

Note #1: Getting started with GRASP0-20040219 and DARC-20040123

Patrick Norrington

June 5, 2009

Contents

1	My machine	2
2	GRASP0	3
2.1	Downloading and extracting files	3
2.2	The README file	3
2.3	The <code>src</code> directory	4
2.4	The <code>grasp0.inc</code> file	4
2.5	Compilation	5
2.6	Testcases	6
2.7	My run of testcases	7
2.8	Testcase script <code>grasp0-tests</code>	9
2.8.1	Testcase 1	11
2.8.2	Testcase 2	11
2.8.3	Testcase 3	12
2.8.4	Testcase 4	12
2.8.5	Testcase 5	13
2.9	Testcase results	13
2.10	The <code>src-machine-dependent</code> directory	14

3	DARC	14
3.1	Downloading and extracting files	14
3.2	The README file	15
3.3	The src directory	15
3.4	The darc.inc file	16
3.5	The stgfjj.inc file	18
3.6	Compilation	19
3.7	Testcases	20
3.8	My run of testcase	26
3.9	Script xsummary	28
3.10	Fe XXII testcase	28
3.11	Optimisation	31
3.12	Test of dstg0	31
4	Updates	31

1 My machine

I am using an iMac running Mac OS X version 10.4.11. This description should be appropriate for any Linux system or Cygwin (<http://www.cygwin.com/>) on a Windows machine. I use the zsh command shell but there is nothing fancy here so bash or ksh should be fine.

The fortran compilers that I have are: ifort

```
Intel(R) Fortran Compiler for applications running on Intel(R) 64,  
Version 10.1    Build 20080312 Package ID: m_fc_p_10.1.014  
Copyright (C) 1985-2008 Intel Corporation. All rights reserved.
```

and gfortran

```
Target: i686-apple-darwin8  
Thread model: posix  
gcc version 4.2.3
```

As gfortran is free I will mainly refer to it. The commercial ifort should give better optimisation and performance. But free is good (like this software!).

2 GRASPO

2.1 Downloading and extracting files

Download the tar file GRASPO-20040219.tar. Create a directory GRASPO and move the tar file into it. Within the directory extract the files from the tar file.

```
>> mkdir GRASPO
>> mv GRASPO-20040219.tar GRASPO
>> cd GRASPO
>> tar xf GRASPO-20040219.tar
>> ls -la
```

should give

```
-rw-r--r--  1 phn  admin  1863680 Jun  1 10:17 GRASPO-20040219.tar
-rw-r--r--  1 phn  admin    1179 Feb 19  2004 README
drwxr-xr-x 11 phn  admin    374 Feb 19  2004 doc-ftnchek
drwxr-xr-x  4 phn  admin    136 Feb 19  2004 doc-manual
drwxr-xr-x  7 phn  admin    238 Feb 19  2004 src
drwxr-xr-x  8 phn  admin    272 Feb 19  2004 src-machine-dependent
drwxr-xr-x  5 phn  admin    170 Feb 19  2004 testcase
```

2.2 The README file

The README file is:

```
=====
tree for GRASPO-2004-0219
=====
```

Tree for directory GRASPO-2004-0219

GRASPO-2004-0219

```
"      doc-ftnchek          ... output from static analyser ftnchek
"      doc-manual          ... manual, needs some revision
"      src                 ... Fortran source with compilation script
"      src-machine-dependent ... replacement for machine dependent routines
"      testcase           ... C IV with 1s hole, see README
"      "      A           ... non-relativistic CSF input
"      "      B           ... relativistic CSF input
```

Changes from previous version:

1. Physical constants updated. See GRASP routine.
2. Added routine DATR so that relativistic CSF input can be used as described in original 1980 CPC write-up.
3. NJSYM is used for calculating recoupling coefficients.
4. Dummy versions of the system-specific routines CALEN and QUARTZ are used.

2.3 The src directory

Change directory to src. You should have these files.

```
-rwxr-xr-x  1 phn  admin    185 Feb 19  2004 COMPILE-ALL
-rw-r--r--  1 phn  admin   96433 Feb 19  2004 blas.f
-rw-r--r--  1 phn  admin  806797 Feb 19  2004 grasp0-2004-0219.f
-rw-r--r--  1 phn  admin    929 Feb 19  2004 grasp0.inc
-rw-r--r--  1 phn  admin  241985 Feb 19  2004 lapack.f
```

2.4 The grasp0.inc file

The file `grasp0.inc` sets parameters that are used to dimension the code. This means that a recompilation is needed if a dimension is exceeded. Of course one can choose large values at the start and in practice this is not too tedious. It does reflect the fact that the code is written in fortran 77 (plus extensions). A conversion to fortran 90 with its ability to use `ALLOCATE` for array dimensions (and nicer layout) is desirable but not urgent.

```
C
C  grasp0.inc
C  This file is INCLUDED in the grasp0 and angul modules.
C  It is used to set dimensions.
C
C  MXNC  relativistic CSFs
C  MXNE  exchange terms
C  MXNG  radial grid points for /WAVE/
C  MXNM  angular coefficients
C  MXNO  Lagrange multipliers
C  MXNP  radial grid points
C  MXNW  relativistic orbitals
```

```
C  MXNX  dimension for EOL ( = 1 for AL,EAL,OL)
C
C  Note:
C
C  1. angul only uses MXNC, MXNW
```

```
C-----
      INTEGER MXNC,MXNE,MXNG,MXNM,MXNO,MXNP,MXNW,MXNX
      PARAMETER (MXNC = 500)
      PARAMETER (MXNE = 300)
      PARAMETER (MXNG = 10000)
      PARAMETER (MXNM = 10000)
      PARAMETER (MXNO = 50)
      PARAMETER (MXNP = 400)
      PARAMETER (MXNW = 35)
      PARAMETER (MXNX = 1)
C-----
```

NOTE

The code `angul` referred to in `grasp0.inc` is a stripped down version of GRASP0 that only calculates the angular coefficients.

2.5 *Compilation*

`COMPILE-ALL` is a simple compilation shell script using the `g77` fortran compiler.

```
# compile all source files
# I have chosen optimisation 02

g77 -02 -c *.f

# link to form executable
# use an optimised lapack/blas library if you have one

g77 -02 -o grasp0.exe *.o
```

NOTE

This is LAPACK (version 2.0). This lapack/blas code is quite old now. The comment about optimised libraries on your system may not be helpful because it is possible that such libraries on your system would not be compatible.

The `COMPILE-ALL` shell script is not of much use for me as the fortran compilers that I have are `ifort` and `gfortran`. Therefore I will compile using

```
>>> gfortran -O0 -o ../grasp0.exe *.f
```

This works for me without any compiler error/warning messages. The executable is moved up a level for running the testcases. You can of course place it anywhere accessible.

NOTE

I have deliberately chosen -O0 (no compiler optimisation) because there have been optimisation problems with GRASP0. You should check it out with -O0 first and then test results as you ramp up the optimisation. Rather than agonise about squeezing the last drop of performance out of the compiler you might just choose -O1.

2.6 Testcases

There are just two testcases here with a slight difference in input. Testcase A uses non-relativistic CSF input data while testcase B uses relativistic CSF input data.

This is the README file from the testcase directory:

```
The testcase is C IV (Li-like) with CSFs 1s(2s^2,2p^2)
and a point nucleus.
```

```
In example A the input parameter IOP (3rd number on
card 2) is set to 2 so that non-relativistic CSF input
is used. The relativistic CSFs are generated from them.
```

```
In example B IOP is set to -1 and the relativistic CSFs
are input explicitly in the form described in the original
1980 CPC write-up of the program.
```

Of course the same results should be obtained.

```
To help with identifying the correct form of CSF input
when IOP is -1, you can first use IOP > -1 and set ANG
option 10 which prints out the equivalent IOP=-1 input.
```

You can compare the input data for the two cases.

Testcase A:

```
C IV Li-like      1s ( 2s^2 , 2p^2 ) point nucleus
2 3 2 12.011 ! CSFs,orbitals,IOP,ATW
1S 1
```

```
2S 2 0
2P 0 2
ANG 1 10
-1
MCP
MCBP
MCDF
```

```
6
OL 2
BENA 4 5 6 8 11
1E-3
STOP
```

Testcase B:

```
C IV Li-like 1s ( 2s^2 , 2p^2 ) point nucleus
9 4 -1 12.011 ! CSFs,orbitals,IOP,ATW
1S 1
2S 2 0 0 0 0 0 0 0 0
2P- 0 0 1 0 1 1 0 1 2
2P 0 2 1 2 1 1 2 1 0
ANG 1
2 2 5/2
3 1 5/2
4 2 3/2
5 1 3/2
6 0 3/2
7 0 1/2
8 1 1/2
MCP
MCBP
MCDF

6
OL 2
BENA 4 5 6 8 11
1E-3
STOP
```

2.7 My run of testcases

Each run of GRASP0 requires an input file `GRASP.INP`. It produces a number of output files including the formatted files `GRASP.OUT` and `GRASP.PUN`.

The latter is abbreviated output. Therefore to avoid any over-writing it is best to run GRASP0 each time in a separate directory.

For testcase A (starting in the GRASP0 directory):

```
>> mkdir testA
>> cp testcase/A/GRASP.INP testA
>> cd testA
>> ../grasp0.exe
```

gives

```
-rw-r--r--  1 phn  admin  23020 Jun  1 14:30 BENA.DAT
-rw-r--r--  1 phn  admin   176 Feb 19  2004 GRASP.INP
-rw-r--r--  1 phn  admin  89654 Jun  1 14:30 GRASP.OUT
-rw-r--r--  1 phn  admin  27021 Jun  1 14:30 GRASP.PUN
-rw-r--r--  1 phn  admin  23368 Jun  1 14:30 MCDF.DAT
-rw-r--r--  1 phn  admin   9732 Jun  1 14:30 MCP.DAT
```

GRASP0 uses PRINT statements to write to the screen. This just tells you that the calculation is progressing fine. For a small example such as this it is not so helpful but it might be more useful for larger cases. You can of course pipe this output to a file and run a large job in the background using

```
>> grasp0.exe > grasp0.out &
```

Using nohup allows you to close the terminal window and log out

```
>> nohup grasp0.exe
```

The output is written to nohup.out. I haven't tested if this works on a Windows/Cygwin machine.

The .DAT files are unformatted output.

Comparing (using diff) GRASP.OUT from directory testcase/A with the file in directory testA gives:

```
643c643
<  1S    2S    -3.9375811E-13
---
>  1S    2S    -3.9430464E-13
918c918
<  1S    2S     5.7766902E-10
---
>  1S    2S     5.7766906E-10
```

```
995,998c995,998
<  2  4  P 1/2  even   7   1.000   8.280271798028E-01
<  3  4  P 3/2  even   4  -1.000   8.286931063989E-01
<  4  4  P 5/2  even   2   1.000   8.297989403877E-01
<  5  2  D 3/2  even   6   1.000   1.152782462712E+00
---
>  2  4  P 1/2  even   7   1.000   8.280271798025E-01
>  3  4  P 3/2  even   4  -1.000   8.286931063987E-01
>  4  4  P 5/2  even   2   1.000   8.297989403874E-01
>  5  2  D 3/2  even   6   1.000   1.152782462711E+00
1000,1001c1000,1001
<  7  2  P 1/2  even   8   1.000   1.202130270554E+00
<  8  2  P 3/2  even   5   1.000   1.203467021184E+00
---
>  7  2  P 1/2  even   8   1.000   1.202130270553E+00
>  8  2  P 3/2  even   5   1.000   1.203467021183E+00
1018c1018
< excess (static potential+residual charge) =  3.6415315E-14
---
> excess (static potential+residual charge) =  3.5527137E-14
```

Similarly for testcase B.

```
571c571
<  1S   2S   -3.9375811E-13
---
>  1S   2S   -3.9430464E-13
846c846
<  1S   2S    5.7766902E-10
---
>  1S   2S    5.7766887E-10
928c928
< excess (static potential+residual charge) =  3.6415315E-14
---
> excess (static potential+residual charge) =  3.7303494E-14
```

The differences are small (at machine accuracy level). We can be confident that the code is working fine.

2.8 Testcase script grasp0-tests

`grasp0-tests` is a shell script that runs 5 testcases.

Just typing `grasp0-tests` gives the following information:

Usage: grasp0-tests <dir>

Runs the GRASPO test cases in directory <dir>.

Uses the executable grasp0.exe.

If <dir> is set to input then the input files are written to <dir>.

The GRASPO test cases:

1. Li-like C IV 1s (2s2, 2p2)
point nucleus OL 2
2. Hg ground state (6s2)
finite (Fermi) nucleus AL
3. N ground state 1s2 2s2 (2p3) J=3/2
point nucleus OL 1
4. He-like F VIII 1s2p, 1s3p, 1s4p, 2s2p, 2s3p, 2p3s
uniform nucleus EAL
5. Be-like Fe XXIII 1s2 (2s2, 2s2p, 2p2)
point nucleus EAL OSCL

Patrick Norrington
1 July 2008

If you specify a directory name as an argument to the script then the calculations are run in that directory. After the calculations are complete the GRASP.OUT files for each testcase are concatenated into a single file grasp0.out in the directory. This can be compared with later runs of the script.

By typing

```
>> grasp0-tests input
```

a directory input is created and the input files are written to it. These files are useful templates.

NOTE

The executable actually has the name \$GRASP/grasp0.exe. This means that you must specify the global variable \$GRASP. If you are in the directory containing grasp0.exe then you can use

```
>> export GRASP=$PWD
```

2.8.1 Testcase 1

The first testcase is the same as testcase A (except it uses ANG option 10 to get the relativistic CSF input data).

```
C IV Li-like      1s ( 2s^2 , 2p^2 ) point nucleus
2 3 2 12.011 ! CSFs,orbitals,IOP,ATW
  1S  1
  2S  2 0
  2P  0 2
ANG  1
-1
MCP
MCBP
MCDF

6
OL  2
BENA 4 5 6 8 11
1E-3
STOP
```

2.8.2 Testcase 2

```
Hg ground state finite (Fermi) nucleus
1 14 2 200.59 ! CSFs,orbitals,IOP,ATW
  1S
  2S
  2P
  3S
  3P
  3D
  4S
  4P
  4D
  4F
  5S
  5P
  5D
  6S
ANG
-1
MCP
MCBP
MCDF
```

```
80
VNUC 2 1.183615194E-4 1.0393E-5
AL 4
BENA

STOP
```

2.8.3 Testcase 3

```
N grnd state (1s2 2s2 2p3) J=3/2
1 3 2 14.0067 ! CSFs,orbitals,IOP,ATW
 1S
 2S
 2P 3
ANG
3/2
MCP
MCBP
MCDF

7
OL 1
BENA

STOP
```

2.8.4 Testcase 4

```
He-like Fluorine uniform nucleus
6 6 2 18.998403 ! CSFs,orbitals,IOP,ATW
 1S 1 1 1 0 0 0
 2S 0 0 0 1 1 0
 2P 1 0 0 1 0 1
 3S 0 0 0 0 0 1
 3P 0 1 0 0 1 0
 4P 0 0 1 0 0 0
ANG
-1
MCP
MCBP
MCDF

9
VNUC 1 18
```

EAL
BENA

STOP

2.8.5 Testcase 5

```
Fe XXIII
3 3 2 55.8 ! CSFs,orbitals,IOP,ATW
 1S
 2S 2 1 0
 2P 0 1 2
ANG 1
-1
MCP
MCBP
MCT 1
MCDF
```

26
EAL
BENA

OSCL

```
END
Fe XXIII
3 3 2 55.8 ! CSFs,orbitals,IOP,ATW
 1S
 2S 2 1 0
 2P 0 1 2
ANG 1
-1
MCT 2
OSCL
```

STOP

2.9 Testcase results

I used the following commands to test various optimisation for the two compilers on my machine. There is nothing too tricky here except the use of export to define the global variable and the use of ln -s for a soft link.

```
cd src
```

```
gfortran -O0 -o ../grasp0-gfortran-00.exe *.f
gfortran -O1 -o ../grasp0-gfortran-01.exe *.f
gfortran -O2 -o ../grasp0-gfortran-02.exe *.f
ifort -O0 -o ../grasp0-ifort-00.exe *.f
ifort -O1 -o ../grasp0-ifort-01.exe *.f
ifort -O2 -o ../grasp0-ifort-02.exe *.f
ifort -fast -o ../grasp0-ifort-fast.exe *.f
cd ..
export GRASP=$PWD
rm grasp0.exe ; ln -s grasp0-gfortran-00.exe grasp0.exe ; grasp0-tests test1
rm grasp0.exe ; ln -s grasp0-gfortran-01.exe grasp0.exe ; grasp0-tests test2
rm grasp0.exe ; ln -s grasp0-gfortran-02.exe grasp0.exe ; grasp0-tests test3
rm grasp0.exe ; ln -s grasp0-ifort-00.exe grasp0.exe ; grasp0-tests test4
rm grasp0.exe ; ln -s grasp0-ifort-01.exe grasp0.exe ; grasp0-tests test5
rm grasp0.exe ; ln -s grasp0-ifort-02.exe grasp0.exe ; grasp0-tests test6
rm grasp0.exe ; ln -s grasp0-ifort-fast.exe grasp0.exe ; grasp0-tests test7
diff test1/grasp0.out test2/grasp0.out > diff-test1-test2
diff test1/grasp0.out test3/grasp0.out > diff-test1-test3
diff test1/grasp0.out test4/grasp0.out > diff-test1-test4
diff test1/grasp0.out test5/grasp0.out > diff-test1-test5
diff test1/grasp0.out test6/grasp0.out > diff-test1-test6
diff test1/grasp0.out test7/grasp0.out > diff-test1-test7
```

As before the differences were small. We would expect the ifort -fast version to run quickest.

2.10 The src-machine-dependent directory

The routines in this directory are not compatible with GRASP0. Apologies. I put in the DARC versions by mistake.

I have modified the -sun versions and these work with both the gfortran and ifort compilers. The testcases work as before. The timings suggest the effect of optimisation but really these examples are too small to be significant.

3 DARC

3.1 Downloading and extracting files

Download the tar file DARC-20040123.tar. Create a directory DARC and move the tar file into it. Within the directory extract the files from the tar file.

```
>> mkdir DARC
>> mv DARC-20040123.tar DARC
>> cd DARC
>> tar xf DARC-20040123.tar
>> ls -la
```

should give

```
-rw-r--r--  1 phn  admin  6236160 Jan 23  2004 DARC-20040123.tar
-rwxrwxrwx  1 phn  admin    856 Jan 23  2004 README
drwxrwxrwx 50 phn  admin   1700 Jun  2 09:13 doc-ftnchek
drwxrwxrwx  4 phn  admin    136 Jun  2 09:13 doc-manual
drwxrwxrwx  8 phn  admin    272 Jun  2 09:13 doc-quick-reference
drwxrwxrwx  5 phn  admin    170 Jun  2 09:13 scripts
drwxrwxrwx 21 phn  admin    714 Jun  2 15:58 src
drwxrwxrwx  8 phn  admin    272 Jun  2 09:13 src-machine-dependent
drwxrwxrwx  8 phn  admin    272 Jun  2 09:14 testcase
```

3.2 The README file

The README file is:

```
=====
tree for DARC-OXQUB-2004-0123
=====
```

Tree for directory DARC-OXQUB-2004-0123:

```
DARC-OXQUB-2004-0123
"      doc-ftnchek      ... output from static analyser ftnchek
"      doc-manual      ... manuals, needs some revision
"      doc-quick-reference ... quick reference text files
"      scripts        ... scripts referred to in testcase
"      src            ... Fortran source with compilation script
"      src-machine-dependent ... replacement for machine dependent routines
"      testcase       ... Ne II electron scattering, see README
"      "              calc-H-file ... calculation of H-file
"      "              calc-coul  ... Coulomb asymptotic calculation
```

3.3 The src directory

Change directory to src. You should have these files.

```

total 7016
-rwxr-xr-x  1 phn  admin    940 Jan 23  2004 COMPILE-ALL
-rw-r--r--  1 phn  admin  96433 Jan 23  2004 blas.f
-rw-r--r--  1 phn  admin   2792 Jan 23  2004 darc.inc
-rw-r--r--  1 phn  admin  33772 Jan 23  2004 dstg0.f
-rw-r--r--  1 phn  admin 136929 Jan 23  2004 dstg1-ints.f
-rw-r--r--  1 phn  admin 183103 Jan 23  2004 dstg1-orbs.f
-rw-r--r--  1 phn  admin 644773 Jan 23  2004 dstg2-njgraf.f
-rw-r--r--  1 phn  admin 538561 Jan 23  2004 dstg2-njsym.f
-rw-r--r--  1 phn  admin 262895 Jan 23  2004 dstg3-asyck.f
-rw-r--r--  1 phn  admin 143245 Jan 23  2004 dstg3-coul.f
-rw-r--r--  1 phn  admin 215259 Jan 23  2004 dstg3-vpm.f
-rw-r--r--  1 phn  admin   87159 Jan 23  2004 dstg4.f
-rw-r--r--  1 phn  admin 111443 Jan 23  2004 dstgh-hsldr.f
-rw-r--r--  1 phn  admin   88422 Jan 23  2004 dstgh-lapack.f
-rw-r--r--  1 phn  admin 241984 Jan 23  2004 lapack.f
-rw-r--r--  1 phn  admin 291602 Jan 23  2004 pdstg3-asyck.f
-rw-r--r--  1 phn  admin 172443 Jan 23  2004 pdstg3-coul.f
-rw-r--r--  1 phn  admin 293300 Jan 23  2004 stgfjj.f
-rw-r--r--  1 phn  admin   1081 Jan 23  2004 stgfjj.inc

```

3.4 The darc.inc file

The file `darc.inc` sets parameters that are used to dimension the code. It is used by all modules except `stgfjj`.

```

C-----
C  darc.inc
C  This file is INCLUDED in the DARC modules.
C  It is used to set dimensions.
C  The following table indicates where the dimensions are used.
C
C          dstg0  dstg1  dstg2  dstgh  dstg3  pdstg3  dstg4
C
C  MXCH                *      *      *      *
C  MXHD                *      *      *      *
C  MXI1                 *      *
C  MXI2                 *      *
C  MXNA                *      *      *      *
C  MXNB                 *      *      *      *
C  MXNC                *
C  MXNK                 *      *      *      *
C  MXNL                *      *      *      *
C  MXNM                *
C  MXNP                 *      *

```

```

C   MXNW      *           *
C   MXP1      *         *
C   MXNAST                    *
C   MXNREC                    *
C   MXNSYM                    *
C   MXNENE                    *
C
C   Note:
C
C   1. dstg3 means dasym,dasypck,dcoul,dvpm
C   2. pdstg3 means pasypck,pcoul
C
C           asycoul   asypck   asyvpm   hslldr
C
C   MXCH      *           *           *
C   MXHD                    *
C   MXNA                    *           *
C   MXNK      *           *           *
C
C-----
C   MXCH      channels
C   MXHD      size of Hamiltonian
C   MXI1      radial integrals
C   MXI2      marks position of radial integrals
C   MXNA      lambda-values for asymptotic coefficients
C   MXNB      continuum orbitals per KAPPA value
C   MXNC      relativistic configurations
C   MXNK      KAPPA values
C   MXNL      target levels or correlation functions
C   MXNM      MCP angular coefficients
C   MXNP      points in radial mesh
C   MXNW      relativistic orbitals
C   MXP1      mesh points used to store orbitals
C   MXNAST   target states
C   MXNREC   records
C   MXNSYM   symmetries per energy point
C   MXNENE   energy points
C-----
C           INTEGER MXCH,MXHD,MXI1,MXI2,MXNA,MXNB,MXNC
C           INTEGER MXNK,MXNL,MXNM,MXNP,MXNW,MXP1
C           INTEGER MXNAST,MXNREC,MXNSYM,MXNENE
C           PARAMETER (MXCH = 200)
C           PARAMETER (MXHD = 4000)
C           PARAMETER (MXI1 = 1000000)
C           PARAMETER (MXI2 = 900)
C           PARAMETER (MXNA = 4)

```

```
PARAMETER (MXNB = 40)
PARAMETER (MXNC = 600)
PARAMETER (MXNK = 100)
PARAMETER (MXNL = 300)
PARAMETER (MXNM = 130000)
PARAMETER (MXNP = 1000)
PARAMETER (MXNW = 150)
PARAMETER (MXP1 = 100000)
PARAMETER (MXNAST = 50)
PARAMETER (MXNREC = 40000)
PARAMETER (MXNSYM = 60)
PARAMETER (MXNENE = 2000)
```

C-----

NOTE

The comments in `darc.inc` refer to an earlier release of the code. For example `dasypck` means `dstg3-asyck` and `pasypck` means `pdstg3-asyck`. The module `dasym` (`dstg3-asy`) is not in this release.

3.5 The stgfjj.inc file

The file `stgfjj.inc` is used by the `stgfjj` module only.

```
C
C BUT = Buttle coefficients (DARC only)
C CHF = channels
C DEG = degenerate channels
C EST
C LMX = multipoles
C LP1 = small l values
C MNP = R-matrix poles
C MSH = energy-mesh points
C NRG = terms in Buttle fit
C PTS = outer-region radial points
C SA1
C SA2
C SLP = S, L, PI cases
C TAR = target states
C TET = coefficients for THETA
C
C     INTEGER XBUT
C     INTEGER XCHF
C     INTEGER XDEG
C     INTEGER XEST
C     INTEGER XLMX
```

```
INTEGER XLP1
INTEGER XMNP
INTEGER XMSH
INTEGER XNRG
INTEGER XPTS
INTEGER XSA1
INTEGER XSA2
INTEGER XSPL
INTEGER XTAR
INTEGER XTET
PARAMETER (XBUT = 60)
PARAMETER (XCHF = 150)
PARAMETER (XDEG = 30)
PARAMETER (XEST = 1000)
PARAMETER (XLMX = 10)
PARAMETER (XLP1 = 99)
PARAMETER (XMNP = 3000)
PARAMETER (XMSH = 5000)
PARAMETER (XNRG = 80)
PARAMETER (XPTS = 600)
PARAMETER (XSA1 = 1)
PARAMETER (XSA2 = 1)
PARAMETER (XSPL = 20)
PARAMETER (XTAR = 100)
PARAMETER (XTET = 200)
```

3.6 **Compilation**

COMPILE-ALL is a simple compilation shell script using the g77 fortran compiler.

```
# compile all source files
# I have chosen optimisation 02

g77 -02 -c *.f

# link to form executables
# dstg0, dstgh-hsldr and dstg4 do not require lapack/blas
# use an optimised lapack/blas library if you have one

g77 -02 -o dstg0.exe dstg0.o

g77 -02 -o dstg1-ints.exe dstg1-ints.o blas.o lapack.o
g77 -02 -o dstg1-orbs.exe dstg1-orbs.o blas.o lapack.o
```

```
g77 -O2 -o dstg2-njgraf.exe dstg2-njgraf.o blas.o lapack.o
g77 -O2 -o dstg2-njsym.exe dstg2-njsym.o blas.o lapack.o

g77 -O2 -o dstg3-asympck.exe dstg3-asympck.o blas.o lapack.o
g77 -O2 -o dstg3-coul.exe dstg3-coul.o blas.o lapack.o
g77 -O2 -o dstg3-vpm.exe dstg3-vpm.o blas.o lapack.o

g77 -O2 -o pdstg3-asympck.exe pdstg3-asympck.o blas.o lapack.o
g77 -O2 -o pdstg3-coul.exe pdstg3-coul.o blas.o lapack.o

g77 -O2 -o dstg4.exe dstg4.o

g77 -O2 -o dstgh-hsldr.exe dstgh-hsldr.o
g77 -O2 -o dstgh-lapack.exe dstgh-lapack.o blas.o lapack.o

g77 -O2 -o stgfjj.exe stgfjj.o blas.o lapack.o
```

NOTE

This is LAPACK (version 2.0).

The COMPILE-ALL shell script is not of much use for me as the fortran compilers that I have are ifort and gfortran. Therefore I changed g77 -O2 to gfortran -O0 in COMPILE-ALL. This works for me and the only compiler error/warning messages were about the use of PAUSE

```
dstg4.f:2399.72:
```

```
      IF (jstack.gt.NSTACK)pause 'NSTACK too small in indexx'
```

1

```
Warning: Deleted feature: PAUSE statement at (1)
```

```
dstg4.f:2484.72:
```

```
      IF (jstack.gt.NSTACK)pause 'NSTACK too small in indexx'
```

1

```
Warning: Deleted feature: PAUSE statement at (1)
```

NOTE

I have deliberately chosen -O0 (no compiler optimisation).

3.7 Testcases

There is just one testcase here.

This is the README file from the testcase directory:

 Test case

Ne II electron scattering

3 target states using 1s, 2s, 2p-, 2p orbitals
 14 continuum orbitals per ang.mom.

eigen-energies

level	J	parity	dominant		a.u.	Ryd.
			CSF	mix		
1	3/2	o	1	1.000	-1.279645753902E+02	-2.559291507805E+02
2	1/2	o	2	1.000	-1.279607689527E+02	-2.559215379054E+02
3	1/2	e	3	1.000	-1.268739238798E+02	-2.537478477596E+02

eigen-energies relative to the lowest

level	J	parity	dominant		a.u.	Ryd.
			CSF	mix		
1	3/2	o	1	1.000	0.00000000E+00	0.00000000E+00
2	1/2	o	2	1.000	3.80643755E-03	7.61287509E-03
3	1/2	e	3	1.000	1.09065151E+00	2.18130302E+00

16 symmetries

Sym.	Kmin	Kmax			chan.	orbs.	corr.	Ham.dim.
1	1	3	J = 0	even	3	14	1	43
2	1	4	J = 0	odd	3	14	0	42
3	1	6	J = 1	even	7	14	0	98
4	1	5	J = 1	odd	7	14	0	98
5	2	7	J = 2	even	8	14	0	112
6	1	8	J = 2	odd	8	14	0	112
7	3	10	J = 3	even	8	14	0	112
8	4	9	J = 3	odd	8	14	0	112
9	6	11	J = 4	even	8	14	0	112
10	5	12	J = 4	odd	8	14	0	112
11	7	14	J = 5	even	8	14	0	112
12	8	13	J = 5	odd	8	14	0	112
13	10	15	J = 6	even	8	14	0	112

14	9	16	J = 6	odd	8	14	0	112
15	11	18	J = 7	even	8	14	0	112
16	12	17	J = 7	odd	8	14	0	112

STEP 1 --- create H-file in directory calc-H-file

Input files

TARGET.INP

ORBS.INP

INTS.INP

DSTG2.INP

DSTGH.INP

Run the executables

> dstg1-orbs.exe

> dstg1-ints.exe

> dstg2-njgraf.exe

> dstgh-lapack.exe

Output files

ORBS.OUT

INTS.OUT

DSTG2.OUT

DSTGH.OUT

The script xsummary can be run to create the file summary
which is a summary of these output files.

Data files

The size is given.

All except DSTGHF.DAT are unformatted.

13048 ORBS.DAT
4232544 ORBITALS.DAT

38224 DSTG1.DAT
1290560 INTEGRAL.DAT

741132 DSTG2.DAT

258300 DSTGHF.DAT

STEP 2 --- run dstg3-coul in directory calc-coul

Input file

DSTG3.INP

Run executable

(after linking to the H-file)

```
>> ln -s ../calc-H-file/DSTGHF.DAT DSTGHF.DAT
>> dstg3-coul.exe
```

Output file

DSTG3.OUT

Data files

These are formatted.

DSTG3.DAT (contains collision strengths)
PHASE.SH (contains eigenphase sums)

STEP 3 --- run dstg4

To analyse the results you can use the dstg4 module.

Input file

DSTG4.INP

Run executable

>> dstg4.exe < DSTG4.INP

Output files

DSTG4.OUT
DSTG4.PUN (lists input data only)

Data files

These are unformatted.

DAT1.DAT
DAT2.DAT
DAT3.DAT

The following file (a script) contains the collision strengths summed over the available symmetries.

DSTG3.RES

STEP 4 --- processing DSTG3.RES

Use the script draw-omega to create a gnuplot input file.
Leave off the -p option if you just want to view the data in gnuplot.

>> . ./DSTG3.RES
>> draw-omega -p INDEX
>> gnuplot draw-omega.plt
>> ps2pdf draw-omega.ps

The following files result from running the script DSTG3.RES
An INDEX file giving info on the x..... files.
An x..... file for each transition.

INDEX
x001002
x001003
x002003

The following file results from running the script draw-omega
Consider this as sample gnuplot commands for displaying the data

draw-omega.plt

The following file results from running gnuplot

draw-omega.ps

The following file results from running ps2pdf

draw-omega.pdf

STEP 5 --- processing PHASE.SH

Use the script draw-phase to create a gnuplot input file.
Leave off the -p option if you just want to view the data
in gnuplot.

```
>> . ./PHASE.SH
>> draw-phase -p INDEXP
>> gnuplot draw-phase.plt
>> ps2pdf draw-phase.ps
```

The following files result from running the script PHASE.SH
An INDEXP file giving info on the pha... files.
A pha... file for each symmetry.

```
INDEXP
pha001
pha002
pha003
pha004
pha005
pha006
pha007
pha008
pha009
pha010
pha011
pha012
pha013
pha014
pha015
pha016
```

The following file results from running the script draw-phase

Consider this as sample gnuplot commands for displaying the data

```
draw-phase.plt
```

The following file results from running gnuplot

```
draw-phase.ps
```

The following file results from running ps2pdf

```
draw-phase.pdf
```

3.8 My run of testcase

To avoid any over-writing it is best to run DARC each time in a separate directory.

For convenience you can specify the global variable `$DARC`. If you are in the directory containing the executables then you can use

```
>> export DARC=$PWD
```

For the testcase (starting in the DARC directory):

```
>> mkdir testrun
>> cp testcase/calc-H-file/*.INP testrun
>> cp testcase/calc-coul/*.INP testrun
>> cd testrun
>> $DARC/dstg1-orbs.exe
>> $DARC/dstg1-ints.exe
>> $DARC/dstg2-njsym.exe
>> $DARC/dstgh-lapack.exe
>> $DARC/dstg3-coul.exe
>> $DARC/dstg4.exe < DSTG4.INP
```

gives

```
-rw-r--r--  1 phn  admin  307280 Jun  2 17:07 DAT1.DAT
-rw-r--r--  1 phn  admin   76908 Jun  2 17:07 DAT2.DAT
-rw-r--r--  1 phn  admin  204800 Jun  2 17:07 DAT3.DAT
-rw-r--r--  1 phn  admin   38224 Jun  2 17:07 DSTG1.DAT
-rw-r--r--  1 phn  admin  741132 Jun  2 17:07 DSTG2.DAT
-rw-r--r--  1 phn  admin    609 Jun  2 17:07 DSTG2.INP
```

```
-rw-r--r-- 1 phn admin 47550 Jun 2 17:07 DSTG2.OUT
-rw-r--r-- 1 phn admin 576160 Jun 2 17:07 DSTG3.DAT
-rw-r--r-- 1 phn admin 139 Jun 2 17:07 DSTG3.INP
-rw-r--r-- 1 phn admin 19735 Jun 2 17:07 DSTG3.OUT
-rw-r--r-- 1 phn admin 25590 Jun 2 17:07 DSTG3.RES
-rw-r--r-- 1 phn admin 46 Jun 2 17:07 DSTG4.INP
-rw-r--r-- 1 phn admin 25873 Jun 2 17:07 DSTG4.OUT
-rw-r--r-- 1 phn admin 247 Jun 2 17:07 DSTG4.PUN
-rw-r--r-- 1 phn admin 70 Jun 2 17:07 DSTGH.INP
-rw-r--r-- 1 phn admin 43233 Jun 2 17:07 DSTGH.OUT
-rw-r--r-- 1 phn admin 258300 Jun 2 17:07 DSTGHF.DAT
-rw-r--r-- 1 phn admin 1290560 Jun 2 17:07 INTEGRAL.DAT
-rw-r--r-- 1 phn admin 30 Jun 2 17:07 INTS.INP
-rw-r--r-- 1 phn admin 10460 Jun 2 17:07 INTS.OUT
-rw-r--r-- 1 phn admin 4232544 Jun 2 17:07 ORBITALS.DAT
-rw-r--r-- 1 phn admin 13048 Jun 2 17:07 ORBS.DAT
-rw-r--r-- 1 phn admin 220 Jun 2 17:07 ORBS.INP
-rw-r--r-- 1 phn admin 32055 Jun 2 17:07 ORBS.OUT
-rw-r--r-- 1 phn admin 180976 Jun 2 17:07 PHASE.SH
-rw-r--r-- 1 phn admin 43540 Jun 2 17:07 TARGET.INP
```

NOTE

I ran the testcase in a single directory. For larger calculations with a number of separate calculations in the external region (for different energy ranges) you would adopt the setup described in the README file.

DARC uses PRINT statements to write to the screen.

The .DAT files are unformatted output with the exception of DSTGHF.DAT and DSTG3.DAT.

Comparing (using diff) DSTG3.RES from directory `testcase/calc-coul` with the file in directory `testrun` gives:

```
2.00000000E-02 2.81936790E-01 | 2.00000000E-02 2.81936791E-01
1.20000000E-01 3.08772977E-01 | 1.20000000E-01 3.08772978E-01
1.30000000E-01 3.11309181E-01 | 1.30000000E-01 3.11309182E-01
1.60000000E-01 3.18771765E-01 | 1.60000000E-01 3.18771766E-01
1.70000000E-01 3.21212732E-01 | 1.70000000E-01 3.21212733E-01
2.30000000E-01 3.35415042E-01 | 2.30000000E-01 3.35415043E-01
4.20000000E-01 3.76550694E-01 | 4.20000000E-01 3.76550693E-01
4.70000000E-01 3.86666505E-01 | 4.70000000E-01 3.86666504E-01
5.70000000E-01 4.06226930E-01 | 5.70000000E-01 4.06226929E-01
7.70000000E-01 4.42939307E-01 | 7.70000000E-01 4.42939308E-01
8.00000000E-01 4.48171478E-01 | 8.00000000E-01 4.48171477E-01
8.30000000E-01 4.53329311E-01 | 8.30000000E-01 4.53329310E-01
1.06000000E+00 4.90283738E-01 | 1.06000000E+00 4.90283737E-01
```

1.29000000E+00	5.22668503E-01		1.29000000E+00	5.22668504E-01
1.31000000E+00	5.25297569E-01		1.31000000E+00	5.25297568E-01
1.67000000E+00	5.73203678E-01		1.67000000E+00	5.73203677E-01
1.68000000E+00	5.74967322E-01		1.68000000E+00	5.74967321E-01
2.05000000E+00	5.23952847E-01		2.05000000E+00	5.23952846E-01
2.09000000E+00	6.16260591E-01		2.09000000E+00	6.16260590E-01
2.30000000E+00	6.05193736E-01		2.30000000E+00	6.05193735E-01
2.39000000E+00	6.09816373E-01		2.39000000E+00	6.09816372E-01
2.48000000E+00	6.13857332E-01		2.48000000E+00	6.13857331E-01
2.69000000E+00	6.21027193E-01		2.69000000E+00	6.21027192E-01
3.24000000E+00	6.26651813E-01		3.24000000E+00	6.26651814E-01
3.56000000E+00	6.23113351E-01		3.56000000E+00	6.23113350E-01
3.92000000E+00	6.14248603E-01		3.92000000E+00	6.14248602E-01
3.95000000E+00	9.63270333E-01		3.95000000E+00	9.63270332E-01
3.96000000E+00	9.65894962E-01		3.96000000E+00	9.65894961E-01

The differences are small.

NOTE

Here we used `dstg2-njsym` rather than `dstg2-njgraf`. Although it is slower I have a preference for the `njsym` version. It is good to have two versions anyway as a check. If the testcase is run with `dstgh-hsldr` rather than `dstgh-lapack` then the results show greater differences. But still you would be happy with the agreement.

3.9 Script xsummary

There is a slight problem with this script that I have corrected.

3.10 Fe XXII testcase

The shell script `darctest-fe23` runs this testcase.

Just typing `darctest-fe23` gives the following information:

```
Usage: darctest-fe23 <dir>
```

```
Runs a DARC case in directory <dir>
```

```
If dir is set to input then only the input data is written
```

```
DARC calculation
```

```
electron + Fe XXIII, 10 state, 20 symmetries
```

```
dstg1-orbs
```

```
|
dstg1-ints
|
dstg2-njsym
|
dstgh-lapack
|
stgfjj
|
dstg4
```

Patrick Norrington
1 July 2008

If you specify a directory name as an argument to the script then the calculations are run in that directory.

By typing

```
>> darctest-fe23 input
```

a directory `input` is created and the input files are written to it. These files are useful templates.

NOTE

The executables actually have the name `$DARC/dstg1-orbs.exe` for example. This means that you must specify the global variable `$DARC`. If you are in the directory containing the executables then you can use

```
>> export DARC=$PWD
```

Also the script `xsummary` should be in this directory.

Below is a listing of the files generated when I ran the script in the directory `fe23-gfortran-00`.

```
=====
tree for fe23-gfortran-00
=====

fe23-gfortran-00
" calc
" stgfjj1
" stgfjj2

=====
```

list for fe23-gfortran-00

=====

```
--          108 -- Jun  3 10:50 -- DSTG4.INP
--        41165 -- Jun  3 10:52 -- calc.out
--       23092 -- Jun  3 10:52 -- calc.sum
--          74 -- Jun  3 10:54 -- stgfjj1.out
--     550170 -- Jun  3 10:54 -- stgfjj1.res
--          74 -- Jun  3 10:54 -- stgfjj2.out
--     45765 -- Jun  3 10:54 -- stgfjj2.res
```

fe23-gfortran-00/calc:

```
--        63644 -- Jun  3 10:50 -- DSTG1.DAT
--    23424044 -- Jun  3 10:50 -- DSTG2.DAT
--         769 -- Jun  3 10:50 -- DSTG2.INP
--       75524 -- Jun  3 10:50 -- DSTG2.OUT
--          73 -- Jun  3 10:50 -- DSTGH.INP
--     164945 -- Jun  3 10:52 -- DSTGH.OUT
--    4991137 -- Jun  3 10:52 -- DSTGHF.DAT
--    3305792 -- Jun  3 10:50 -- INTEGRAL.DAT
--          30 -- Jun  3 10:50 -- INTS.INP
--     12572 -- Jun  3 10:50 -- INTS.OUT
--    7967008 -- Jun  3 10:50 -- ORBITALS.DAT
--     24200 -- Jun  3 10:50 -- ORBS.DAT
--         223 -- Jun  3 10:50 -- ORBS.INP
--     48934 -- Jun  3 10:50 -- ORBS.OUT
--     38500 -- Jun  3 10:50 -- TARGET.INP
```

fe23-gfortran-00/stgfjj1:

```
--    14560365 -- Jun  3 10:54 -- DSTG3.DAT
--         2551 -- Jun  3 10:54 -- DSTG4.LIS
--    141642 -- Jun  3 10:54 -- DSTG4.OUT
--         262 -- Jun  3 10:54 -- DSTG4.PUN
--          18 -- Jun  3 10:52 -- DSTGHF.DAT -> ../calc/DSTGHF.DAT
--     591212 -- Jun  3 10:54 -- OMEGA
--    189007 -- Jun  3 10:54 -- OMJJ
--         252 -- Jun  3 10:50 -- STGF.INP
--    2789127 -- Jun  3 10:54 -- STGF.OUT
--    1042556 -- Jun  3 10:54 -- STGF.PUN
```

fe23-gfortran-00/stgfjj2:

```
--    393485 -- Jun  3 10:54 -- DSTG3.DAT
--         2551 -- Jun  3 10:54 -- DSTG4.LIS
--     9256 -- Jun  3 10:54 -- DSTG4.OUT
--         262 -- Jun  3 10:54 -- DSTG4.PUN
--          18 -- Jun  3 10:54 -- DSTGHF.DAT -> ../calc/DSTGHF.DAT
```

```
--      16169 -- Jun  3 10:54 -- OMEGA
--      13355 -- Jun  3 10:54 -- OMJJ
--         254 -- Jun  3 10:50 -- STGF.INP
--      50733 -- Jun  3 10:54 -- STGF.OUT
--      15132 -- Jun  3 10:54 -- STGF.PUN
```

=====

The modules given above in the Fe XXIII testcase are the modules of choice for production work. This version of stgfj works only for targets that are ions.

3.11 Optimisation

These are timings in seconds for dstgh-lapack for the Fe XXIII testcase. This involves diagonalising 20 matrices. They have been run 3 times each to test for reproducibility.

```
gfortran -O0    86.44 86.01 86.16
gfortran -O1    16.26 16.11 16.10
gfortran -O2    21.12 15.28 15.21

ifort -O0      125.05 88.43 90.52
ifort -O1      12.92 13.04 12.93
ifort -O2      14.97 10.21  9.73
ifort -fast     8.91  8.98  8.98
```

-O1 optimisation seems a safe choice.

3.12 Test of dstg0

This module is used to generate the file TARGET.INP that contains the bound SPFs (single particle functions). It is required input to dstg1-orbs.

As a test you can use MCDF.DAT from `grasp0-tests` testcase 5 (Fe XXIII) and run dstg0 interactively. The result can be compared with the supplied TARGET.INP in the `darctest-fe23` script. They are not identical because they were generated with different compilers/optimisations. However the differences should be small.

4 Updates

The files in the tar file `update01.tar` are:

```
-rw-r--r--    1 phn  admin    3227 Jun  1 17:49 CALEN.f
-rw-r--r--    1 phn  admin    3317 Jun  2 16:06 QUARTZ.f
-rwxr-xr-x    1 phn  admin   43601 Jun  3 10:49 darctest-fe23
-rw-r--r--    1 phn  admin   23092 Jun  3 10:52 darctest-fe23-gfortran-00-calc.sum
-rw-r--r--    1 phn  admin  550170 Jun  3 10:54 darctest-fe23-gfortran-00-stgfjj1.res
-rw-r--r--    1 phn  admin   45765 Jun  3 10:54 darctest-fe23-gfortran-00-stgfjj2.res
-rwxr-xr-x    1 phn  admin    3815 Jun  1 13:49 grasp0-tests
-rw-r--r--    1 phn  admin  265934 Jun  1 18:01 grasp0-tests-gfortran-00.out
-rwxr-xr-x    1 phn  admin    9333 Jun  1 21:59 xsummary
```

You should extract the files in a separate directory.

```
>> mkdir update01
>> mv update01.tar update01
>> cd update01
>> tar xf update01.tar
```

Briefly:

- CALEN.f and QUARTZ.f are the GRASP0 versions that work with gfortran and ifort.
- xsummary is the corrected version.
- grasp0-tests is the GRASP0 testcase script.
- darctest-fe23 is the DARC testcase script.
- Sample output is given from executables compiled with gfortran -O0.