



**HAL**  
open science

## Odyssée Version 1.6 The User's Reference Manual

Christèle Faure, Yves Papegay

► **To cite this version:**

Christèle Faure, Yves Papegay. Odyssée Version 1.6 The User's Reference Manual. [Technical Report] RT-0211, INRIA. 1997, pp.50. inria-00069960

**HAL Id: inria-00069960**

**<https://inria.hal.science/inria-00069960>**

Submitted on 19 May 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

*Odyssée Version 1.6*  
***The User's Reference Manual***

Christèle Faure — Yves Papegay

**N° 0211**

Novembre 1997

THÈME 2



*rapport  
technique*





## Odyssée Version 1.6 The User's Reference Manual

Christèle Faure\* , Yves Papegay†

Thème 2 — Génie logiciel  
et calcul symbolique  
Projet SAFIR

Rapport technique n° 0211 — Novembre 1997 — 50 pages

**Abstract:** *Odyssée* is an *automatic differentiation* software performing transformation of FORTRAN 77 codes which implements both the forward and reverse mode of automatic differentiation. The current version of *Odyssée* is version 1.6. It includes the *Odyssée* engine, an interpreter of the command language and a MOTIF-like graphical user interface.

This document introduces the main functionalities of *Odyssée* through an illustrative example. It presents the basic concepts and objects of *Odyssée*, and completely describes the command language of *Odyssée* and its graphical user interface.

**Key-words:** Odysee, automatic differentiation, fortran, program transformation

\* Email : [Christele.Faure@sophia.inria.fr](mailto:Christele.Faure@sophia.inria.fr), URL : <http://www.inria.fr/safir/WHOSWHO/Christele.Faure>

† Email : [Yves.Papegay@sophia.inria.fr](mailto:Yves.Papegay@sophia.inria.fr), URL : <http://www.inria.fr/safir/Yves.Papegay>

Unité de recherche INRIA Sophia Antipolis

2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)

Téléphone : 04 93 65 77 77 - International : +33 4 93 65 77 77 — Fax : 04 93 65 77 65 - International : +33 4 93 65 77 65  
à partir du 01/01/1998

Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

# *Odyssée* Version 1.6

## Manuel de l'utilisateur

**Résumé :** *Odyssée* est un logiciel de différentiation automatique de programmes FORTRAN 77 qui implémente à la fois les modes direct et inverse de différentiation automatique. La version courante d'*Odyssée* est la version 1.6. Elle est composée d'une boîte à outils, d'un interprète du langage de commande qui permet d'accéder à ces outils et d'une interface graphique à ce langage.

Ce rapport présente les principales fonctionnalités d'*Odyssée* ainsi que les concepts et les objets de base. De plus, il documente complètement le langage de commande d'*Odyssée* et son interface graphique.

**Mots-clés :** *Odyssée*, différentiation automatique, fortran, transformation de programme

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>The Odyssée Concepts</b>	<b>9</b>
2.1	Odyssée and the Fortran 77 Code . . . . .	9
2.2	Odyssée and the Dependencies . . . . .	9
2.3	Odyssée and the File System . . . . .	12
<b>3</b>	<b>The Odyssée Command Language</b>	<b>15</b>
3.1	The Internal Library Related Commands . . . . .	17
3.2	The Information Base Related Commands . . . . .	22
3.3	The Differentiation Related Commands . . . . .	25
3.4	The Units Related Commands . . . . .	27
3.5	The System Related Commands . . . . .	28
3.6	The Odyssée Global Variables . . . . .	29
<b>4</b>	<b>The Odyssée Graphical User Interface</b>	<b>33</b>
4.1	The File Menu . . . . .	35
4.2	The Display Area . . . . .	38
4.3	The Differentiation Menu . . . . .	39
4.4	The Tools Menu . . . . .	40
4.5	The Options Menu . . . . .	46
4.6	Other Items . . . . .	48



# Chapter 1

## Introduction

### What is Odyssée?

Odyssée is an *automatic differentiation* software performing transformations of FORTRAN 77 codes:

Given a set of FORTRAN 77 units which compute a numerical function  $f$ , and a list of input variables, it produces FORTRAN 77 subroutines which compute the derivatives of  $f$  with respect to those variables.

Compared to other automatic differentiation tools<sup>1</sup>, the key features of Odyssée are:

- to implement both the forward and reverse mode of automatic differentiation,
- to be able to deal with *black-box* units, i.e. subroutines or functions whose code is not available.

In forward mode, Odyssée generates an augmented FORTRAN 77 code which computes the function and one directional derivative (tangent line) of it. In reverse mode, it generates a FORTRAN 77 code for the computation of a gradient (cotangent line).

Odyssée analyzes the FORTRAN 77 code to detect what are all the *active variables*, those whose values depend on the variables with respect to which the program is differentiated. But Odyssée may also treat calls of subroutines or functions whose code is not available, called *black-box* units. For that purpose, it uses a data base which contains, for each unit, the information on dependencies between the output and input variables of this unit.

---

<sup>1</sup>For an introduction to the automatic differentiation and a survey of the existing tools and their functionalities, visit the web page <http://www.mcs.anl.gov/Projects/autodiff/index.html> of the Computational Differentiation Project at Argonne National Laboratory



## Miscellanies

The development of *Odyssée* began in 1991 at the Sophia Antipolis research center of INRIA by members of the SAFIR team. The SAFIR team is a joint team of INRIA<sup>2</sup>, CNRS<sup>3</sup> and UNSA<sup>4</sup>.

The current version of *Odyssée* is version 1.6. It includes the *Odyssée* engine, an interpreter of the command language and a MOTIF-like graphical user interface. The engine and the interpreter of *Odyssée* have been written in the Objective-CAML<sup>5</sup> language, its interface has been written in the TCL-TK language.

Only the binary versions are distributed outside INRIA and UNSA. It runs on several UNIX platforms. It is available on request without charge for educational and non-profit research and for internal evaluation. For this purpose, or for any other information, the development team of *Odyssée* can be contacted by e-mail at [odyssee@sophia.inria.fr](mailto:odyssee@sophia.inria.fr).

This document describes completely the command language of *Odyssée* and the menus of its graphical user interface. Other information can be found at the *Odyssée* www site at the URL <http://www.inria.fr/safir/SAM/Odyssee/odyssee.html>.

## Getting Started

When *Odyssée* has been successfully installed, typing `odyssee` or `xodyssee` are the two ways to run it.

The command `odyssee` starts the interpreter of the *Odyssée* command language<sup>6</sup>. After the banner, you will get a special prompt `ODYTOP>`: *Odyssée* is waiting for input.

```

      ODYSSEE
      copyright INRIA - UNSA
      Version 1.6 - Fri Oct 25 11:28:42 MET DST 1996

      Sourcing .odysseerc...
      OK
      ODYTOP>
```

If you prefer to start the graphical user interface of *Odyssée*, use the command `xodyssee`<sup>7</sup> and wait until the *Odyssée* window appears like in figure 1.1.

In both cases, once *Odyssée* has been started, three steps are necessary to differentiate a FORTRAN 77 program:

<sup>2</sup>INRIA is the French national institute for research in computer science and control (<http://www.inria.fr/welcome-eng.html>)

<sup>3</sup>CNRS is the French national center for scientific research (<http://www.cnrs.fr/index.html>)

<sup>4</sup>UNSA is the university of Nice Sophia Antipolis (<http://www.unice.fr>)

<sup>5</sup><http://pauillac.inria.fr/ocaml/index.html>

<sup>6</sup>otherwise check if the shell variable `PATH` is correctly set.

<sup>7</sup>in case of problems, check that the shell variables `DISPLAY` and `ODYSSEE` are correctly set.

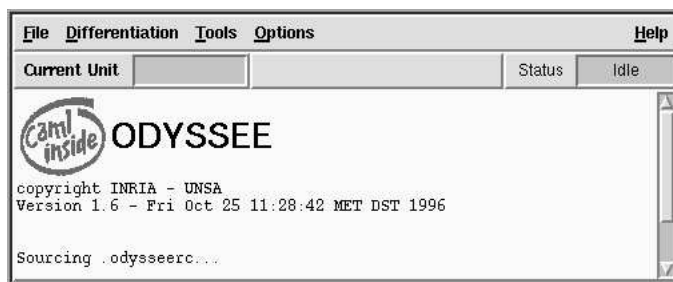


Figure 1.1: Odyssée initial window

1. to load the input FORTRAN 77 code into the *internal library*,
2. to select the *head unit* in the internal library, the differentiation method, and the *active input variables* – the variables with respect to which the input code will be differentiated,
3. to save the code generated by Odyssée in a file and/or to visualize it on the screen.

## A Sample FORTRAN 77 Code

In his document we will always consider the same example to illustrate how the commands and the menus are working. It is described below.

Let  $\phi$  be the numerical function of two real variables defined by:

$$\phi : (s, t) \longmapsto \frac{\sqrt{\exp(s^2) + \frac{\sin s}{s}} - \sqrt{\exp(t^2) - \frac{\sin t}{t}}}{1 + \sqrt{\exp(s^2) + \frac{\sin s}{s}} + \sqrt{\exp(t^2) - \frac{\sin t}{t}}}$$

It may be implemented by the FORTRAN 77 code shown in figure 1.2 page 8 which is a collection of six compilation units. To compute numerical values, this code should be linked to subroutines dedicated to the input of numerical data and to the output of the result. These routines are not presented here, as they are not useful for differentiation.

```
double precision function f (t)

double precision t

f = t*t
f = exp(f)
return
end

double precision function g (t)

double precision t

g = 1
if (t .ne. 0.d0) then
  g = sin(t)/t
endif
return
end

subroutine sub1 (x,y)

double precision x,y

y = f(x) + g(x)
y = sqrt(y)
return
end

subroutine sub2 (x,y)

double precision x,y

y = f(x) - g(x)
y = sqrt(y)
return
end

subroutine sub0 (u,v)

common zn,zd
double precision u,v,z1,z2,zn,zd

call sub1 (u,z1)
call sub2 (v,z2)
zn = z1 - z2
zd = 1 + z1 + z2
return
end

subroutine main (i1,i2,o)

common zn,zd
double precision i1,i2,o,zn,zd

call sub0 (i1,i2)
o = zn/zd
return
end
```

Figure 1.2: Example of fortran Code

## Chapter 2

# The Odyssée Concepts

### 2.1 Odyssée and the Fortran 77 Code

A **compilation unit** is a part of a file of FORTRAN 77 code that can be compiled separately. Therefore a compilation unit is one of the following FORTRAN 77 entities: a subroutine, a function, a program, or a block data. Odyssée differentiate only subroutines, or functions.

The **internal library** is the structure where Odyssée stores compilation units:

- the units loaded by Odyssée,
- the units generated by Odyssée (results of a differentiation process).

When a new session of Odyssée starts, the internal library is empty.

The **head unit** is the routine chosen by the user that computes the mathematical function to be differentiated. Usually, a mathematical function is represented as a collection of compilation units. In order to differentiate it, the user has to give Odyssée the name of the head unit.

### 2.2 Odyssée and the Dependencies

The **variables** of a compilation unit is the collection of all the variables appearing in this unit. It includes (see figure 2.1 page 11):

- formal parameters or dummy arguments : the arguments of subroutines and functions,
- global variables : parts of common blocks,

- local variables : other variables.

One must notice that *Odyssée* considers the function name as a variable, as it is attached to the output value of the corresponding function.

**The input and output variables** of a compilation unit are global or dummy variables.

- *input variables* are the non-local variables which should have a value attached to before the unit could be run,
- *output variables* are the non-local variables which are modified when the unit is run.

Note that some variables may be at the same time input and output variables (see figure 2.2 page 11).

**The dependencies** are defined between the input and the output variables of a compilation unit: an output variable depends on an input variable when it is necessary to know the value of the input variable to be able to compute the output variable. For each compilation unit, it is possible to define a function  $f$  from the set  $\mathcal{O}$  of its output variables to the set  $\mathcal{P}(\mathcal{I})$  of the subsets of the set  $\mathcal{I}$  of its input variables:

$$f : \begin{array}{l} \mathcal{O} \longrightarrow \mathcal{P}(\mathcal{I}) \\ o \longmapsto \{i_1, \dots, i_n\} \end{array}$$

such as  $i \in f(o)$  if and only if it is necessary to know the value of  $i$  to compute the value of  $o$ .

In fact, it is even possible to define two such functions, one concerning the *control-dependencies*, the other concerning the *value-dependencies*. The following example shows the difference between control-dependencies and values-dependencies:

In the FORTRAN 77 instruction

```
if (x.gt.0.) then
  z = y
end if
```

the variable  $z$  is control-dependent on the variable  $x$  and value-dependent on the variable  $y$ .

The only dependencies considered by *Odyssée* are the value-dependencies.

**The information base** is the structure where *Odyssée* stores, for each unit all the necessary information on the non-local variables and on the dependencies between them:

- its input dummy variables,
- its input global variables,

```

subroutine sub0 (u,v) ← formal parameters

common zn,zd ← global variables
double precision u,v,z1,z2,zn,zd

call sub1 (u,z1)
call sub2 (v,z2)
zn = z1 - z2
zd = 1 + z1 + z2
return
end
    
```

Figure 2.1: different variables

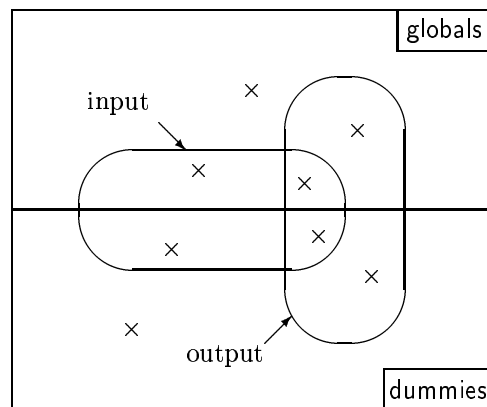


Figure 2.2: Input / output variables

- its output dummy variables,
- its output global variables,
- the dependencies between its output variables and its input variables.

The entry in this base for each unit is filled with default values when the unit is loaded and is really computed when the system performs the analyse of the code. These entries may also be changed by the user through an *information file* (see section 3.2).

**An active variable** is a variable that depends on the variables with respect to which the mathematical function must be differentiated (subset of input variables of the head unit).

**The active information base** is a restriction of the information base that concerns only active variables. The active information base is also attached to a diff computation. It does not exist before the first differentiation and is re-computed at any evaluation of a diff command. Only the information contained in the current active information base may be accessed.

## 2.3 Odyssée and the File System

Odyssée reads or writes information through several kinds of text files. Odyssée considers five different types for files: fortran, include, batch, ibasis and odyssee and a special file: .odysseerc.

Three global variables control each type of file T: T\_ext for the name extension, T\_dir for the directory where the files have to be written by Odyssée, and T\_path for a list of path from where Odyssée reads the files.

- The fortran files are the files containing FORTRAN 77 code read as input of a load command or generated by a diff computation and written by a getlibrary, a getunit, a getprogram or a getdiffprogram.
- The include files are the files included by the FORTRAN 77 code to be loaded. Note that they are in-lined in the code of the compilation units in the library.
- The batch files contain valid Odyssée commands and are executed through the load-batch command.
- The ibasis files contain Odyssée commands that modify the information base (setdummys, setglobals, setinout, setabstract). If an ibasis file of name  $\langle unit \rangle\_Bl.\langle extension \rangle$  where  $\langle extension \rangle$  is the value of the Odyssée global variable `ibasis_ext` exists, it is automatically loaded by Odyssée during a diff command applied to  $\langle unit \rangle$ .
- The odyssee files are the files where Odyssée writes information about the internal library like in the listunits, libgraph or getgraph command.

- The `.odysserc` file is a special batch-file automatically loaded at the beginning of any session of *Odyssée*. It can also be reloaded by the `loadodysserc` command. It is generally used to configure or customize *Odyssée*.





## Chapter 3

# The Odyssée Command Language

The Odyssée *character set* consists of the 52 lower case and upper case letters, the 10 digits and the special characters of the table 3.1.

\	backslash
␣	blank
-	dash
.	dot
"	double quotes
#	sharp
/	slash
_	underscore
(	left parenthesis
)	right parenthesis

Table 3.1: special characters

Tokens of the Odyssée command language are *strings*, *identifiers*, *lists* and *numbers*:

- A string is a sequence of any printable characters enclosed in double quotes.
- An identifier is a letter, a dot or a slash followed by zero or more letters, digits, dots, dashes, underscores or slashes. `true` and `false` are the identifiers corresponding to the two boolean constants. These constants, the command identifiers and the option names are reserved words of the language. They are listed in the table 3.2.
- A list is a sequence of tokens separated by white spaces, for clarity those tokens may be enclosed between parentheses.
- A number is a signed integer. On most computer architectures, it may have any value in the range  $-2^{30} \dots (2^{30} - 1)$ .

commands			others
clear	getsymbols	quit	-cl
clearaddedunits	getunit	remove	-cotangent
computeibasis	getvar	roundoffmain	-h
copyright	help	roundoffunit	-head
diff	libgraph	save	-o
exit	listunits	saveibasis	-opt
getabstract	load	setabstract	-optloops
getactiveinout	loadbatch	setdummys	-output
getaddedunits	loadibasis	setglobals	-tangent
getdependency	loadodysseerc	setinout	-tl
getdiffprogram	makeblock	setvar	-variables
getgraph	preprocess	shell	-vars
getinout	print	slice	
getlibrary	print2	stratlist	false
getprogram	printunit	version	true

Table 3.2: reserved words

An input of the *Odyssée* interpreter consists of a single statement usually referred to as a *command line*. However a line, i.e. a sequence of characters followed by a newline character, may contain an incomplete statement. And a statement may be continued from line to line by preceding the newline character by a backslash. In this case, both the backslash and the newline are ignored by *Odyssée*.

When the parser of *Odyssée* “sees” a sharp character anywhere on a line, any following character until the next newline is treated as a *comment*.

A command line is a sequence of tokens separated by white spaces – a *white space* is either a blank or a tabulation. The first token of a command line should be a valid *command name*. Each command has its own syntax described in one of the following sections.

Except in a few cases – namely when reading arguments of an help command, *Odyssée* does not distinguish between an identifier and a string.

## 3.1 The Internal Library Related Commands

### 3.1.1 Modifying the Internal Library

**clear**

Resets the internal library to the empty set.

**load**  $\langle file_1 \rangle$  [ $\langle file_2 \rangle$  ...  $\langle file_N \rangle$ ]

Loads the compilation units from the specified FORTRAN 77 file(s) in the internal library.

If the name  $\langle file_i \rangle$  has an extension, then Odyssée is looking for the file  $\langle file_i \rangle$ . Otherwise, Odyssée is looking for the file  $\langle file_i \rangle.\langle extension \rangle$  where  $\langle extension \rangle$  is the value of the Odyssée global variable `fortran_ext`.

load returns an error message if no file of the given name(s) are found in one of the directory listed in the Odyssée global variable `fortran_path`. Odyssée also returns an error message if any file found contains anything else but valid FORTRAN 77 compilation units.

Note that, if there is an include statement in the loaded code, it is necessary to have properly set the Odyssée global variable `include_path`.

**remove**  $\langle unit \rangle$ 

Removes the compilation unit of name  $\langle unit \rangle$  from the internal library.

remove returns an error message if  $\langle unit \rangle$  is not the name of a compilation unit of the internal library.

**clearaddedunits**

Resets the list of the added units to the empty list.

### 3.1.2 Displaying FORTRAN 77 Code

getlibrary, getunit, getprogram and getdiffprogram are used to view units from the internal library.

Those four commands may all be invoked with an optional  $\langle file \rangle$  parameter. If this parameter is present, their output is printed in a file, otherwise it is displayed on the standard input. If the name  $\langle file \rangle$  has an extension, then Odyssée is writing in the file of name  $\langle file \rangle$ . In the other case, Odyssée is writing in the file of name  $\langle file \rangle.\langle extension \rangle$  where  $\langle extension \rangle$  is the value of the Odyssée global variable `fortran_ext`. getlibrary, getunit, getprogram and getdiffprogram are writing FORTRAN 77 files in the directory defined by the Odyssée global variable `fortran_dir`. They do not produce any warning message if the file of name  $\langle file \rangle$  or of name  $\langle file \rangle.\langle extension \rangle$  already exists in this directory.

The format of the output of these four commands may be tuned by resetting Odyssée global variables like `all_uppercase`, `do_indentation`, `if_indentation`, `continuation_char` or `blank_line_after_statement`, see section 3.6 for further details.

**getlibrary** [*file*]

Displays all the compilation units from the internal library.

```

ODYTOP> load exemple
ODYTOP> getlibrary
      SUBROUTINE main (i1, i2, o)

      COMMON // zn
      COMMON // zd
      DOUBLE PRECISION i1, i2, o, zn, zd

      CALL sub0(i1, 12)
      o = zn/zd
      ...

```

**getprogram** *unit* [*file*]

Displays the set of the compilation units corresponding to the program whose head unit is the compilation unit of name *unit* (see figure 3.1 page 19).

Depending on the computation performed by *Odyssée* during the session, *getprogram* may add comments to the original code of the units giving the informations stored in its active information base.

*getprogram* returns an error message if *unit* is not the name of a compilation unit of the internal library.

**getdiffprogram** *unit* [*file*]

Displays the set of the compilation units corresponding to the program whose head unit is the compilation unit of name *unit*, provided that this unit is the head unit of a program generated by a *diff* command (see figure 3.2 page 20).

Even if some initial units – not generated by a *diff* command – belong to the program, *getdiffprogram* only displays the units generated by the *diff* command. The code of these units includes some comments providing informations stored in the active information base of *Odyssée*.

*getdiffprogram* returns an error message if the name *unit* is not the name of a compilation unit of the internal library generated by a *diff* command.

```
ODYTOP> getprogram sub2
COD Odysee 1.6 - Fri Oct 25 11:28:42 MET DST 1996
```

```
COD Program : sub2
```

```
COD Unit : sub2
```

```
  SUBROUTINE sub2 (x, y)

  DOUBLE PRECISION x, y

  y = f(x)-g(x)
  y = sqrt(y)
  RETURN
  END
```

```
COD Unit : f
```

```
  FUNCTION f (t)

  DOUBLE PRECISION t
  DOUBLE PRECISION f

  f = t*t
  f = exp(f)
  RETURN
  END
```

```
COD Unit : g
```

```
  FUNCTION g (t)

  DOUBLE PRECISION t
  DOUBLE PRECISION g

  g = 1
  IF (t.NE.0.d0) THEN
    g = sin(t)/t
  END IF
  RETURN
  END
```

Figure 3.1: `getprogram sub2`

```
ODYTOP> diff -tl -h g -vars t
ODYTOP> getdiffprogram gtl
COD Odyssee 1.6 - Fri Oct 25 11:28:42 MET DST 1996
```

```
COD Program: gtl
COD with respect to dummies: t
COD strategy: Standard
```

```
COD Unit: gtl
COD Differentiated from the unit: g
COD Active IN dummies: t
COD Active OUT dummies: g
COD Dependencies between OUT and IN:
COD g <-- t
```

```
      SUBROUTINE gtl (t, g, tt1, gt1)

      DOUBLE PRECISION t
      DOUBLE PRECISION g
      DOUBLE PRECISION sd01s
      DOUBLE PRECISION gt1
      DOUBLE PRECISION tt1
      DOUBLE PRECISION sd01st1

      g = 1
      IF (t.NE.0.d0) THEN
        gt1 = 0d0
        sd01st1 = 0d0
        sd01st1 = 0d0
        sd01st1 = tt1*cos(t)
        sd01s = sin(t)
        gt1 = 0d0
        gt1 = (sd01st1*t-sd01s*tt1)/t**2
        g = sd01s/t
      ELSE
        gt1 = 0d0
        sd01st1 = 0d0
      END IF
      RETURN
      END
```

Figure 3.2: getdiffprogram gtl

**getunit** *<unit>* [*<file>*]

Displays the compilation unit of name *<unit>*.

getunit returns an error message if *<unit>* is not the name of a compilation unit in the internal library.

```

ODYTOP> getunit sub2
      SUBROUTINE sub2 (x, y)

      DOUBLE PRECISION x, y

      y = f(x)-g(x)
      y = sqrt(y)
      RETURN
      END

```

**printunit** *<unit>*

Displays the compilation unit on the standard output in an extended format with statements numbers.

printunit returns an error message if *<unit>* is not the name of a compilation unit in the internal library.

```

ODYTOP> printunit sub2
[ Top ]      SUBROUTINE sub2 (x, y)

[ Decl ]      DOUBLE PRECISION x, y, f, g

[ Stat 1 ]      y = f(x)-g(x)
[ Stat 2 ]      y = sqrt(y)
[ Stat 3 ]      RETURN
[ Bottom ]     END

```

### 3.1.3 Displaying Information

listunits, getgraph, and libgraph are used to visualize informations on the contents of the internal library, getaddedunits to visualize the list of the new added units.

The three first commands may all be invoked with an optional *<file>* parameter. If this parameter is present, their output is printed in a file, otherwise it is displayed on the standard output. If the name *<file>* has an extension, then Odyssée is writing in the file of name *<file>*. In the other case, Odyssée is writing the file of name *<file>.<extension>* where *<extension>* is the value of the Odyssée global variable `odyssee_ext`. listunits, getgraph, and libgraph are writing FORTRAN 77 files in the directory defined by the Odyssée global variable `odyssee_dir`. They do not produce any warning message if the file of name *<file>* or of name *<file>.<extension>* already exists in this directory.



**listunits** [*file*]

Displays the list of the compilation units of the internal library: for each compilation unit, its name and its type (Function or Subroutine) are listed.

```
ODYTOP> load exemple
ODYTOP> listunits
Function f
Function g
Subroutine main
Subroutine sub0
Subroutine sub1
Subroutine sub2
```

**getaddedunits**

Displays the list of the names of the new added units.

**libgraph** [*file*]

Displays a representation of the call graph of all the compilation units in the internal library.

```
ODYTOP> libgraph

main +- sub0 +- sub1 +- f
      +- g
      +- sub2 +- f
      +- g
```

**getgraph** (*unit*) [*file*]

Displays the sub-graph of the call graph whose head unit is the unit of name *unit*.

getgraph returns an error message if the name *unit* is not the name of a compilation unit of the internal library.

```
ODYTOP> getgraph sub1

sub1 +- f
      +- g
```

## 3.2 The Information Base Related Commands

### 3.2.1 Displaying Information

getinout and getabstract are used to visualize informations stored in the information base.

Those two commands may both be invoked with an optional *file* parameter. If this parameter is present, their output is printed in a file, otherwise it is displayed on the standard

input. If the name  $\langle file \rangle$  has an extension, then Odyssée is writing in the file of name  $\langle file \rangle$ . In the other case, Odyssée is writing the file of name  $\langle file \rangle.\langle extension \rangle$  where  $\langle extension \rangle$  is the value of the Odyssée global variable `odyssee_ext`. `getinout`, and `getabstract` are writing FORTRAN 77 files in the directory defined by the Odyssée global variable `odyssee_dir`. They do not produce any warning message if the file of name  $\langle file \rangle$  or of name  $\langle file \rangle.\langle extension \rangle$  already exists in this directory.

**getinout**  $\langle unit \rangle$  [ $\langle file \rangle$ ]

Displays the input and the output variables of the compilation units belonging to the program whose head unit is the unit  $\langle unit \rangle$ .

`getinout` returns an error message if the name  $\langle unit \rangle$  is not the name of a compilation unit of the internal library.

```
ODYTOP> getinout sub0
```

```
Unit : sub0
IN  dummys: v u
IN  globals: zd zn
OUT dummys: v u
OUT globals: zd zn
```

```
Unit : sub1
IN  dummys: y x
OUT dummys: y x
```

```
Unit : f
IN  dummys: f t
OUT dummys: f t
```

```
Unit : g
IN  dummys: g t
OUT dummys: g t
```

```
Unit : sub2
IN  dummys: y x
OUT dummys: y x
```

**getabstract**  $\langle unit \rangle$  [ $\langle file \rangle$ ]

Displays the dependencies between the input and the output variables of the compilation units belonging to the program whose head unit is the unit  $\langle unit \rangle$ .

`getabstract` returns an error message if the name  $\langle unit \rangle$  is not the name of a compilation unit of the internal library.

```
ODYTOP> getabstract sub0
```

```
Unit : sub0
COD Dependencies between OUT and IN:
```

```

u <-- zd zn v u
v <-- zd zn v u
zn <-- zd zn v u
zd <-- zd zn v u

```

```

Unite : sub1
x <-- y x
y <-- y x

```

```

Unite : f
t <-- f t
f <-- f t

```

```

Unite : g
t <-- g t
g <-- g t

```

```

Unite : sub2
x <-- y x
y <-- y x

```

### 3.2.2 Modifying the Information Base

`computeibasis` makes *Odyssée* analyze the code to find the dependencies between variables. This computation is automatically performed before any differentiation.

`setdummys`, `setglobals`, `setinout`, `setabstract` allow the user to provide *Odyssée* with information on the variables of the compilation units. They are particularly useful to provide information on variables of black-box subroutines whose code is not available. They can also be used to overwrite information computed by *Odyssée*.

These four commands are not supposed to be used at the interpreter level: they are normally used inside information base files which are automatically loaded by the system before any differentiation. Any invocation of one of those four commands at the interpreter level will have no effect on further `computeibasis` or `diff` commands.

When the user wants to modify an entry of the information database attached to the head unit  $\langle unit \rangle$ , he has to define an information file: a file of name  $\langle unit \rangle\_B1.\langle extension \rangle$  where  $\langle extension \rangle$  is the value of the *Odyssée* global variable `ibasis_ext` and to store it in one of the directory of the *Odyssée* global variable `ibasis_path`.

#### **computeibasis** $\langle unit \rangle$

Computes the information base of the program whose head unit has the name  $\langle unit \rangle$ .

#### **setdummys** $\langle unit \rangle$ $\langle var\_list \rangle$

Sets the list of the arguments of the unit of name  $\langle unit \rangle$  to the list of variables  $\langle var\_list \rangle$ .

**setglobals**  $\langle unit \rangle$   $\langle var\_list \rangle$

Sets the list of the global variables of the unit of name  $\langle unit \rangle$  to the list of variables  $\langle var\_list \rangle$ .

**setinout**  $\langle unit \rangle$   $\langle var\_list\_1 \rangle$   $\langle var\_list\_2 \rangle$   $\langle var\_list\_3 \rangle$   $\langle var\_list\_4 \rangle$

Sets the lists of the input and output dummies and global variables of the unit of name  $\langle unit \rangle$ .

The list of input dummies is set to the list of variables  $\langle var\_list\_1 \rangle$ , the list of output dummies to the list  $\langle var\_list\_2 \rangle$ , the list of input global to the list  $\langle var\_list\_3 \rangle$ , and the list of output global to  $\langle var\_list\_4 \rangle$ .

As a side effect, setinout sets the dependencies between variables to their default value: each output variable depends on all the input variables.

**setabstract**  $\langle unit \rangle$   $\langle var\_list\_list \rangle$

Sets the lists of the dependencies between the input and the output variables of the compilation unit of name  $\langle unit \rangle$ .

## 3.3 The Differentiation Related Commands

### 3.3.1 The diff Command

The command `diff` is used to differentiate a program stored in the internal library. `diff` produces new compilation units and stores them in the internal library. The syntax of `diff` is the following:

**diff**  $\langle arguments\_expression \rangle$

$\langle arguments\_expression \rangle$  is made up of several necessary and optional arguments given as:

$-\langle argument\_name \rangle$  [ $\langle argument\_value \rangle$ ]

To run properly, `diff` requires at least three arguments:

- the name of the head unit to differentiate
- the list of the active input variables
- the method of differentiation chosen by the user

The table below gives the list of the corresponding argument names, their syntax and their actions.

-variables	-vars	-variables $\langle var\_list \rangle$ or -vars $\langle var\_list \rangle$ Introduces the list of the active variables for differentiation
-head	-h	-head $\langle unit \rangle$ or -h $\langle unit \rangle$ Introduces the head unit of the program to differentiate. diff will return an error message if the name $\langle unit \rangle$ is not the name of a compilation unit of the internal library.
-tangent	-tl	Sets the differentiation method to direct mode. The generated code will compute the tangent line of the function.
-cotangent	-cl	Sets the differentiation method to reverse mode. The generated code will compute the cotangent line of the function.

The tables below give the list of the optional arguments of diff, their syntax and their actions.

-output	-o	-output $\langle file \rangle$ or -o $\langle file \rangle$ Defines the output of the diff command. By default, the FORTRAN 77 code generated by diff is stored in the internal library but is neither displayed nor saved in a file. Giving std as file name tells <i>Odyssée</i> to display the code on the standard output. Otherwise the code is printed in a file using the same rule than other commands displaying FORTRAN 77 code (see section 3.1.2).
-optloops	-opt	-optloops $\langle unit \rangle$ or -opt $\langle unit \rangle$ Tells the diff command to use a special method that produces an optimized code for subroutines with loops. This method can be used only if the files opt_rev.f and opt_utils.f have been load in the internal library. These two files are included in the sub-directory fortran of the <i>Odyssée</i> distribution.

### 3.3.2 Other Commands

A side effect of the evaluation of the diff -head  $\langle unit \rangle$  -vars  $\langle var\_list \rangle$  command line is to compute of the active dependencies i.e. to propagate the active input variables  $\langle var\_list \rangle$  through the compilation units of the program which head unit has name  $\langle unit \rangle$ . These

dependencies are stored in the active information basis attached to the diff computation. `getactiveinout` can be used to visualize the information of the last computed active information basis.

**getactiveinout** *<unit>* [*<file>*]

Displays the dependencies between the active input variables and the active output variables of each unit of the program which head unit has name *<unit>*. If the *<file>* parameter is given, the output of `getactiveinout` is redirected to a file, it is displayed on the standard output otherwise.

If the name *<file>* has an extension, then Odyssée is writing in the file of name *<file>*. In the other case, Odyssée is writing the file of name *<file>*.*<extension>* where *<extension>* is the value of the Odyssée global variable `odyssee_ext`. When writing in a file, `getactiveinout` are writing in the directory defined by the Odyssée global variable `odyssee_dir`. It does not produce any warning message if the file of name *<file>* or of name *<file>*.*<extension>* already exists in this directory.

### 3.4 The Units Related Commands

**slice** *<unit>* *<var\_list>*

Generates an unit equivalent to the unit of name *<unit>* where the code is restricted to the computation of the variables of the list *<var\_list>*.

**makeblock** *<unit>* *<main>* *<sub>* *<number\_list>*

Builds the subroutine of name *<sub>* with the statements of the routine of name *<unit>* which number are in the list *<number\_list>*. Create the routine of name *<main>* which is equivalent to the routine of name *<unit>* but contents a call to the subroutine *<sub>*.

```

ODYTOP> printunit sub0
[ Top ]      SUBROUTINE sub0 (u, v)

[ Decl ]     COMMON // zd, zn
[ Decl ]     DOUBLE PRECISION u, v, z1, z2, zn, zd

[ Stat 1 ]   CALL sub1(u, z1)
[ Stat 2 ]   CALL sub2(v, z2)
[ Stat 3 ]   zn = z1-z2
[ Stat 4 ]   zd = 1+z1+z2
[ Stat 5 ]   RETURN
[ Bottom ]   END

ODYTOP> makeblock sub0 sub0s sub3 (3 4)

ODYTOP> getunit sub0s

```

```

SUBROUTINE sub0s (u, v)

EXTERNAL sub3
COMMON // zd, zn
DOUBLE PRECISION u, v, z1, z2, zn, zd

CALL sub1(u, z1)
CALL sub2(v, z2)
CALL sub3(z2, z1)
RETURN
END

ODYTOP> getunit sub3
SUBROUTINE sub3 (z2, z1)

COMMON // zd, zn
DOUBLE PRECISION z1, z2

zn = z1-z2
zd = 1+z1+z2
RETURN
END

```

## 3.5 The System Related Commands

### 3.5.1 Loading Batch Files

**loadbatch**  $\langle batch_1 \rangle$  [ $\langle batch_2 \rangle \dots \langle batch_N \rangle$ ]  
Attempts to execute the specified batch file(s).

If the name  $\langle batch_i \rangle$  has an extension, then *Odyssée* is looking for the file  $\langle batch_i \rangle$ . Otherwise, *Odyssée* is looking for the file  $\langle batch_i \rangle.\langle extension \rangle$  where  $\langle extension \rangle$  is the value of the *Odyssée* global variable `batch_ext`.

`loadbatch` returns an error message if no file of the given name(s) is found in one of the directory listed in the *Odyssée* global variable `batch_path`. *Odyssée* also returns an error message if any file found contains anything else but a sequence of valid command lines of this language.

If the execution of a batch file produces an error then the command lines of the next files are not executed.

**loadodyseerc**

Attempts to execute the special init batch file `HOME/.odyseerc`

### 3.5.2 Configuration of the System

**getvar** *<var>*

Displays the value of the Odyssée global variable *<var>* (see 3.6).

**setvar** *<var>* *<value>*

Sets the value of the variable *<var>* to *<value>*.

setvar returns an error message if the type of *<value>* is not compatible with the predefined type of *<var>* (see 3.6).

### 3.5.3 Miscellaneous Commands

**help** [*<command>*]

help without argument displays a list of all the commands of the Odyssée language.

help followed by a string which is the name of a command of the Odyssée language displays a short description of this command including its syntax.

help followed by anything else produces an error message.

**shell** *<string>*

Sends *<string>* to the shell.

**version**

Displays the current version number of Odyssée.

**copyright**

Displays the copyright banner of Odyssée.

**exit**

Terminates Odyssée without saving anything.

**quit**

Terminates Odyssée without saving anything.

## 3.6 The Odyssée Global Variables

In this section, all the Odyssée global variables are described. Initialized to their default values when Odyssée starts, these variables allow the user to change the configuration of the system.



### 3.6.1 Methods of Automatic Differentiation

`minimal_split` is the only global variable used for the configuration of the automatic differentiation method. It is a boolean variable whose default value is true. When false, *Odyssée* splits all the algebraic expression in the input code into elementary operations, otherwise, the constants and the arguments of the functions are isolated in local variables.

### 3.6.2 Format of Generated FORTRAN 77 Code

Variable	Type	Default	Comment
<code>all_uppercase</code>	boolean	true	When false, only the FORTRAN 77 keywords of the generated code are displayed with uppercase letters, otherwise, the whole code is written with uppercase letters.
<code>do_indentation</code>	integer	3	Tunes the indentation of the body of a do loop in the generated code. Unit is a white space.
<code>if_indentation</code>	integer	2	Tunes the indentation of the body of a if statement in the generated code. Unit is a white space.
<code>continuation_char</code>	string	:	Defines the continuation char.
<code>blank_line_after_stat</code>	boolean	false	When true, an empty line is inserted after any statement in the FORTRAN 77 generated code.
<code>parenthesis_level</code>	integer	0	Tunes the level of parenthesis in the generated instructions. Default correspond to a minimal level of parenthesis.

### 3.6.3 Input and Output through Files

Variable	Type	Default	Comment
fortran_ext	string	f	Extension of the FORTRAN 77 files (read and writen)
fortran_path	list of strings	(.)	List of the paths used to read FORTRAN 77 files.
fortran_dir	string	.	Name of the directory where Odyssée writes the FORTRAN 77 files.
include_ext	string	inc	Extension of the include files (read and writen)
include_path	list of strings	(.)	List of the paths used to read include files of the FORTRAN 77 code.
include_dir	string	.	Name of the directory where Odyssée writes the include files.
batch_ext	string	batch	Extension of the batch files (read and writen)
batch_path	list of strings	(.)	List of the paths used to read batch files.
batch_dir	string	.	Name of the directory where Odyssée writes the batch files.
ibasis_ext	string	bi	Extension of the information base files (read and writen)
ibasis_path	list of strings	(.)	List of the paths used to read information base files.
ibasis_dir	string	.	Name of the directory where Odyssée writes the information base files.
odyssee_ext	string	od	Extension of the file to be written which may not be read again by Odyssée.
odyssee_dir	string	(.)	Name of the directory where Odyssée writes the files which are not supposed to be read again.
home_dir	string	HOME	Home dir of the user (Odyssée point of view).

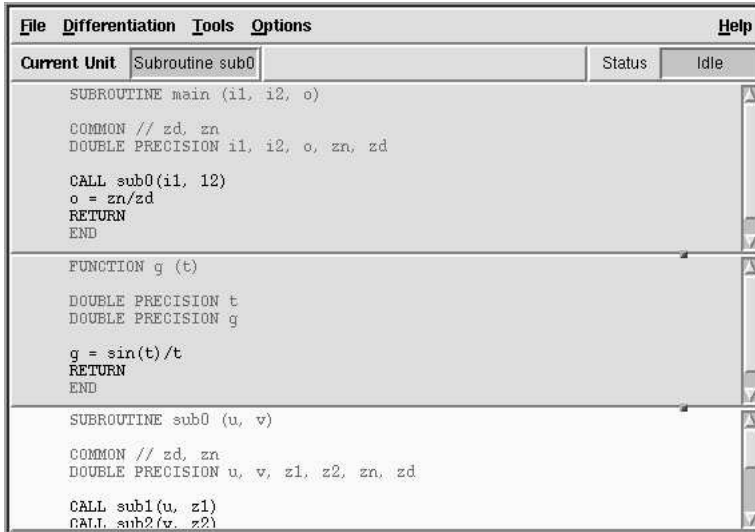


## Chapter 4

# The Odyssée Graphical User Interface

The graphical user interface has been designed with the help of the TK toolkit. Therefore it has a MOTIF look-and-feel with pop-up windows, menus and sub-menus, highlighting of pre-selected and selected items, radio buttons, scrollable windows ... see figure 4.1.

The main window of the Odyssée graphical user interface consists in three different parts :



The screenshot shows a graphical window titled "File Differentiation Tools Options Help". Below the title bar is a "Current Unit" field containing "Subroutine sub0", a "Status" field containing "Idle", and an "Idle" button. The main area of the window is a scrollable text editor displaying Fortran code. The code is organized into three sections: a main subroutine, a function, and another subroutine. The main subroutine "main" calls "sub0". The function "g" calculates the sine of t divided by t. The subroutine "sub0" calls "sub1" and "sub2".

```
File Differentiation Tools Options Help
Current Unit Subroutine sub0 Status Idle
SUBROUTINE main (i1, i2, o)
COMMON // zd, zn
DOUBLE PRECISION i1, i2, o, zn, zd
CALL sub0(i1, i2)
o = zn/zd
RETURN
END
FUNCTION g (t)
DOUBLE PRECISION t
DOUBLE PRECISION g
g = sin(t)/t
RETURN
END
SUBROUTINE sub0 (u, v)
COMMON // zd, zn
DOUBLE PRECISION u, v, z1, z2, zn, zd
CALL sub1(u, z1)
CALL sub2(v, z2)
```

Figure 4.1: the Odyssée graphical user interface window

- the menubar with four active menus used for sending commands to *Odyssée*, and a Help menu,



- the status bar with an active button to select the *current unit* in the internal library, and two information windows displaying the name of the current unit and the status of the engine of *Odyssée*.



- the display frame, which can be divided in several views, used for the visualization of the FORTRAN 77 units of the internal library

```

SUBROUTINE main (i1, i2, o)
COMMON // zd, zn
DOUBLE PRECISION i1, i2, o, zn, zd

CALL sub0(i1, i2)
o = zn/zd
RETURN
END

FUNCTION g (t)
DOUBLE PRECISION t
DOUBLE PRECISION g

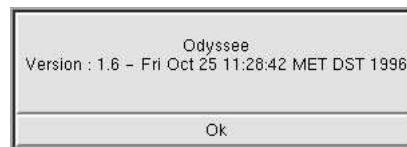
g = sin(t)/t
RETURN
END

SUBROUTINE sub0 (u, v)
COMMON // zd, zn
DOUBLE PRECISION u, v, z1, z2, zn, zd

CALL sub1(u, z1)
CALL sub2(v, z2)

```

In the current version of *Odyssée*, the Help menu is restricted to one item **About a** which provides the informations on the version of *Odyssée* in a pop-up window as shown below (Click on Ok to dismiss it).



<b>L</b> oad source	<b>l</b>
S <b>a</b> ve source	<b>s</b>
<b>C</b> lear library	<b>c</b>
S <b>a</b> ve s <b>e</b> ttings	<b>t</b>
<hr/>	
<b>A</b> dd view	<b>a</b>
<b>R</b> emove current view	<b>r</b>
<b>Q</b> uit	<b>q</b>

Figure 4.2: the File menu

<b>L</b> oad source	<b>l</b>
S <b>a</b> ve source	<b>s</b>
<b>C</b> lear library	<b>c</b>
S <b>a</b> ve s <b>e</b> ttings	<b>t</b>
<hr/>	
<b>A</b> dd view	<b>a</b>
<b>R</b> emove current view	<b>r</b>
<b>Q</b> uit	<b>q</b>
<hr/>	
/0/safir/papegay/Odysee/doc/exemples/exemple.f	

Figure 4.3: the extended File menu

## 4.1 The File Menu

There is seven items in the File menu when beginning a new session of *Odyssée* (see figure 4.2).

The Save item stays grayed and the Save feature is disabled as long as the internal library is empty. When FORTRAN 77 files are loaded in the internal library of *Odyssée*, the File menu is extended by items labeled with the names of these files (see figure 4.3).

Selecting one of those items is a shortcut to reloading the corresponding file in the internal library of *Odyssée*.

### 4.1.1 The Load source Button

The **L**oad source **l** item invokes the load command of the language of *Odyssée* to load in the internal library the compilation units of selected file(s). A dialog window like in figure 4.4 appears on the screen for the selection of the file(s) to load.

In the display area of the dialog-window are listed the files of the directory whose name appears in the Pathname field and matches the value of the Selection pattern field.

Files are selected by clicking on them, then the name(s) of selected file(s) are grayed in the display area and appear(s) in the Filename field.

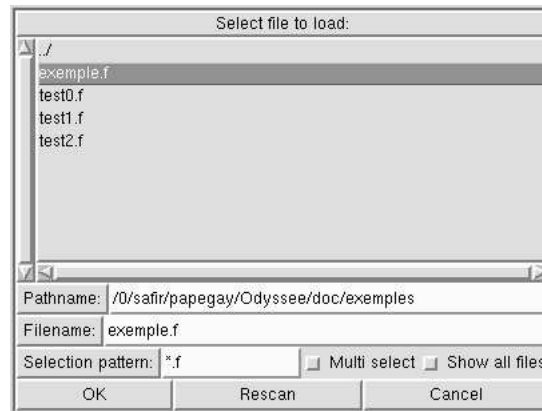


Figure 4.4: the Load source dialog window

When the selected file is a directory, its name is appended to the pathname by double-clicking on it and then the files corresponding to the selection pattern are listed in the display area.

Clicking on the Selection pattern button offers a selection of possible extensions.

Pathname, Filename and Selection pattern fields can also be edited in the corresponding frames.

Multiple selection is allowed when the Multi select check-button is activated. Double-clicking on a name will append it to the Filename list.

Activation of the Show all files check-button causes *Odyssée* to list all the files of the Pathname directory – even the files that start with a dot – in the display area.

Clicking on the Cancel button interrupts the loading process and causes *Odyssée* to return to its previous state.

The Rescan button is used for updating the list of files selectable.

Once the selection has been done, clicking on the Ok button invokes the load command with the selected file(s) as argument(s).

An error window is popped up if no file of the selected name is found in the pathname directory or if any found file is a syntactically incorrect FORTRAN 77 file.

#### 4.1.2 The Save source Button

When enabled, the **Save source** **s** item invokes the `getunit` command of the command language of *Odyssée* to save the current unit in the selected file. A dialog window like in figure 4.5 appears on the screen for the selection of the file.

In the display area of the dialog window, the files of the directory which name has the value of the Pathname field and matching the value of the Selection pattern field are listed.

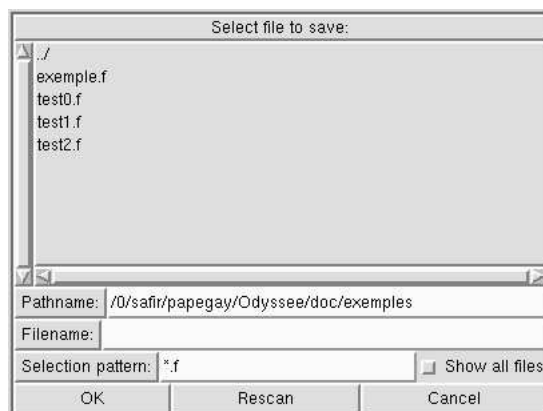


Figure 4.5: the Save source dialog window

A file can be selected by clicking on its name, then the name of the selected file is grayed in the display area and appears in the Filename field.

When the selected file is a directory, its name is appended to the pathname by double-clicking on it and then the files corresponding to the selection pattern are listed in the display area.

Clicking on the Selection pattern button offers a selection of possible extensions.

Pathname, Filename and Selection pattern fields can also be edited in the corresponding frames.

Activation of the Show all files check-button causes *Odyssée* to list all the files – even the files that start with a dot – of the pathname directory in the display area.

Clicking on the Cancel button interrupts the saving process and causes *Odyssée* to return to its previous status.

The Rescan button is used for updating the list of selectable files.

Once the selection has been done, clicking on the Ok button invokes the save command with the current unit and the selected file as argument.

If the file already exists, it will be overwritten without any warning.

### 4.1.3 Other Buttons

Selecting the **Clear library** **c** resets the internal library of *Odyssée* to the empty set.

The **Save settings** **t** saves the current values of several *Odyssée* global variables (governing paths and extension names) into a batch file of name `.xodysserc.batch` in the directory where *Odyssée* has been started. Note that this file is different from the `.odysserc` file and is not loaded automatically when starting a new session of *Odyssée*.



The views management buttons will be described in the section 4.2 which presents the display area and the features to control this area.

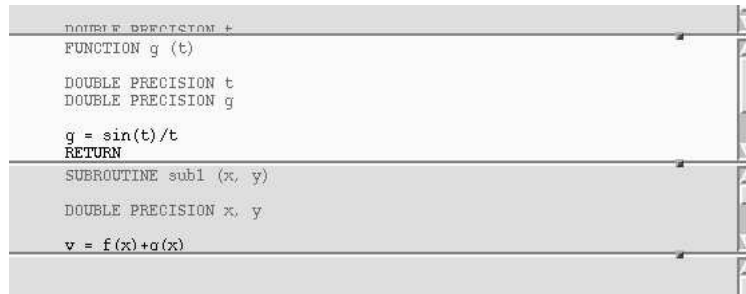
Clicking on **Quit** **q** terminates *Odyssée* without saving anything but a pop-up window appears for confirmation.

## 4.2 The Display Area

To visualize several compilation units at the same time, the display area can be divided in several views.

The **Add view** **a** button adds a new view in the display area. As the size of the display area is controlled by the window manager, the existing views are resized when adding a new view: the size of each view is set to the size of the display area divided by the number of views.

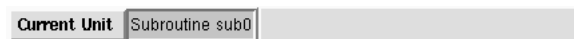
The current view can be selected by clicking on it. All the views, except the *current view*, are grayed. The respective size of the views can be changed by the user by moving the separators lines (using a small drop-and-drag button on the line) as shown below:



The information of the Current Unit of the status bar concerns the current view.

The **Remove current view** **r** button removes the current view of the display area. Whatever the size of the remaining views, they are resized: the size of each view is set to the size of the display area divided by the number of views. The view on top of the display area is set as the current view, and the attached compilation unit becomes the current unit.

The left part of the status bar consists in a button labeled Current unit and a window where the name of the current unit is displayed:



By definition, the current unit is the compilation unit displayed in the current view.

Clicking on the Current unit button invokes the listunits command and pulls up a menu like in figure 4.6 with as many (up to ten) items as units in the internal library of *Odyssée*. Clicking on one of those items selects the corresponding compilation unit as current unit. If the internal library contents more than ten compilation units then the only ten first units are displayed and a More... item allows to access a new menu with other units.

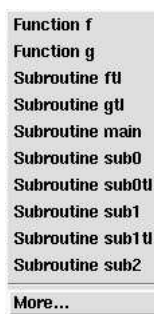
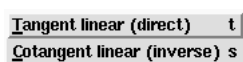


Figure 4.6: a sample of the Current unit button

### 4.3 The Differentiation Menu



The two items of the Differentiation menu correspond to a call to the `diff` command on the current unit with different options for the method of differentiation, respectively `-tl` for direct (tangent line computation) and `-cl` for inverse (cotangent line computation).

The active input variables of the head unit are required prior to differentiating FORTRAN 77 code. For this purpose, invocation of any of these two items causes *Odyssée* to open a new dialog window (see figure 4.7).

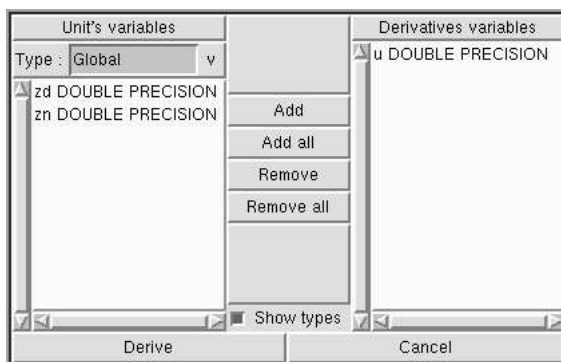


Figure 4.7: the dialog window for Differentiation menu

The active input variables with respect to which *Odyssée* will differentiate the current unit have to be selected by the user.

The right part of the dialog window is the Unit's variables frame. In this frame is displayed the list of the variables of the current unit.

The variables are listed by type and only the variables of types Global, Local or Dummy appearing in the frame Type are displayed – only the Global variables in figure 4.7.

To change the type of the variables to be displayed, click on the *v* button and select the item of the menu corresponding to the selected type. Note that it is impossible to differentiate with respect to a local variable.

The selected active input variables are listed on the left part of the dialog window: the Derivatives variables frame.

To add a variable or a list of variables displayed in the Unit's variables frame to the Derivatives variables frame, select their names by clicking on them and click on the Add button.

Use Add all button to add all the variables displayed in the Unit's variables frame to the Derivatives variables frame.

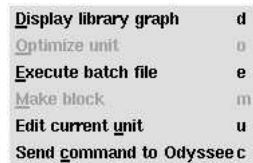
To remove a variable displayed in the Derivatives variables frame from the list of the selected active input variables, select its name by clicking on it and click on the Remove button.

Use Remove all button to remove all the variables displayed in the Derivatives variables frame from the list of the selected active input variables.

The Show types check-button commands the display of the FORTRAN 77 type of the variables.

Click on the Derive button to send the diff command to the *Odyssée* engine, on the Cancel button to abort it.

## 4.4 The Tools Menu



The Tools menu consists of six items but in the current version of *Odyssée*, only five of them are implemented. That is why the Optimize unit stays grayed.

### 4.4.1 The Display library graph Button

The **Display library graph** **d** button invokes the libgraph command and displays a representation of the call graph of all the compilation units in the internal library in a pop-up window as shown in figure 4.8.

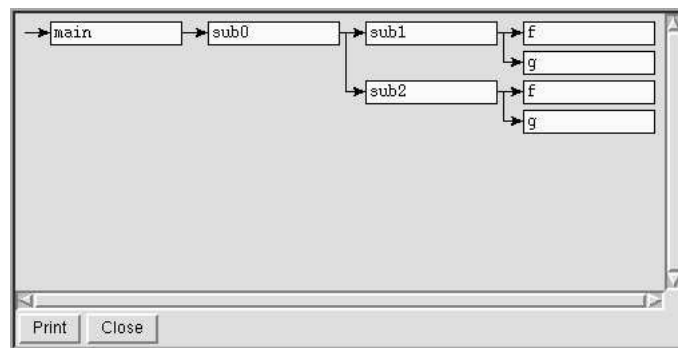


Figure 4.8: a sample of a call graph

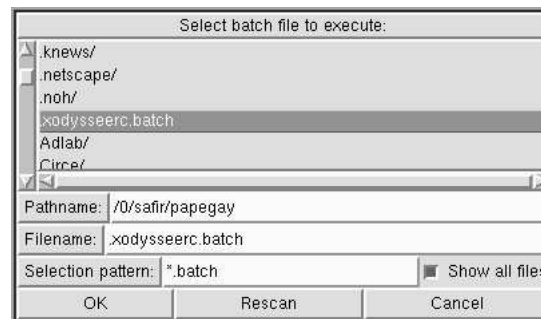


Figure 4.9: the Execute batch file dialog window

#### 4.4.2 The Execute batch file Button

The **Execute batch file** button is used to execute the sequence of commands of a batch-file into *Odyssée*. Selection of the batch file is made through a dialog window like in figure 4.9.

In the display area of the dialog-window are listed the files of the directory whose name has the value of the Pathname field and matching the value of the Selection pattern field.

Files are selected by clicking on them, then the name(s) of selected file(s) are grayed in the display area and appear(s) in the Filename field.

When the selected file is a directory, its name is appended to the pathname by double-clicking on it and then the files corresponding to the selection pattern are listed in the display area.

Clicking on the Selection pattern button offers a selection of possible extensions.

Pathname, Filename and Selection pattern fields can also be edited in the corresponding frames.

Activation of the Show all files check-button causes *Odyssée* to list all the files of the pathname directory – even the files that start with a dot – in the display area.

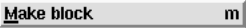
Clicking on the Cancel button interrupts the loading process and causes *Odyssée* to return to its previous status.

The Rescan button is used for updating the list of selectable files.


Once the selection has been done, clicking on the Ok button invokes the loadbatch command with the selected file as argument and the command lines of the batch-file are executed by *Odyssée*.

An error window is popped up if no file with the selected name is found in the pathname directory or if any files found contain anything else but valid *Odyssée* command lines.

### 4.4.3 The Make block Button

Clicking on the  button is only possible when statements of the current unit have been selected, otherwise the item stays greyed. It invokes the menublock command to build a subroutine with the selected statements of the current unit and a main routine which is equivalent to the current unit and contents a call to the subroutine (see exemple on figure 4.10). Names of the main routine and of the subroutine are prompted in a pop-up window.

### 4.4.4 The Edit current unit Button

Clicking on the  button opens a window for editing the FORTRAN 77 code of the current unit (see figure 4.11).

The editing facilities are the features of a very simplified emacs-like editor accepting the control characters described in the table below where “~” stands for “control”.

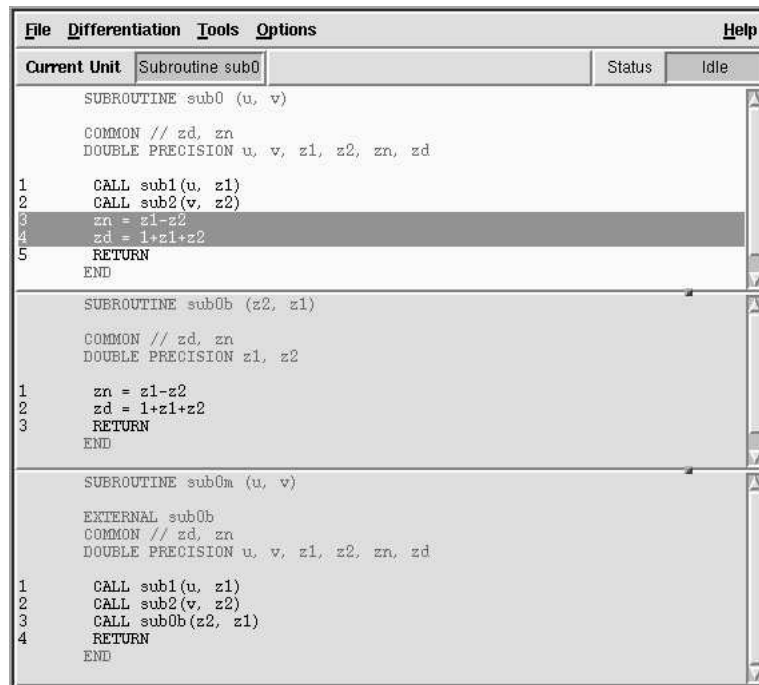


Figure 4.10: An exemple of use of Make block button

```

SUBROUTINE maint1 (il, i2, o, i1ttl, ottl)

DOUBLE PRECISION il, i2, o
DOUBLE PRECISION i1ttl
DOUBLE PRECISION ottl
EXTERNAL sub0tl
COMMON // zd, zn
COMMON // zdttl, znttl
DOUBLE PRECISION zn, zd
DOUBLE PRECISION znttl, zdttl

znttl = 0d0
zdttl = 0d0
C Completion of the following Call

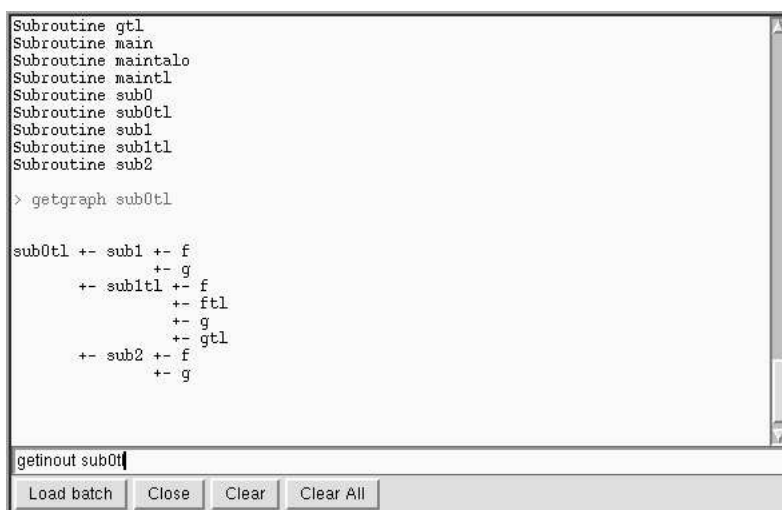
znttl = 0d0
zdttl = 0d0
CALL sub0tl(il, i2, i1ttl)
ottl = 0d0
ottl = (znttl*zd-zn*zdttl)/zd**2
o = zn/zd
RETURN
END

```

Figure 4.11: the editing window for FORTRAN 77 code

characters	action
~f	forward character
~b	backward character
~a	beginning of line
~e	end of line
~p	previous line
~n	next line
~d	delete character
~h	delete backward
~t	transpose characters
~k	kill line
~o	open line

The effect of the Save modifications button is to replace the original unit by the result of the edition inside the internal library of *Odyssée*: if the name of the routine has been modified during edition, then the result of the edition is saved under the new name as a new unit of the internal library, otherwise, the original routine is overwritten.



```
Subroutine gtl
Subroutine main
Subroutine maintalo
Subroutine maintl
Subroutine sub0
Subroutine sub0tl
Subroutine sub1
Subroutine sub1tl
Subroutine sub2

> getgraph sub0tl

sub0tl +- sub1 +- f
        +- g
      +- sub1tl +- f
            +- ftl
            +- g
            +- gtl
      +- sub2 +- f
            +- g
```

getinout sub0tl

Load batch Close Clear Clear All

Figure 4.12: the interpreter window

If the result of the edition is not a valid FORTRAN 77 compilation unit, then *Odyssée* pops up an error window.

Clicking on the Cancel button aborts the edition process without changing anything.

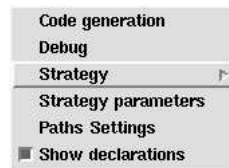
#### 4.4.5 The Send command to Odysee Button

The **Send command to Odysee** button allows to use the command language of *Odyssée* without exiting the graphical user interface. Commands are entered and results are displayed in an *Odyssée* command window like shown in figure 4.12.

In the upper part of the window are displayed the result of the *Odyssée* commands in the same format than they would be on an interpreter window. The lower part of the window is used for editing a line of commands. The following control characters are recognized for edition (^f: forward character, ^b: backward character, ^a: beginning of line, ^e: end of line).



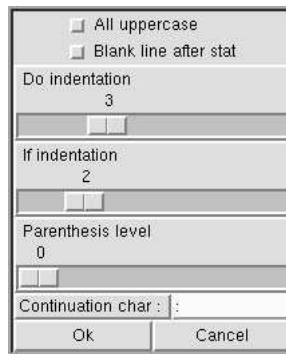
## 4.5 The Options Menu



The six buttons of the Options are used for setting global variables of *Odyssée*.

### 4.5.1 The Code generation Button

The **Code generation** button is used to control the format of the FORTRAN 77 code printed by *Odyssée*. It pops up the window below:



When the All uppercase check-button is not on, only the FORTRAN 77 keywords of the printed code are written with uppercase letters, otherwise, the whole code is written with uppercase letters included the names of the variables.

When the Blank line after stat check button is checked, an empty line is inserted after any statement in the FORTRAN 77 printed code.

The Do indentation slide button tunes the indentation of the body of a do loop in the printed code. The unit is a white space.

The If indentation slide button tunes the indentation of the body of a if statement in the printed code. The unit is a white space.

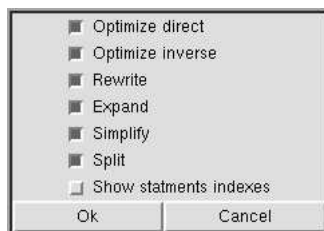
The Parenthesis level slide button tunes the level of parenthesis in the printed instructions. Zero corresponds to a minimal level of parenthesis

The Continuation char input field allows to enter the FORTRAN 77 continuation character used when breaking lines of code.

### 4.5.2 The Debug Button

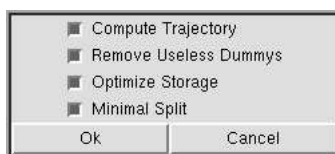
The **Debug** button is for developers purpose only. It allows to suppress (or to add) parts of the differentiation process executed by the diff command.

The Debug dialog box containing seven check-buttons is reproduced below:



### 4.5.3 The Strategy parameters Button

The **Strategy parameters** button is used for controlling the strategy used by the differentiation algorithm of Odyssée. Four strategy options can be chosen by checking buttons of the dialog window:

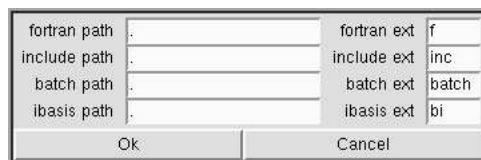


In the current version, only the strategy associated to the Minimal split button is implemented.

When the button is not checked, Odyssée splits all the algebraic expression in the input code into elementary operations, otherwise, the constants and the arguments of the functions are isolated in local variables.

### 4.5.4 The Path settings Button

The **Paths Settings** button is used to set the path and the extensions of the names of files for input or output. It pops up the window below:



The several input fields allow the user to redefine the paths and the extensions of files read or written by Odyssée (FORTRAN 77 units, include FORTRAN 77 files, batch files, information base files).

## 4.6 Other Items

The Strategy button is not yet implemented.

The  **Show declarations** button is a check button which control the display of the FORTRAN 77 compilation units. The declarations of variables inside the FORTRAN 77 routines are shown only when this button is checked.

# Index

active information base, 12  
active input variables, 7  
active variable, 12  
active variables, 5

black-box, 5

character set, 15  
clear, 17  
clearaddedunits, 17  
command line, 16  
command name, 16  
comment, 16  
compilation unit, 9  
computeibasis, 24  
control-dependencies, 10  
copyright, 29  
current unit, 34  
current view, 38

dependencies, 10  
diff, 25

exit, 29

getabstract, 23  
getactiveinout, 27  
getaddedunits, 22  
getdiffprogram, 18  
getgraph, 22  
getinout, 23  
getlibrary, 18  
getprogram, 18  
getunit, 21  
getvar, 29

head unit, 7, 9  
help, 29

identifiers, 15  
information base, 10  
information file, 12  
input and output variables, 10  
input variables, 10  
internal library, 7, 9

libgraph, 22  
lists, 15  
listunits, 22  
load, 17  
loadbatch, 28  
loadodyseerc, 28

makeblock, 27

numbers, 15

output variables, 10

printunit, 21

quit, 29

remove, 17

setabstract, 25  
setdummys, 24  
setglobals, 25  
setinout, 25  
setvar, 29  
shell, 29

slice, 27

strings, 15

value-dependencies, 10

variables, 9

version, 29

white space, 16



---

Unité de recherche INRIA Sophia Antipolis  
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique  
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)

Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

---

Éditeur  
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)  
<http://www.inria.fr>  
ISSN 0249-6399