n: number of nonzero elements.
- zeros: number of zero elements.
- mom: non-centered moments,  $\langle s^n \rangle$
- momabs: non-centered absolute moments,  $\langle |s^n| \rangle$
- cmom: centered moments,  $\langle (s - \langle s \rangle)^n \rangle$
- cmomabs: centered absolute moments,  $\langle |s - \langle s \rangle|^n \rangle$
- skewness: normalized 3rd order non-centered moment.
- flatness: normalized 4th order non-centered moment.
- skewnessc: normalized 3rd order centered moment.
- flatnessc: normalized 4th order centered moment.
- history: remind for which field **statf** has been called.

For the "centered" quantities, the brackets  $\langle \rangle$  means spatial and ensemble average over the whole fields.

If F is a vector field, use [STATX,STATY] = **statf**(F), where STATX and STATY are the statistics for each component of the vector field.

... = **statf**(F,MAXORDER) specifies the maximum order for the moments. MAXORDER=6 is used by default (although its convergence may not be guaranteed).

If the fields contain 0 (zero) values, they are considered as erroneous and are not included into the computations.

## Examples

```
statrot = statf(vec2scal(filterf(loadset,1),'rot'));
```

## See Also

[loadvec](#), [vec2scal](#), [histf](#), [corrff](#), [vsf](#).

[Previous: spaverf](#)

[Next: statvec\\_disp](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# statvec\_disp

Displays statistics of vector fields.

## Description

**statvec\_disp**(V) displays statistics of a set of vector fields as a function of the field number. 4 subplots are drawn: mean, standard deviation, rms and energy of each component.

**statvec\_disp**, without input argument, uses the current set.

**statvec\_disp**(V,SMOOTH) smoothes the statistics over SMOOTH consecutive vector fields. Use SMOOTH=1 for no smoothing (by default). Use SMOOTH=0 for smoothing over the whole fields (display only one point).

## Example

```
statvec_disp(loadset,10);
```

## See Also

[statf](#), [histf](#), [histvec\\_disp](#).

[Previous: statf](#)

[Next: subaverf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# subaverf

Subtract the spatial or ensemble average of a field

## Description

`FF = subaverf(F)` subtracts the spatial average from the vector/scalar field `F`. If `F` is an array of fields, `subaverf(F)` subtracts the individual spatial average from each field.

`FF = subaverf(F,AXIS)` subtracts the spatial average (computed by [spaverf](#)), averaged over the direction `AXIS`, where `AXIS='X'`, `'Y'` or `'XY'`. Default is `'XY'`.

`FF = subaverf(F,'e')` subtracts from each `F` the field averaged over all `F`, as computed from [averf](#). The field subtracted is an "ensemble" or "phase" or "temporal" average.

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

## Example

```
showvec(subaverf(loadset,'e'));
```

Note: To subtract both an ensemble average and a spatial average, use `FF = operf('-', F, averf(spaverf(F,'...')));`

## See Also

[averf](#), [spaverf](#), [azaverf](#).

[Previous: statvec\\_disp](#)

[Next: surfheight](#)

2005-2007 [PIVMat Toolbox 1.60](#)



# surfheight

Surface height

## Description

$H = \text{surfheight}(R, H_0, N)$  computes the surface height  $H$ , given the displacement field  $R$ , the mean height  $H_0$  and the refraction index  $N$  ( $N=1.33$  by default). The displacement field is obtained from the correlation of a reference image (flat surface) and a distorted image.

$H = \text{surfheight}(\dots, \text{MODE})$  specifies the integration mode:

- 1 : small slope and arbitrary amplitude (by default)
- 2 : small slope and small amplitude

**surfheight** is based on INTGRAD2.M by J. D'Errico.

## Example

```
r = loadvec('*.vc7');  
h = surfheight(r, 10);  
showscal(h);
```

## See Also

[showvec](#), [showscal](#), [vec2scal](#), [gradientf](#).

[Previous: subaverf](#)

[Next: truncf](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# truncf

Truncate a vector/scalar field.

## Description

TF = **truncf**(F) truncates the vector/scalar field(s) F to the largest centered square.

TF = **truncf**(F, 'nonzero') truncates the vector/scalar field(s) F to the smallest rectangular area excluding zero (erroneous or masked) elements.

TF = **truncf**(F, CUT) eliminates bands of width CUT along the borders of the field (CUT is 0 by default). This is useful when boundary effects are expected (e.g., using [bwfilterf](#)). By default, CUT is specified in mesh units, unless the option 'phys' is specified (e.g., **truncf**(F, CUT, 'phys')).

If no output argument, the result is displayed by [showvec](#) or [showscal](#).

## See Also

[extractf](#), [rotatef](#), [shiftf](#), [bwfilterf](#).

[Previous: surfheight](#)

[Next: vec2mat](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# vec2mat

Convert DaVis files into MAT files

## Description

**vec2mat**(FILENAME) converts the DaVis images and vector fields matching FILENAME into Mat-files, with the same filename but replacing the original suffix by '.mat'. All file formats accepted by [loadvec](#) are accepted (VEC/VC7/IMX/IM7/IMG/TXT/SET). Wildcards (\*) and brackets ([]) may be used (see [expandstr](#)). The resulting MAT-Files can be reloaded using [loadvec](#) (e.g., V=[loadvec](#)('B00001.mat')), or directly using the MATLAB's LOAD function: LOAD('B00001.mat') creates the variable V in the workspace. See [loadvec](#) for the definition of the structure V.

**vec2mat**, without input argument, converts all the DaVis files of the current directory into MAT files.

Reloading MAT-Files converted by **vec2mat** instead of loading the original VEC files saves time (up to a factor 10). It may also be useful to further process the files on platforms other than Windows, which can not directly handle DaVis files (but the conversion itself must be carried out on a Windows platform, since it uses [loadvec](#) and the readimx DLL).

## Example

```
vec2mat('dir*/*.vec') converts all the vec files in the directories matching 'dir*' into MAT-files. These files may be further loaded using v=loadvec('dir*/*.mat').
```

## See Also

[loadvec](#), [loadset](#), LOAD.

2005-2007 [PIVMat Toolbox 1.60](#)

# vec2scal

Converts a vector field into a scalar field

## Description

`S = vec2scal(V)` returns scalar field(s) `S` given by the norm of the vector field(s) `V`. The vector field(s) `V` can be obtained from [loadvec](#) or [loadset](#) and the scalar field(s) `S` can be displayed by [showscal](#).

`S = vec2scal(V,MODE)` specifies the conversion `MODE`:

<code>norm</code>	<code>norm (ux^2 + uy^2)^(1/2)</code> ; by default.
<code>ux, uy</code>	<code>x</code> or <code>y</code> component of the vector field
<code>en, ken</code>	kinetic energy, <code>norm^2 / 2</code>
<code>rad, deg</code>	velocity angle ( <code>tan(angle)=vy/vx</code> ), in rad or deg
<code>curl (or rot)</code>	curl (vorticity field)
<code>absrot</code>	absolute value of the curl
<code>div</code>	2D divergence ( <code>dux/dx + duy/dy</code> )
<code>ens</code>	enstrophy (=square of vorticity)
<code>strain</code>	norm of the strain rate, <code>sqrt(s1^2 + s2^2)</code> , where <code>s1</code> and <code>s2</code> are the 2D strain eigenvalues.
<code>q</code>	Q-criterion, <code>Q = rot^2 - (strain^2)/2</code>
<code>eps</code>	squared strain rate, <code>s1^2 + s2^2</code>
<code>duxdx, duxdy, duydx, duydy</code>	spatial derivatives ( <code>du_i / dx_j</code> )

Adding '-' (minus sign) before `MODE` (e.g., '-rot') inverts the result.

The resulting scalar field `S` contains the following fields:

<code>x,y:</code>	vectors containing the X and Y coordinates
<code>w:</code>	matrix of scalar elements
<code>namex, unitx, namey, unity:</code>	strings for the name and unit of coord
<code>namew, unitw:</code>	strings for the name and unit of the matrix w
<code>name:</code>	name of the VEC file from which originates <code>V</code>
<code>setname:</code>	name of the current directory
<code>history:</code>	Remind from which command <code>S</code> has been obtained

The scalar fields built from derivatives (e.g., `rot`, `div`, `ens` etc.) are computed from 2nd-order centered differences.

Specify '`rot1`', '`div1`' etc. to use 1st-order finite differences (in this case, the resulting field is decreased by 1 unit, and the `x` and `y` coordinates are interpolated).

`S = vec2scal(FILE)` is a shortcut for `S = vec2scal(loadvec(FILE))`.

By default, vectors with a zero component are considered as erroneous, and are not used for the computation of the derivative fields (`rot`, `div`, `eps`, `duxdx`, ...). If however you want to keep them in the

computation, specify **vec2scal**(V,MODE,'keepzero').

Limitation: This option works only with 1st-order derivatives (rot1, div1...) - it is not yet implemented for 2nd-order derivatives (rot, div...).

**vec2scal**(...), without output argument, shows the result with [showscal](#).

## Examples

```
showscal(vec2scal(v,'div'));  
showscal(vec2scal(filterf(loadset,2),'rot'));  
stat_rot = statf(vec2scal(v,'rot'));  
vec2scal *.vec
```

## See Also

[showvec](#), [showscal](#), [gradientf](#), [operf](#).

[Previous: vec2mat](#)

[Next: vortex](#)

2005-2007 [PIVMat Toolbox 1.60](#)

# vortex

Vector field of a centered vortex.

## Description

$V = \mathbf{vortex}$  returns a 128x128 vector field containing at the center a Burger's **vortex** (Gaussian vorticity distribution) of radius 8 mm and vorticity 1 s<sup>-1</sup>.

$V = \mathbf{vortex}(N,R,W)$  returns a N\*N vector field containing a **vortex** of radius R (in mm) and vorticity W (in s<sup>-1</sup>).

$V = \mathbf{vortex}(\dots, \text{MODE})$  specifies the vorticity distribution shape: 'burgers' (by default) or 'rankine' (top-hat vorticity). For the 'burgers mode', a divergence may also be specified by  $V = \mathbf{vortex}(\dots, \text{'burgers'}, D)$ , with D (in s<sup>-1</sup>) of order of W (D=W by default).

## Example

```
showvec(vortex);
```

## See Also

[showvec](#), [randvec](#).

# vsf

Structure functions of a vector field, histograms of increments

## Description

STFUN = **vsf**(F) computes the longitudinal and transverse structure functions of the vector field F, and the histograms of the longitudinal and transverse increments. If F is an array of fields, the structure functions are averaged over the fields.

Longitudinal (resp. Transverse) structure functions of order n are the moments of the vector increments as a function of the separation r,  $\langle (v(x+r) - v(x))^n \rangle$ , where v is the projection of the velocity along (resp. perpendicular) to r.

The structure STFUN contains the following fields:

- r : separation lengths (in mesh units)
- unitr: unit of separation length (e.g., 'mm')
- scaler : scale (in physical units) of a unit separation
- lsf, tsf : R-by-ORDER matrix containing the longitudinal and transverse structure functions of order ORDER.
- lsfabs, tsfabs : idem as lsf and tsf, with absolute values.
- skew\_long, skew\_trans: long. and trans. skewness factor.
- flat\_long, flat\_trans: long. and trans. flatness factor.
- bin : bins for the histograms
- binwidth: width of a bin
- hlvi, htvi : (R x BINVEC) matrix containing the longitudinal and transverse histograms of velocity increments.
- pdflvi, pdftvi: normalized long. and trans. histograms (PDF).
- history: remind for which field **vsf** has been called.

The resulting structure STFUN may be displayed using [vsf\\_disp](#).

STFUN = **vsf**(F, 'PropertyName', PropertyValue, ...) also specifies the optional input arguments:

- 'bin', BIN : vector where velocity increments should be bined. By default, 1000 bins are taken, chosen from the rms of the first field.
- 'r', R : list of separations delta\_r over which increments are computed, given in mesh units (should be smaller than the largest extent of the field). If not specified, a default separation list is used.
- 'maxorder', ORDER: maximum order of structure function (usually between 4 and 8). ORDER=4 is taken by default



(although its convergence may not be guaranteed).

By default, **vsf** considers the values 0 as erroneous, and does not include them in the histogram. If however the values 0 are to be included, specify **vsf**(..., '0');

## Example

```
stfun = vsf(filterf(loadset,1), 'maxorder', 6);  
vsf_disp(stfun, 'hist', 'skew');
```

## See Also

[vsf\\_disp](#), [histf](#), [corrff](#).

[Previous: vortex](#)

[Next: vsf\\_disp](#)

2005-2007 [PIVMat Toolbox 1.60](#)

## PIVMat Function Reference

# vsf\_disp

Displays velocity structure functions computed from VSF.

## Description

**vsf\_disp**(STFUN,'Property',...) displays the statistics from the structure STFUN created by [vsf](#), as specified by the following properties:

- 'stfun': displays the structure functions
- 'hist': displays the histograms
- 'skewness': displays the skewness factor
- 'flatness': displays the flatness factor

(each plot is displayed in a new figure).

Additional properties may be specified:

- 'norm': normalize the structure functions and the histograms
- 'skip': skip the last dr values (which may be noisy)

**vsf\_disp**(STFILE,'Property',...) does the same, loading the Mat-file STFILE saved by [vsf](#).

## Example

```
v=loadset;  
st=vsf(filterf(v,1));  
vsf_disp(st,'hist','skew');
```

## See Also

[vsf](#).

[Previous: vsf](#)

2005-2007 [PIVMat Toolbox 1.60](#)