Assignment 1.

Suggestion: Always use “implicit none” as the second line of the program to help the program detect undeclared variables.

```
Program ....
implicit none
...
```

1. Write a program that prints

```
        2 to the power of 20 is xxx
```

where xxx is replaced by the value of 2^{20}.

2. Use appropriate loops for programs that calculate the following sums:

    a) 1 + .5 + .25 + ...
    b) 1 + .5 + .25 + ... +xxx, where xxx>10^{-3}.
    c) 55+56+...+100

3. Read an integer variable i and a real variable x from the console repeatedly . Depending on the value of the integer variable, print the following:

integer	action
1	logarithm of x
2	sine of x
3	square of x
“else”	“quit the program”

Use the DO loop without arguments and the CASE statement.

Note: To read variables x and y from the keyboard, type
read*,x,y

4. Browse through a list of Fortran functions in your manual. By writing simple programs, determine differences between int, nint, floor and ceiling.

5. Print the following number:
1.0/3.0**(-i), i=1,10
in F, E and G formats using specifications 10.3 and 12.6.

Optional

6. Create a file with a pedigree line that contains animal ID, sire ID, dam ID and year of birth:
HOL234HOL001HOL9831998
Write a programs that will read the IDs and year of birth into separate variables.

7. Write a program that

- reads an alphanumeric variable of length up to 10,
- for each character of that variable, writes a numeric equivalent of that character.

The numeric equivalent of character c is ichar(c).

Assignment 2.

1. Set matrices A and B to:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 1 \\ 4 & 8 \end{bmatrix}$$

Calculate C=A*B as matrix multiplication using loops and matmul. Print. Do results differ?

Write the sum of A as a whole, along dimension 1, and along dimension 2.

Print the first column of A.

Count how many numbers in A+B are larger than 5 and compute their sum; do not use any loop.

2. Initialize matrix x(100,100) to x_{ij}=100*i+j-1. Create an internal function

$$tr(n,m) = \sum_{i=n}^{m} x_{ii}$$

and calculate tr(1,10) and tr(90,100).

3. In the following data (to be read from a file), a line containing the dimensions of a 2-dimensional matrix is followed by the data values of this matrix.

```
 2
1.5 3.2
2.7 3.3
4
4 3 2 1
5 4 3 2
6 5 4 3
7 6 5 4
1
10
```

Create the file above. Write a program using the dynamic memory allocation that will:

a) read the dimension
b) allocate a real matrix with the given dimensions
c) read that matrix
d) print that matrix
e) deallocate the matrix
f) continue until the end of file

Optional

4. Create a vector x of 100,000 reals and initialize x(i)=i+1.0/3.0. Store it in files in two formats:
   a) formatted
   b) unformatted

Then read the files and accumulate the sums. Compare timings of either reading using the CPU_TIME subroutine.
Close files with options to delete them.

3. Write to file the following:

```
1 : printed  1 number(s)
1 2 : printed  2 number(s)
...
1 2 3 4 5: printed 10 number(s)
```

without using more than 2 write statements. Use clause ADVANCE='NO'.

Assignment 3.

1. Read descriptions of array functions. Create examples for: maxval, maxloc, pack, reshape, size and unpack.

2. Let A and b be:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$$

Write a function that calculates the quadratic form: b'Ab. Make this function:

a) external,
b) internal,
c) module

3. Write a function that calculates a sample from a pseudo-normal distribution using a formula:

$$\sum_{i=1}^{12} u_i - 6, \quad u_i \sim UN(0,1)$$

Samples from the UN(0,1) distribution are generated by subroutine random_number().

Write a subroutine that, given a vector with samples, calculates mean and SD

Write a program that asks for a number of samples to be generated, generates those samples from a) UN(0,1) and b) N(0,1), and then calculates their mean and SD.

4. Modify function that generates normal samples to accept parameters for mean and variance. Make this parameters optional with default mean=0 and variance=1.

Optional

5. Create a vector x of 100,000 reals and initialize x(i)=i+1.0/3.0. Store it in files in two formats:
   a) formatted
   b) unformatted

Then read the files and accumulate the sums. Compare timings of either reading using the CPU_TIME subroutine.
Close files with options to delete them.

6. Write a subroutine printnice_r(x,n), that will print a real matrix x(n,n) with nicely aligned columns.

7. Modify function printnice
   a) putting it in a module as an internal procedure
   b) eliminating the n argument
   a) adding an optional argument, which is format for one line

Assignment 4.

1.Write subrouties printnice_int and printnice_real that print real and integer matrices. Overload so that printnice prints both. Test.

2. Put main program and modules form previous assignment in separate files. Write Makefile to compile.

3. Check whether subroutine random_number is elemental; this can be done as part of assignment from yesterday.

4. The program contains the data on animals in structure animal. Create a subroutine that given a variable of type animal will print the information on that animal.

```
module animals
type calf
  character (10) :: id
  integer :: year_of_birth
  character:: sex
end type
end module

program registration
use animals
implicit none
type (calf) :: a,b,c

a=calf('small',1997,'M')
! b is also a male born in 1997 but has a different name
b=a
b%id='large'
```

Write subroutine print_am that prints ID, year of birth and sex, with syntax as below:

```
call print_am(a)
call print_am(b)
```

Optional

3. This assignment tests data structures for sparse vectors, operations on data vectors, and operator overloading. All definitions and procedures need to reside in module sparse_vec. Write:
a) data structure for sparse vector
b) subroutine that creates sparse vector from a dense vector
c) subroutine that prints a sparse vector
d) subroutine that multiplies two sparse vectors.

Provide interface to c) with "=" operator
Provide interface to d) with "+" operator

To the last assignment, add interfaces to "+" and "*" that allow to add or multiply sparse vector by a regular vector.

Assignment 5.

1. Run program lsq.f90. This program with related data sets can be found in directory listed on board. Modify the program so that it can read a parameter file containing: name of data file, number of effects and number of levels. The parameter file could contain the following:

```
data_pr1            !name of the data file
2                   !number of effects
2 3                 !number of levels for each trait
```

2. Apply the modified program to:
   a) the original data
   b) the data as below

management	animal	treatment	record
1	1	1	12
1	2	2	14
1	3	1	13
2	3	1	16
2	2	2	18
3	4	1	15
3	3	2	16
3	2	2	19

Optional

3. Think how the parameter file can be simplified so that the number of effects is not give, i.e., the above parameter file simplifies to:

```
data_pr1            !name of the data file
2 3                 !number of levels for each trait; defines the number of traits
```

hint:when the reading is unsuccessful, for example because of too few numbers, the variable associated with iostat becomes $\neq 0$.

4. Examine differences between lsq.f90 and lsqmt.f90. Run the latter. Change R_0 to these matrices:

$$R_0 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \begin{bmatrix} 10 & 0 \\ 0 & 20 \end{bmatrix}$$

and compare solutions. Should they be different?

Assignment 6.

1. Write a program that creates the inverse of the animal relationship matrix. Use the following pedigree:

animal	sire	dam
1	3	4
2	3	4
3	5	-
4	-	-

2. Incorporate the program above in lsq.f90. Run it with data of the yesterday's assignment and pedigrees as above assuming var(e)=**I**10 and var(animal)=

a) **I**2.

b) **A**2

Compare solutions.

3. Modify parameters in blup.f90 to support the same models as above. Compare solutions. Estimable functions should be identical.

Optional

4. Modify parameters in blup.f90 for a maternal model using parameters from 2b. Possibly compare solutions against another program.
Hint: If the dam is not identified, create a phantom dam.

5. In 3, make management autocorrelated with var(management)=5 and ρ=.8. Then, vary the autocorrelation to 0.98 and 0.9 and compare solutions for the management effects.

Assignment 7

DENSEOP

1. This assignment involves module DENSEOP, which is described at http://nce.ads.uga.edu/~ignacy/newprograms.html. An example of the use of this module is in directory /home/ads-guest6/cdf90/testdense. Because of complexity, the program inside the directory needs to be compiled by the command:

```
make
```

For make to work, copy file Makefile as well as all the other files from the same directory.

Examine contents of kind.f90 lapack90r.f90 denseop.f90. They are in directory /home/ads-guest6/f90/libs.

Consider the following system of equations Ax=b:

$$\begin{bmatrix} 36 & 30 & 18 \\ 30 & 41 & 23 \\ 18 & 23 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 150 \\ 181 \\ 106 \end{bmatrix}$$

Using dense matrix module DENSEOP and precision (r8):

a) calculate Cholesky decomposition A=LL' and check its correctness by matrix multiplication,

b) calculate eigenvalues and eigenvectors: A=VDV, and verify that VV'=I,

c) calculate the determinant of A

d) solve for x using both symmetric and general solvers; confirm the correctness of solutions by multiplication

e) Calculate the inverse of A and verify that AA^{-1}=I.

2. Repeat solving in different data formats: dense and half stored, single and double precision. To create a half stored matrix, convert a full-stored matrix.

SPARSEM, SPARSEOP

3. The assignments include the sparse matrix module SPARSEM. An example of the use of this module is in file test.f90 in directory directory /home/ads-guest6/cdf90/testsparse. Because of complexity, the program inside this directory program needs to be compiled by the command:

```
make
```

For make to work, copy file Makefile as well as all the other files from the same directory.

For programs with name other than test, make and Makefile can still be used, but replace "test" with the name of your program in file Makefile.

For Windows, the following files need to be compiled in this particular order:
kind.f90 sparse2.f sparse.f90 fspak90.f90 fspak.f fspaksub.f sparssub.f second.f

4. Examine data structures for different matrix formats in file sparse.f90. Examine procedures implementing **zerom**. They are in directory /home/ads-guest6/f90/libs.

5. Consider the following system of equations Ax=b:

$$\begin{bmatrix} 36 & 30 & 18 \\ 30 & 41 & 23 \\ 18 & 23 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 150 \\ 181 \\ 106 \end{bmatrix}$$

a) Set this system of equation with densem, dense_symm, and sparse_hash matrices,
b) convert the format sparse_hash to sparse_ija,
c) print all matrices,
c) solve by iteration using the four data types,
d) print sparse matrices in internal format (printm(x,'internal')

Optional

6. Consider the following matrix A of dimension n x n:

$$a_{ii} = 20.0+\mathrm{mod}(i,3), \quad a_{ij} = a_{ji} = \begin{cases} \mathrm{rnd}-.5 & \text{for } j=1,\ 1+p,\ 1+2p,\ ..,\ n \\ 0 & \text{otherwise} \end{cases}$$

where rnd is a random number distributed in the range <0,1), p=max(1,n/10), and the vector of right hand sides is generated as:

$y_i = 20.0+\mathrm{mod}(i,6)$

Random number x can be generated by subroutine random_number(x). ATTENTION: Matrix X may not be positive definite for large n!

Using modules sparsem and sparseop, write a program that:

a) sets up a matrix A and vector y,

b) solves the system of equations A x = y

- iteratively
- by FSPAK90 (sparse storage only)

for n=10 and n=500, and densem and sparse_hash matrix types. With FSPAK90, the sparse_hash matrix needs to be converted to sparse_ija matrix.

For n=500, record CPU time, which under UNIX can be obtained by preceding the executable by command time, e.g.,

```
time a.out
```

Additionally, for each format:

c) print the contents of the matrix (n=10) or its section A(120:125,120:125) (n=500)

e) find the determinant of A

4. Modify program BLUP.F90 so that the coefficient matrix is stored and solved as a sparse matrix structure.

Assignment 8

Create a directory f90 and download file progsf90.tar.gz. This file is available at /home/ads-guest6/cdf90 or at http://nce.ads.uga.edu/~ignacy/newprograms.html, by clicking on "package". Uncompress this file into a newly created directory by:

```
tar xzvf  progsf90.tar.gz
```

Then compile all the libraries and programs:

```
make linux-intel
make all
```

Each program has its own directory and its own Makefile. To recompile any program separately, go to its directory and type

```
make
```

For Windows installation, see the end of this page.

1. Read:
   a) Readme, Installation, and Makefile files in the main directory,
   b) Readme in directory libs. Identify programs that implement reading the parameters, sparse matrix factorization, and hash function.

2. Run blupf90 with examples specified in Appendices A-C with 3 solving options:
   a) by default iteration (PCG),
   b) by Gauss-Seidel iteration,
   c) by FSPAK.

For details how to change the solving options, see blupf90.f90 and readme.blupf90 in the blupf90 directory. Also look follow links to programs and FAQ at http://nce.ads.uga.edu/html/projects/projects.html.
The parameter files start or end with letters ex. Examples are stored in the directory examples.

3. Prepare parameter file for a data set used with an assignment for lsq.f90 and run blupf90.

Optional

3. Modify blupf90.f90 so that it calculates accuracies for the animal effect. Accuracies for an animal in equation i in a single-trait situation is defined as:

$$1 - LHS^{ii}/\sigma_a^2$$

where LHS^{ii} is the i-th diagonal of the inverse of the left hand side.

Suggestion: Invert by FSPAK and obtain elements of the inverse by function "getm" from module sparsem.

Assignment 9

Parameter files for this exercises are in directory example.

Files ending with *99 contain data files for up to 14 traits. Parameter file exmr91 uses these files for a single-trait model ...((conatin

Calculate estimates of variance components by remlf90 and airemlf90 using the parameter file exmr99s1. Record the number of rounds and CPU time. Extend the model to 2 traits by adding the observations in column 4 (parameter file exmr99s2). Repeat the computations for AIREMLF90 only. How much slower is REMLF90 and how longer are the computations in the two-trait case?

4. Change variances in parameter file exmr99s1 to very small small values (approximately 10 times smaller than now) and very large values (10 times bigger). Check convergence of REMLF90 and AIREMLF90.

Optional

5. The data file that are is used in parameter files exmr99* contains data fields for 14 traits. Try extending the number of traits in analyses beyond 3. At what number of traits is the algorithm becoming unstable? For a number of traits that it starts to be unstable, try REMLF90.

6. Convert REMLF90 to using the faster single-trait formula for the variance components, as described in the notes. Compare its efficiency for exmr99s1 and exmr99s2.

Running programs with a larger number of traits can be done in groups so as not to overload the computer.

Assignment 11

4. Read the documentation on gibbs sampling programs in subdirectories gibbs, gibbs1, gibbs2 and gibbs3.

5. Run gibbsf90 and gibbs2f90 with a single trait example exmr99s1 (1 traits) and exmr99s (3 traits) for 100 rounds. Compare speed.

6. Run gibbs1f90 for a 3-trait example; use the number of samples 1000 and burn-in 200. Run program postgibbsf90. For graphical output, postgibbsf90 requires a plotting package GNUPLOT and X Windows (e.g., as provided by X emulation packages or by free VNC).

Attention: For reasonable results, usually the number of samples needs to be in the range

Assignment 12

1. Read Readme* in directory thrgibbs1.
    a) identify subroutines that approximate liabilities and thresholds
    b) generate the data using program gen_thr.f90.
    c) estimate parameters using thrgibbs1f90. Compare with assumed values.
    d) repeat b-c for a number of designs and parameters. How many records are necessary for good estimates with different heritabilities?

Warning*:* *gen_thr may not work on some systems. where random_number is not treated as an elemental function. In this case, use gen_thr_ibm.f90.*

2. Convert data in exmr99s to result in one trait binary, one with 3 categories, and one linear. Estimate parameters.

Use postgibbsf90 to determine the number of needed samples.

Optional

3. Estimate parameters of 1bc using cblup90reml.f90 (quasi-REML approach).

4. Modify gibbsf90 to a single trait threshold model for binary data. Assume a threshold of 0. Test.

Suggestions: The residual variance needs to be fixed at 1 and not estimated. Subroutine "predict_missing_values" should be modified to sample from the interval -∞:0 for category 1 and 0..∞for category 2.

5, Listen to Shogo's presentation on renumf90.