Introduction to IDL®

A Kiewit Minicourse

Simon Shepherd
Thayer School of Engineering
simon@thayer.dartmouth.edu
http://thayer.dartmouth.edu/~simon

IDL

Agenda

What is IDL?

How to run IDL

Simple IDL Commands

Arrays in IDL

Data I/O

Basic Plotting

Built-in Analysis Functions

Mapping

Example

Scientific Computing Today

Traditional programming languages:

Basic, Pascal, FORTRAN, C, C++, Java

Scripting languages: sh, csh, bash, Tcl, Perl

Numeric/graphics: IDL, Matlab, Octave

Symbolic: Maple, Mathematica, MathCad

Dataflow: LabView, Simulink, AVS, DX, OpenDX

What is IDL?

Interactive Data Language

IDL

What is IDL?

Interactive Data Language

Proprietary software distributed by Research Systems, Inc. of Boulder, CO now a division of Kodak.

IDL

What is IDL?

Interactive Data Language

Proprietary software distributed by Research Systems, Inc. of Boulder, CO now a division of Kodak.

Grew out of programs written for analysis of data from NASA missions such as Mariner and the International Ultraviolet Explorer.

What is IDL?

Interactive Data Language

Proprietary software distributed by Research Systems, Inc. of Boulder, CO now a division of Kodak.

Grew out of programs written for analysis of data from NASA missions such as Mariner and the International Ultraviolet Explorer.

Oriented toward use by scientists and engineers in the analysis and visualization of multi-dimensional data sets.

What is IDL?

Interactive Data Language

Proprietary software distributed by Research Systems, Inc. of Boulder, CO now a division of Kodak.

Grew out of programs written for analysis of data from NASA missions such as Mariner and the International Ultraviolet Explorer.

Oriented toward use by scientists and engineers in the analysis and visualization of multi-dimensional data sets.

Platform Independent: Unix, linux, Windows, Macintosh

What is IDL?

Interactive Data Language

Offers all the power, adaptability, and programmability of high level languages like FORTRAN, C, and C++.

IDL

What is IDL?

Interactive Data Language

Offers all the power, adaptability, and programmability of high level languages like FORTRAN, C, and C++.

But it adds two capabilities which are essential for modern data analysis: interactivity and graphics display.

IDL

What is IDL?

User-written .pro files

User-written functions in C or Fortran

Numerical and graphical .pro files

User interface
Core numerics and graphics

Ways to Run IDL

Interactively

- Development Environment (DE)
- Command Line

Batch mode

Ways to Run IDL

Batch mode

```
prompt> idl filename
```

where *filename* contains a list of IDL commands

commands are interpreted as if they were entered at the command line

No multiline statements such as

```
begin
...
end
```

Ways to Run IDL

Command Line

```
prompt> idl

IDL> command

IDL>
```

each *command* entered is interpreted and executed
and the prompt is returned

Ways to Run IDL

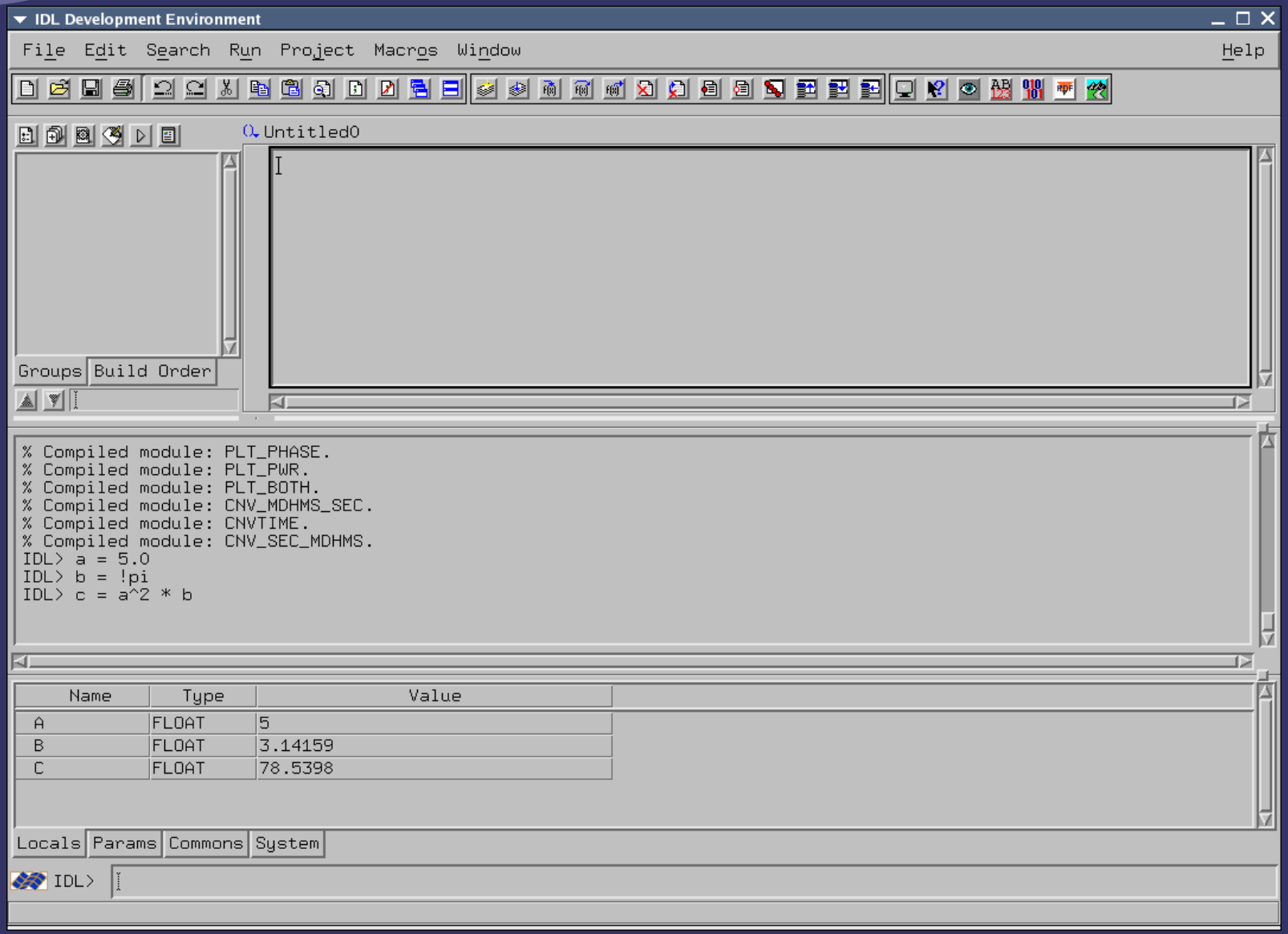
Development Environment

```
prompt> idlde
```

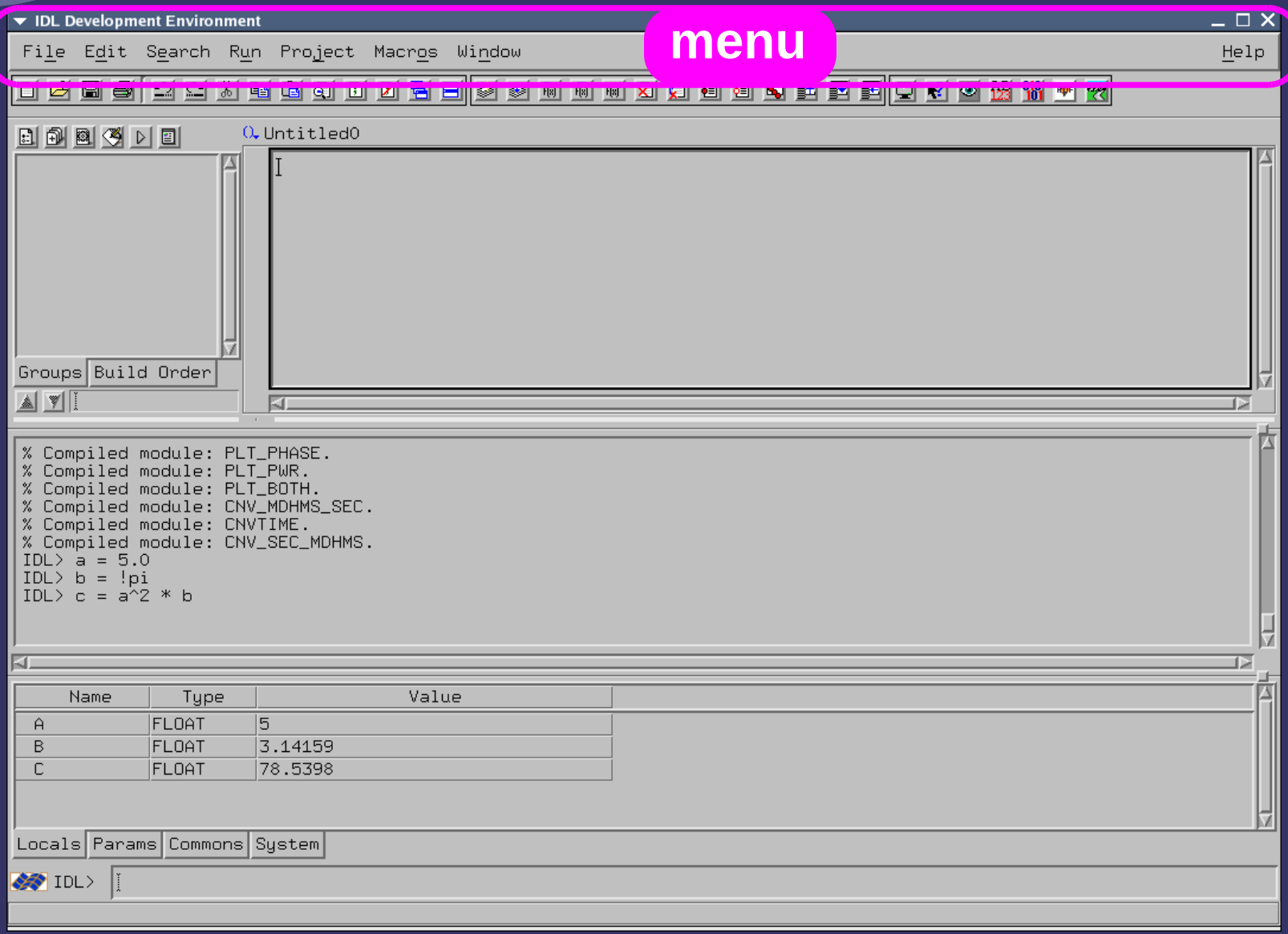
or

select from desktop, panel, toolbar, or launcher

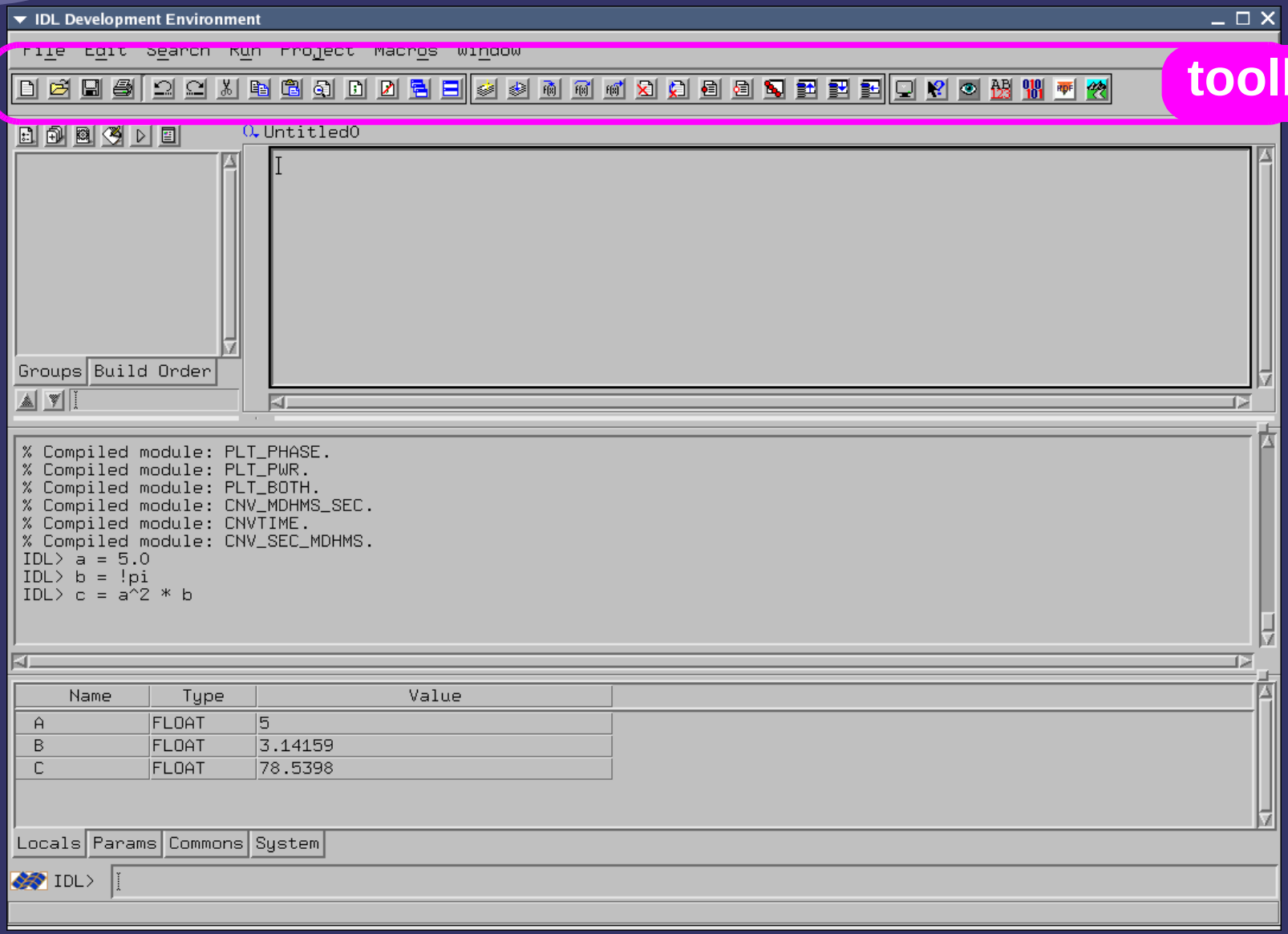
Graphical User Interface (DE)



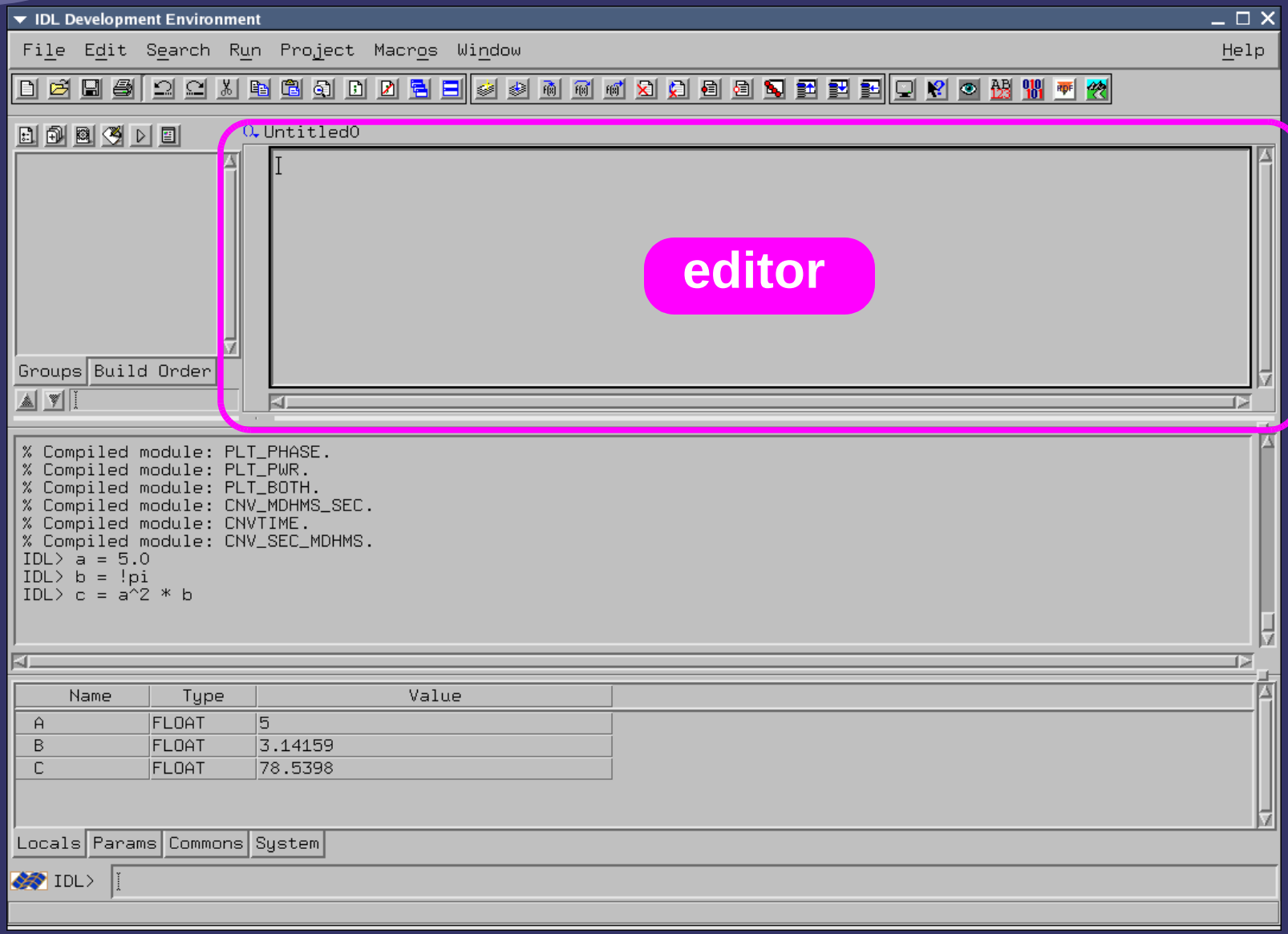
Graphical User Interface (DE)



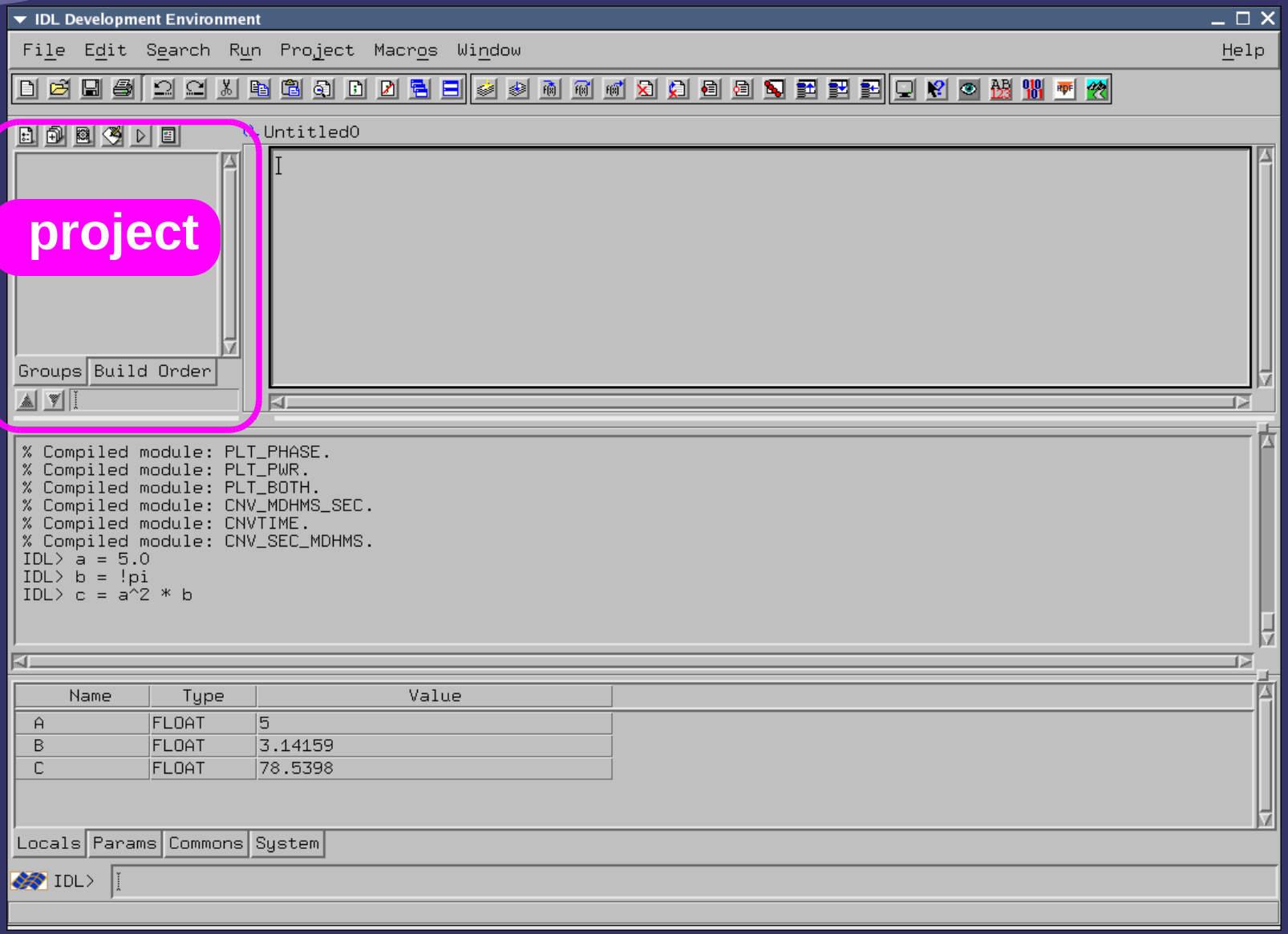
Graphical User Interface (DE)



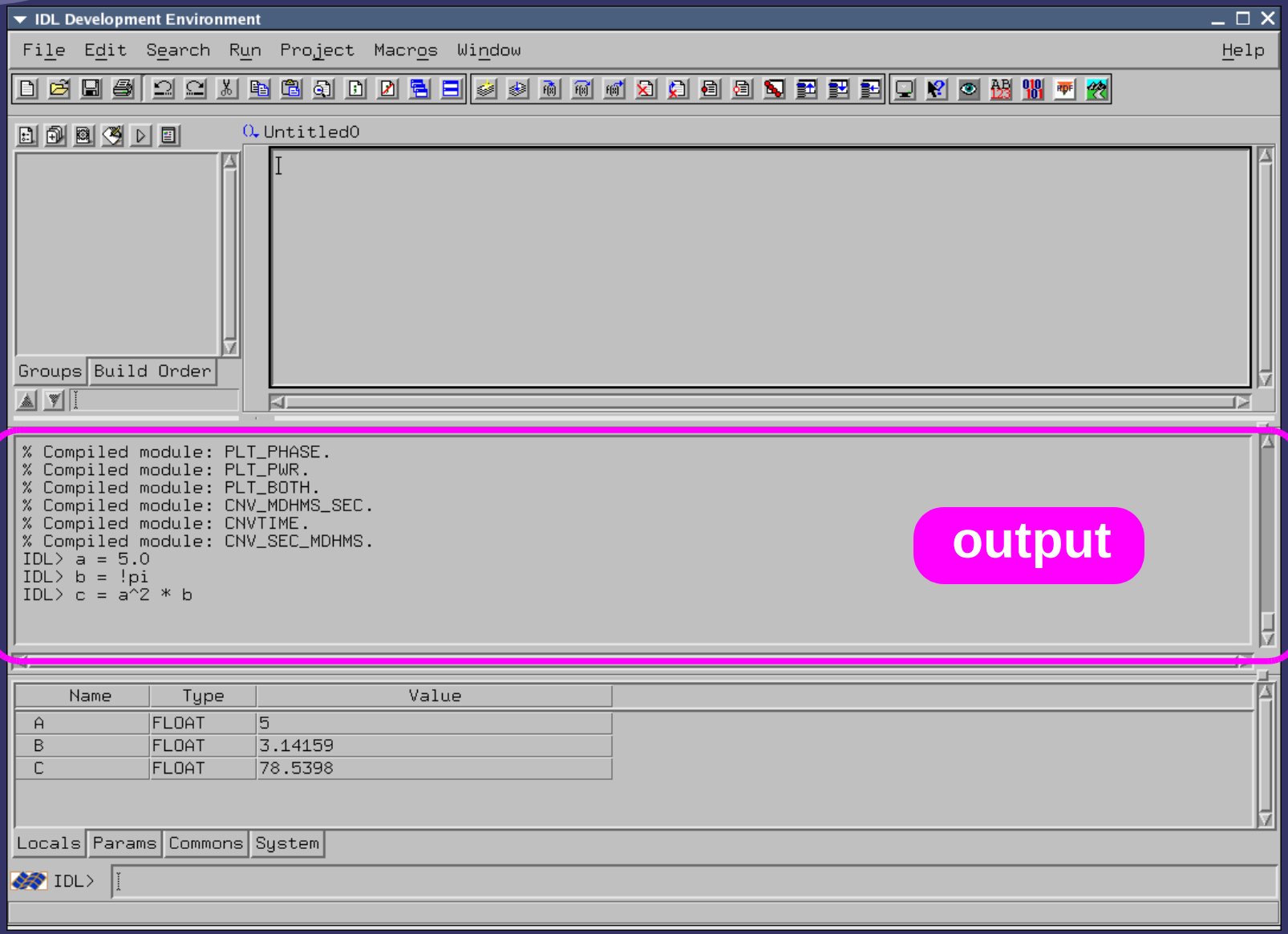
Graphical User Interface (DE)



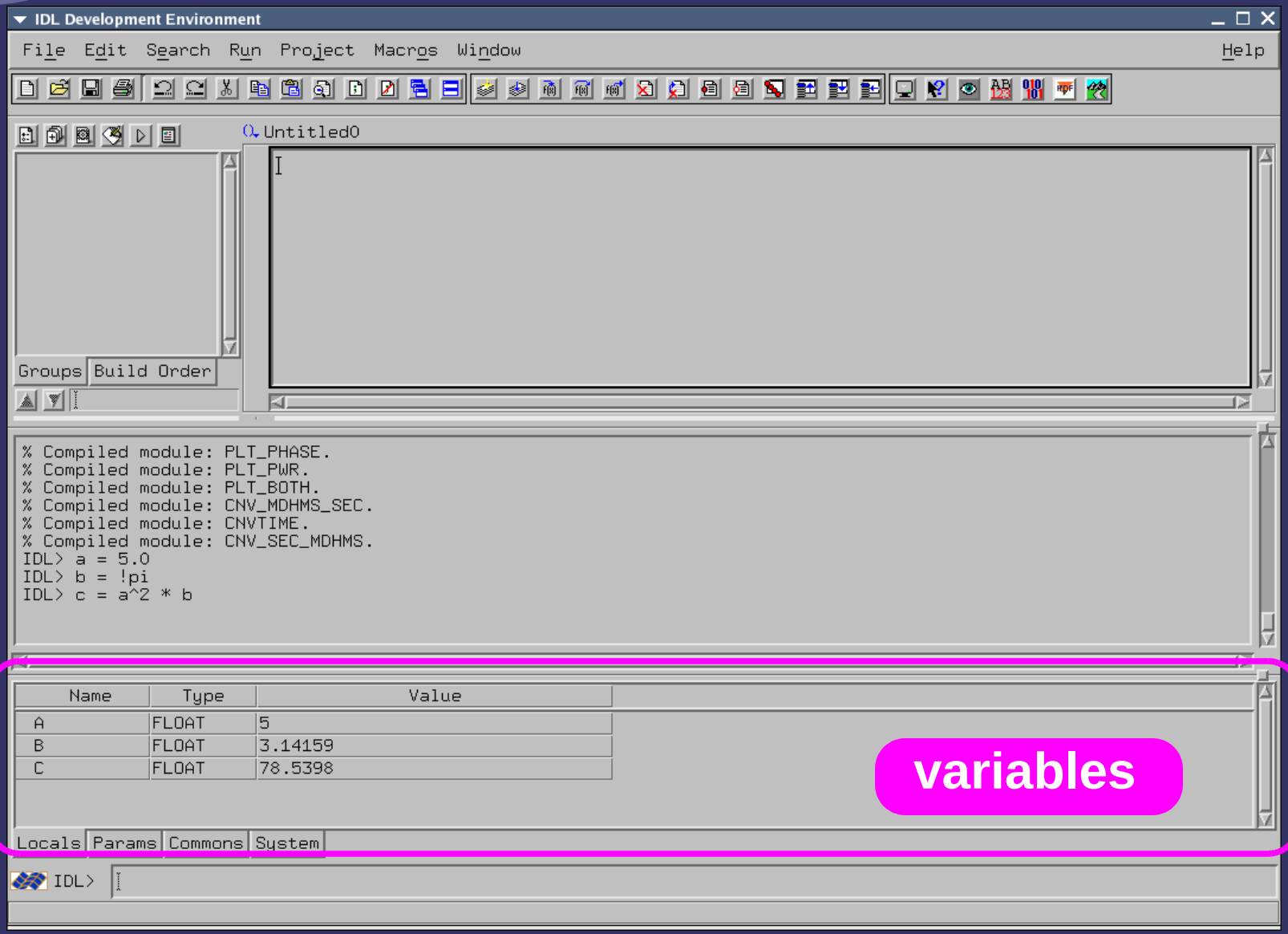
Graphical User Interface (DE)



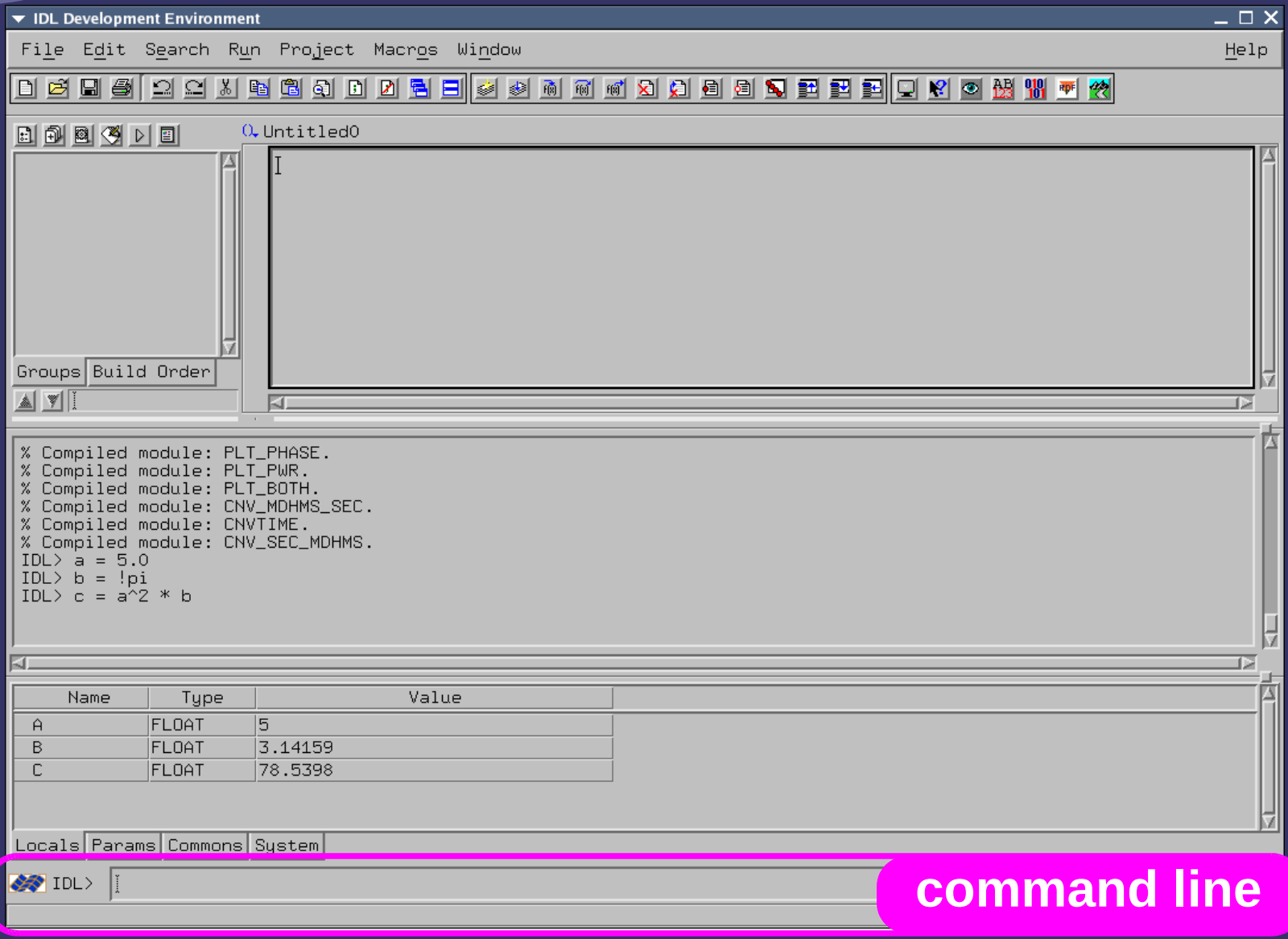
Graphical User Interface (DE)



Graphical User Interface (DE)



Graphical User Interface (DE)



Using IDL as a calculator

```
IDL> a = 3              ; assignment statement
                        ; semicolon is used for comments
IDL> print, a + 4 ; built-in print command
       7
IDL> print, a + 4.; variables dynamically typed
       7.00000          ; take on highest precision


IDL> print, a + 4D; double precision
       7.0000000

IDL> print, a * 1e-9     ; scientific notation
       3.00000e-09
```

Note: variable names are case insensitive (A = a)

IDL

Data Types

1	Byte	*n*B	8-bit unsigned
2	Integer	*n*	16-bit signed
3	Longword	*n*L	32-bit signed
4	Floating Point	*n.n*	32-bit ($+10^{38}$)
5	Double-Precision	*n.n*D	64-bit ($+10^{38}$)
6	Complex	COMPLEX(*n.n,n.n*)	real/imaginary pairs
7	String	*'ssss'*	0-32k in length
8	Struct		User-defined Structure
9	D-P Complex	DCOMPLEX(*n.n,n.n*)	real/imaginary pairs double-precision
10	Pointer		Pointer
11	ObjRef		Object Reference
12	Unsigned Int	*n*U	16-bit unsigned
13	Unsigned Long	*n*UL	32-bit unsigned
14	64-bit Long	LONG64(*n*)	64-bit signed longword
15	64-bit Unsigned Long	ULONG64(*n*)	64-bit unsigned longword

Unary and Compound Operators

```
IDL> a = 3          ; assignment statement

IDL> a += 5         ; C compound operators
IDL> a = a + 5      ; equivalent but rather tedious
IDL> print, a
       8

IDL> a++            ; C unary operators
IDL> print, a
       8

IDL> print, a--     ; careful...
       8
IDL> print, a
       7
```

Arrays in IDL

Array-oriented language

operations are performed on arrays

Any data type (but not mixed)

Dynamically sized (and resized)

Vectors in IDL

```
IDL> vec = [3,5,1,4,5]  ; assignment statement

IDL> print, vec   ; 5 element vector
       3       5       1       4       5
```


```
IDL> vec[3] = 9          ; square brackets to access

IDL> print, vec[3]       ; elements of array
       9
```

Matrices in IDL

```
IDL> mat = [[3,5,1,4,5],$  ; $ is line continuation
IDL>        [8,3,2,9,1]]
IDL> print, mat                ; 5-column 2-row array
       3       5       1       4       5
       8       3       2       9       1

IDL> help, mat                 ; useful command...
MAT             INT       = Array[5, 2]

IDL> print, mat[3,1]           ; access elements of matrix
       9
```

column row

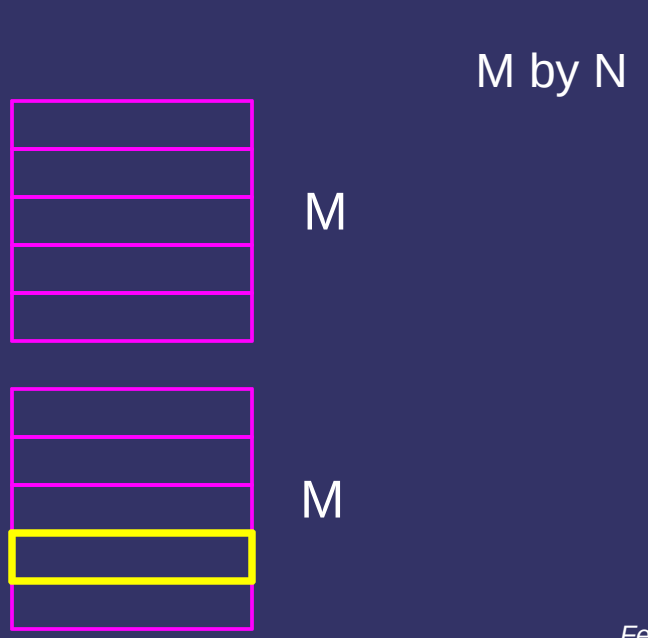
Matrices in IDL

```
IDL> print, mat[3,1]     ; access elements of matrix
       9
```

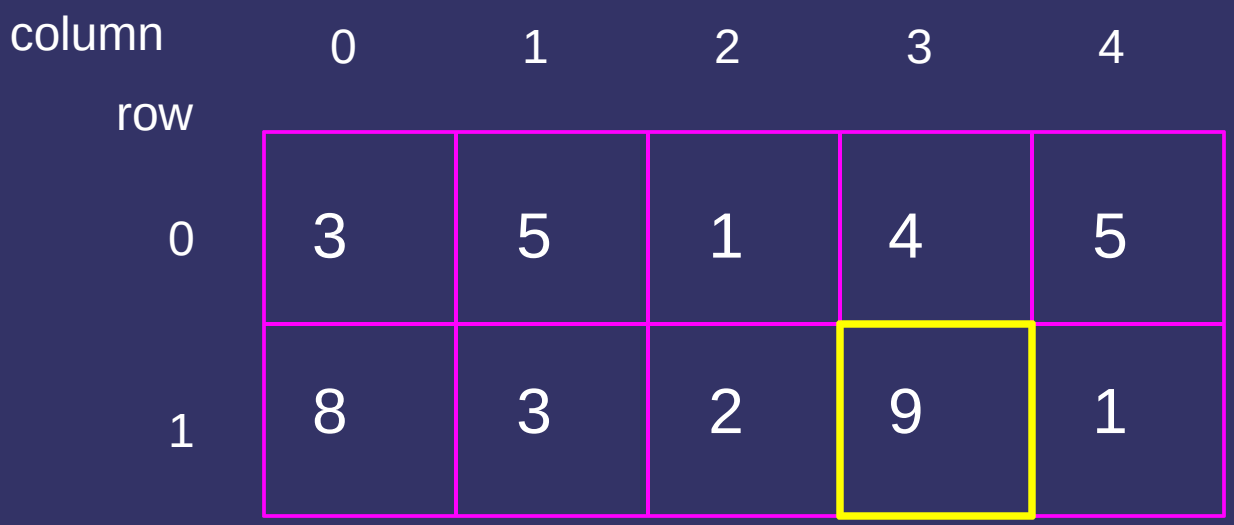
column row

```
IDL> print, mat[5*1+3]   ; equivalent...
       9
```

column row format...



Matrices in IDL



M by N

5 by 2

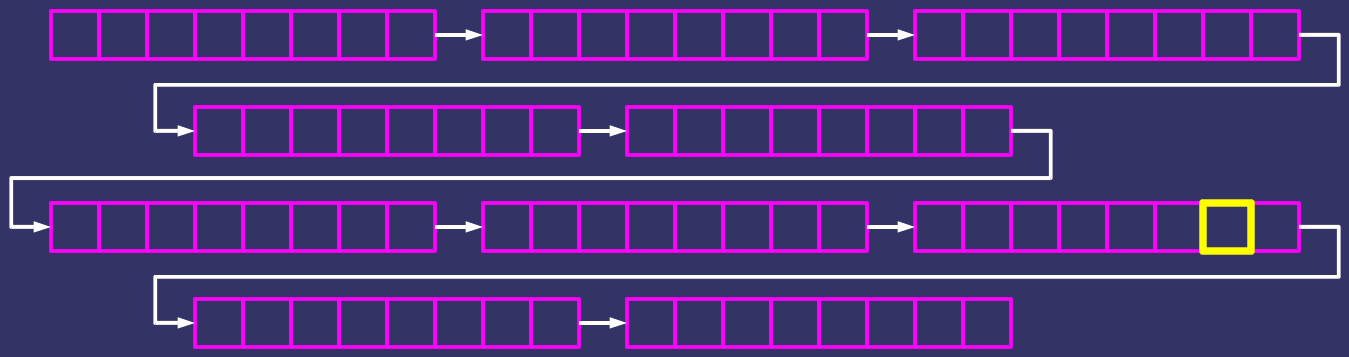
```
; nth row:    n*M     mth column: m
               1*5       +          3     =   8
```

```
mat[3,1]   =   mat[5*1+3]
```

Declaring Matrices

```
IDL> mat = [[3,5,1,4,5],[8,3,2,9,1]] ; explicitly
IDL> mat = fltarr(5,2)                ; all zeros
IDL> arr = dblarr(8,5,2)              ; other types
```

P by M by N array



```
arr[6,2,1] = arr[n*P*M + m*P + p]
                 40    + 16   + 6
```

Filling Up Arrays

```
IDL> darr = dblarr(8,5,2)                    ; all zeros
IDL> iarr = indgen(8,5,2)                    ; sequential
IDL> print, iarr
       0       1       2       3       4       5       6       7
       8       9      10      11      12      13      14      15
      16      17      18      19      20      21      22      23
      24      25      26      27      28      29      30      31
      32      33      34      35      36      37      38      39

      40      41      42      43      44      45      46      47
      48      49      50      51      52      53      54      55
      56      57      58      59      60      61      62      63
      64      65      66      67      68      69      70      71
      72      73      74      75      76      77      78      79
```

also: `bindgen()`, `lindgen()`, `findgen()`, `dindgen()`, `cindgen()`, `dcindgen()`, `uindgen()`, `ulindgen()`, `sindgen()`

Accessing Array Elements

```
IDL> mat = [[3,5,1,4,5],[8,3,2,9,1]] ; 5 by 2

IDL> print, mat[3,*]                 ; print column m=3
       4
       9

IDL> print, mat[2:4,0]               ; print columns 2-4
       1       4       5             ; of row 0


IDL> q = [3,2,0]                     ; another intarr

IDL> print, mat[q,0]                 ; index with another
       4       1       3             ; array or indices...
```

Searching Arrays

```
IDL> mat = [[3,5,1,4,5],[8,3,2,9,1]] ; 5 by 2

IDL> q = where(mat le 2, nq)  ; find all elements
                              ; in mat that are <= 2

IDL> print, q                 ; indices of the array
           2           7           9

IDL> print, mat[q]            ; values of the array
      1       2       1

IDL> print, nq                ; optional, but good
           3                  ; to check...
```

also: `eq`, `ge`, `gt`, `lt`

Array Size

```
IDL> mat = [[3,5,1,4,5],[8,3,2,9,1]] ; 5 by 2

IDL> print, n_elements(mat[0,*])    ; vector
           2

IDL> print, n_elements(mat)    ; total number for
          10                   ; multi-dimensional

IDL> print, size(mat)                ; more information
        2         5         2         2        10
```

dimensions — 1st — 2nd — data type — total elements

elements in *x* dimension

Array Calculations

```
IDL> n = 100                          ; declare num

IDL> x = findgen(n+1)*2*!pi/n         ; x = [0,2π] in
                                      ; steps of 2π/100

IDL> y = cos(x)                       ; element-by-element
                                      ; assignment

IDL> y = print, y[0:5]
      1.00000     0.998027     0.992115     0.982287     0.968583     0.951057
```

No looping necessary
cleaner looking code
faster execution

Note: `!pi` is a system variable

Also: system structures such as `!p`, `!x`, `!y`, etc.

e.g. `!p.multi = [0,ncol,nrow,nz,order]`

IDL

Array Calculations

Using C

```
#define N 101
int n;
float pi=3.1415926;
float x[N], y[N];

dx = 2*pi/N;
for(n=0;n<N;n++) {
    x[n] = n*dx;
    y[n] = cos(x[n]);
}
```

Array Calculations

Using C

```
#define N 101
int n;
float pi=3.1415926;
float x[N], y[N];

dx = 2*pi/N;
for(n=0;n<N;n++) {
    x[n] = n*dx;
    y[n] = cos(x[n]);
}
```

Using IDL (looping)

```
n = 101
dx = 2*!pi/n
for i=0,n-1 do begin
    x[n] = n*dx
    y[n] = cos(x[n])
endfor
end
```

Array Calculations

Using C

```
#define N 101
int n;
float pi=3.1415926;
float x[N], y[N];

dx = 2*pi/N;
for(n=0;n<N;n++) {
    x[n] = n*dx;
    y[n] = cos(x[n]);
}
```

Using IDL (looping)

```
n = 101
dx = 2*!pi/n
for i=0,n-1 do begin
    x[n] = n*dx
    y[n] = cos(x[n])
endfor
end
```

Using IDL (array)

```
n = 101
x = findgen(n)*2*!pi/(n-1)
y = cos(x)
```

Array Calculations

n times slower than C

Using C

```
#define N 101
int n;
float pi=3.1415926;
float x[N], y[N];

dx = 2*pi/N;
for(n=0;n<N;n++) {
    x[n] = n*dx;
    y[n] = cos(x[n]);
}
```

Matlab: 23 looping

21 array

Using IDL (looping) 58

```
n = 101
dx = 2*!pi/n
for i=0,n-1 do begin
    x[n] = n*dx
    y[n] = cos(x[n])
endfor
end
```

Using IDL (array) 8

```
n = 101
x = findgen(n)*2*!pi/(n-1)
y = cos(x)
```

Executive Commands

(or Dot Commands)

file loop.pro

```
n = 101
dx = 2*!pi/n
for i=0,n-1 do begin
   x[n] = n*dx
   y[n] = cos(x[n])
endfor
end
```

to execute commands in loop.pro

```
IDL> .run loop.pro
IDL>
```

Other executive commands: (can only be used at the IDL command line)

.compile, .edit,

.run, .go, .continue, .out, .return, .run, .skip, .step, .stepover, .trace

Reading/Writing Files in IDL

Read in data from file:

```
openr, readf
```

Interactively with the DE

Write data to a file:

```
openw, writef
```

Reading Data from a File

```
IDL> filename = 'ssn.dat'                          ; declare filename
IDL> nlines   = 3072                               ; cheat...
IDL> record = {time:0.0, ssn:0.0}                  ; declare struct
IDL> data   = replicate(record, nlines)            ; make array of structs
IDL> openr, ifile, filename, /get_lun              ; open file for read
IDL> readf, ifile, data                            ; read block of data
IDL> free_lun, ifile                               ; clean up...
IDL> print, data[0:9].time                         ; access range of data
     1749.05      1749.13      1749.21      1749.29      1749.38      1749.46
     1749.54      1749.63      1749.71      1749.80
```

Note: must know number of lines ...

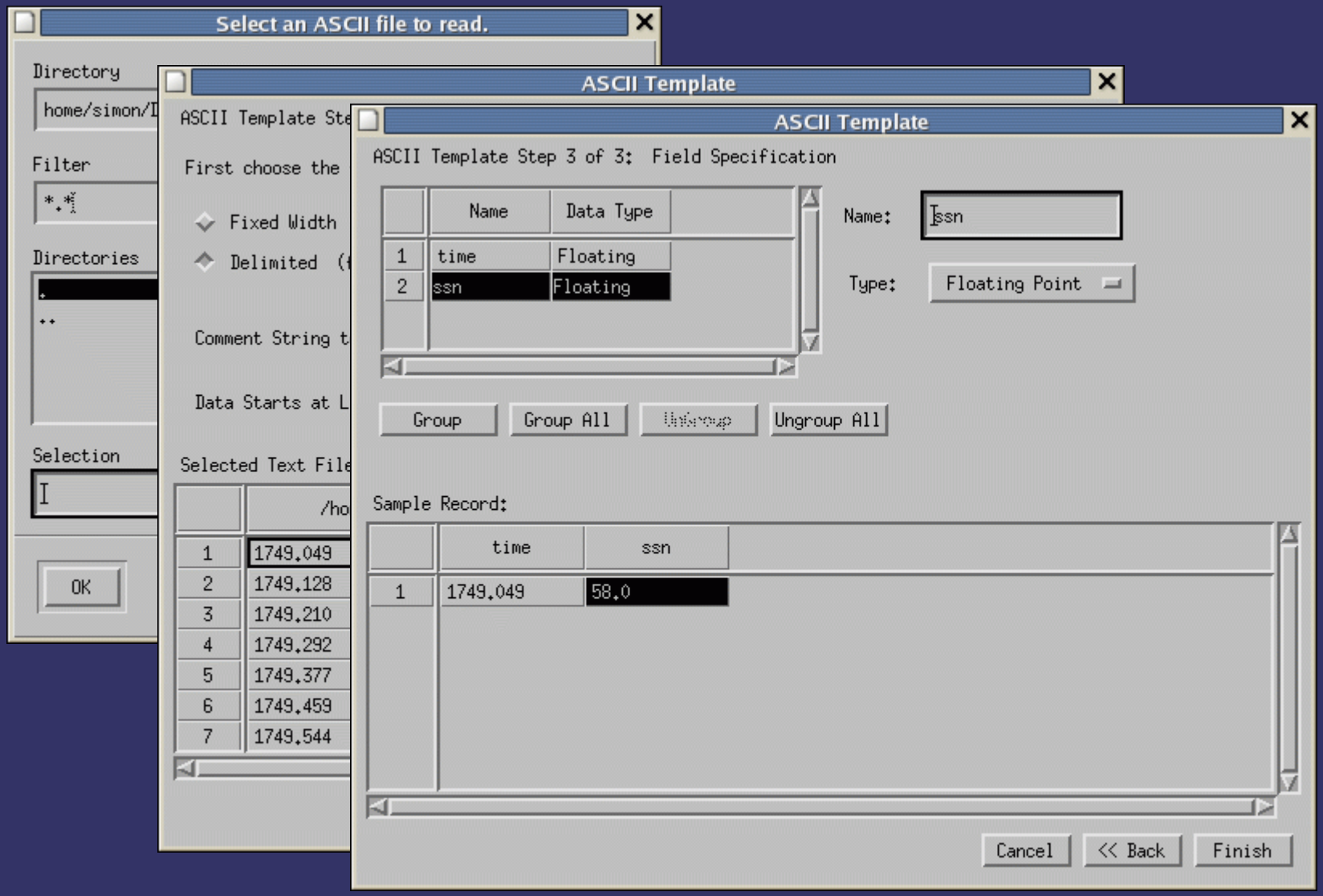
can do with

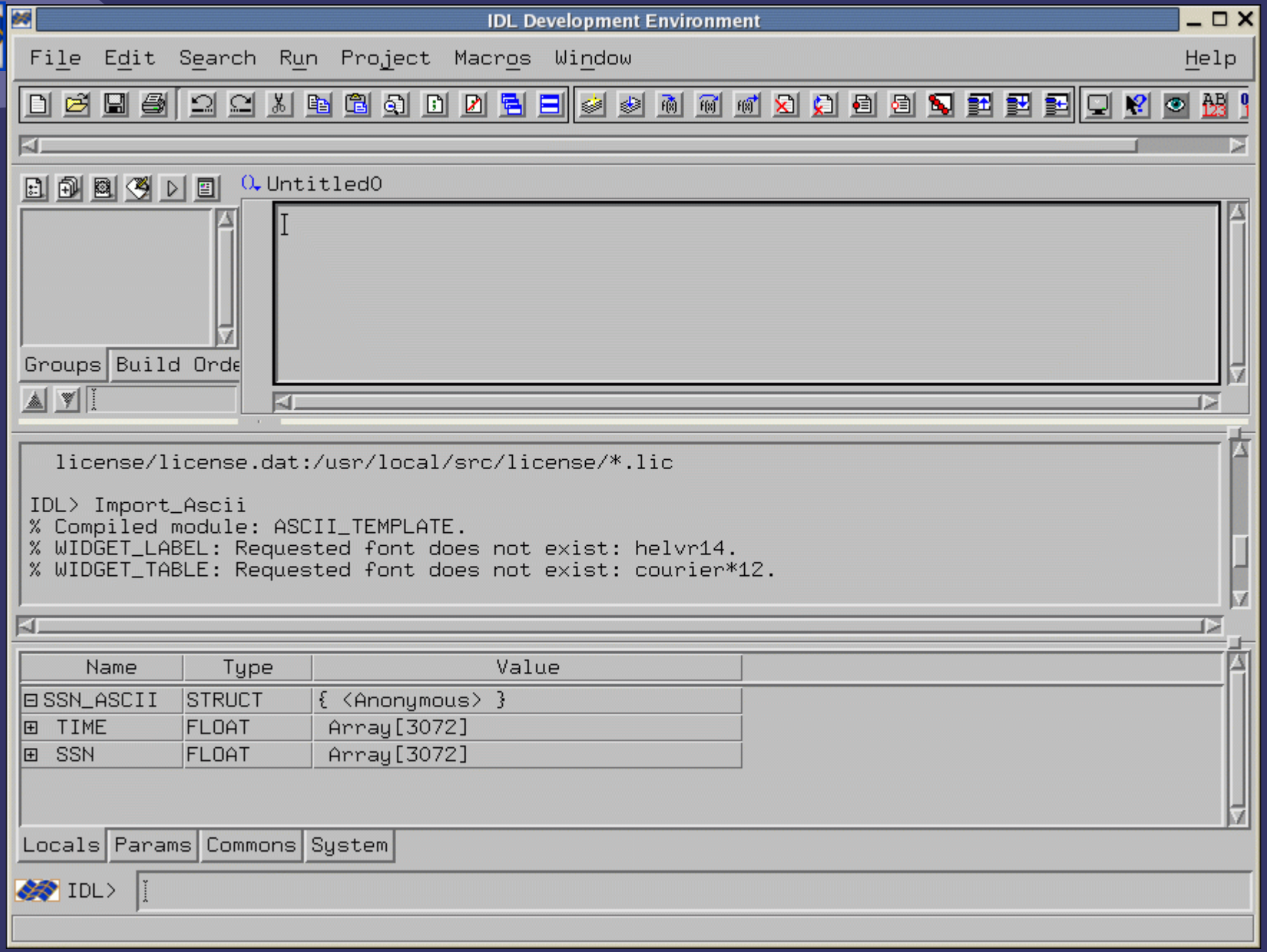
```
spawn, 'wc -l '+filename
```

or by looping

```
while not eof(ifile) do begin
readf, ifile, record
       data[i++] = record
endwhile
```

Reading Data from a File



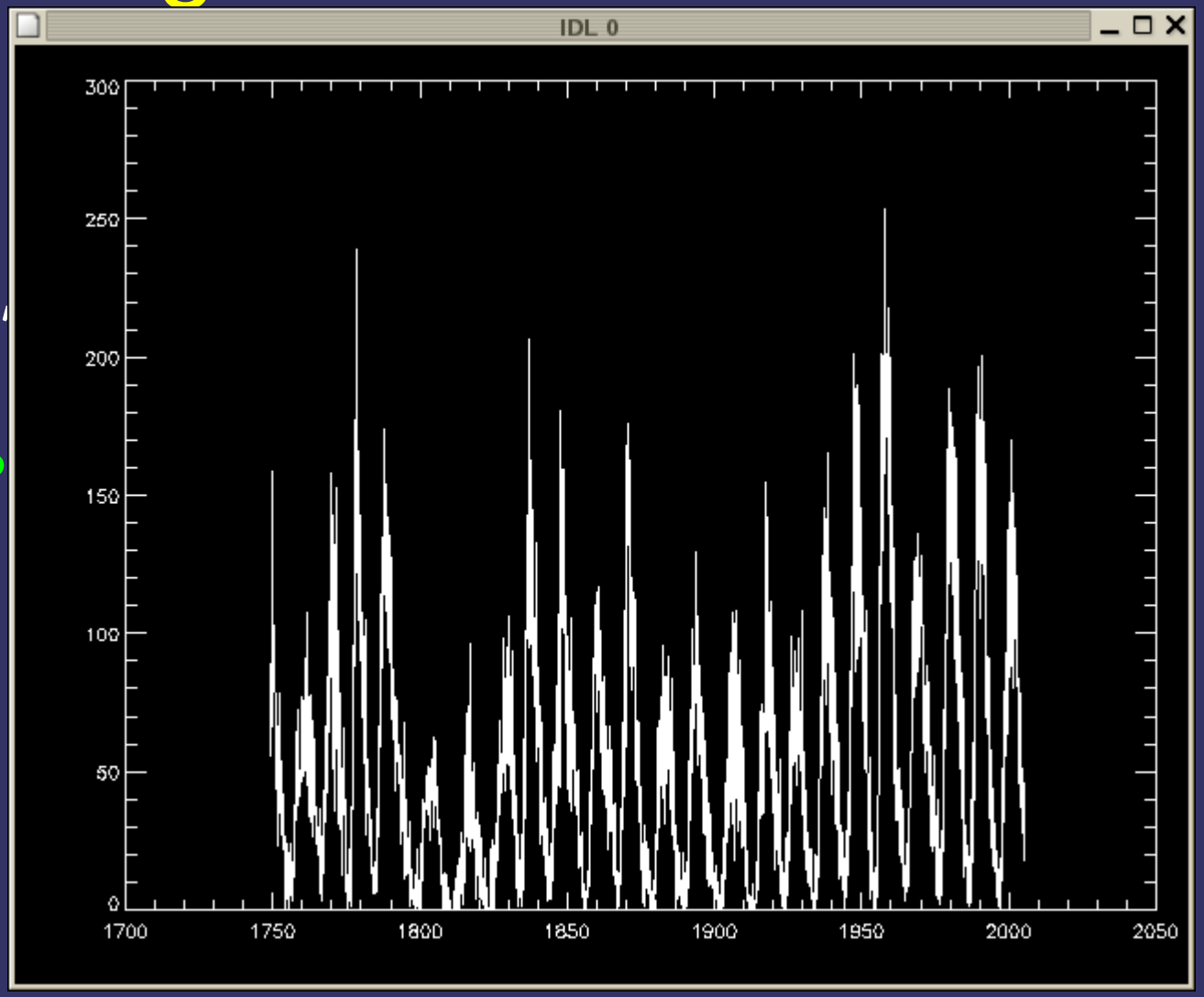

IDL Development Environment
File Edit Search Run Project Macros Window Help
Untitled0
Groups Build Orde
license/license.dat:/usr/local/src/license/*.lic
IDL> Import_Ascii
% Compiled module: ASCII_TEMPLATE.
% WIDGET_LABEL: Requested font does not exist: helvr14.
% WIDGET_TABLE: Requested font does not exist: courier*12.
Name Type Value
SSN_ASCII STRUCT { <Anonymous> }
TIME FLOAT Array[3072]
SSN FLOAT Array[3072]
Locals Params Commons System
IDL>

Plotting Features in IDL

IDL has

```
plot,
```

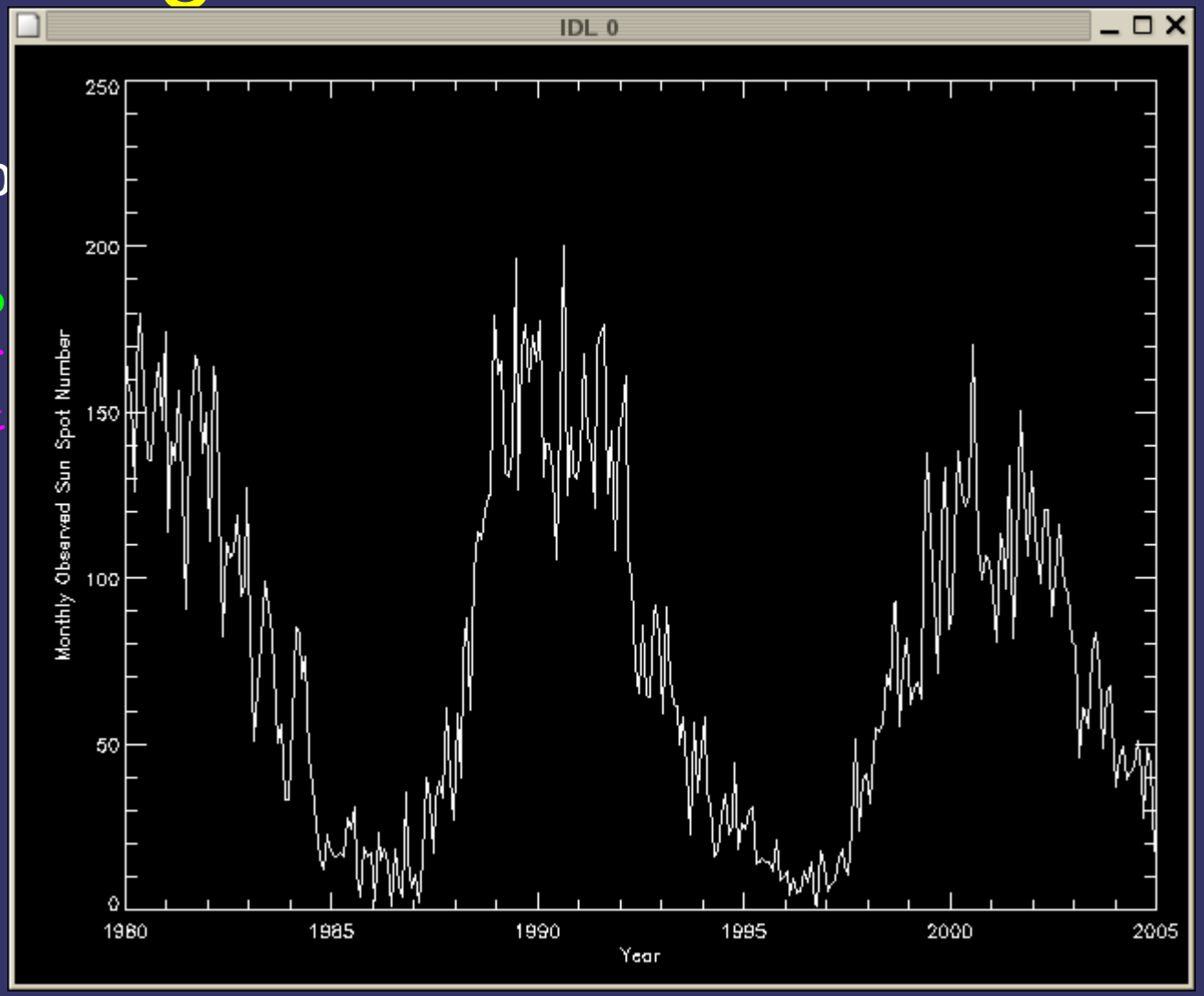
```
IDL> plo
```



Plotting Features in IDL

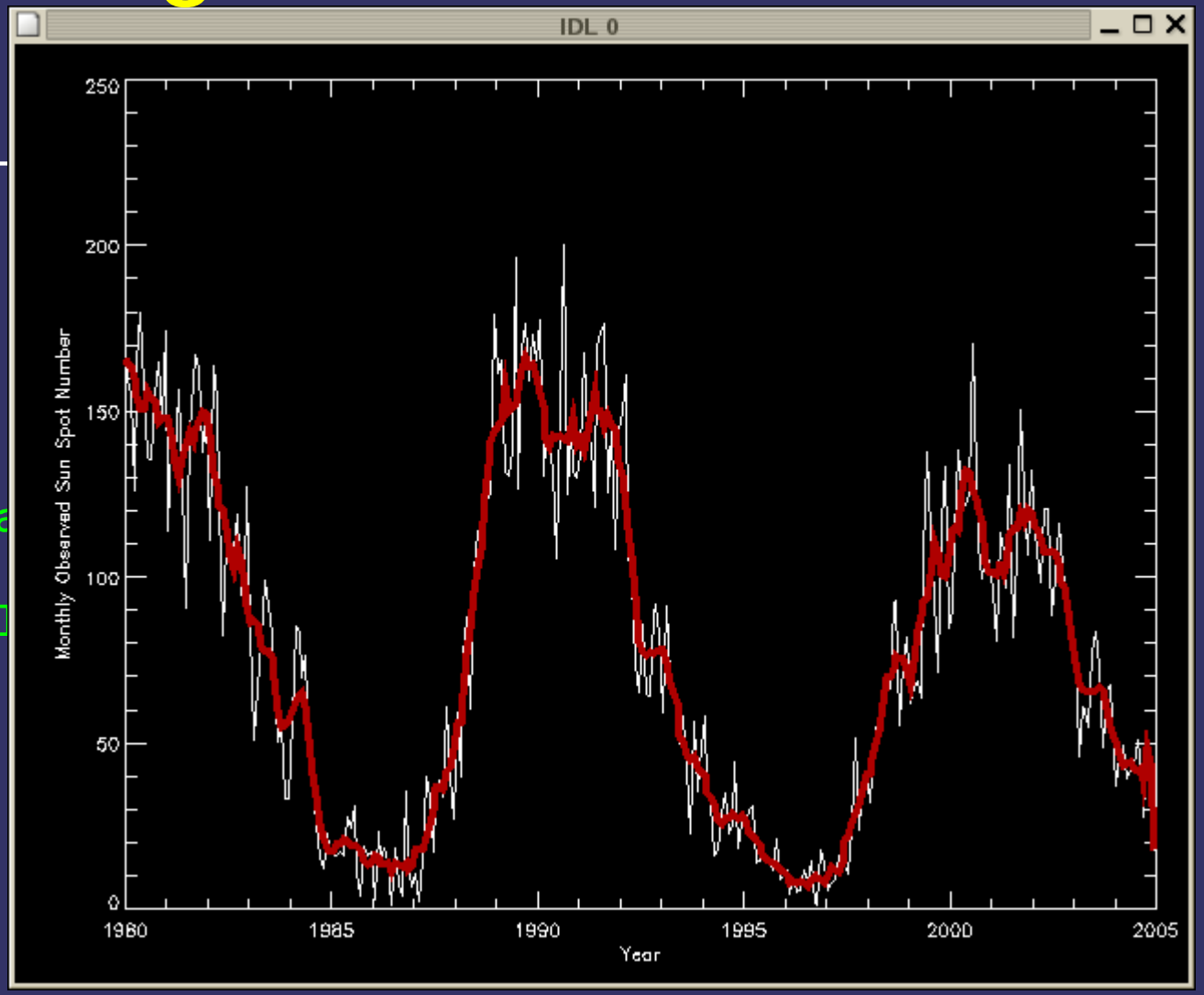
Many op

```
IDL> plo
       xr
       yt
```



Plotting Features in IDL

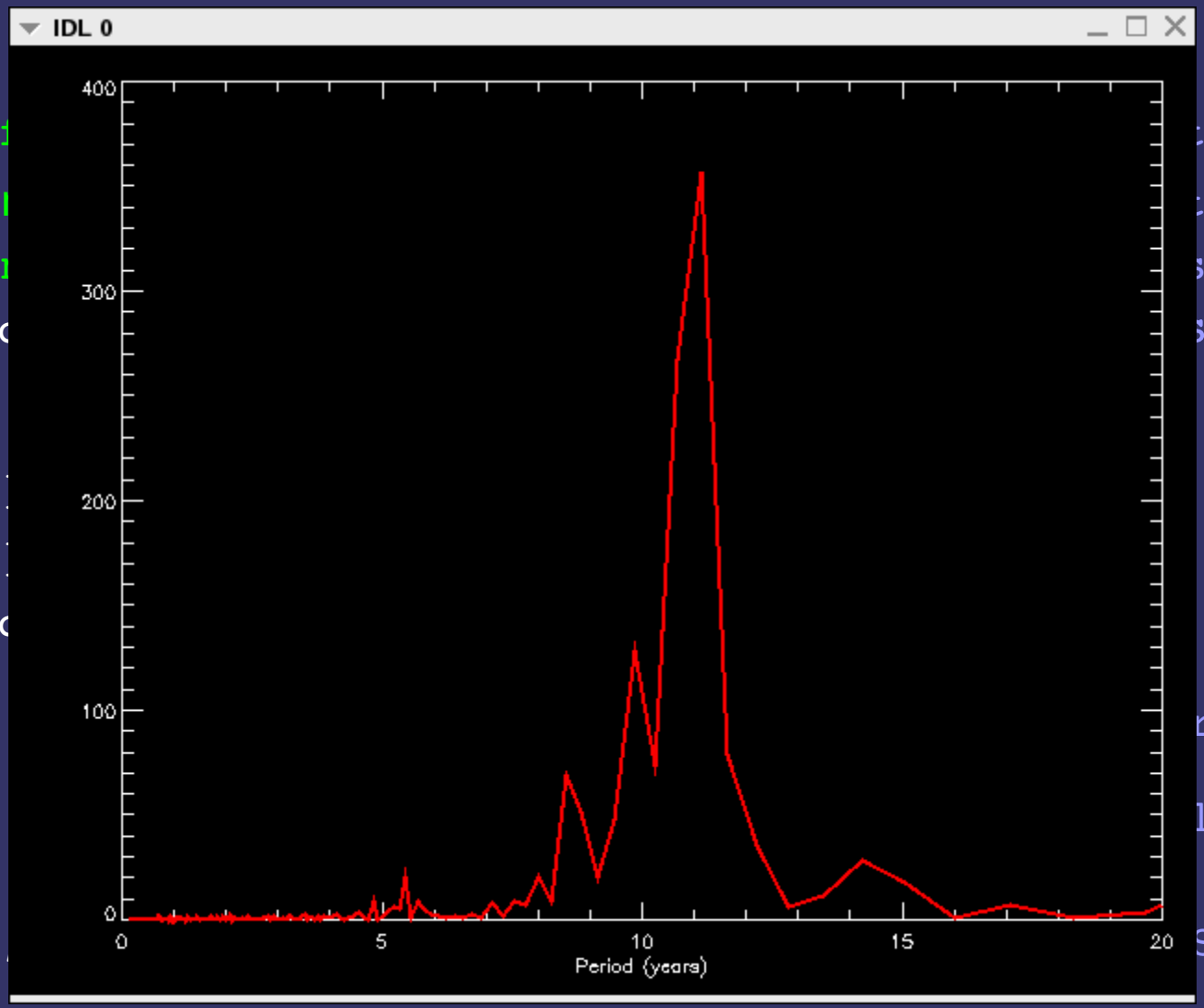
```
IDL> sm_
IDL> loa
IDL> opl
```



Spectral Analysis

Estimate the

```
IDL> H = fi
IDL> ndat = n
IDL> q = in
IDL> p = no
IDL> PSD =
IDL> PSD[0]
IDL> PSD[q]
IDL> PSD[no
IDL> per =
IDL> plot,
IDL> oplot,
```



IDL

Discrete Math Functions

Correlation Analysis

Curve and Surface Fitting

Differentiation and Integration

Eigenvalues and Eigenvectors

Gridding and Interpolation

Linear Systems including LAPACK Routines

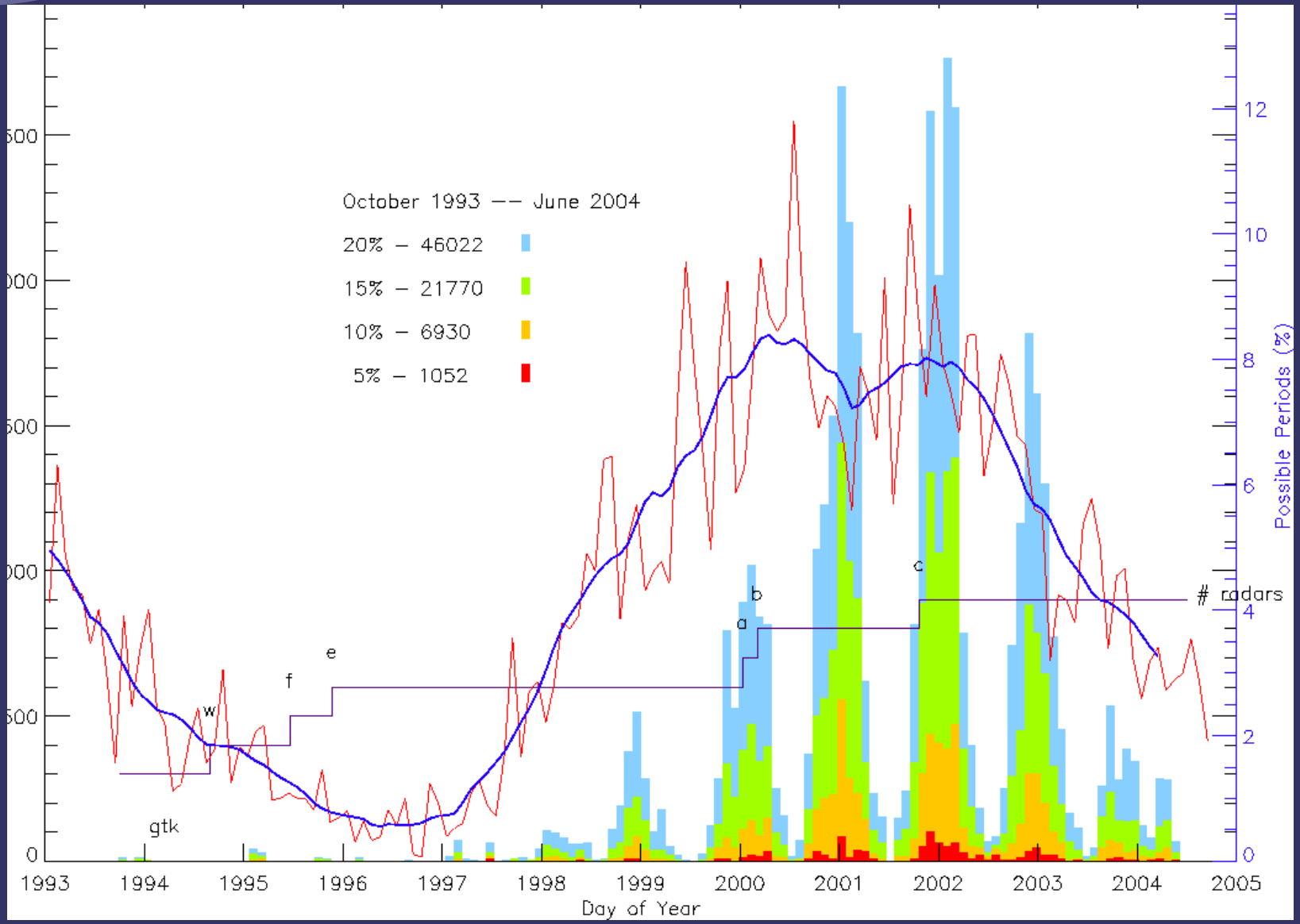
Multivariate Analysis

Nonlinear Equations

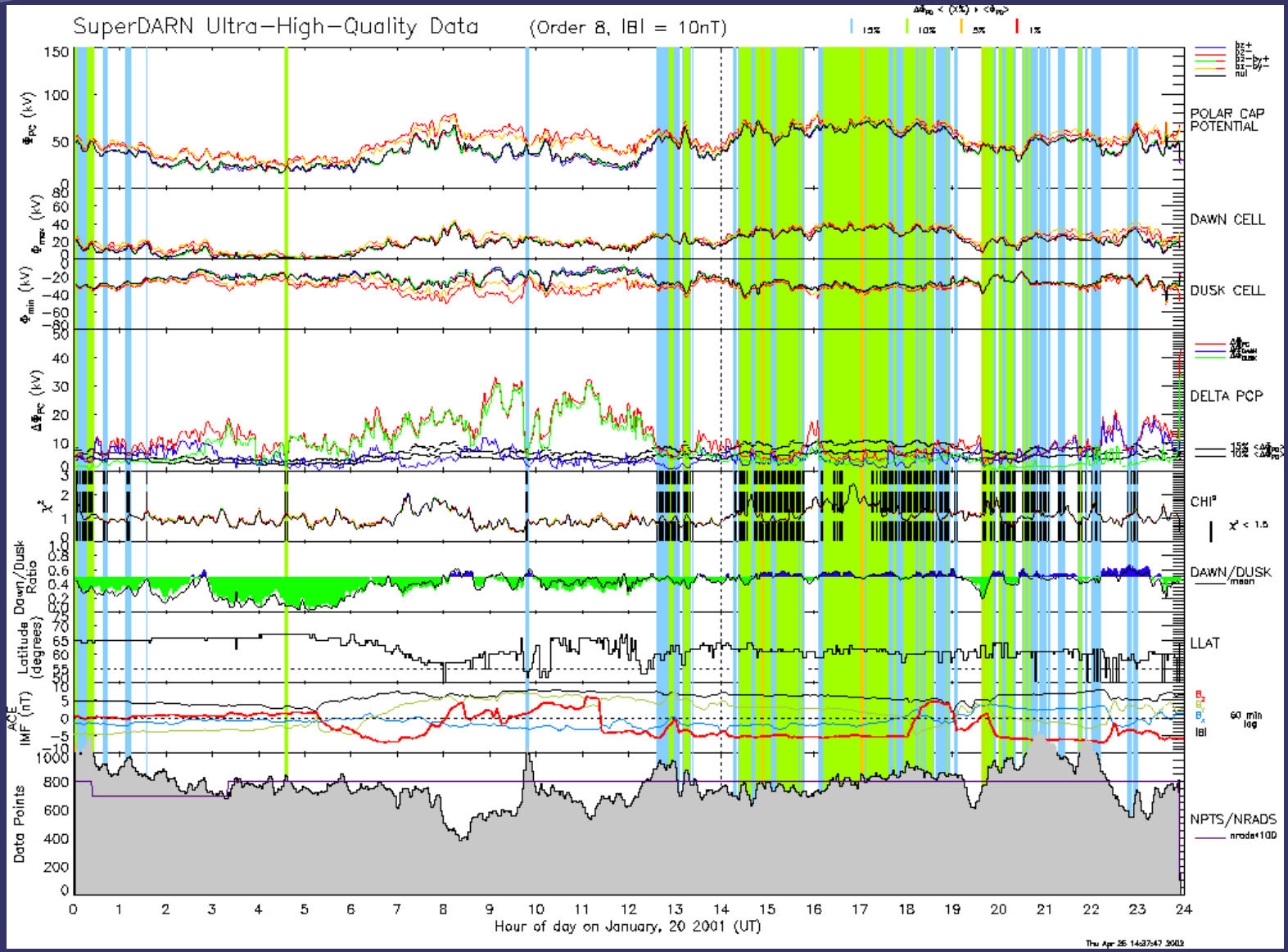
Optimization

Statistics and Probability

More Complicated Plots



More Complicated Plots



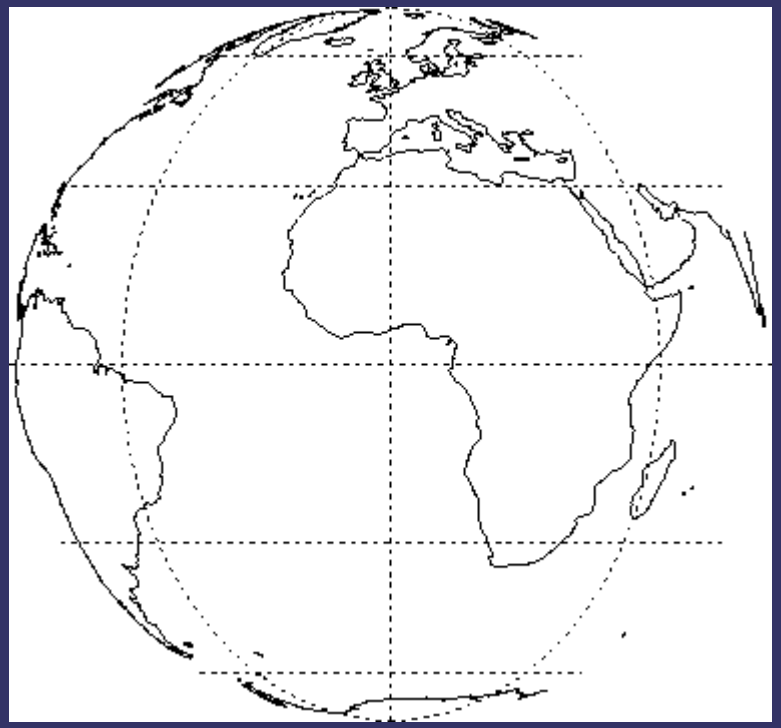
Mapping Features in IDL

IDL has built-in mapping capabilities

```
IDL> map_set, /orthographic,/grid, /continent,$
IDL>          /noborder
```

orthographic map projection

basic view

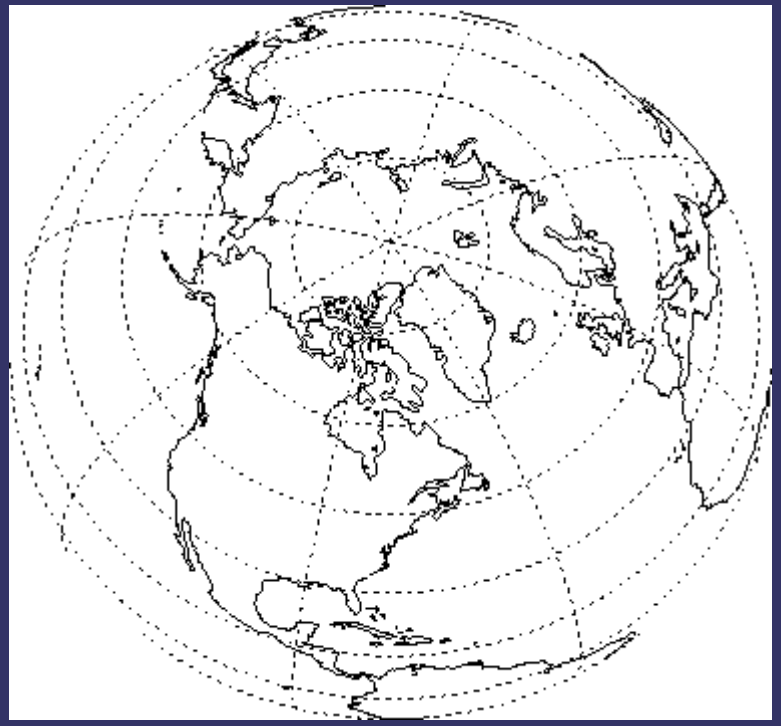


Mapping Features in IDL

```
IDL> map_set, 70, -75, /orthographic,/grid, $
IDL>          /continent,/noborder
```

orthographic map projection

rotated view



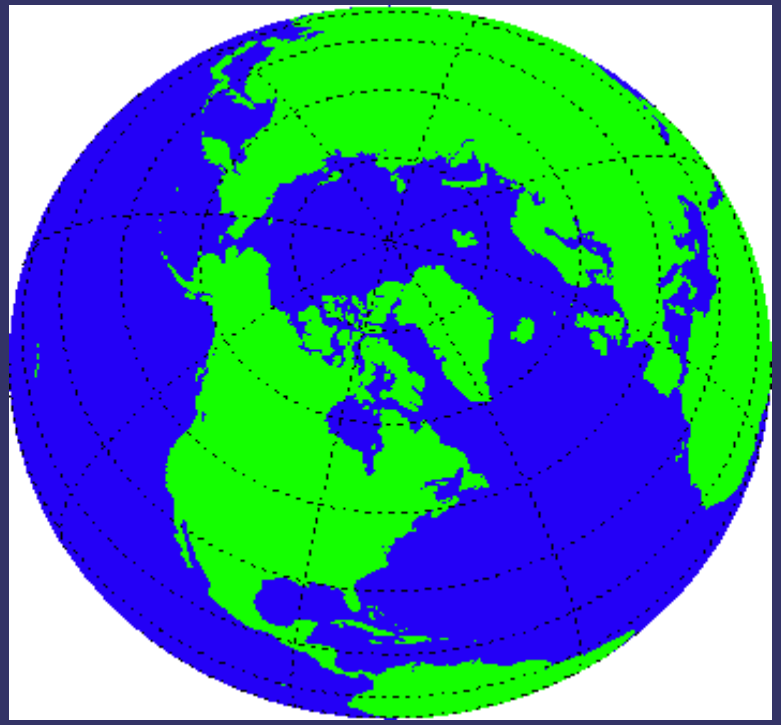
Mapping Features in IDL

```
IDL> map_set, 70, -75, /orthographic,/grid, $
IDL>          /continent,/noborder, $
IDL>          e_continents={fill:1, color:150}, $
IDL>          e_horizon={fill:1, color:50}
```

orthographic map projection

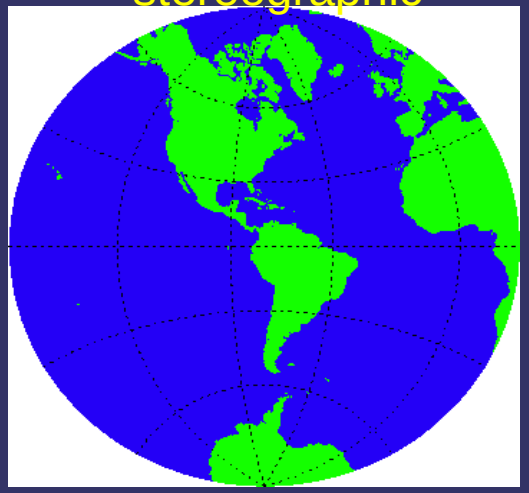
rotated view

colored

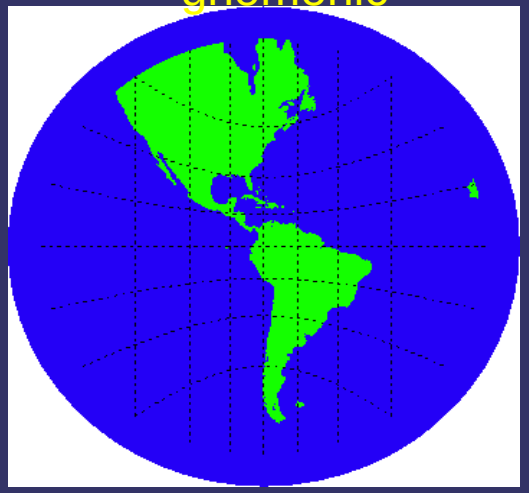


Map Projections in IDL

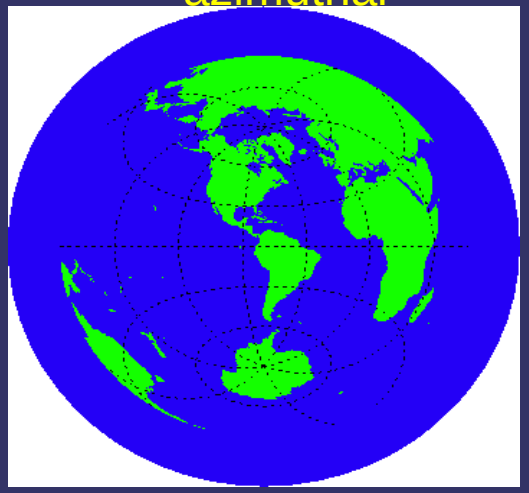
stereographic



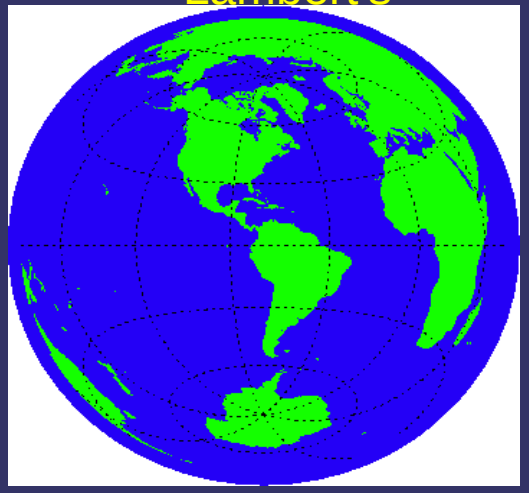
gnomonic



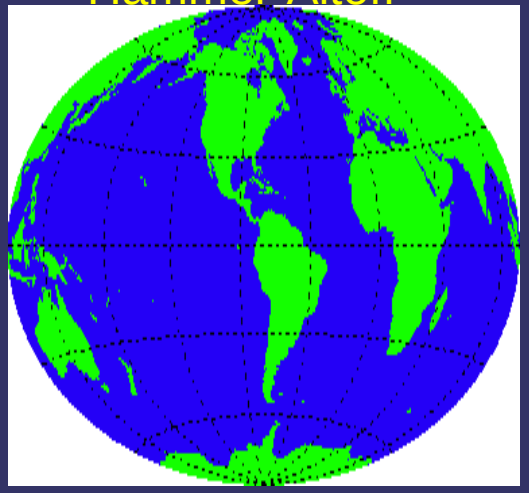
azimuthal



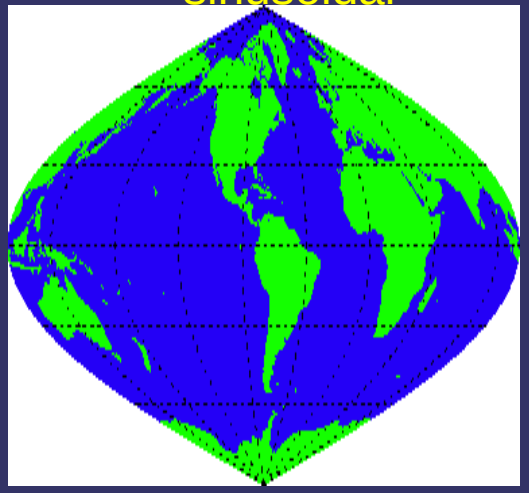
Lambert's



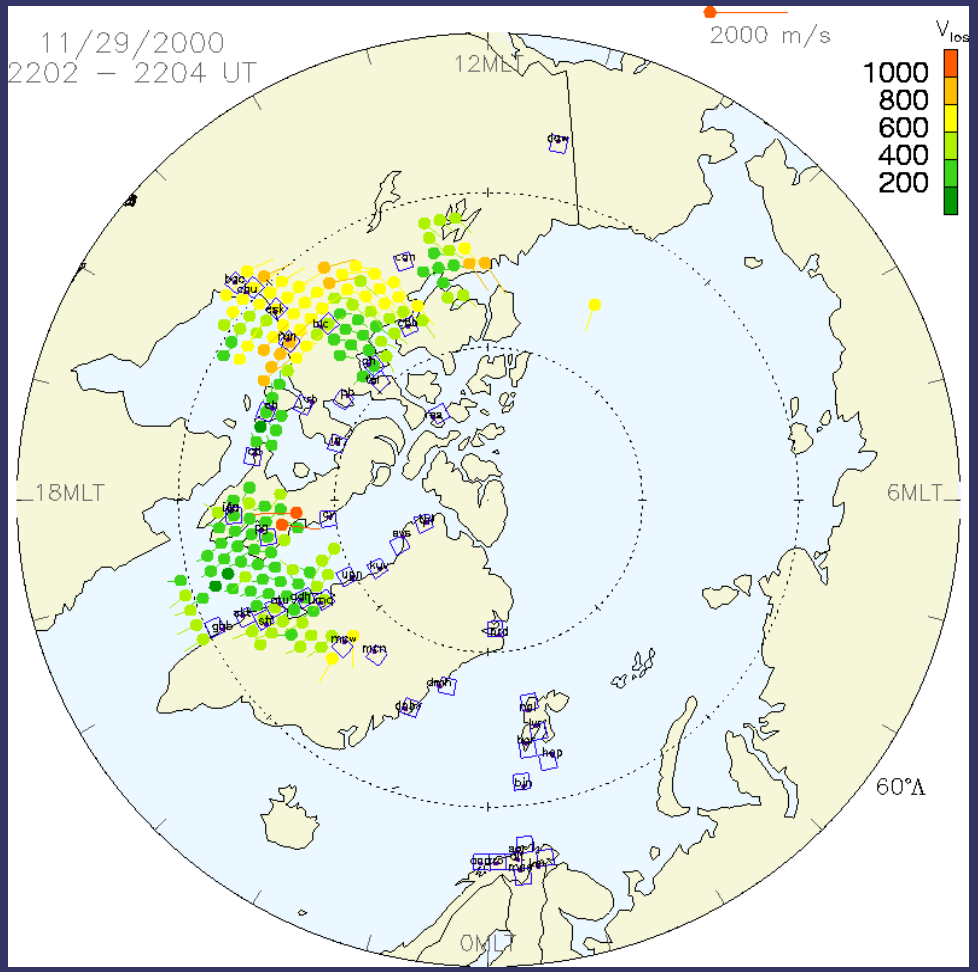
Hammer-Aitoff



sinusoidal



Data on Maps


11/29/2000
2202 – 2204 UT
2000 m/s
V_{los}
1000
800
600
400
200
12MLT
18MLT
6MLT
0MLT
60°Λ

Further Topics to Explore

Writing IDL Programs

Calling external functions (C and FORTRAN) from IDL

IDL Widgets

Plotting 2D and 3D

Object Graphics

Debugging

Data Structures

How to Get IDL Help

Online:

```
prompt> idlhelp
```

Help menu in the DE

Internet:

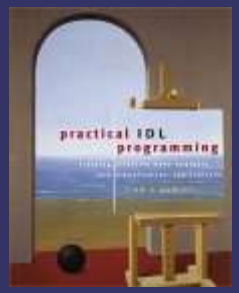
http://www.dfanning.com/documents/tips.html

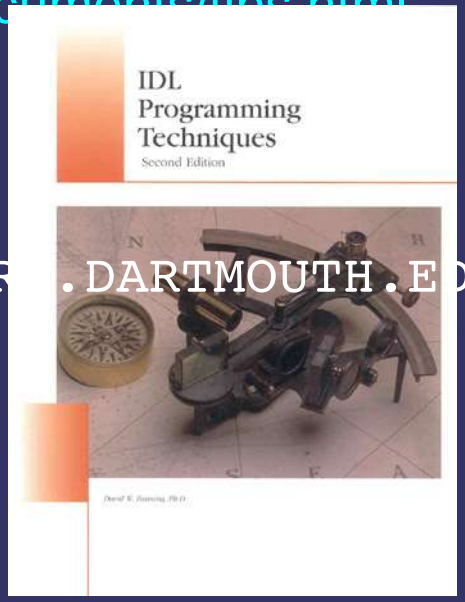
Listserve:

IDL users at Dartmouth

IDL-USERS@LISTSERV.DARTMOUTH.EDU

Books:





IDL

Where is IDL at Dartmouth?

Download onto your own computer: ~25 licenses