Fortran

Tomáš Kroupa

20. května 2014

Abstrakt

Zde uvedeme návod k instalaci a řadu tipů a triků a řešení zapeklitých drobností. Předem připravme čtenáře na to, že bez zkoušení si vlastních subroutin a hledání na jiných místech, není pravděpodobně možnost vše níže napsané pochopit. Proto, když něco nepochopíte, nevztekejte se, že je to špatně napsané, věřte, že autor na všechno musel přijít sám. A to uznejte, že takto „špatně“ napsaný pomocný text je i tak dar z nebes;-)!

Tento studijní materiál je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

Obsah

1 Instalace **2**

2 Datové typy **4**

3 Struktura subroutin **4**

4 Common block **8**

5 Tisk a čtení textových souborů **9**

Tento studijní materiál je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

1 Instalace

Instalace Fortranu 2011 je popsána níže. Jedná se o kompiler, který má nakoupený KME. Návod k instalaci je převzatý z našich informací na google disk. Systémové proměnné (PATH a LIB) lze nalézt tak, že kliknete levým myšítkem na nabídku start → pravým myšítkem na počítač → levým myšítkem na vlastnosti → pravým myšítkem na upřesnit nastavení systému → pravým myšítkem na Proměnné prostředí → pak si vytvoříte nebo najdete stávající systémovou proměnnou a upravíte. Občas se hodí si celý obsah proměnné zkopírovat do textového souboru pomocí ctrl+c a ctrl+v, tam upravit a opět pomocí ctrl+c a ctrl+v vložit upravené zpět (jen pozor ať to zpět vkládáte bez enterů, musí se to vkládat jako jedna řádka). Postup pro instalaci je uveden ve zdrojových datech 1.

Zdrojová data 1: Instalace Fortran compileru.

```
Vsechno nutne k instalaci je na euleru.
Ve woknech namapovane treba na z:\sw\inst\Fortran\2011\Win.

Spustit soubor w_fcompxe_2011.1.127.exe

Pri instalaci vybrat variantu

1. Alternative instalation (nazev bez zaruky)
2. Instalovat s license file

Licence

Normalne by mel fungovat soubor lic.lic
(pokud ho system nebude chtit pouzit, tak pouzijte license.lic).

!V pripade potizi s licenci, pak je potreba ten licence.lic nahrat
    do slozky,
kde ho hledal napriklad Abaqus pri prekladu subroutiny!

Nasledne klasicky pridat cesty do path a lib

Celkem vzato normalne to funguje takto (win 7, 64bit):

PATH:
c:\Program Files (x86)\Intel\ComposerXE-2011\bin\intel64\;
C:\Program Files (x86)\Intel\ComposerXE-2011\redist\intel64\mkl;
C:\Program Files (x86)\Intel\ComposerXE-2011\redist\ia32\mkl;
C:\Program Files (x86)\Intel\ComposerXE-2011\redist\intel64\mpirt;
C:\Program Files (x86)\Intel\ComposerXE-2011\redist\intel64\
    compiler;
C:\Program Files (x86)\Intel\ComposerXE-2011\redist\ia32\mpirt;
C:\Program Files (x86)\Intel\ComposerXE-2011\redist\ia32\compiler;
C:\Program Files (x86)\Intel\ComposerXE-2011\compiler\lib;
C:\Program Files (x86)\Common Files\Microsoft Shared\VSA\9.0\VsaEnv
    ;

... vsechny ostatni cesty ...

c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\Bin\amd64\;
c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\Bin\;

LIB (Pravdepodobne bude treba vytvorit):
c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\Bin\amd64\;
c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\Bin\;
c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\PlatformSDK\
    Lib\x64\;
c:\Program Files (x86)\Intel\ComposerXE-2011\compiler\lib\intel64\;
c:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\Lib\amd64\
```

2 Datové typy

Narozdíl od Maltabu a konec konců i Pythonu je Fortran velmi striktní, co se týče definice a dodržování datových typů. Integer musí být integer, floating point (REAL) musí být floating point typ a navíc přesně definovaný. Například REAL není REAL*8, REAL je single precision a REAL*8 je double precision. V případě potřeby lze tvořit i ALLOCATABLE proměnné, tedy proměnné, kterým lze přiřadit velikost až během výpočtu. Jak to udělat a kdy se to hodí bych nechal na čtenáři a `www.google.com` :-).

Důležité informace 1: Základní zaklínadlo ve Fortranu při použití Abaqusu.

Abaqus na „Windows“ strojích a při zapnutí „full precision“ v Abaqus/CAE by měl využívat datový typ REAL*8 (DOUBLE PRECISION).

Lze definovat dokonce často používané konstanty jako parametr, odkažme na [**?**]. Detailnější informace ohledně datových typů ve Fortranu lze začít čerpat například na [**?**] nebo v [**?**]. Za každý další užitečný detail ohledně šikovné a vychytralé práce s datovými typy z průběhu vaší práce budou všichni vděční.

3 Struktura subroutin

V této kapitole si na několika ukázkách vysvětlíme, jak jsou subroutiny strukturované. Subroutina je v podstatě podprogram, který se přilinkuje ke stávající části Abaqusího řešiče. Dále se vše zkompiluje a následuje výpočet. Subroutin můžete používat kolik chcete, i své vlastní. V případě, že je to subroutina, kterou volá přímo Abaqus, téměř vždy se musí někde v Abaqus/CAE zakliknout, nebo zadat do .inp souboru řádek, že se ta daná subroutina má během výpočtu volat.

Jako ukázku vezměme subroutinu umat, ve které lze definovat vlastní materiálový model pro Abaqus Standard. Abaqus obsahuje ještě explicitní modul, v tom lze definovat vlastní materiál v subroutině vumat. Pro případ umat subroutiny se zakliknutí, neboli sdělení řešiči, že se má subroutina používat, provede výběrem materiálu jako user material. Dále je třeba nezapomenout na zadání počtu depvar (vždy studujte manuál a testujte a testujte vše a pořád!). A v neposlední řadě je také třeba zadat v definici jobu, kde lze nalézt textový soubor s příponou .for, ve kterém jsou zdrojové kódy subroutin sepsané.

Ve zdrojových datech 2 je vidět začátek subroutiny umat. První část obsahuje klíčové slovo SUBROUTINE a další je název subroutiny UMAT. Obojí může být jak malými, tak velkými písmeny - Fortran není asi vůbec „case sensitive“. Dále je zde vidět seznam proměnných, některé jsou jen vstupní, jiné mohou být výstupní a další vstupně-výstupní. Detailní popis toho, které proměnné musí a které mohou a které nemohou být definovány v příslušných subroutinách je vždy nutné zjisti v manuálu Abaqusu.

V téměř všech subroutinách pro Abaqus se vyskytuje řádka ukázaná ve Zdrojových datech 3. Tento kus kódu obsahuje jen klasické informace v Marcovském stylu, které jsou ukázané ve Zdrojových datech 4.

Zde je tedy jen řečeno, že všechny proměnné, které začínají písmeny od A do H a od O do Z pokud není uvedeno jinak budou typu REAL*8 a zbytek standardně integer. Co a jestli vůbec něco dalšího obsahuje se nepodařilo zatím

Zdrojová data 2: Začátek subroutiny umat.

```
      SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,
     2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
     3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
     4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)

      INCLUDE 'ABA_PARAM.INC'

      CHARACTER*80 CMNAME

      INTEGER ipvt
      DIMENSION ipvt(3)

      REAL*8 rcond,z
      DIMENSION z(3)

      REAL*8 STRESS,DDSDDE,SPD,PROPS,
     1 E,VU,SY0,FZT0,FES0,FES1,FEP0,FM,FN,
     2 eelas,eplas,eqplas,sred,dsred,
     3 syield,dsyield,yieldstress,
     4 fv,tolerance,residuum,error
     5 FRK,dFRK,FRMK,FB,FD,FXT,FXK,FY,
     6 gaussmatrix,FDD,FBFD,FYMFBFD,FYMFBFDFDT,
     7 FBA,FMKA

      DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
     2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
     3 PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3),
     4 FRK(3),dFRK(3,3),FRMK(3),FB(3,3),FD(3),FXT(3),FXK(3),FY(3),
     5 gaussmatrix(3,4),FBFD(3),FYMFBFD(3),FYMFBFDFDT(3,3),
     6 FBA(3,3),FMKA(3)

      ... ZDROJOVY KOD programu ...

      RETURN
      END
```

Zdrojová data 3: „INCLUDE“ řádka v Abaqus.

```
      INCLUDE 'ABA_PARAM.INC'
```

Zdrojová data 4: „INCLUDE“ řádka v MSC.Marc.

```
      IMPLICIT REAL *8 (A-H,O-Z)
```

vygooglit. Dále popišme více ukázku ze subroutiny umat (Zdrojová data 2). Zde je vidět hlavička subroutiny, spousty vstupních nebo výstupních proměnných, Include část, definice datového typu string (CHARACTER) a další definice použitých proměnných, pak následuje samotný kód programu a na závěr musí být subroutina ukončena návratovou hláškou RETURN a slovem END. RETURN způsobí, že se hodnoty parametrů subroutiny zase pošlou z ní zpět do nadřazeného programu, ze kterého byla subroutina volána. Toto lze udělat i jinak, v některých situacích výhodněji, proto odkažme na [**?**].

Důležité informace 2: Maximální počet znaků identifikujících promněnnou ve Fortranu.

Maximální počet znaků identifikujících promněnnou může být 31!

Důležité informace 3: Prázdné znaky na začátku řádky.

Prvních 6 znaků na každé řádce jsou mezery! Až poté začíná zdrojový kód! Vyjímky jsou, že první znak není mezera a to znamená, že na řádce je komentář. Dále, že šestý znak je číslo, nebo hvězdička, nebo asi fungují i jiné znaky, to znamená, že předchozí řádek pokračuje. Také lze dát na poslední prázdné znaky na začátku řádek návěští a na něj se z programu odkazovat. Za tímto návěštím musí být klícové slovo continue. Takto lze tovřit i cykly.

Důležité informace 4: Maximální počet znaků na řádce.

Maximální počet znaků na řádce je 60. (úplně přesně nevím, stálo by za to pořádně vygooglit)

Při práci se subroutinami je čas od času nutné číst výsledky již během analýzy. To lze udělat tak, že si ke standarnímu souboru s výsledky .odb necháte vypisovat ještě .fil soubor a pro čtení výsledků z .fil souboru použijete subroutinu POSFIL. Soubor .fil se bude vypisovat pokud v Abaqus/CAE zapíšete v nabídce model → Edit Keywords na správné místo několik klíčových slov. Která přesně a kam přesně si již každý čtenář musí najít sám, liší se to případ od případu. Snad jen umístění bývá většinou těsně před koncem KEYWORDS. Zdrojová data **??** ukazují část kódu, který čte .fil soubor s výsledky až na konec a pak ho převine na začástek, toto je nutné, protože .fil soubor je sekvenční, aby se v něm vše dělo maximálně rychle. Když cyklus dojede na konec souboru vyskočí na návěští 190 a pak se soubor převine na začátek.

Zdrojová data 5: Práce se subroutinou POSFIL.

```
!     Finds current increment in .fil file
      CALL POSFIL(KSTEP,KINC,ARRAY,JRCD)

!     Up to the end of the file
      do K1=1,999999
        call DBFILE(0,ARRAY,JRCD)
        if (JRCD .NE. 0) GO TO 190
      enddo
  190 continue
!     Rewind of file into the begining
      call DBFILE(2,ARRAY,JRCD)
```

4 Common block

Vpřípadě, že budete programovat i jen trochu složitější model, popřípadě budete tvořit komplikovanou analýzu, může se stát a ono se to stane, že budete potřebovat mít některé proměnné tak říkajíc vždy při ruce. Tedy, že bude dobré je deklarovat jako globální proměnné. Tohoto lze dosáhnout pomocí tzv. Common blocku. V následující ukázce je vidět, jak je lze definovat. Jedná se o subroutinu UEXTERNALDB, ve které lze definovat vše možné, tato subroutina je v Abaqusu právě proto, aby v ní šlo definovat cokoliv obslužného a pomocného a šikovného atd. Kde všude se subroutina volá lze vidět v obrázku **??**. Připoměnmě jen, že při použití globálních proměnných, musí být v každé subroutině, ve které je cheme používat, tyto proměnné definovány stejně. A dále, **každý datový typ by měl mít vlastní COMMON BLOCK!**.

Zdrojová data 6: Subroutina UEXTERNALDB a COMMON BLOCK.

```
      SUBROUTINE UEXTERNALDB(LOP,LRESTART,TIME,DTIME,KSTEP,KINC)

      INCLUDE 'ABA_PARAM.INC'

      DIMENSION TIME(2)

      CHARACTER*256 curdir,curfile,unistr,unistr2,str_line

      LOGICAL flag_load
      COMMON /FLG/ flag_load
      DIMENSION flag_load(100,4)

      COMMON /IDS/ no_strips,nl_geom,no_steps,id_step
      DIMENSION no_steps(100),id_step(100)

      COMMON /NUM/ grip_load,dgrip_load,ext_disp,
     1 ext_disp_max,fR_disp_node,a_FEA,
     2 c_ext_n1,c_ext_n2,c_disp_n,di_lo_n,
     3 ext_disp_p,fR_disp_p,dul_l_turn,
     4 dperc_l_turn
      DIMENSION grip_load(100),dgrip_load(100),ext_disp(100),
     1 ext_disp_max(100),fR_disp_node(100),
     2 c_ext_n1(100,3),c_ext_n2(100,3),
     3 c_disp_n(100,3),a_FEA(100),di_lo_n(100,3),
     4 ext_disp_p(100),fR_disp_p(100),dul_l_turn(100),
     5 dperc_l_turn(100)
```

5 Tisk a čtení textových souborů

Opět se zastavíme u čtení a tvoření textových souborů, protože to je vždy důležité. Stále si člověk potřebuje něco vytisknout a načíst a tak pořád dokola. Jaká čísla (identifikátory) používat pro soubory lze nalézt v *Abaqus Analysis User's manual, 3.6 File extension definitions*. Shrnuto a podrtženo, používejte čísla nad 100. Ukázka jak otevřít soubor a zapsat do něj je ve zdorjových datech 7. Dodejme, že .dat soubor pro Standard analýzu[1] má id = 6.

Proč je ve zdrojových datech 7 divné volání subroutiny GETOUTDIR[2]? Protože je to subroutina, která načte cestu do aktuálního adresáře a uloží jí do proměnné curdir. Ta je navíc vždy oříznuta funkcí trim, aby neobsahovala zbytečné mezery. V curdir je pak uložena natvrdo cesta do pracovního adresáře. Toto je nutné, protože Abaqusí analýza probíhá ve scratchovém adresáři a ten není stejný jako pracovní a proto, pokud nebudete mít zjištěnou pevnou cestu do pracovního adresáře, by se vám tvořily soubory ve scratchovém adresáři a na závěr analýzy by se smazaly.

Zdrojová data 7: Tisk do souboru.

```
!       Nacteni soucasneho pracovniho adresare
        call GETOUTDIR( curdir, LENOUTDIR )

!       Oriznuti stringu s cestou o prebytecne mezery na konci
!       A pridani nazvu souboru
        curfile = trim(curdir) // '\info'

!       Pridani pripony souboru
        curfile = trim(curfile)//'.abqd'

!       Otevreni souboru
        open(105,FILE=curfile)

!       Zapisovani do souboru
        write(105,*)'Current directory'
        write(105,*)trim(curdir)
        write(105,*)'Current file for info'
        write(105,*)trim(curfile)
        write(105,*)'------------------------'
```

Zdorjová data 8 obsahují ukázku, jak číst z textového souboru. Proměnná curfile je string složený z oříznutého curdir, ke kterému je připojeno jméno souboru a pak ještě přípona[3]. Dále následuje otevření souboru s parametry pro čtení atp. V ukázce se načte jedna řádka s komentářem do stringu a pak další řádka s integer číslem.

[1] Číslování souborů pro explicitní modul Abaqusu se liší.

[2] Tato subroutina je standardní součástí Fortranu.

[3] Přípona je autorem zvolená, tak aby po srovnání souborů podle přípon ve Windows, byly tyto soubory na začátku. Mimochodem zkratka .abqp znamená abaquspython - je to totiž soubor obsahující data pro výpočet, který byl vytvořen v pythonu.

Zdrojová data 8: Tisk do souboru.

```
!       Otevreni souboru
        curfile = trim(curdir) //'\tensile_data'
        curfile = trim(curfile)//'.abqp'
        open (unit=1001,file=curfile,status='old',action='read')
!       Cteni dvou radek  a kontrolni vypis do souboru 105
        read(1001,*)str_line
        write(105,*)trim(str_line)
        read(1001,*)no_strips
        write(105,*)no_strips
```

Tato prezentace je spolufinancována Evropským sociálním fondem a státním rozpočtem České republiky v rámci projektu č. **CZ.1.07/2.2.00/28.0206 „Inovace výuky podpořená praxí“.**

Tento studijní materiál je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.