De JurassicFortran à Fortr'&Furious

Vincent Lafage

[1]IJCLab, Laboratoire de Physique des 2 Infinis Irène Joliot-Curie
Université Paris-Saclay

12 juillet 2023

UrFortran *Propédeutique du Fortran...*

- 1954 : *Automatic Coding System for the IBM 704* dated October 15, 1956, first presented in February 1957 and first delivered in April 1957
- FORmula TRANslator : langage de haut niveau gérant la première abstraction, l'abstraction des expressions :
  Le compilateur transforme les expressions en arbre d'évaluation
- pas le plus ancien compilateur, mais presque, et en tous cas le plus ancien à subsister
- le plus ancien compilateur optimiseur
- format fixe (cartes perforées)
- orienté calcul et pas système (relativement haut niveau = abstraction vis-à-vis du modèle mémoire)
- peu de types économie de moyen déclarations implicites

Don't piss off old people : the older we get,the less life in prison is a deterrent

Historique

- 1966, *a.k.a.* `FORTRAN IV` première standardisation (ANSI X3.9-1966)
  (paléo FORTRAN)
- 1977 (ANSI X3.9-1978 FORTRAN, ISO 1539 :1980) influence de la programmation structurée :
  (archéo FORTRAN)
  paradigme **procédural** langage de haut niveau gérant la deuxième abstraction, l'abstraction du flux de contrôle
- 1989 `f2c` premier traducteur open source de `FORTRAN` vers `C`
- 1988 ? Rendez-vous manqué avec l'histoire
- 1991 `FORTRAN` ⇒ `Fortran`
  `Fortran 90` : langage de haut niveau gérant la troisième abstraction, l'abstraction des données
  paradigme **modulaire** + gestion des types dérivés `struct/record`
- 1995, Feb 17 : `g77` premier compilateur open source de `FORTRAN`
- *Free Format*
- 1997 `Fortran 95` : allocation dynamique
- early 2000—2006 : `g95` premier compilateur open source de `Fortran`
- 2003, Jan—2005, Apr 20 : `gfortran` fork de `g95`

Missing In Fortran

- héritage multiple ⇒ Facade Pattern
- first-class functions ⇒ `real, external, pointer :: f_ptr`.
  Alternative : `procedure (f), pointer :: f_ptr`
  `https://gcc.gnu.org/onlinedocs/gfortran/Working-with-C-Pointers.html`
- gestion d'exception (au-delà des exceptions IEEE 754)
- `assert`
- généricité (polymorphisme paramétrique) ⇒ `Fortran202Y`
  `https://github.com/j3-fortran/generics/tree/main`
- `const` ? (≠ `parameter` ∼ `consteval`) ⇒ …, `intent (in) ::` ou …, `protected ::`
- `C` binding to variadic functions
- coroutines
- …

BASIC programming is not basic programming

Fortran programming is not FORTRAN programming

Aspect sociolinguistique

- Style (supposé) de FORTRAN :
  - format fixe
  - pas d'indentation
  - ALL CAPS !
  - identifiant trop courts (6 caractères)
  - typage implicite
  - répétitions des déclarations et des COMMON
  - fonctions trop longues
  - GOTO à outrance, GOTO arithmétique, code spaggheti
- ⇒ AdaTran (DOI:10.1145/126551.126611) : de l'Ada écrit (supposément) comme du FORTRAN

https://jackkrupansky.medium.com/what-is-adatran-or-adatran-or-adatran-6bab09c44b93

1. The use of the Ada programming language as if it were the FORTRAN programming language.
2. Coding in the syntax of Ada with the style and semantics of FORTRAN.
3. Ada code written by a FORTRAN programmer.
4. A subset of Ada taught to FORTRAN programmers.
5. Ada code generated by an automated translation of FORTRAN code.
6. Ada code generated by automated translation of any non-Ada programming language.
7. The use of any advanced technology as if it were an older technology with which one is more familiar and experienced.

Paradigme procédural/structuré

- `GOTO label … label:`
- `IF (…) THEN… ELSE… ENDIF`
- ```
  SELECT CASE (selector-expression)
  CASE (label-list-1)
     statements-1
    ..............
  CASE (label-list-n)
     statements-n
  CASE DEFAULT
     statements-DEFAULT
  END SELECT
  ```
- `DO variable = from, to{, step} … ENDDO`
- `DO … ENDDO`
- `DO WHILE (condition) … ENDDO`
- `EXIT {construct-name}`
- `CYCLE {construct-name}`
- `CONTINUE`
- Fortran 2008

```
{label:} BLOCK
          ...lines of code...
         END BLOCK
```
```

Harnessing Array la chasse aux boucles est ouverte

- opérations sur tableaux : $+$, $-$, $*$ (HADAMARD), / (*element-wise*)
- applications de fonctions sur tableaux (mapping)
  - les fonctions intrinsèques
  - les fonctions `ELEMENTAL` : `PURE` $+$
- réduction de tableaux : `SUM`, `PRODUCT(ARRAY, DIM, MASK)`
- `DOT_PRODUCT(VECTOR1, VECTOR2)`
- `MATMUL`
- `RESHAPE`
- ...
- `FORALL (variable = from:to{:step})`
- `DO CONCURRENT (variable = from:to{:step})`

Build

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

```
#!/usr/bin/make
SHELL = /bin/sh
MAIN ?= base_gray_scott_from_make_90

# FC = the Fortran compiler to use
FC ?= gfortran

ifeq ($(FC), ifx)
  MAIN:=$(MAIN)_ifx
  OBJ = objintel
  FFLAGS = -g -traceback -check bounds -warn all -Wextra -Wpedantic -Wconversion -Wshadow
else ifeq ($(FC), nvfortran)
  MAIN:=$(MAIN)_nv
  OBJ = objnv
  FFLAGS = -g -Wall -Wextra -Werror
else # ifeq ($(FC), gfortran)
  MAIN:=$(MAIN)_gnu
  OBJ = obj
  FFLAGS = -g -Wall -Wextra -Wpedantic -Wconversion -Wshadow -Werror -fimplicit-none -finit-real=s
endif

ARCHI := $(shell LC_ALL=C uname -m)
HDF_INCLUDES   = /usr/include/hdf5/serial # package libhdf5-dev
HDF_LIB    = /usr/lib/$(ARCHI)-linux-gnu/hdf5/serial # package libhdf5-dev libhdf5…-103 / libhdf5-1
```

Build

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

```
…
#DEFINES       = -DVERSION=\'$(shell LC_ALL=C hg id -n):$(shell LC_ALL=C hg id -i)\' -DLASTDATE=\'
#DEFINES       = -DVERSION=\'$(shell LC_ALL=C hg tip --template '{parent}'|sed -e 's/ //')\' -DLAS
DEFINES       = -DVERSION=\'$(shell LC_ALL=C git describe --abbrev=12 --always --dirty=+)\' -DLAST
CPPFLAGS      = $(DEFINES) -DCPP_VERSION=\'$(CPP_VERSION)\'
CPPFLAGS     += -I$(HDF_INCLUDES)

LDFLAGS=$(FFLAGS)
LDLIBS=-L$(HDF_LIB) -lhdf5

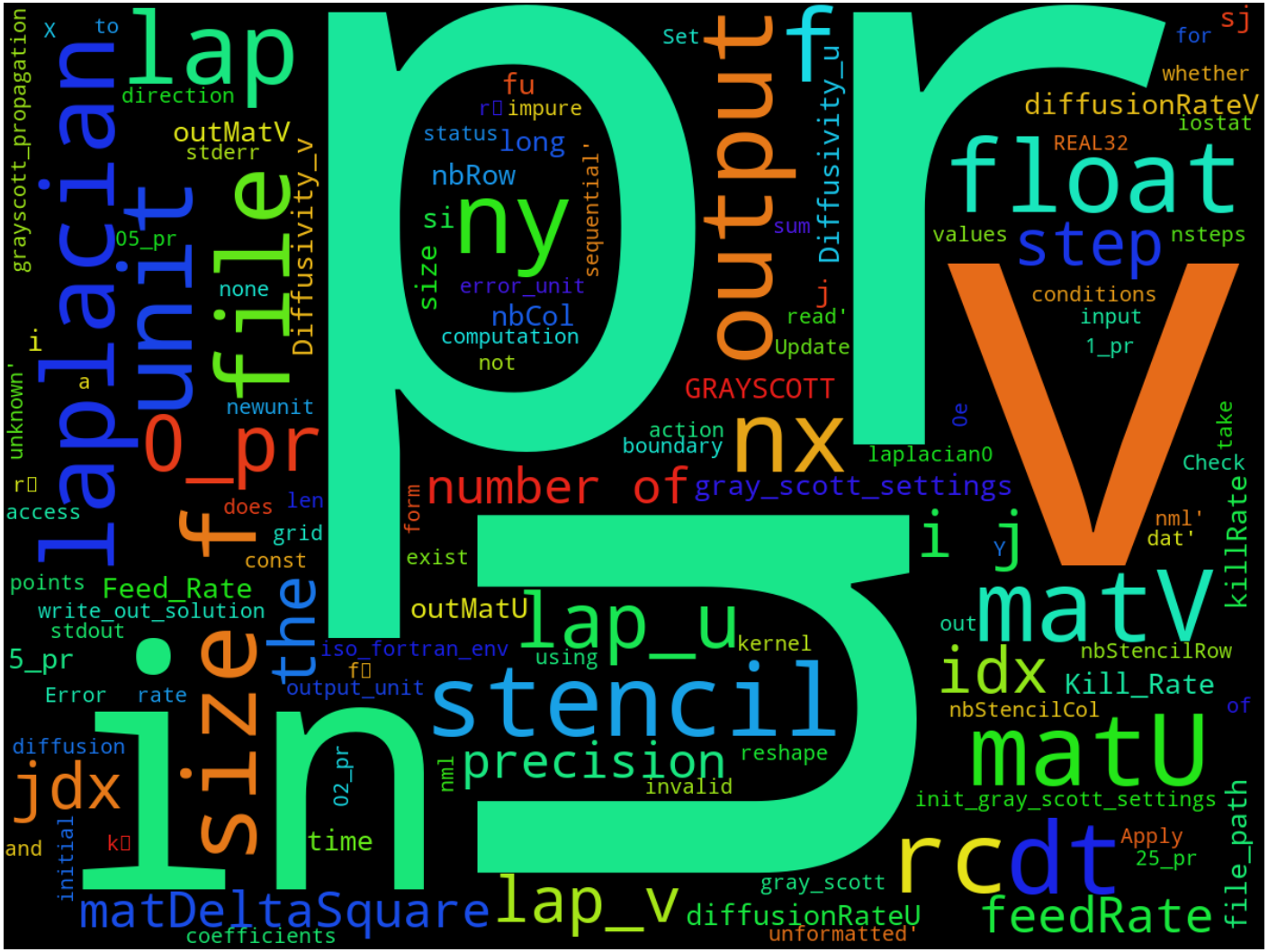
SOURCES=gray_scott.f90
PRJS= $(SOURCES:.f=.prj)
```

Build

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

```
lafage@ll-lafage:~/GrayScott$ make help
# target: base_gray_scott_from_make_90 - build main executable (default target)
# target: help - Display callable targets.
# target: clean - clean all intermediate build files
# target: clobber - clean all intermediate build files, executable and coredump
# target: display - timed execution of main executable applied on data
# target: run - timed execution of main executable applied on data
# target: perf - perf evaluation on monocore execution of main executable applied on data
# target: maqao - vectorization analysis of main executable
# target: memcheck - memchecked execution of main executable applied on data
# target: callgrind - profiled execution of main executable applied on data
# target: cachegrind - profiled execution of main executable applied on data
# target: doc - Doxygen documentation from literate source code.
# target: check - static check of Fortran source code (linter).
# target: sloccount - SLOC count of Fortran source code.
# target: wordcloud - display wordcloud of Fortran source code.
# target: validate_typo - check number of typo mistakes ridden files, by category.
```

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto



Build

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

```
…
# target: base_gray_scott_from_make_90 - build main executable (default target)
$(MAIN): $(SOURCES) $(OBJ)
	cd $(OBJ) ; $(FC) $(LDFLAGS) $(CPPFLAGS) ../$< -o ../$@ $(LDLIBS) ; cd ..

$(OBJ):
	mkdir -p $(OBJ)

# target: help - Display callable targets.
help:
	@egrep "^# target:" $(MAKEFILE_LIST)

# target: clean - clean all intermediate build files
clean:
	$(RM) -f *.o *.mod

# target: clobber - clean all intermediate build files, executable and coredump
clobber:
	$(RM) *.o *.mod core $(MAIN)

# target: display - timed execution of main executable applied on data
display: $(MAIN)
	strings ./$<|grep -E '(build_date|revision_date|revision|compiler_version| version):'

# target: run - timed execution of main executable applied on data
run: $(MAIN)
	time ./$<

# target: perf - perf evaluation on monocore execution of main executable applied on data
perf: $(MAIN)
```

Source

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

```
module precision
  use, intrinsic :: iso_fortran_env, only: pr => REAL32
end module precision

module gray_scott_settings
  use precision
  integer, parameter ::          &
   nx = 960,                     & ! number of grid points in X direction
   ny = 540,                     & ! number of grid points in Y direction
   nsteps = 3400,                & ! number of time steps to take
   input  = 10,                  & ! logical unit number of output file
   output = 11                     ! logical unit number of output file
  real (pr) ::                   &
    dt = 0.1_pr,                 & ! time step
    Diffusivity_u = 2.0e-5_pr,   & ! _ur: diffusion rate of u
    Diffusivity_v = 1.0e-5_pr,   & ! _vr: diffusion rate of v
    Feed_Rate = 0.02_pr,         & ! _rf
    Kill_Rate = 0.05_pr            ! _rk

  ! Stencil coefficients
  real (pr), dimension (-1:1, -1:1), parameter :: &
    stencil = reshape ([ real (pr) :: 0, 1, 0, 1, -4, 1, 0, 1, 0], [3, 3])

  namelist /GRAYSCOTT/ dt, Diffusivity_u, Diffusivity_v, Feed_Rate, Kill_Rate

contains
```

Source

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

uvrr

```
  subroutine init_gray_scott_settings
    use, intrinsic :: iso_fortran_env, only: stderr => error_unit, stdout => output_unit
    character (len=*),  parameter     :: file_path = 'gray_scott_settings.nml'
    integer                           :: fu, rc
    ! Check whether file exists.
    inquire (file=file_path, iostat=rc)
    if (rc /= 0) then
        write (stderr, '("Error: input file ", a, " does not exist")') file_path
        return
    end if
    ! Open and read Namelist file.
    open (action='read', file=file_path, iostat=rc, newunit=fu)
    read (nml=GRAYSCOTT, iostat=rc, unit=fu)
    if (rc /= 0) write (stderr, '("Error: invalid Namelist format")')
    write (nml=GRAYSCOTT, unit=stdout)
    close (unit=fu)
  end subroutine init_gray_scott_settings

  function laplacian (f, stencil) result (lap)
    real (pr), dimension (:, :), intent (in) :: f
    real (pr), dimension (-1:1, -1:1), intent (in) :: stencil
    real (pr), dimension (size (f, 1), size (f, 2)) :: lap
    integer :: i, j, si, sj

    lap = 0.0_pr

    ! Apply the stencil kernel for the laplacian computation
```

Source

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

uvrr

```
    do j = 2, size (f, 2) - 1
      do i = 2, size (f, 1) - 1
        do sj = -1, 1
          do si = -1, 1
            lap (i, j) = lap (i, j) + stencil (si, sj) * f (i+si, j+sj)
          end do
        end do
      end do
    end do
  end function laplacian

  pure subroutine init_fields (u, v)
    real (pr), dimension (:, :), intent (inout) :: u, v
    ! Set initial conditions
    u = 1.0_pr
    v = 0.0_pr
    u (nx/2:nx/2 + 15, ny/2:ny/2 + 15) = 0.5_pr
    v (nx/2:nx/2 + 15, ny/2:ny/2 + 15) = 0.25_pr

    ! Set boundary conditions
    ! u (:, 1) = 0.5_pr
    ! v (:, 1) = 0.25_pr
  end subroutine init_fields
```

Conf

https://gitlab.in2p3.fr/lafage/GrayScottFortranTuto

```
&GRAYSCOTT
 DT= 1.0,                  ! time step
 DIFFUSIVITY_U=  0.1,      ! ur: diffusion rate of u
 DIFFUSIVITY_V=  0.05,     ! vr: diffusion rate of v
 FEED_RATE=  0.014,        ! rf
 KILL_RATE=  0.054,        ! rk
 /
```