Introduction to computational plasticity using FORTRAN

Youngung Jeong

Changwon National Univ.

FORTRAN

- FORMULA TRANSLATION
- Compile, Object Files, Executable
- Much faster than other advanced languages
- The contents you'll learn
  - Algorithm
  - Numerical approach for plasticity theories
  - Other numerical techniques

REF: https://web.stanford.edu/class/me200c/

Installation

- GNU FORTRAN is recommended for beginners.
  - Available open to everyone (free)
  - I am quite familiar with (~10 years of experience)
- Intel FORTRAN is recommended as well.
  - But it is a commercial product; you should pay.
  - Nevertheless, if you are a student, you could request for a license for a year for free.
  - It could be faster than GFORTRAN (when optimized)

Choice of your editor

- Whatever tickles your fancy.
- Some might claim "Syntax highlighter is for newbies." If you agree, you could use:

  c:\> notepad <path-to-file-name>
- to which I do not agree. There are many advantages of using advanced editor when you are programming. Syntax highlighter generally helps reducing 'errors'.

- I personally use EMACS, which I **DO NOT** recommend my students to learn - it takes a lot of efforts and time to get used to it.
  - (*Unlike* many other cases, once you get used to it, you'll love it)
- For Windows users, I'd recommend
  - NOTEPAD ++ (https://notepad-plus-plus.org),
  - ATOM (https://atom.io)
  - sublime text (https://www.sublimetext.com) …
- For those who look for more advanced features ..
  - EMACS (https://www.gnu.org/software/emacs)
  - VIM (https://www.vim.org)and so forth

Compile and Linking

Translating a source code (to a machine code then) to an **object file**

Linking objects files to build your '**executable**' program

Compile and Linking

Compile:

c:\> gfortran -c <path-to-file> -o <path-to-object>

Linking:

c:\> gfortran –o <path-to-executable> <path-to-object 1> … <path-to-object n>

Example:

Suppose you wrote a program consisting of two files <a.f> and <b.f>. Below sequence should be performed in order for you to obtain an executable file <c.exe>:

c:\> gfortran –c a.f –o a.o

c:\> gfortran –c b.f –o b.o

c:\> gfortran –o c a.o b.o

COLUMN position rules

- COL.1 : BLANK or ‘C’ or “*” for comments
- COL.1-5 : statement label (reference to a specific statement line; optional)
- COL. 6. : continuation of previous line (optional)
- COL 7-72: statements

- Comments can be written by two methods:
  - 1) Use COL 1 rule
  - 2) Use ! symbol to comment out the following cols.

Your first FORTRAN program

```
program hi
write(*,*) “Hello, world”
end program
```

1. Let’s save this file to <hi.f>
2. Compile it
3. And … run!

```
c:\> gfortran –c hi.f –o hi.o
c:\> gfortran –o hi hi.o
c:\> hi
“Hello, world”
```

Tip: case-sensitivity applies only to character variables.

Make (optional for advanced students)

- Make programs are usually to reduce the time required for ‘building’ the program
- Some programs may require a lot number of objects
- In many cases, only a few objects among many are revised when ‘incrementally’ developing program
- Make programs are useful for such occasions

https://www.gnu.org/software/make/

Types

- INTEGER (정수)
- REAL (실수)
- COMPLEX (복소수)
- LOGICAL (Boolean, 참 .true. 혹은 거짓 .false.)
- CHARACTER (문자)

Constants

- 1 0 -100 + 327 +15
- 1.0 1. -0.25 2e6 -5.3e-4
- -5.d-4 -5.e-4
- .true. .false.
- 'ABS' "ABC" "1!@" "Hello, world"

Expressions

- OEPRAND(피연산수) OPERATOR(연산자) OPERAND

Ex) *x* + *y*

Types of operator

* (multiplication, 곱하기)
+ (addition, 더하기)
- (subtraction, 빼기)
/ (division, 나누기)
** (exponential, 지수)

Assignment

- Variable = expression
- X = 5*1e-5+3.**2
- Y = (X+X**2)/3.
- Y = (Y+2)

- RULE: Evaluate on the right first, then assign the result to the left.
- In that sense, the equating symbol (=) is more like an arrow (←)

Logical expressions

A .gt. B (or, A>B)

A .ge. B (or, A>=B)

A .lt. B (or, A<B)

A .le. B (Or, A<=B)

A.eq.B (or, A==B)

A.ne.B (or, A!=B)

Others logical expressions with combination of

.and.

.or.

Examples:

(A.eq.B).or.(A.eq.C)

(A.eq.C).and.(C.eq.D)

IF statement

- Following format:

```
IF (a logical expression) THEN
    statements A
ELSEIF (another logical expression) then
    statements B
ELSE
    statements C
ENDIF
```

DO-END Loops

```
DO i=1, n
   statements …
ENDDO
```

- Other loops:

```
  DO WHILE (logical expression)
      statements …
  ENDDO
```

Arrays

- One dimensional
- REAL A(20) 1 axis, 20 slots
- REAL B(20,3) two axes, with 20 and 3 slots
- INTEGER C(0:19,3,4) three axes, 20, 3, and 4 slots (total 20x3x4 slots)

Sub programs

- FUNCTION
- SUBROUTINE

INTRINSINC functions

- FUNCTIONS that come along with FORTRAN
- Examples:
  - ABS
  - MIN
  - SQRT
  - SIN
  - COS
  - ASIN
  - ACOS
  - ATAN
  - ATAN2
  - LOG
  - EXP …

WRITING your own functions (EX)

```
REAL FUNCTION addition(a,b)
real a, b
addition =a +b
return
END FUNCTION
```

Main program to use your function

main.f

```
PROGRAM main
real addition
real  a,b,c
a=3.
b=203.3
c=addition(a,b)
Write(*,*) c
End
```

add.f

```
REAL FUNCTION addition(a,b)
real a, b
addition =a +b
return
END FUNCTION
```

C:\> gfortran –c main.f –o main.o

C:\> gfortran -c add.f –o add.o

C:\> gfortran –o myprogram main.o add.o

C:\> myprogram

C:\> gfortran –o myprogram main.f add.f

Consider writing a batch file

- Compile and execution is usually done quite repeatedly, while all the tasks requires you type in command line prompt.
- In order for you to boot your speed and reduce the tedious typing of the same commands, I would recommend you to write your sequence of commands to a batch file.

FILE I/O

- OPENING/CLOSING a file
- OPEN(list-of-specifiers)
- UNIT, FILE, STATUS, …

Example (OUTPUT):

```
OPEN(30, FILE='dummy.txt',status='unknown')
WRITE(30, *) 'Hello, world'
CLOSE(30)
```

Example (INPUT):

```
OPEN(3, FILE='list-of-integers.txt', status='old')
READ(3,*) i
WRITE(*,*) i     ! Print the integer to screen (standard output)
CLOSE(3)
```

SIMPLE I/O

- READ(*,*) variable (or constant)
- WRITE(*,*) variable (or constant)

- In combination with WRITE(*,*), you might want to consider formatting the output..
  - Example:
  - WRITE(*,'(3f7.2)') 3.,4.,5.

I/O Format

- F
- E
- I
- A
- X
- Combination of the above…