Project tools and Loop Calculations with DIANA and aITALC

Alejandro Lorca

DESY Zeuthen

Contents

1. Motivation
2. Project tools
3. DIANA and $a\mathring{\imath}$TALC
4. Play a bit! (examples)

I. Motivation

Motivation: writing up

If you don’t have to be a journalist to write good papers. . .

Motivation: writing up

If you don't have to be a journalist to write good papers. . .

. . . then you don't need to be a hacker to write decent software

Motivation: writing up

If you don't have to be a journalist to write good papers. . .

. . . then you don't need to be a hacker to write decent software

Both are scientist's responsabilities

Motivation: decent scientific software

What should be understood by decent software in science?

In particle physics, where is quite frequent to rely on non-commercial programs, the software should

- ✍ have documentation (and if open sourced MANY Comments)

Motivation: decent scientific software

What should be understood by decent software in science?

In particle physics, where is quite frequent to rely on non-commercial programs, the software should

- ✍ have documentation (and if open sourced MANY Comments)
- 🏭 be trivial to compile and easy to execute (not only in authors' machines)

Motivation: decent scientific software

What should be understood by decent software in science?

In particle physics, where is quite frequent to rely on non-commercial programs, the software should

- have documentation (and if open sourced MANY Comments)
- be trivial to compile and easy to execute (not only in authors' machines)
- accept debugging and perform quality/correctness tests

Motivation: decent scientific software

What should be understood by decent software in science?

In particle physics, where is quite frequent to rely on non-commercial programs, the software should

- ✍ have documentation (and if open sourced MANY Comments)
- 🏭 be trivial to compile and easy to execute (not only in authors' machines)
- CE accept debugging and perform quality/correctness tests
- ℹ provide the *'Hello world'* examples

Motivation: decent scientific software

What should be understood by decent software in science?

In particle physics, where is quite frequent to rely on non-commercial programs, the software should

- ✍ have documentation (and if open sourced MANY Comments)
- 🏭 be trivial to compile and easy to execute (not only in authors' machines)
- CE accept debugging and perform quality/correctness tests
- ℹ provide the *'Hello world'* examples

Public free codes $\not\Rightarrow$ Not-working codes

Let's make the effort because every piece of code matters!

II. Project Tools

Project Tools: overview

To handle software projects we'd better be organized. Some UNIX tools extremely facilitate the management of complicated tasks.

I would like to mention some GNU/LINUX tools in

① Version management

② Installation

③ Inter-operability

✓ Enhance the quality of your software and the efficiency of team-work!

Project Tools: Bad version management

John and Mary compute π. Is this directory structure familiar to you?

```
mary@linux:~ > ls -l
total 32
-rw-rw-r-- 1 mary users 108 2005-02-01 16:02 pi_2.f
-rw-rw-r-- 1 john users 141 2005-02-01 18:02 pi_2.old.f
-rw-rw-r-- 1 mary users 132 2005-02-01 18:05 pi_2.old.f~
-rw-rw-r-- 1 mary users 156 2005-04-05 16:03 pi_4.f
-rw-rw-r-- 1 mary users  89 2005-01-18 11:47 pi.f
-rw-rw-r-- 1 john users 171 2005-04-05 16:04 pi.new.f
-rw-rw-r-- 1 john users 108 2005-03-01 16:05 pi-works_I_think.f
```

Project Tools: Bad version management

John and Mary compute π. Is this directory structure familiar to you?

```
mary@linux:~ > ls -l
total 32
-rw-rw-r-- 1 mary users 108 2005-02-01 16:02 pi_2.f
-rw-rw-r-- 1 john users 141 2005-02-01 18:02 pi_2.old.f
-rw-rw-r-- 1 mary users 132 2005-02-01 18:05 pi_2.old.f~
-rw-rw-r-- 1 mary users 156 2005-04-05 16:03 pi_4.f
-rw-rw-r-- 1 mary users  89 2005-01-18 11:47 pi.f
-rw-rw-r-- 1 john users 171 2005-04-05 16:04 pi.new.f
-rw-rw-r-- 1 john users 108 2005-03-01 16:05 pi-works_I
```

Will Mary and John progress much further in this mess?

They need **urgently** something like CVS

Project Tools: Version management CVS

CVS helps you managing your project versions

- keeps clean and visible current developing source files

```
mary@linux:˜ > ls -l
total 4
drwxr-xr-x 2 john users  48 2005-04-05 17:21 CVS
-rw-r--r-- 1 mary users 108 2005-04-01 16:02 pi.f
```

Project Tools: Version management CVS

CVS helps you managing your project versions

- keeps clean and visible current developing source files

```
mary@linux:~ > ls -l
total 4
drwxr-xr-x 2 john users  48 2005-04-05 17:21 CVS
-rw-r--r-- 1 mary users 108 2005-04-01 16:02 pi.f
```

- you should ‘commit’ modifications when done!

Project Tools: Version management CVS

CVS helps you managing your project versions

- keeps clean and visible current developing source files

```
mary@linux:~ > ls -l
total 4
drwxr-xr-x 2 john users  48 2005-04-05 17:21 CVS
-rw-r--r-- 1 mary users 108 2005-04-01 16:02 pi.f
```

- you should ‘commit’ modifications when done!
- you can ‘tag’ the whole project → Automatic unique version

Project Tools: Version management CVS

CVS helps you managing your project versions

- keeps clean and visible current developing source files

```
mary@linux:˜ > ls -l
total 4
drwxr-xr-x 2 john users  48 2005-04-05 17:21 CVS
-rw-r--r-- 1 mary users 108 2005-04-01 16:02 pi.f
```

- you should ‘commit’ modifications when done!
- you can ‘tag’ the whole project → Automatic unique version
- records all commited changes into a safe ‘Repository’ back-up

Project Tools: Version management CVS

CVS helps you managing your project versions

- keeps clean and visible current developing source files

```
mary@linux:~ > ls -l
total 4
drwxr-xr-x 2 john users  48 2005-04-05 17:21 CVS
-rw-r--r-- 1 mary users 108 2005-04-01 16:02 pi.f
```

- you should ‘commit’ modifications when done!
- you can ‘tag’ the whole project → Automatic unique version
- records all commited changes into a safe ‘Repository’ back-up
- no need to stamp yourself the files, use `$Id:$` instead!

Project Tools: Version management CVS

CVS helps you managing your project versions

- keeps clean and visible current developing source files

```
mary@linux:~ > ls -l
total 4
drwxr-xr-x 2 john users  48 2005-04-05 17:21 CVS
-rw-r--r-- 1 mary users 108 2005-04-01 16:02 pi.f
```

- you should ‘commit’ modifications when done!
- you can ‘tag’ the whole project → Automatic unique version
- records all commited changes into a safe ‘Repository’ back-up
- no need to stamp yourself the files, use `$Id:$` instead!

No magic: Requires a bit of discipline and the right policy

Project Tools: Compilation with Make

In ‘multiple files’ codes you need to know the rules about how the interactions between the elements are. This might be clear to the authors but not to other end-users.

Project Tools: Compilation with Make

In ‘multiple files’ codes you need to know the rules about how the interactions between the elements are. This might be clear to the authors but not to other end-users.

Getting used to Makefile is always a good idea!, you just

- Establish ‘targets’ which are your final goals
- Specify the ‘requisites’ on which your targets rely (recompilation)
- Give generic or specific ‘rules’ for compilation (commands)
- Easy handle the file names and script functions
- Delete unnecesary (intermediate) objects: `make clean`

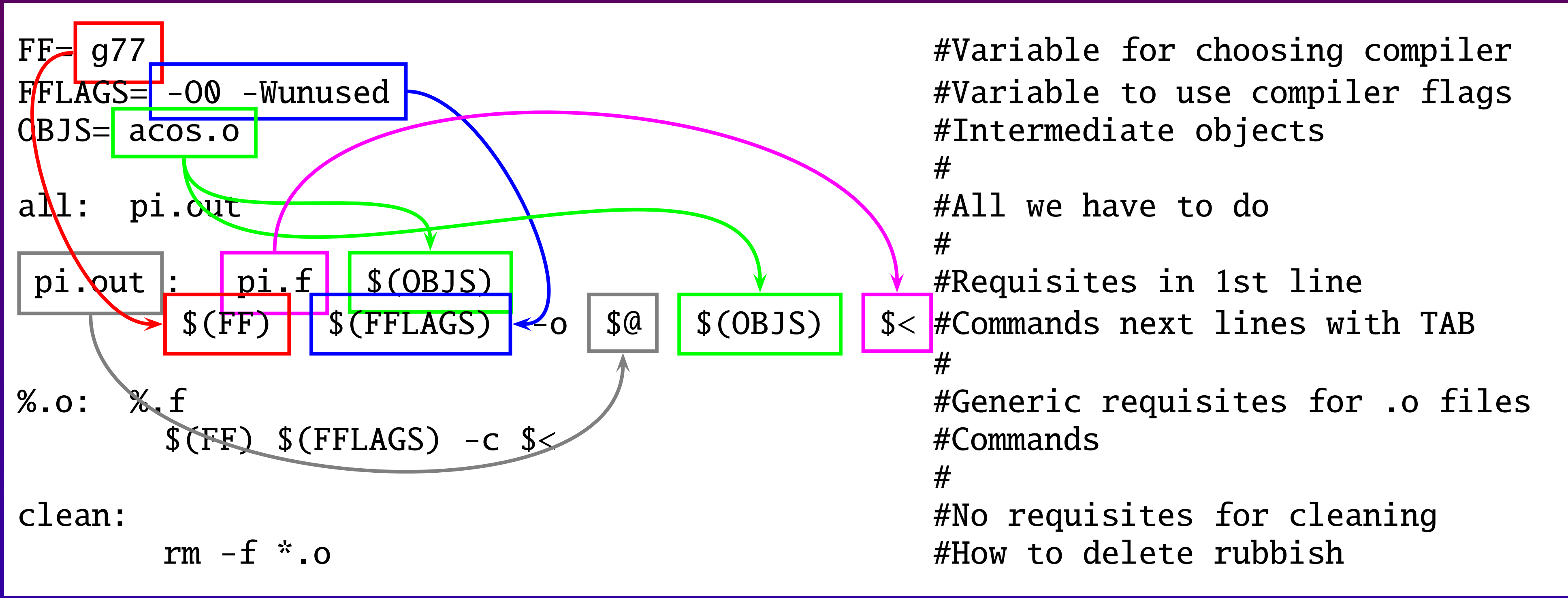
Project Tools: Compilation with Make

In ‘multiple files’ codes you need to know the rules about how the interactions between the elements are. This might be clear to the authors but not to other end-users.

```
FF= g77                                                #Variable for choosing compiler
FFLAGS= -O0 -Wunused                                   #Variable to use compiler flags
OBJS= acos.o                                           #Intermediate objects
                                                       #
all:  pi.out                                           #All we have to do
                                                       #
 pi.out :   pi.f   $(OBJS)                             #Requisites in 1st line
        $(FF)   $(FFLAGS)  -o  $@   $(OBJS)   $<       #Commands next lines with TAB
                                                       #
%.o:  %.f                                              #Generic requisites for .o files
        $(FF) $(FFLAGS) -c $<                          #Commands
                                                       #
clean:                                                 #No requisites for cleaning
        rm -f *.o                                      #How to delete rubbish
```

Project Tools: Compilation with Make

In ‘multiple files’ codes you need to know the rules about how the interactions between the elements are. This might be clear to the authors but not to other end-users.



```
FF= g77                                              #Variable for choosing compiler
FFLAGS= -O0 -Wunused                                 #Variable to use compiler flags
OBJS= acos.o                                         #Intermediate objects
                                                     #
all:  pi.out                                         #All we have to do
                                                     #
pi.out :  pi.f  $(OBJS)                              #Requisites in 1st line
        $(FF) $(FFLAGS) -o  $@  $(OBJS)  $<          #Commands next lines with TAB
                                                     #
%.o:  %.f                                            #Generic requisites for .o files
        $(FF) $(FFLAGS) -c $<                        #Commands
                                                     #
clean:                                               #No requisites for cleaning
        rm -f *.o                                    #How to delete rubbish
```

Looks difficult but you just have to practice!

Project Tools: Inter-operability with Autoconf

Do you usually get errors when installing other's codes? (Yes/No)

Typical avoidable errors:

- `Library 'libwhatever.a' not found`
- `invalid option -- X, Try --help for more information`
- `Version X.Y does not allow such operation`

Project Tools: Inter-operability with Autoconf

Do you usually get errors when installing other's codes? (Yes/No)

Typical avoidable errors:

- `Library 'libwhatever.a' not found`
- `invalid option -- X, Try --help for more information`
- `Version X.Y does not allow such operation`

With AUTOCONF you have a chance of reducing configuration errors by checking which libraries are needed, optimal flags, minimal version

↑ Ideal if you work in C or C++

↓ Adaptation tests require a bit of shell scripting

Hint: Have a look to a 'unorthodox' `configure.in` coming with *a*ÎTALC

III. DIANA and *a*ITALC

Diana: Feynman Diagram Analizer

Developed at U.Bielefeld 1997-2004 (Fleischer and Tentyukov)

Diana: Feynman Diagram Analizer

Developed at U.Bielefeld 1997-2004 (Fleischer and Tentyukov)

- C program, based on Nogueira's FORTRAN generator QGRAF2
- Command line: requires a driver file and model file
- High portability, running in many UNIX systems
- Front-end topology editor (tedi) included for GNU/LINUX

`http://www.physik.uni-bielefeld.de/~tentukov/diana.html`

Diana: Feynman Diagram Analizer

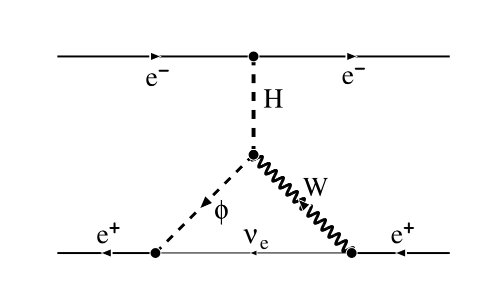
Developed at U.Bielefeld 1997-2004 (Fleischer and Tentyukov)

What do we ask?

```
SET _processname = Bhabha
\Begin(model,EWSM.model)
\Begin(process)
ingoing le(;p1),Le(;p4);
outgoing le(;-p2),Le(;-p3);
loops = 1;
options = onshell,notadp;
*\excludevertex(Le,le,H)
SET MakeEps = "!"
...
```

What does Diana answer?

Bhabha626.eps



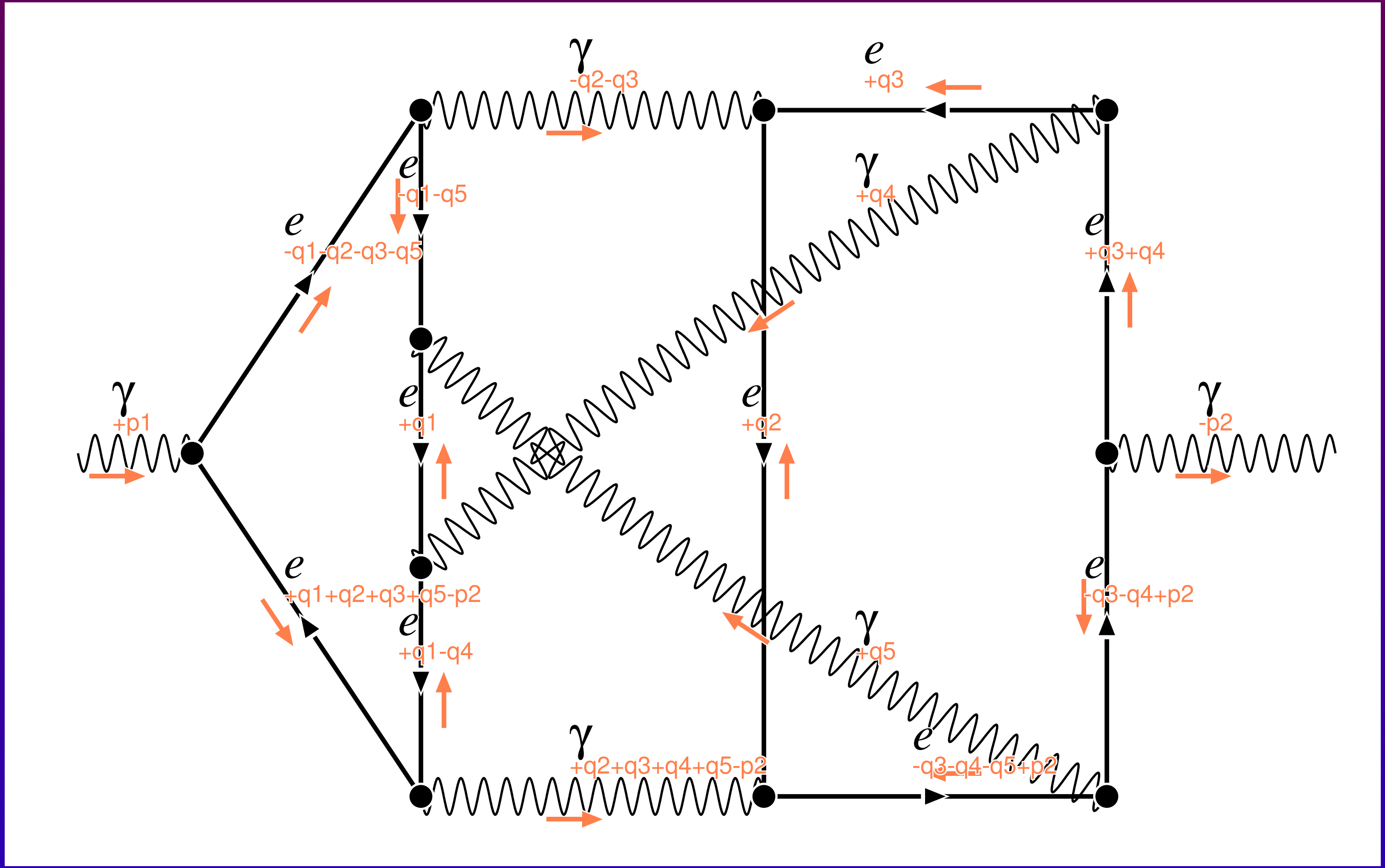
```
G Amplitude =
(-1)*F(1,1,1,0,0)*(-i_)*e/2/sw*Mle/MW*F(2,2,1,-1,0)*
(-i_)*e/2/sqrt2/sw*Mle/MW*FF(3,2,+q,Mne)*i_*
F(3,2,mu1,1,-1,1)*(+i_)*e/2/sqrt2/sw*SS(4,0)*i_*
SS(1,2)*i_*VV(2,mu2,mu1,-q-k2,2)*i_*
V(4,mu2,+p1+p2-(+q+k1),1)*(-i_)*e/2/sw;
```

```
#define COUNTER "626" #define LINE "4"
#define LOOPTYPE "c" ...
```

Diana: Feynman Diagram Analizer

Developed at U.Bielefeld 1997-2004 (Fleischer and Tentyukov)

Do you imagine doing 5-loop QED calculations?



Computing: aITALC

an Integrated Tool for Automated Loop Calculations

Computing: aITALC

an Ịntegrated Tool for Automated Loop Calculations

- Restricted to automated 2 → 2 fermions (EWSM and QED)
- GNU/LINUX tool, GPL licensed, free available since 29.10.04
- `http://www-zeuthen.desy.de/theory/aitalc`
- Submitted to CPC: Lorca and Riemann. `hep-ph/0412047`

Computing: aITALC

an I̊ntegrated Tool for Automated Loop Calculations

- Restricted to automated 2 → 2 fermions (EWSM and QED)
- GNU/LINUX tool, GPL licensed, free available since 29.10.04
- `http://www-zeuthen.desy.de/theory/aitalc`
- Submitted to CPC: Lorca and Riemann. `hep-ph/0412047`

Three structural blocks:

Diagram generation	Algebra simplification	Numerical evaluation
DIANA 2.35 (QGRAF)	FORM 3.1	FORTRAN 77 LOOPTOOLS (FF)

Computing: aITALC

an I̊ntegrated Tool for Automated Loop Calculations

- Restricted to automated 2 → 2 fermions (EWSM and QED)
- GNU/Linux tool, GPL licensed, free available since 29.10.04
- `http://www-zeuthen.desy.de/theory/aitalc`
- Submitted to CPC: Lorca and Riemann. `hep-ph/0412047`

Three structural blocks: all running under Make environment

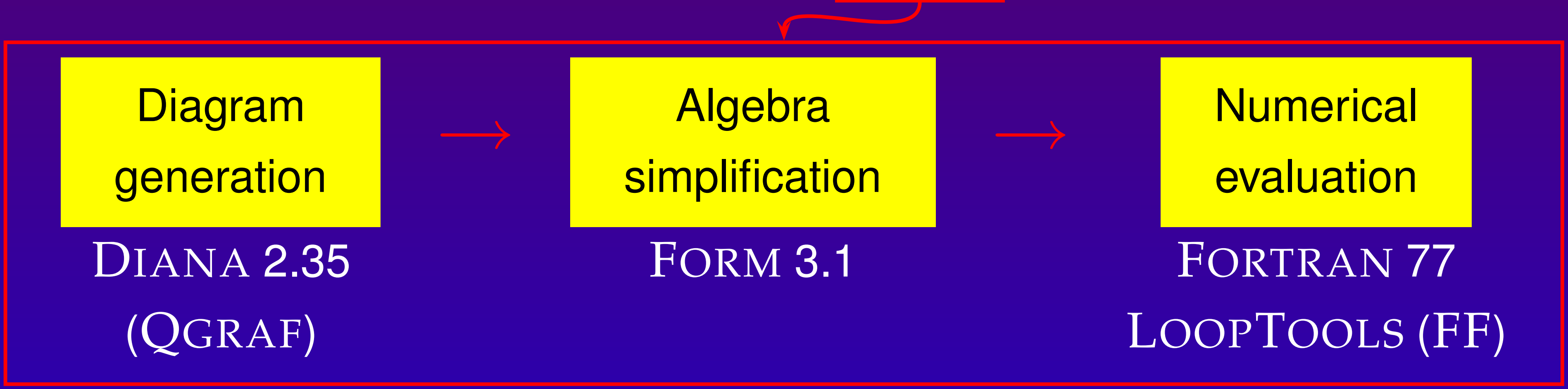


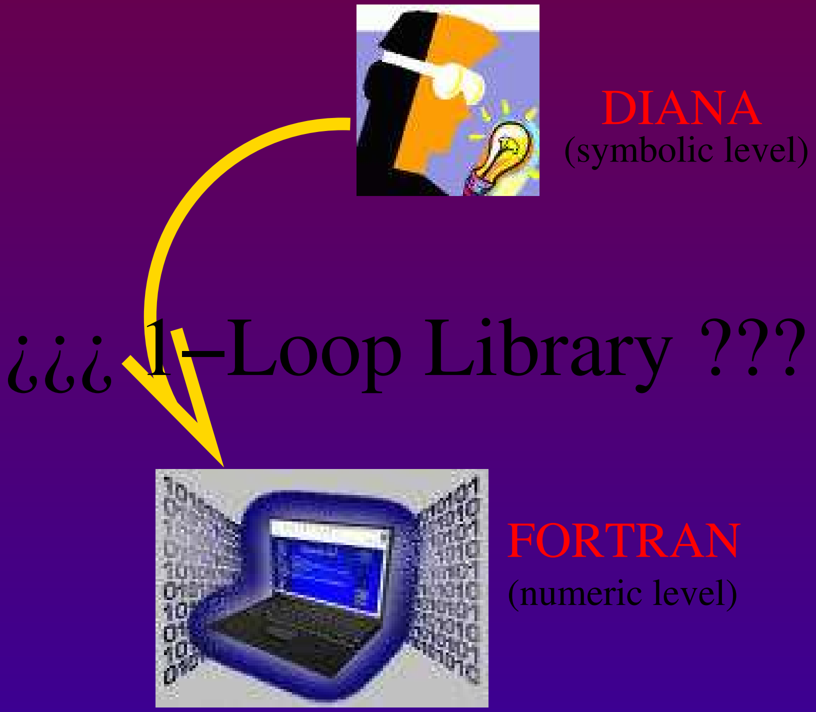
Diagram generation → Algebra simplification → Numerical evaluation

Diana 2.35 (Qgraf)

Form 3.1

Fortran 77 LoopTools (FF)

Computing: aITALC algebra



DIANA
(symbolic level)

¿¿¿ 1–Loop Library ???

FORTRAN
(numeric level)

Written in FORM

```
#call feynmanrules()
...
#call tracefermiloops()
#call integration()
#call chisholm()
#call dimensionfour()
#call gammaalgebra()
#call onshell()
#call diracequation()
#call massiveformfactors()
.end
```

These general procedures perform all algebra simplification

✓ Write automatically FORTRAN subroutines from DIANA output

Computing: aITALC numerical

For numerical evaluation language FORTRAN 77 is used

Computing: aITALC numerical

For numerical evaluation language FORTRAN 77 is used

The code is decomposed into different routines

- **Local**: Process-dependent automatically generated (`me, ff`)
- **Global**: Fixed coming with the distribution (`renorm.`)
- **External**: LOOPTOOLS package (evaluation of loop integrals)

Computing: aITALC numerical

For numerical evaluation language FORTRAN 77 is used

The code is decomposed into different routines

- **Local**: Process-dependent automatically generated (`me, ff`)
- **Global**: Fixed coming with the distribution (`renorm.`)
- **External**: LOOPTOOLS package (evaluation of loop integrals)

Executable file $\boxed{\texttt{main.out}}$

Input → parameter list, control flags.

Output ← tables for differential and integrated cross sections and forward-backward asymmetries

Tests ✔ ultraviolet and infrarred finiteness against parameter variation. Quadruple precision

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - CVS for managing your code versioning

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - CVS for managing your code versioning
  - Makefile to compile

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - CVS for managing your code versioning
  - Makefile to compile
  - Autoconf to adapt your code to the machine configuration

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - ► CVS for managing your code versioning
  - ► Makefile to compile
  - ► Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - ► CVS for managing your code versioning
  - ► Makefile to compile
  - ► Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.
  - ► Fast and portable

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - ▶ CVS for managing your code versioning
  - ▶ Makefile to compile
  - ▶ Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.
  - ▶ Fast and portable
  - ▶ Difficult to modify and not well documented

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - ▶ CVS for managing your code versioning
  - ▶ Makefile to compile
  - ▶ Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.
  - ▶ Fast and portable
  - ▶ Difficult to modify and not well documented
- *a*I̊TALC offers a integrated solution from symbols to numerics

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - ► CVS for managing your code versioning
  - ► Makefile to compile
  - ► Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.
  - ► Fast and portable
  - ► Difficult to modify and not well documented
- *a*I̊TALC offers a integrated solution from symbols to numerics
  - ► Free, Documented, Installable, Automated, Run!

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - CVS for managing your code versioning
  - Makefile to compile
  - Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.
  - Fast and portable
  - Difficult to modify and not well documented
- $a\mathring{\text{I}}$TALC offers a integrated solution from symbols to numerics
  - Free, Documented, Installable, Automated, Run!
  - Limitation by now to 2 → 2 fermions in EWSM

Conclusions & Outlook

- Interesting tools to handle software projects (Want more? RTFM)
  - CVS for managing your code versioning
  - Makefile to compile
  - Autoconf to adapt your code to the machine configuration
- DIANA is a powerful tool, intended to deal with many Feynman diagrams and their analyses.
  - Fast and portable
  - Difficult to modify and not well documented
- aITALC offers a integrated solution from symbols to numerics
  - Free, Documented, Installable, Automated, Run!
  - Limitation by now to 2 → 2 fermions in EWSM
  - Part of contributions required for precise collider predictions (hard photon, QCD, kin. cuts . . .)

IV. Play a bit!

`~alorca/public/capp_examples`