

CHS FORTRAN library Project

What's the aim?

The goal is to provide a library of general Fortran modules.

What is already there?

Alphabetical ordered modules:

Name	One-Line description
mo_anneal	Optimization algorithm Simulated Annealing & estimation of initial temperature
mo_append	Appends vectors and/ or matrixes like bash "cat" and "paste"
mo_boxcox	Box-Cox transformation, inverse transformation & estimating best exponent for transformation
mo_combinatorics	Combinatorial algorithms, e.g. binomial coefficient and k-subsets
mo_constants	Mathematical and physical constants
mo_corr	Correlation function with Fast Fourier Transform, Auto- & Crosscorrelations via direct calculation
mo_dds	Dynamically Dimensioned Search (DDS) and Modified DDS (MDDS)
mo_elemeffects	Calculation of parameter's Elementary Effect on a model/ function output
mo_errormeasures	Distance or error measures between two datasets, e.g. bias, RMSE
mo_file_utils	Utilities for file handling, e.g. search free unit
mo_finish	Routine to end program gracefully
mo_fit	Linear & polynomial fitting, and general fit with singular value decomposition
mo_histo	Histogram of data (useable also for variogram)
mo_integrate	Integration routines
mo_interpol	Linear interpolation for irregular grids
mo_julian	Converts Julian Day into Day, Month and Year, and vice versa; Standard and IMSL convention
mo_kernel	Kernel regression and kernel density estimation for PDF and CDF
mo_kind	Definition of number precision
mo_mad	Median absolute deviation test
mo_mcmc	Monte Carlo Markov Chain sampling of parameter distribution around optimum
mo_message	Write out message; works with num2str from mo_string_utils
mo_minpack	Optimization package minpack (F90 interfaces)
mo_moment	1st to 4th moments, central and mixed central moments
mo_mvn	Generating Multivariate Normal Random Number with given Covariance
mo_ncread	Reading nc files using the netcdf4 library
mo_ncwrite	Writing nc files using the netcdf4 library
mo_nelmin	Minimizes a function using the Nelder-Mead algorithm with the Applied Statistics algorithms No. 047.
mo_nml	Routines to handle namelist files
mo_nr	Main numerical recipes module containing the interfaces
mo_nrutil	Numerical recipes utilities module
mo_ode_solver	Iterative methods for the approximation of solutions of Ordinary Differential Equations (ODE)
mo_opt_functions	Test functions for optimization routines
mo_percentile	Median, Percentiles
mo_poly	Tests if a given 2D point lies inside, outside, or on edge/vertex of a 2D polygon, compute area and center of mass

CHS FORTRAN library Project

[What's the aim?](#)

[What is already there?](#)

[Note on Numerical Recipes](#)

[How to checkout the complete FORTRAN lib?](#)

[How to contribute to the FORTRAN lib?](#)

[License](#)

[Note on Numerical Recipes License](#)

mo_remap	Remaps a grid to another grid
mo_sce	Optimization routine Shuffled Complex Evolution
mo_sort	Quicksort arrays or indices
mo_string_utils	Utilities for strings
mo_template	Module template demonstrating the coding standard of the library
mo_timer	Cpu time routines to allowing setting of multiple CPU timers
mo_xor4096	Generating Uniform/ Gaussian Random Numbers using the xor4096 algorithm
mo_xor4096_apps	Wrapper functions for Random number generator xor4096

Categorical ordered modules:

<i>Category</i>	
<i>Name</i>	<i>One-Line description</i>
Date/ Time	
mo_julian	Converts Julian Day into Day, Month and Year, and vice versa; Standard and IMSL convention
mo_timer	Cpu time routines to allowing setting of multiple CPU timers
Input/ Output	
mo_file_utils	Utilities for file handling, e.g. search free unit
mo_ncread	Reading nc files using the netcdf4 library
mo_ncwrite	Writing nc files using the netcdf4 library
mo_nml	Routines to handle namelist files
Optimization / Fitting	
mo_anneal	Optimization algorithm Simulated Annealing & estimation of initial temperature
mo_dds	Dynamically Dimensioned Search (DDS) and Modified DDS (MDDS)
mo_elemeffects	Calculation of parameter's Elementary Effect on a model/ function output
mo_fit	Linear & polynomial fitting, and general fit with singular value decomposition
mo_kernel	Kernel regression and kernel density estimation for PDF and CDF
mo_mcmc	Monte Carlo Markov Chain sampling of parameter distribution around optimum
mo_minpack	Optimization package minpack (F90 interfaces)
mo_nelmin	Minimizes a function using the Nelder-Mead algorithm with the Applied Statistics algorithms No. 047.
mo_opt_functions	Test functions for optimization routines
mo_sce	Optimization routine Shuffled Complex Evolution
Miscellaneous	
mo_append	Appends vectors and/ or matrixes like bash "cat" and "paste"
mo_constants	Mathematical and physical constants
mo_finish	Routine to end program gracefully
mo_kind	Definition of number precision
mo_message	Write out message; works with num2str from mo_string_utils
mo_nr	Main numerical recipes module containing the interfaces
mo_nrutil	Numerical recipes utilities module
mo_poly	Tests if a given 2D point lies inside, outside, or on edge/vertex of a 2D polygon, compute area and center of mass
mo_remap	Remaps a grid to another grid
mo_string_utils	Utilities for strings
mo_template	Module template demonstrating the coding standard of the library
Math	
mo_integrate	Integration routines
mo_interpol	Linear interpolation for irregular grids
mo_mvn	Generating Multivariate Normal Random Number with given Covariance

mo_ode_solver	Iterative methods for the approximation of solutions of Ordinary Differential Equations (ODE)
mo_sort	Quicksort arrays or indices
mo_xor4096	Generating Uniform/ Gaussian Random Numbers using the xor4096 algorithm
mo_xor4096_apps	Wrapper functions for Random number generator xor4096
Statistics	
mo_boxcox	Box-Cox transformation, inverse transformation & estimating best exponent for transformation
mo_combinatorics	Combinatorial algorithms, e.g. binomial coefficient and k-subsets
mo_corr	Correlation function with Fast Fourier Transform, Auto- & Crosscorrelations via direct calculation
mo_errormeasures	Distance or error measures between two datasets, e.g. bias, RMSE
mo_histo	Histogram of data (useable also for variogram)
mo_mad	Median absolute deviation test
mo_moment	1st to 4th moments, central and mixed central moments
mo_percentile	Median, Percentiles

Please update the two tables above, if you uploaded a tested, working module. If your module is still not documented or in beta-version please do NOT list it in the tables.

Note on Numerical Recipes

- The directory [nr_ori](#) contains the original *Numerical Recipes in Fortran 90* routines. It also includes PDFs of the two books *Numerical recipes in Fortran 77* and *Numerical recipes in Fortran90*. There is also a Windows help file *NR9F206H.HLP*.
- The directory [nr_chs](#) contains copies of the routines that are already ported to the standard of this library. To use them: copy *mo_kind.f90*, *mo_constant.f90*, *mo_nrutil.f90*, and *mo_nr.f90* to your code. Also copy the routines needed from the *nr_chs* directory, e.g. *gamma.f90*. Then include in your code for the gamma-example:
USE mo_nr, only: gamma
- To add a new subroutine that is not yet in *nr_chs*, do the following:
 - Copy routine from *nr_ori* to *nr_chs* and move it from *nr_ori* to *nr_ori/in_nr_chs*.
 - In new routine
 - Change *nrtype* to *mo_kind*.
 - Change *nrutil* to *mo_nrutil*.
 - Change *nr* to *mo_nr*.
 - Change *I?B* to *I?*.
 - Add *USE mo_constants* if constants such as *PI* are used.
 - Make *sp* and *dp* versions.
 - Look for the interface in *mo_nr.f90* and change it accordingly, for example add the *dp* version.
 - If you change input/output or similar of the routine, the original documentation in the PDF files is not valid anymore.
Please add the documentation structure from *mo_template.f90* or consider making a completely new module (example *mo_sort.f90*).
- Be aware that the code were acquired under *Numerical Recipes Personal Single-User License*. This license lets you personally use Numerical Recipes code ("the code") on any number of computers, but only one computer at a time. You are not permitted to allow anyone else to access or use the code. You may, under this license, transfer precompiled, executable applications incorporating the code to other, unlicensed, persons, providing that (i) the application is noncommercial (i.e., does not involve the selling or licensing of the application for a fee), and (ii) the application was first developed, compiled, and successfully run by you, and (iii) the code is bound into the application in such a manner that it cannot be accessed as individual routines and cannot practicably be unbound and used in other programs. That is, under this license, your application user must not be able to use Numerical Recipes code as part of a program library or "mix and match" workbench.

How to checkout the complete FORTRAN lib?

To checkout the library in a local directory also called FORTRAN_chs_lib:

```
svn checkout https://svn.ufz.de/svn/chs-svn/FORTRAN_chs_lib/
```

To checkout into a local folder with the local name "local_name", which will be created if it does not exist yet:

```
svn checkout https://svn.ufz.de/svn/chs-svn/FORTRAN_chs_lib/ local_name/
```

This will check out the whole library with modules and test folders. To checkout only the module files:

```
svn checkout --depth=files https://svn.ufz.de/svn/chs-svn/FORTRAN_chs_lib/
```

How to contribute to the FORTRAN lib?

As we discussed and agreed with the Fortran programmers at our internal meeting, there are certain **programming rules** we all have to follow to share modules. A template which follows all the rules you can find under [mo_template.f90](#). Mainly the rules are:

1. Use the [mo_kind.f90](#) to declare your number precisions.

EXAMPLE:

```
use mo_kind, only : i4, dp
integer(i4) :: a      ! To declare a 32-bit integer variable
real(dp)      :: b      ! To declare a double precision real variable
```

2. All subroutines you contribute have to have a single and a double precision version which are combined in an INTERFACE.

EXAMPLE:

```
INTERFACE foo
  MODULE PROCEDURE foo_SP, &      ! Single precision subroutine
                        foo_DP      ! Double precision subroutine
END INTERFACE foo
```

3. Just the interface routines are public, all others are private.

EXAMPLE:

```
PRIVATE
PUBLIC :: foo ! 1-line description of foo here
PUBLIC :: bar ! 1-line description of bar here

INTERFACE foo
  MODULE PROCEDURE foo_SP, foo_DP
END INTERFACE foo
INTERFACE bar
  MODULE PROCEDURE bar_SP, bar_DP
END INTERFACE bar
```

4. If you provide a function/ subroutine based on matrices you should allow for an optional MASK argument, if possible.

EXAMPLE:

```
FUNCTION Mean_dp(A,MASK)
! Calculate average
real(dp), dimension(:), intent(in)      :: A
logical, dimension(:), intent(in), optional :: MASK
real(dp), dimension(size(A)) :: Mean_dp
...
```

```

Mean_dp = sum(A, mask=MASK) / count(MASK)
...
END SUBROUTINE

```

5. Documentation/ Comment the source code
 - a. Give 1-line descriptions after the public definition.
 - b. Documentation:
 - document before the individual routines;
 - do one documentation per interface, i.e. no separate docu. for sp and dp;
 - follow the documentation structure before the function mean_sp in mo_template.f90;
 - break comment lines at column 130 at most.
 - c. Each module has to appear in the **two tables** above. Please include the module name and its one-line description only if the module is fully tested and documented.
6. Coding style
 - a. Sort routines alphabetically in the file and in the public definitions.
 - b. Break lines at column 130 at most.
 - c. Do not use tabs in files.
 - d. Each module should have a test directory : **test/test_mo_xxx**, which includes a test program. Include a check that states if the test worked or not (output message should include "o.k." or "failed", since these strings are grepped by command "make check"). In the subdirectory test/test_mo_xxx you simply link your modules, i.e. you do on the command prompt in the test directory:


```
test/test_mo_xxx> ln -s ../../mo_kind.f90
test/test_mo_xxx> ln -s ../../mo_xxx.f90
```
 - e. The module should be tested with at least two different compilers (of different vendors), e.g. ifort and gfortran
 - f. Use lowercase characters for the module name.

License

This file is part of the UFZ Fortran library.

The UFZ Fortran library is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The UFZ Fortran library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the UFZ Fortran library. If not, see < <http://www.gnu.org/licenses/>>.

Note on Numerical Recipes License

Be aware that some code is under the Numerical Recipes License 3rd edition
< <http://www.nr.com/aboutNR3license.html>>

The Numerical Recipes Personal Single-User License lets you personally use Numerical Recipes code ("the code") on any number of computers, but only one computer at a time. You are not permitted to allow anyone else to access or use the code. You may, under this license, transfer precompiled, executable applications incorporating the code to other, unlicensed, persons, providing that (i) the application is noncommercial (i.e., does not involve the selling or licensing of the application for a fee), and (ii) the application was first developed, compiled, and successfully run by you, and (iii) the code is bound into the application in such a manner that it cannot be accessed as individual routines and cannot practicably be unbound and used in other programs. That is, under this license, your application user must not be able to use Numerical Recipes code as part of a program library or "mix and match" workbench.

Businesses and organizations that purchase the disk or code download, and that thus acquire one or more Numerical Recipes Personal Single-User Licenses, may permanently assign those licenses, in

the number acquired, to individual employees. Such an assignment must be made before the code is first used and, once made, it is irrevocable and can not be transferred.

If you do not hold a Numerical Recipes License, this code is only for informational and educational purposes but cannot be used.

[Goto MainPage](#)