THE UNIVERSITY
of MANCHESTER

An Introduction to AVS Data Interchange Application (ADIA)

Student Workbook

S Larkin, A J Grant, F Lin, N Hill

Manchester and North HPC T&EC

ADVISORY GROUP ON
Computer Graphics

MAN T&EC
The Manchester and North HPC
Training and Education Centre

Edition 2.0 December 1994

Document Editor

This document was compiled and typeset by Steve Larkin, Computer Graphics Unit, Manchester Computing Centre using the document preparation system LaTeX.

The document is published and distributed by the Manchester and North High Performance Computing Training and Education Centre (MAN T&EC).

For further information of MAN T&EC see the following World Wide Web page:

http://www.hpctec.mcc.ac.uk/hpctec/

or send email to:

hpc-info@mcc.ac.uk

or contact Andrew Grant (a.j.grant@mcc.ac.uk) or by mail at:

MAN T&EC
Manchester Computing Centre
University of Manchester
Manchester M13 9PL
UK

©1994 Computer Graphics Unit, Manchester Computing Centre

The use of registered names, trademarks etc. in this document does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Table of Contents

1 Introduction

The AVS Data Interchange Application (ADIA) Tutorial

This workbook is a supplement to the Introductory AVS Course to provide tutorial information on the AVS Data Interchange Application for importing data.

Before starting the exercises

The data files for these exercises are not part of the standard AVS release and should be installed under the directory `/usr/avs/auxdata`[1]. If it is not possible to create this directory then please inform users as the course notes assume this directory exists. The answers to the exercises can be found in appendix A.

Data files and modules needed

You will need the two datafiles `fin.dat` and `arrow.dat` for each exercise and the public domain module `scatter_arrows` from the repository of AVS modules. These files can be obtained from either the International AVS Center (IAC) (`avs.ncsc.org`) or the Computer Graphics Unit (CGU), Manchester Computing Centre (`ftp.mcc.ac.uk`) via anonymous FTP.

If you using the IAC site then cd to the appropriate directory given; if you are using the MCC site then first cd to `pub/cgu/avs`.

```
/usr/avs/auxdata/fin.dat       part of avs-course/adia-dat.tar.Z
/usr/avs/auxdata/arrow.dat     part of avs-course/adia-dat.tar.Z

scatter_arrows module          found under avs_modules/mappers/scatter_arrows
```

You may now proceed with the examples detailed in this workbook.

[1] copies of the source code can be obtained from the Computer Graphics Unit, Manchester Computing Centre, University of Manchester M13 9PL, United Kingdom.

2 Using ADIA to import fin.dat

Aims of this exercise

This exercise will illustrate the general features of ADIA by importing the datafile `/usr/avs/auxdata/fin.dat`. Please read the description of the data before attempting the tutorial.

Description of the Data

The data is a downsized version of the AVS sample dataset `bluntfin.fld` which shows the flow over a blunt fin.

The data file is called `/usr/avs/auxdata/fin.dat` and is an ASCII file which represents an $10 \times 8 \times 8$ irregular grid. Each data element has 5 floating point data components and each element has some associated coordinate data to position it in 3D space. Therefore the AVS field mapping type is `irregular`.

The data file format is as follows:

```
density x-momentum y-momentum z-momentum stagnation x-coord y-coord z-coord
```

and a small extract from the file is shown below:

```
2.171 0 0 0 10.7602  0 0 0
2.1523 0 0 0 10.6423  0.0160473 0.129677 0
1.9742 0 0 0 9.61927  0.0757717 0.266559 0
1.6659 0 0 0 7.96548  0.173189 0.379764 0
1.2728 0 0 0 5.94705  0.299646 0.45922 0
0.86119 0 0 0 3.92577  0.443802 0.497866 0
0.7654 0 0 0 3.6417  0.618551 0.501378 0
0.73957 0 0 0 3.52176  1.08654 0.501378 0
0.55002 0 0 0 2.62795  2.47197 0.501378 0
0.45067 0 0 0 2.20615  6.57433 0.501378 0
```

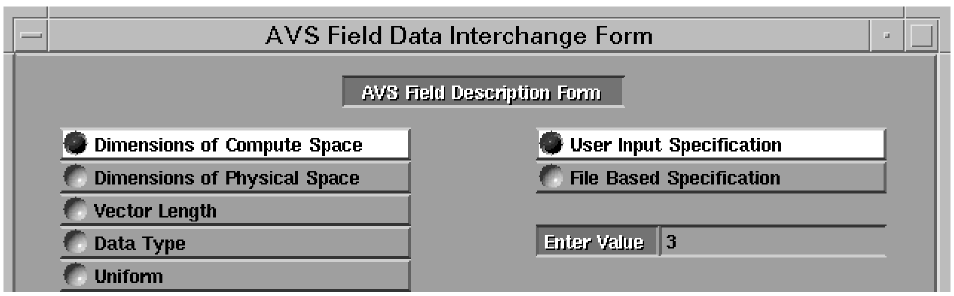
We will use the `line # and word #` method to extract the data and coordinate information. For example to extract all the `y-momentum` data we will start at line 1, move to the third item (word) and loop around (stride) 8 items to extract the next `y-momentum` value i.e., `line 1, word 3, stride 8`.

Starting ADIA

Starting ADIA

To start ADIA drag the module `file_descriptor` (Data Input) from the palette and drop onto the workspace. A popup panel, shown above, should appear after you have placed the module.

Describing the data structure

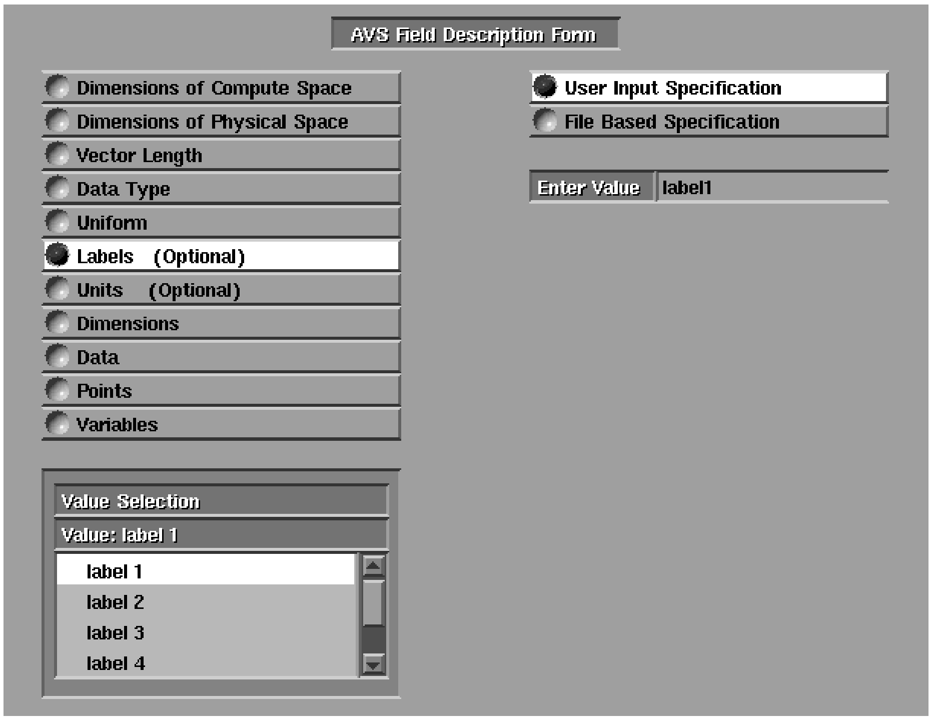


Describing the data structure

On the popup panel select the following menu items and change the value, where necessary, to the values indicated below:

Menu item	New value
`Dimensions of Compute Space`	3
`Dimensions of Physical Space`	3
`Vector Length`	5
`Data Type`	float
`Uniform`	irregular

Describing the data components labels and units

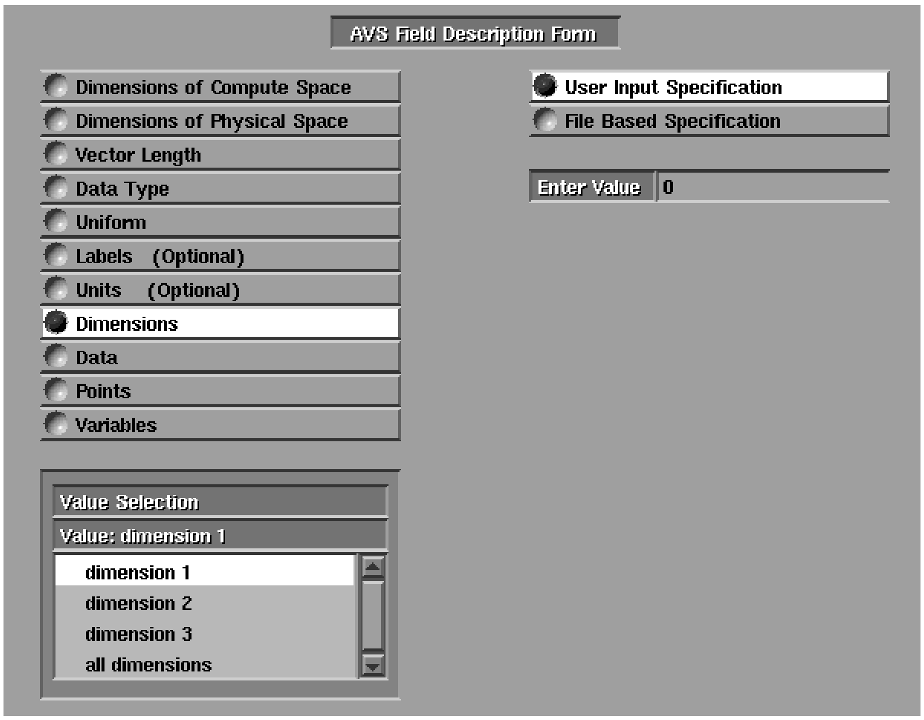


Describing the data components labels and units

1. Select the menu item "`Labels (Optional)`" on the popup menu;
2. In the mini browser which appears below the menu items ensure that "`label 1`" is highlighted;
3. In the panel labelled "`Enter Value`" change "`label1`" to read "`density`";
4. Now in the mini browser select "`label 2`" and change the value in the panel from "`label2`" to "`x-momentum`";
5. Repeat for labels 3–5 changing the values to "`y-momentum`", "`z-momentum`" and "`stagnation`" respectively.

This process can be followed to specify the units aswell but you need not do this now.

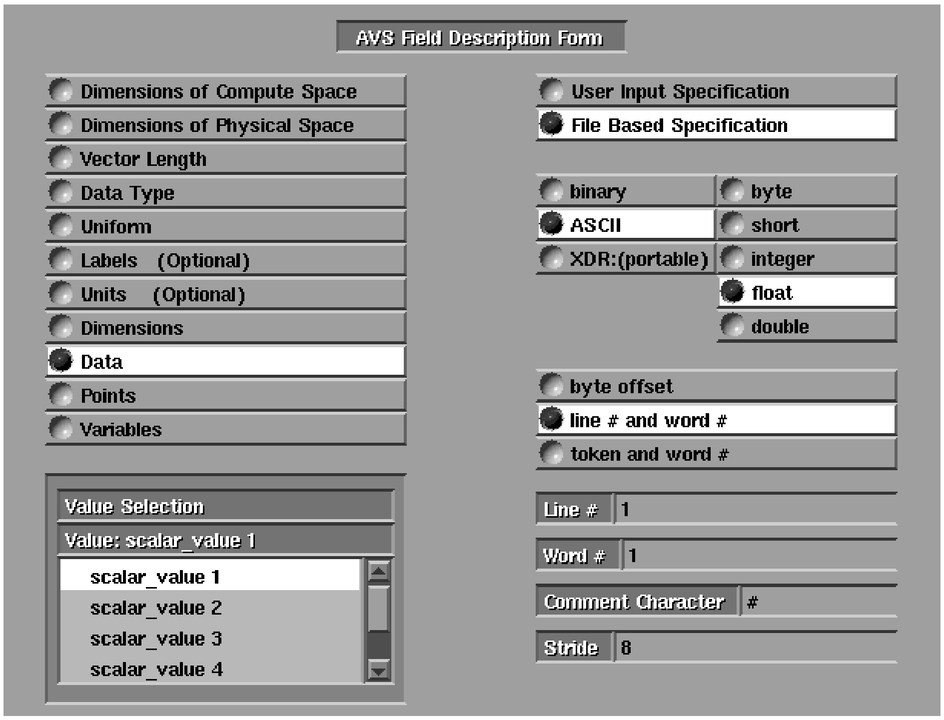
Describing the dimensions



Describing the dimensions

1. Select the menu item "`Dimensions`" on the popup menu;
2. In the mini browser which appears below the menu items ensure that "`dimension 1`" is highlighted;
3. In the panel labelled "`Enter Value`" change "`0`" to read "`10`";
4. Now in the mini browser select "`dimension 2`" and change the value in the panel from "`0`" to "`8`";
5. Do the same for dimension 3 and set this value to 8.

Reading the data components



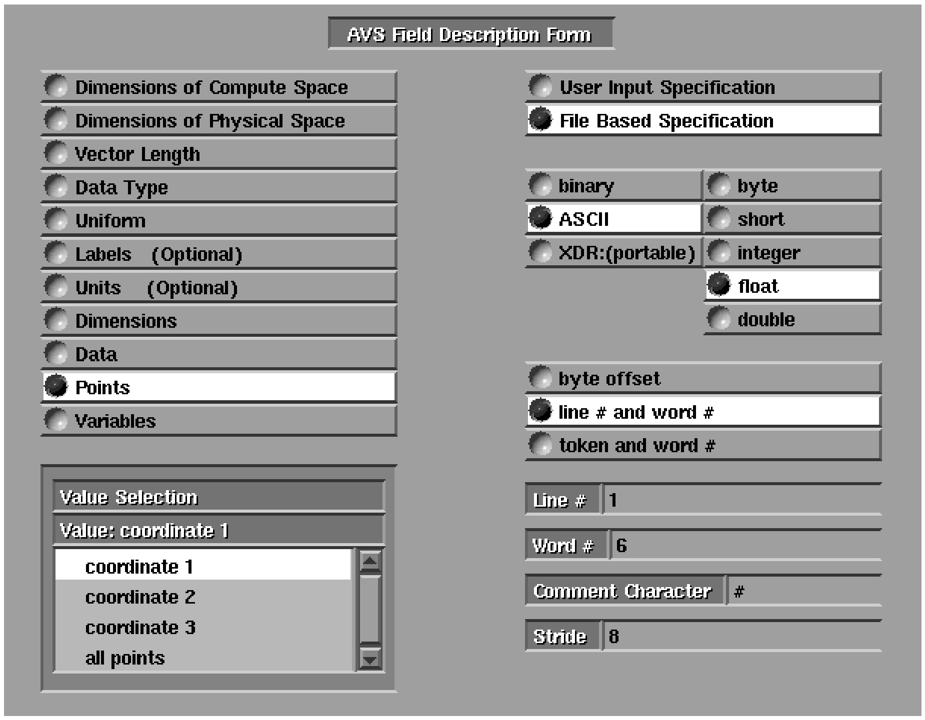
Reading the data components

1. Select the menu item "`Data`";
2. At the top of the popup panel change the setting to "`File Based Specification`";
3. Select the options "`ASCII`" and "`float`" on the top-right of the popup panel;
4. In the mini browser which appears below the menu items ensure that "`scalar_value 1`" is highlighted;
5. Select the option "`line # and word #`";
6. Change the following values:

   `line #` 1
   `word #` 1
   `stride` 8

7. Repeat this for the `scalar_values` 2–5 but alter the `word #` value to be 2,3,4 and 5 respectively.

Reading the data coordinates

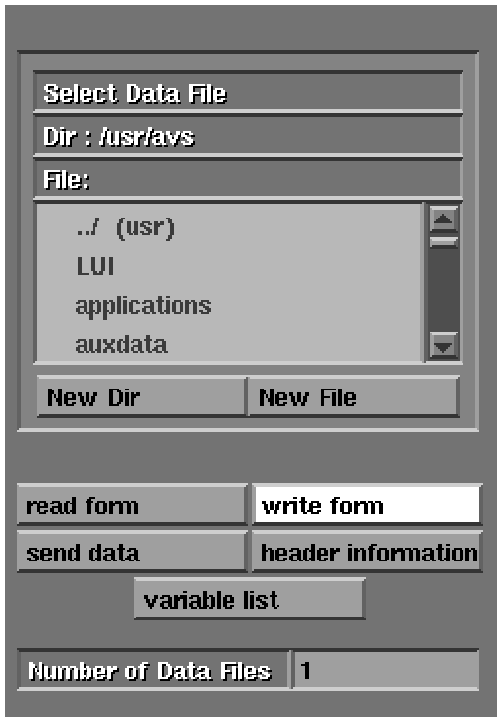


Reading the data coordinates

1. Select the menu item “`Points`”;
2. In the mini browser which appears below the menu items ensure that “`coordinate 1`” is highlighted;
3. At the top of the popup panel change the setting to “`File Based Specification`”;
4. Select the options “`ASCII`” and “`float`” on the top-right of the popup panel;
5. Select the option “`line # and word #`”;
6. Change the following values:

   ```
   line #   1
   word #   6
   stride   8
   ```
7. Repeat this for the `coordinate` 2 and 3 but alter the `word #` value to be 7 and 8 respectively.

Saving the form

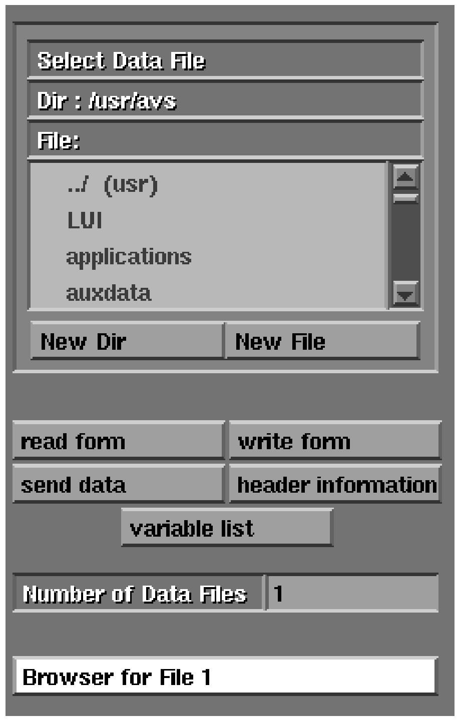


Saving the form

Now return to the network editor and look towards the side of the screen for the parameters associated with the `file_descriptor` module.

1. Select the "`write form`" button underneath the file browser;
2. Click the "`New Dir`" button and change to your home directory;
3. Click on the "`New File`" button and type the name of the file to save the form in e.g., `fin.form`

Importing the data

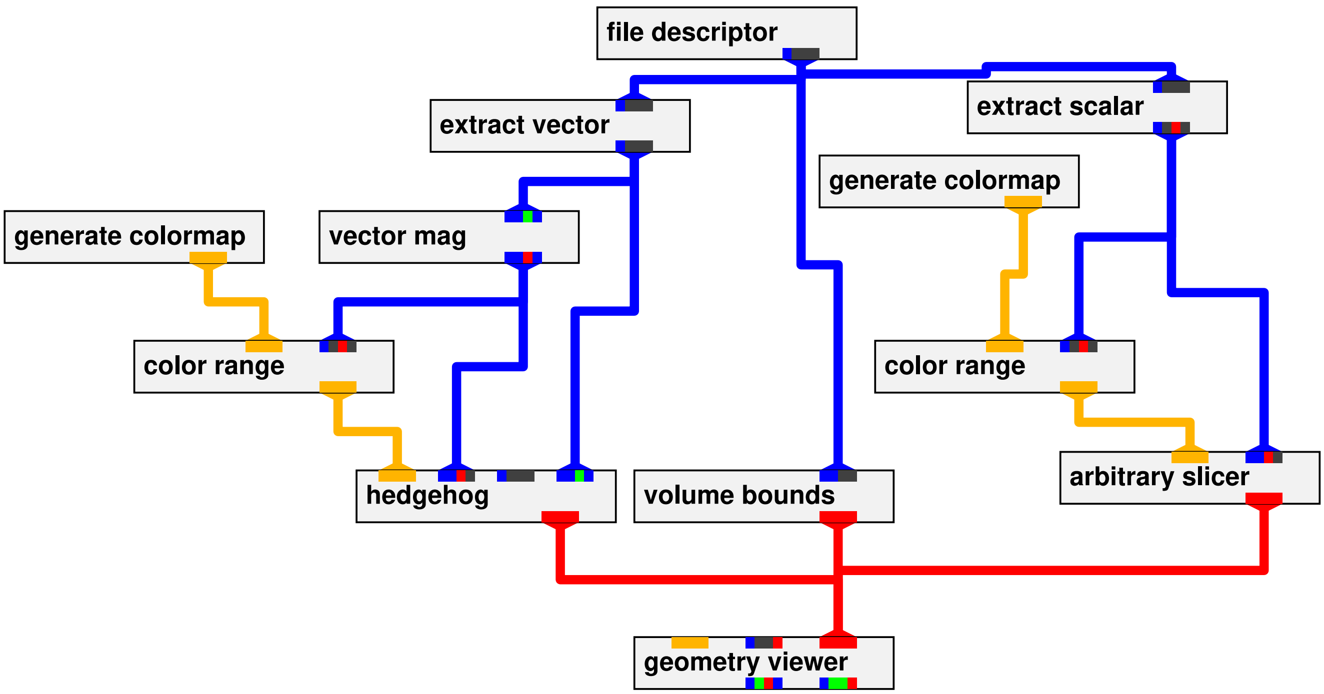


Importing the data

1. Select the "`Browser for File 1`" button underneath the file browser;
2. Click the "`New Dir`" button and change to the directory containing the datafile `fin.dat` e.g., `/usr/avs/auxdata`
3. Select the file `fin.dat`;
4. Click on the button "`send data`" and the data should be imported;

N.B., any errors in the specification of the form you have made will be reported at this stage. To correct them re-edit the relevant items and re-write the form.

Example networks



Example networks

The network above can be used to test that the data is being imported correctly.

3 Using ADIA to import arrow.dat

Aims of this exercise

This exercise will illustrate the keyword features of ADIA by importing the datafile `/usr/avs/auxdata/arrow.dat`. Please read the description of the data before attempting the tutorial.

Description of the Data

The data file is called `/usr/avs/auxdata/arrow.dat` and is an ASCII file which represents a list of scattered 3D vector components. Each data element has 3 floating point data components and each element has some associated coordinate data to position it in 3D space. Therefore the AVS field mapping type is `irregular`.

The data file format is as follows:

```
values = <number of data elements>
x-coord y-coord z-coord x-vel y-vel z-vel
```

and a small extract from the file is shown below:

```
values = 40
0.0000800 0.0000000 0.00 6.2949e-03 0.0000e+00 0.00
0.0000566 0.0000566 0.00 -1.3994e-07 6.2950e-03 0.00
-0.0000000 0.0000800 0.00 -6.2951e-03 0.0000e+00 0.00
-0.0000566 0.0000566 0.00 -1.3994e-07 -6.2950e-03 0.00
```

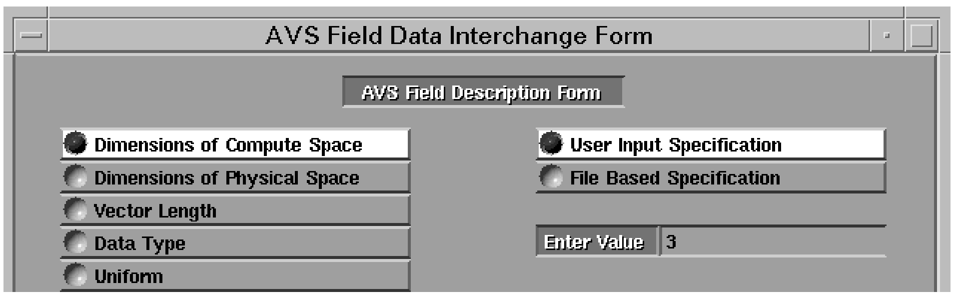
We will use the "`token # word #`" method to extract the length (`dimension 1`) of the datafile. To do this AVS searches for the `token` "`values`" and the length is the third item on that line, `word 3`.

Starting ADIA

Starting ADIA

To start ADIA drag the module `file_descriptor` (Data Input) from the palette and drop onto the workspace. A popup panel, shown above, should appear after you have placed the module.

Describing the data structure

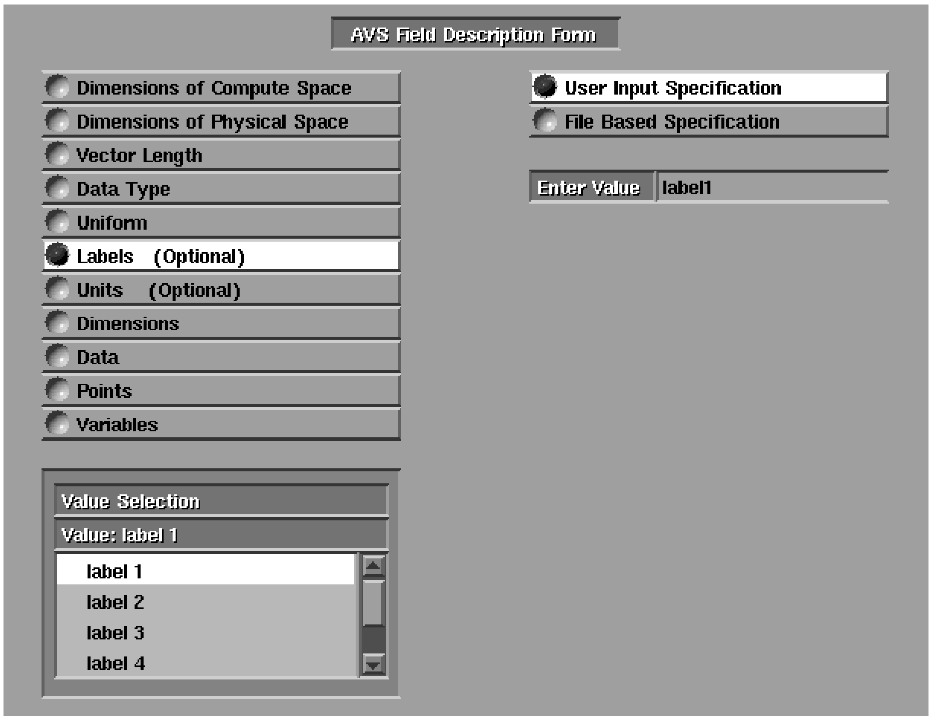


Describing the data structure

On the popup panel select the following menu items and change the value, where necessary, to the values indicated below:

Menu item	New value
`Dimensions of Compute Space`	1
`Dimensions of Physical Space`	3
`Vector Length`	3
`Data Type`	float
`Uniform`	irregular

Describing the data components labels and units

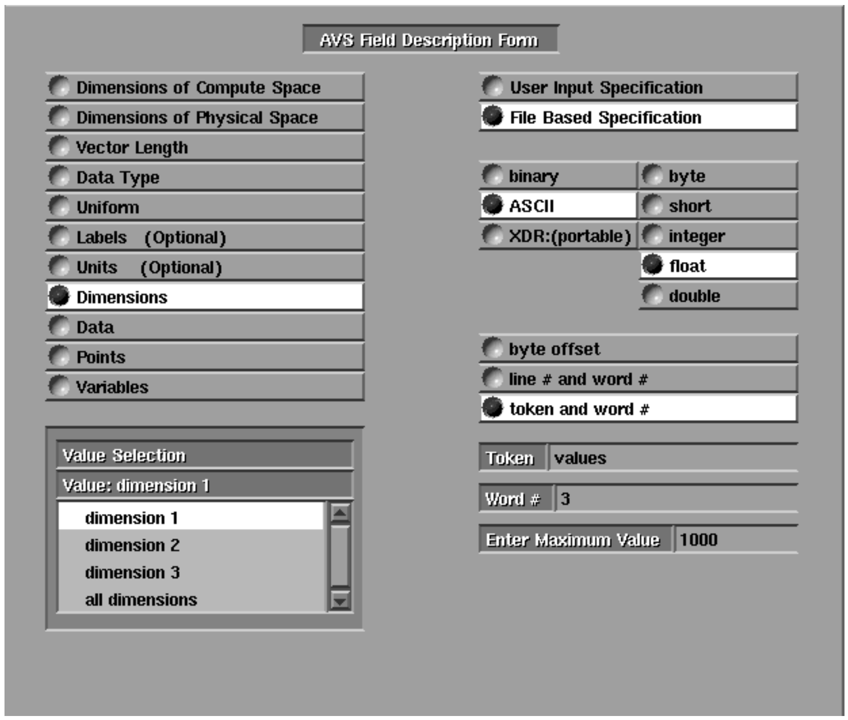


Describing the data components labels and units

1. Select the menu item "`Labels (Optional)`" on the popup menu;
2. In the mini browser which appears below the menu items ensure that "`label 1`" is highlighted;
3. In the panel labelled "`Enter Value`" change "`label1`" to read "`x-vel`";
4. Now in the mini browser select "`label 2`" and change the value in the panel from "`label2`" to "`y-vel`";
5. Repeat for label 3 changing the value to "`z-vel`".

This process can be followed to specify the units aswell but you need not do this now.

Describing the dimensions

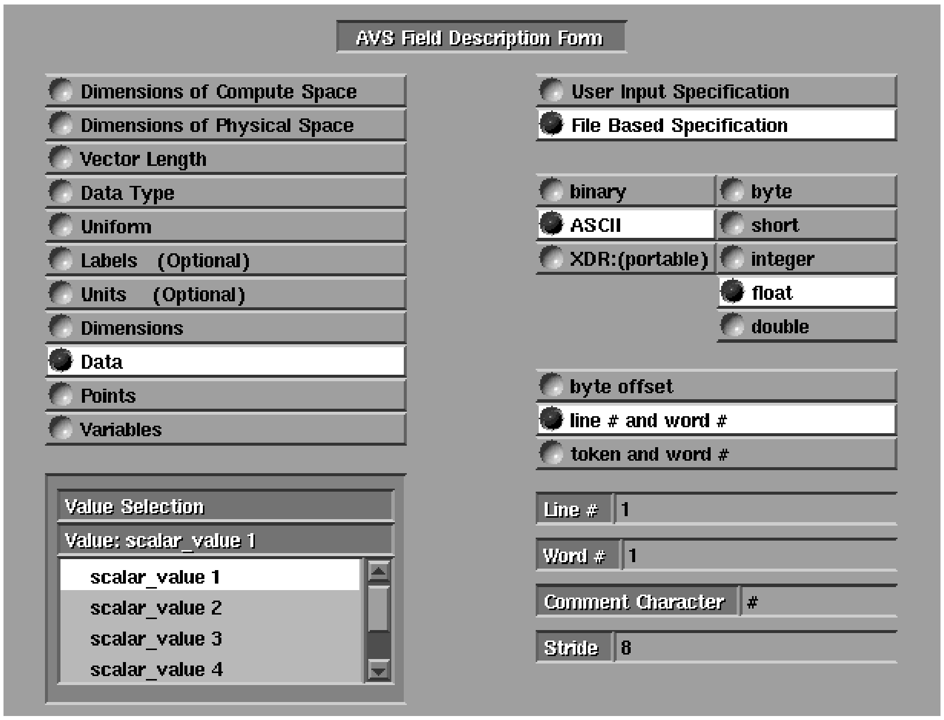


Describing the dimensions

1. Select the menu item "`Dimensions`" on the popup menu;
2. In the mini browser which appears below the menu items ensure that "`dimension 1`" is highlighted;
3. At the top of the popup panel change the setting to "`File Based Specification`";
4. Select the options "`ASCII`" and "`float`" on the top-right of the popup panel;
5. Select the option "`token # and word #`";
6. Change the following values:

```
token #   values
word #    3
```

Reading the data components

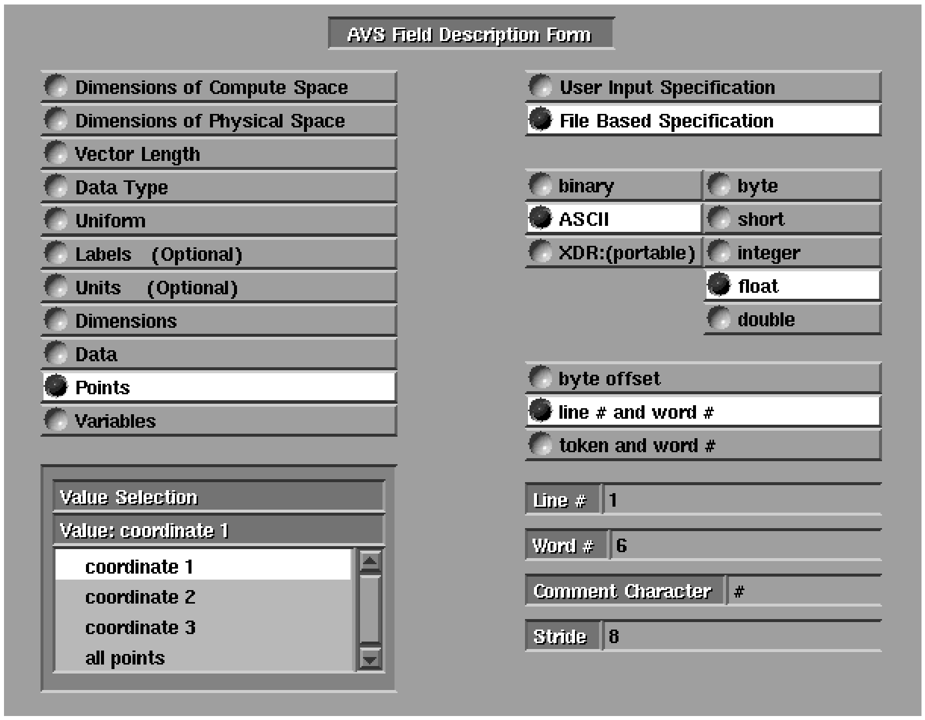


Reading the data components

1. Select the menu item "`Data`";
2. At the top of the popup panel change the setting to "`File Based Specification`";
3. Select the options "`ASCII`" and "`float`" on the top-right of the popup panel;
4. In the mini browser which appears below the menu items ensure that "`scalar_value 1`" is highlighted;
5. Select the option "`line # and word #`";
6. Change the following values:

   ```
   line #   2
   word #   4
   stride   6
   ```
7. Repeat this for the `scalar_values` 2 and 3 but alter the `word #` value to be 5 and 6 respectively.

Reading the data coordinates



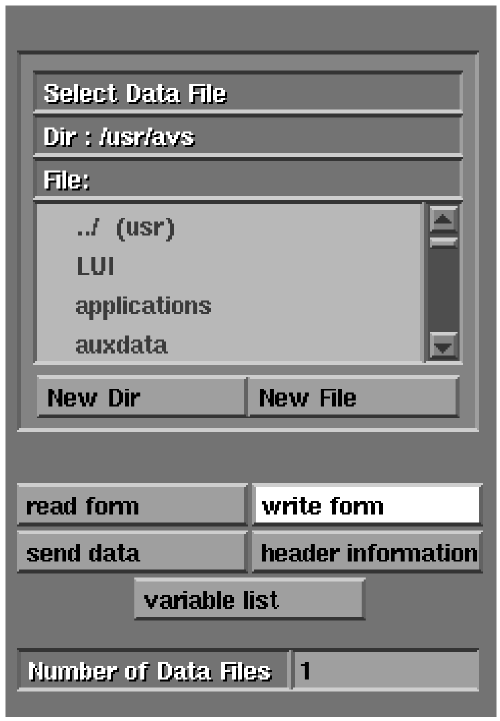
Reading the data coordinates

1. Select the menu item "`Points`";
2. In the mini browser which appears below the menu items ensure that "`coordinate 1`" is highlighted;
3. At the top of the popup panel change the setting to "`File Based Specification`";
4. Select the options "`ASCII`" and "`float`" on the top-right of the popup panel;
5. Select the option "`line # and word #`";
6. Change the following values:

   ```
   line #   2
   word #   1
   stride   6
   ```

7. Repeat this for the `coordinate` 2 and 3 but alter the `word #` value to be 2 and 3 respectively.

Saving the form

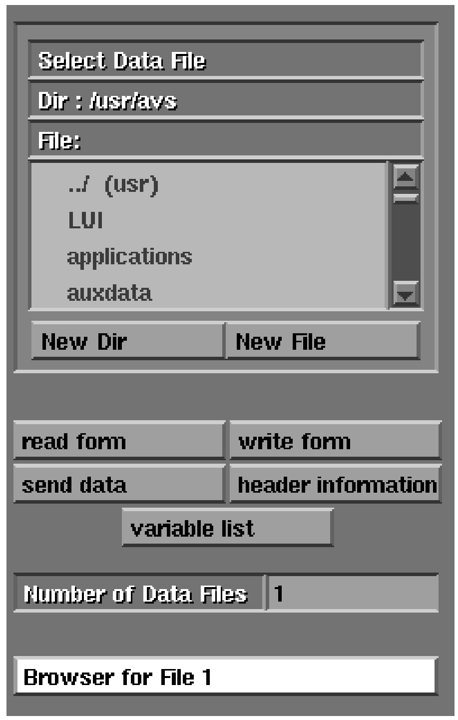


Saving the form

Now return to the network editor and look towards the side of the screen for the parameters associated with the `file_descriptor` module.

1. Select the "`write form`" button underneath the file browser;
2. Click the "`New Dir`" button and change to your home directory;
3. Click on the "`New File`" button and type the name of the file to save the form in e.g., `arrow.form`

Importing the data

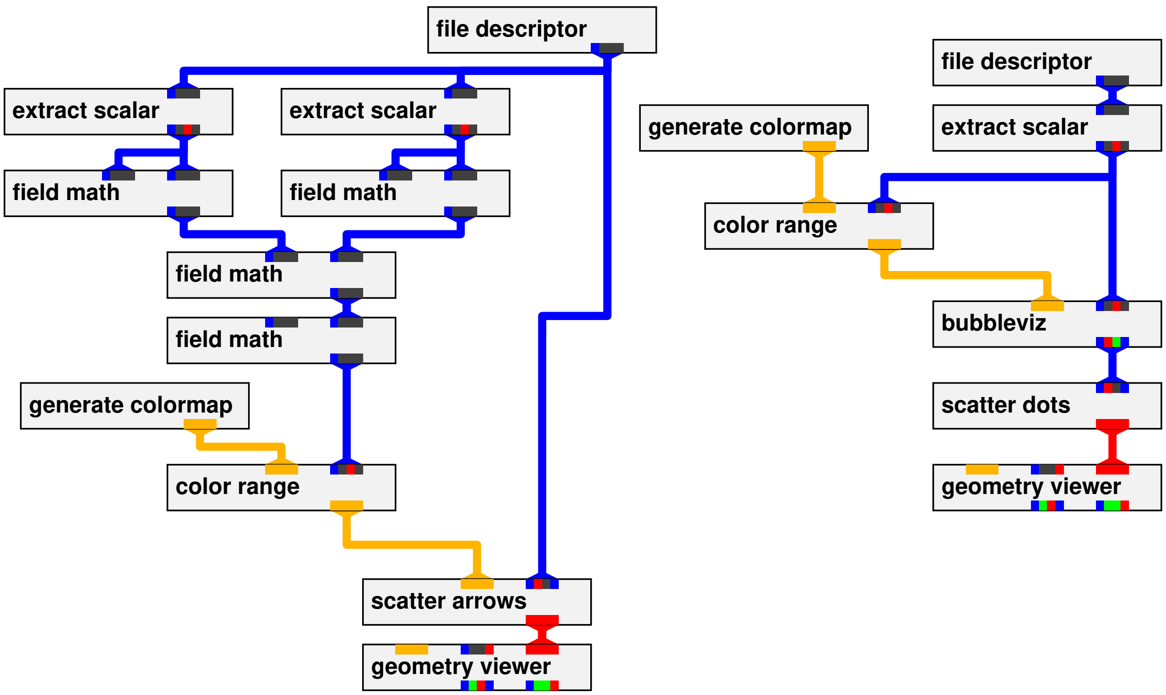


Importing the data

1. Select the "`Browser for File 1`" button underneath the file browser;
2. Click the "`New Dir`" button and change to the directory containing the datafile `arrow.dat` e.g., `/usr/avs/auxdata`
3. Select the file `arrow.dat`;
4. Click on the button "`send data`" and the data should be imported;

N.B., any errors in the specification of the form you have made will be reported at this stage. To correct them re-edit the relevant items and re-write the form.

Example networks



Example networks

The two networks above can be used to test that the data is being imported correctly.

The first network uses the combination of `field_math` modules to provide a 2D version of the `vector_mag` module which currently only imports 3D data. The parameters on the modules should be set to perform the following calculation, $\sqrt{xvel^2 + yvel^2}$.

In the second network set the radius size in `scatter_dots` to be 0.001.

A Answers to the exercises

Form descriptor for `fin.dat`

```
# AVS file descriptor: Version 1.0
#
#
field {
  compute_space 3
  physical_space 3
  vector_length 5
  data_type float
  uniform irregular
  label_1 density
  label_2 x-momentum
  label_3 y-momentum
  label_4 z-momentum
  label_5 stag
  dimension_1 10
  dimension_2 8
  dimension_3 8
  data_1 {
     value        0
     file_number  1
     file_format  ASCII {
       ascii_choice  2
       line_num      1
       word_num      1
       comment       #
     }
     data_type    float
     max_value    ***
     stride       8
  }
  data_2 {
     value        0
     file_number  1
     file_format  ASCII {
       ascii_choice  2
       line_num      1
       word_num      2
       comment       #
```

```
   }
   data_type    float
   max_value    ***
   stride       8
}
data_3 {
   value        0
   file_number  1
   file_format  ASCII {
     ascii_choice  2
     line_num      1
     word_num      3
     comment       #
   }
   data_type    float
   max_value    ***
   stride       8
}
data_4 {
   value        0
   file_number  1
   file_format  ASCII {
     ascii_choice  2
     line_num      1
     word_num      4
     comment       #
   }
   data_type    float
   max_value    ***
   stride       8
}
data_5 {
   value        0
   file_number  1
   file_format  ASCII {
     ascii_choice  2
     line_num      1
     word_num      5
     comment       #
   }
   data_type    float
   max_value    ***
   stride       8
}
coordinate_1 {
   value        0
   file_number  1
   file_format  ASCII {
     ascii_choice  2
     line_num      1
     word_num      6
     comment       #
   }
   data_type    float
   max_value    ***
   stride       8
}
```

```
  coordinate_2 {
     value       0
     file_number  1
     file_format  ASCII {
       ascii_choice  2
       line_num      1
       word_num      7
       comment       #
     }
     data_type    float
     max_value    ***
     stride       8
  }
  coordinate_3 {
     value       0
     file_number  1
     file_format  ASCII {
       ascii_choice  2
       line_num      1
       word_num      8
       comment       #
     }
     data_type    float
     max_value    ***
     stride       8
  }
}
```

Form descriptor for `arrows.dat`

```
# AVS file descriptor: Version 1.0
#
#
field {
  compute_space 1
  physical_space 3
  vector_length 3
  data_type float
  uniform irregular
  label_1 xvel
  label_2 yvel
  label_3 zvel
  dimension_1 {
     value       0
     file_number  1
     file_format  ASCII {
       ascii_choice  3
       token         values
       word_num      3
     }
     data_type    integer
     max_value    100
  }
  data_1 {
     value       0
```

```
    file_number  1
    file_format  ASCII {
      ascii_choice  2
      line_num      2
      word_num      4
      comment       #
    }
    data_type    float
    max_value    ***
    stride       6
 }
 data_2 {
    value        0
    file_number  1
    file_format  ASCII {
      ascii_choice  2
      line_num      2
      word_num      5
      comment       #
    }
    data_type    float
    max_value    ***
    stride       6
 }
 data_3 {
    value        0
    file_number  1
    file_format  ASCII {
      ascii_choice  2
      line_num      2
      word_num      6
      comment       #
    }
    data_type    float
    max_value    ***
    stride       6
 }
 coordinate_1 {
    value        0
    file_number  1
    file_format  ASCII {
      ascii_choice  2
      line_num      2
      word_num      1
      comment       #
    }
    data_type    float
    max_value    ***
    stride       6
 }
 coordinate_2 {
    value        0
    file_number  1
    file_format  ASCII {
      ascii_choice  2
      line_num      2
      word_num      2
```

```
      comment       #
    }
    data_type    float
    max_value    ***
    stride       6
  }
  coordinate_3 {
    value        0
    file_number  1
    file_format  ASCII {
      ascii_choice  2
      line_num      2
      word_num      3
      comment       #
    }
    data_type    integer
    max_value    ***
    stride       6
  }
}
```