

Lecture 3 - Programming

- 1 On Fortran, C/C++ and scripting languages.
- 2 Writing efficient scientific programs.
- 3 Data locality.
- 4 Blocked algorithms.
- 5 Solving linear algebra problems on computers.
- 6 Matrix transformations.
- 7 Mathematical libraries.

Programming languages: Fortran

- fortran 77
 - Traditional language of HPC.
 - Easy to start programming in. Efficient compilers.
 - Unsuitable for writing very large programs.
 - Compilers do very little syntax error checking.
- Fortran 90
 - Can perform operations on entire vectors or matrices.
 - Modules for better program structure and error checking.
 - Becomes more commonspread only now, 16 years since inception.

Programming languages: C/C++

- C: matrix operations are not as easily expressed as in f90.
- C++ – very flexible language (with both + and –). May be convenient for new projects but requires care and effort to link against old f77 code.
- Scripting languages: awk, Perl, Python. Very useful glue and for data extraction. Shell scripts.

Scripting with awk and perl

- print nicely electric field and corresponding energy.

```
.....  
INMA  electric_field_x := 0.0002  
.....  
RESC FINAL ENERGY:      -45071.34094404266  
.....
```

Solution:

```
awk '/electric_field_x :/{f=$4} /FINAL/{print f, $4}
```

Perl power

- Perl allows for more complex operations:

```
perl -pi -e 's,field=0.1,field=0.2,g' input_f
```

- file opening and processing:

```
open FL, 'FNAME' or die;
while(<FL>) {
    print FL if /DIPLLEN/;
}
close FL;
```

```
while($a = <FL>) {
    print FL, $a if $a =~
```

Using compilers

-O works everywhere. Some compilers support `-fast`.

- Intel Fortran Compiler (f95): `ifc`, `efc`, `ifort`
- Intel C++ Compiler: `icc`, `ecc`
 - `-xW` – use SSE2 instructions (Pentium4 only): vector processing.
 - `-tpp7` may be useful.
- `pgf90/pgcc`: `-fast` and `-fastsse`
- `g77/gcc`: `-march=pentium4 -malign-double -O3`
 - portable, *free* compiler of good quality.

Writing efficient programs – priorities

- make it *right*.
- (... nothing, nothing, ...)
- make it fast.
- Use existing libraries: do not reinvent the wheel!
- Scientific programs are difficult to write: document them!
Too much code has been covered by dust because nobody understood it any more.

Data: locality and dependence.

- matrix multiplication (700x700):

```
DO I=1, N                                DO J=1, N
  DO K=1, N                                DO K=1, N
    DO J=1, N                                DO I=1, N
      C(I,J) = C(I,J) + A(I,K)*B(K,I,J) = C(I,J) + A(I,K)
    END DO
  END DO
END DO                                    END DO
END DO                                    END DO
```

6.2s

DGEMM/BLAS/atlas: **0.215s**

1.1s

Solving linear algebra

- use libraries!
 - Basic Linear Algebra Subroutines (BLAS): $V \times V$, $M \times V$, $M \times M$
 - Linear Algebra PACKage (LAPACK): eigenvalue problems etc.
- optimized for hardware (cache size, etc): ATLAS, MKL.
- can exploit symmetry of the problem (symmetric, hermitian, banded).
- `-lf77blas -latlas, -lessl`
- <http://www.netlib.org/>

BLAS – details

Meaning of prefixes S – REAL, D – DOUBLE PRECISION, C – COMPLEX, Z – COMPLEX*16

Matrix types : GE – GEneral, GB – General Band, SY – SYmmetric, SB – Symmetric Band, SP – Symmetric Packed, HE – HErmitian, HB – Hermitian Band, HP – Hermitian Packed, TR – TRiangular, TB – Triangular. Band, TP – Triangular Packed

Options **TRANx** – 'No transpose', 'Transpose', 'Conjugate transpose'
 UPLO – 'Upper triangular', 'Lower triangular'
 DIAG – 'Non-unit triangular', 'Unit triangular'
 SIDE – 'Left', 'Right'

Typical quantum chemistry calculation

```
./dalton dal mol
```

dal – file describing calculation type (HF,DFT, MC-SCF etc).

mol – file describing the molecule to perform the calculation on.

- Some programs have just one file containing both these parts.
- Temporary files are created often to reduce memory usage – make sure the temporary directory is set properly.

Program compilation – makefiles

- large programs consist of many files.
- files need to be compiled in certain order but...
- not all files should be recompiled on single modification and...
- it should be possible to compile files in parallel.

The answer is program `make` – controlled by `makefiles`.

- `makefile` contains variable definitions and dependencies between files.
- some dependencies can be generated automatically.

Typical makefile

```
FORT = f77
CC   = cc
COPTS = -O3 -march=pentium4
FOPTS = -O3 -march=pentium4 -funroll-loops
```

```
program: prog-main.o helpers.o
    $(FORT) -o $@ $^
```

```
prog-main.o: prog-main.f
    $(FORT) $(FOPTS) -c $^
```

```
helpers.o: helpers.c
    $(CC) $(CFLAGS) -c $^
```

Usually, modification of few variables is needed.

Exercise

Use your favorite tool (awk, perl) to extract

- 1 basis set name,
- 2 number of basis functions,
- 3 SCF energy

from the output file of a quantum-chemical program of your choice. If you do not have one, contact me for a test output.