**Lecture Note-Numerical Analysis (1): Introduction**

1. **Sequence of numerical approach to engineering and scientific problems**

* System mathematical modeling

- Mathematical formulation (expression) of physical laws

- Initial and/or boundary condition

- System constraints

- Etc.

* Numerically approximate the system mathematical model using numerical algorithm such as

- Descretization (ex: finite difference method)

- Linearization for nonlinear system

- Interpolation (linear, quadratic, polynomial, etc)

- Integration (line, surface, volume integrals)

- Etc.

* Convert the numerical approximation into a computer program

- Using computer languages such as Fortran, C, Basic, Matlab, etc

- Using software package such as IMSL libraries, BLAS, Lapack, etc

- Naive generation of computer codes (time consuming, involve high probability of mistakes)

- etc

* Run the program in computers to get solutions

- Compliers/linkers

- Consider the limitation of computer hardware/software resources

- etc

* Analyze the computed results

- Accuracy/convergence

- interprete physical meanings

- apply or use the computed results to resolve the engineering and scientific problems

- etc

1. **Objective and Major concerns of numerical analysis**

* Objective

**“Application of the computed results to resolve the engineering and scientific problems”**

* Major concerns for real applications

1. Required accuracy

- Accuracy in mathematical modeling of the system

- Accuracy in numerical analysis

1. Means to overcome physical limits of computing machines

- Floating point handling capability (8-bit, 16-bit, 32-bit, 64 bit machines)

- Number of processors How to use multi-processor machines

- Memory: How to reduce the required memory

- CPU time: How to reduce the computing time

- Software such as compilers: What software to choose

- etc

1. Computational efficiency of numerical methods

- Number of floating point operations

- Numerical stability

- Fast convergence

- etc

1. Others

- Reusability

- Range of applications

1. **Type of Mathematical Problems**

**(3-1) Roots of equations :** 

**(example 1)** 

**(example 2)** 

**(example 3) Roots of system of equation with more than two unknowns such as**

🡪 

with 

**(example 4) General polynomial equation with one unknown variable**

 , 

It generally has n solutions

**(example 4) General nonlinear algebraic equation with multiple unknown variables**

**(n unknowns)**

 , 

**Remarks: Notational convention in this lecture**

1. **A component of the real valued set is represented by.**

* **Real number: *R***
* **Complex number: *C***
* **positive integer (Natural number): *N*, etc.**

1. **A Lower Case Plane Letter represents a scalar or scalar function such as:**

** etc.**

1. **A Lower Case Bold Letter represents a vector or vector function such as:**

** etc.**

1. **An Upper Case Letter represents a matrix such as: **



* 1. **Solution of simultaneous linear algebraic equations (or matrix equation)**

**(example1) Linear algebraic equation with three unknowns** 



🡪 

🡪 where 

**(example 2) Find x satisfying  where **

**🡪 solution: **

**(example3) General linear algebraic equation with n unknowns**



🡪



* 1. **Optimization**

**(example 1) solve the following**

 🡪  and 



* **Unconstrained Optimization Problems**



**1st order Optimization condition**

**🡪 becomes a root-finding problem**

**(example 1) Nonlinear programming problem with an equality constraint**

****

**s. t.  🡪 solution **

**(example 2) Nonlinear Programming problem with one equality constraint and one inequality constraint)**

****

**s. t. **

** 🡪 solution **

* **General form of constrained nonlinear programming problem (Nonlinear Programming):**

**find unknown  which minimizes the function  subject to**



Subject to



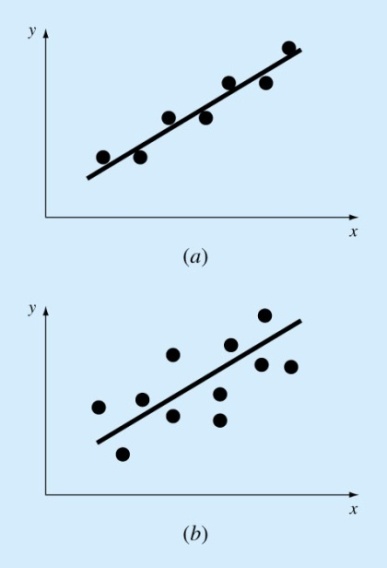
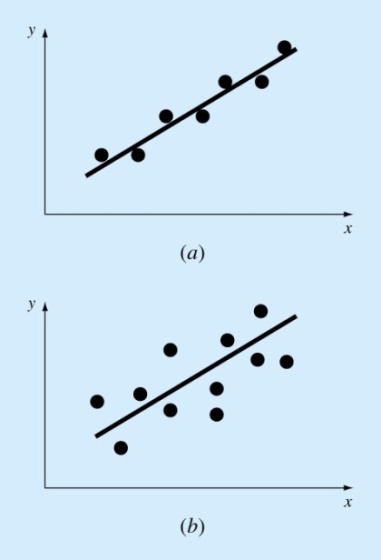
**Where  is an equality constraint and  is an inequality constraint**

**If f(x) is a linear function of x and g(x) and h(x) are also linear function of x, the resultant problem is called linear programming problem**

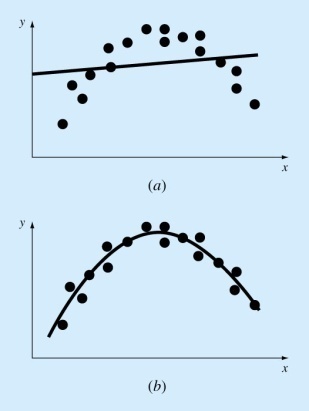
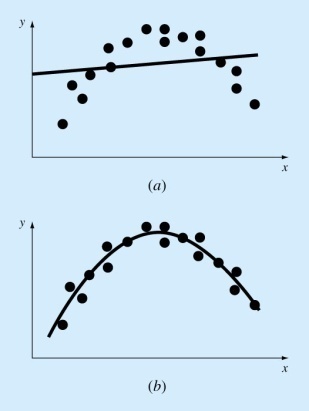
* 1. **Curve fitting (interpolation, least squares approximation)**
* **Regression: approximation curve doesn’t have to pass all given data point but is determined to minimize the approximation error in the sense of**

**Gaussian norm**

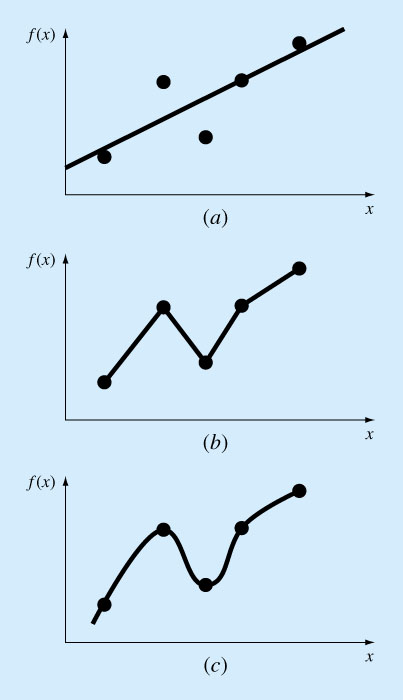
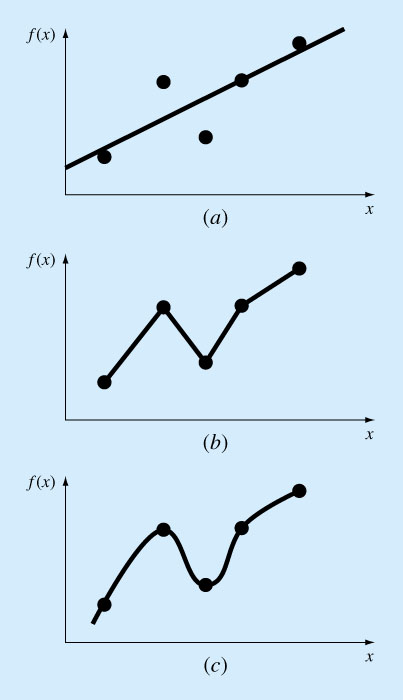
* **Interpolation : approximation curve should pass the given data points**

**** ****

(a) linear regression with small error (b) linear regression with large error

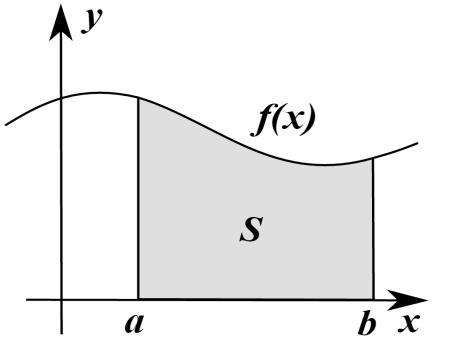
 

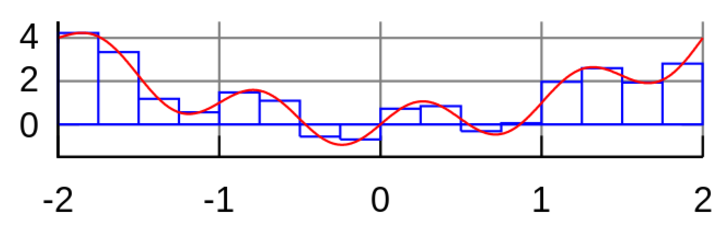
(c) linear regression with large error (d) nonlinear regression

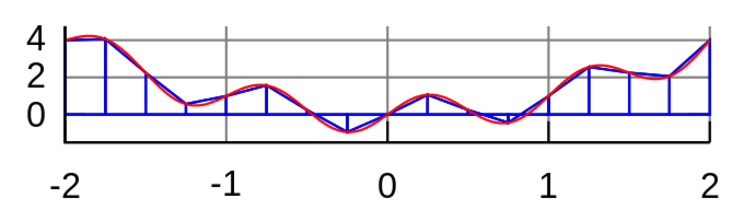
 

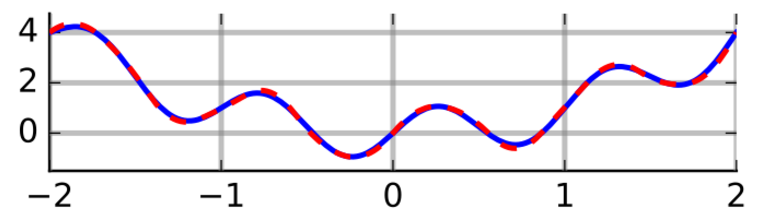
(e) linear interpolation (f) nonlinear interpolation

* 1. **Numerical Integration: Numerical approximation of the following Integral**

 **Line integral:** 







**Surface integral:** 

**(example 1): Application to the solution of a first order ordinary differential equation**



* 1. **Solution of ordinary differential equations such as**



**(example 1)** Problem definition and analytic solution



Homogeneous solution :

Characteristic equation

Characteristic roots 

Solution 



Particular solution 





Solution 



Applying initial conditions to determine coefficient 





Therefore, the solution becomes



**(example 2) Motion of a pendulum hanging on a hinge via a massless string**

****

**Motion equation : moment equation around O**

 where 



 where 

**Initial conditions**



**How to compute the pendulum motion?**

* 1. **Partial differential equations (not covered)**

 with suitable boundary conditions

1. **Pseudocode**

* Formal language : C, C++, Fortran (77,90,99), Matlab, Basic, etc
* Pseudo code is used to describe the general coding procedure using an informal language
* Pseudo code should be translated using one of the formal language for its running in a computer

Pseudo code example 1 to calculate  with 

Pseudo code Description

 save 1 into memory a

 save 2 into memory b

 add the values in memories a and b, save the result into memory y

Pseudo code example 2 to calculate  with 

Pseudo code Description

 save 1 into memory a

 save 5 into memory x

 save 2 into memory b

 replace the value in memory y with the multiplication 

 add b to the values in memories y and replace it with the result

Pseudo code example 3 to calculate  , where x,y,z, are vectors such as



Pseudo code (1) Description

 For each value of, add  and .

Then save the result

Pseudo code (2) Description

do j=1, N Repeat for each value of 



Pseudo code (3) Description

for j=1:1:N Repeat for each value of with an increment 1



Pseudo code (4) Description

Repeat for j=1 until j=N Repeat for each value of 



Pseudo code (5) Description

Repeat for j=1 until j=N Repeat for each value of 



Pseudo code example 4 for logical statement

(1)

If a>0, b=2 if a>1, then set b with 2

(2) if a>0, then

b=2

else

b=3

end if

(3) if a>0, b=2

Otherwise, b=3

(4) if a>0, b=2

Else if a=0

b = 0

Otherwise

b=3

end if

1. **Tips for building an efficient numerical analysis program**
2. Memory saving in calculating 

High memory

 🡪 Memory required: 

Minimum memory

 🡪 Memory required: 

Computer Program

MATLAB: 

FORTRAN: 

1. Number of floating point operations in calculating 

- 

Number of additions: 5 Number of multiplications: 5+4+3+2= 14

- 

Number of additions: 5 Number of multiplications: 5

1. Miscellaneous for basic numerical operations in computers

- Calculation of  with given values of a constant  and an array

Naive calculation: 

Other method set 



- Calculation of with given values of a constantand an array

Naive calculation: 

Other method (1) set 



Other method (2) set 



- Calculation of 

Naive calculation: 

Other method (1) when ( )s are not used at the later part of program

Other method (2) Set 

- Matrix Calculation of  with a given constant  and



Inefficient calculation procedure (number of multiplications: NxN+NxN)



Normal calculation procedure (number of multiplications: NxN+N)



- Matrix Calculation of  with a given constant  and



1. Sequence of operation (1) ; multiplication





1. Sequence of operation (2) ; multiplication 





- Calculation of Exponential  with integer index

Using syntax such as  Fortran

 Matlab

Normal calculation (1)

 🡨 

Normal calculation (2)

 🡨 

**“Compare**

**number of multiplications/divisions**

**number of additions/subtractions**

**and run time for each algorithm”**

1. **Examples of math intrinsic function in Fortran**

|  |  |
| --- | --- |
| **Name** | **Description** |
| [ACOS](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfacos.htm) | Returns the arc cosine of the argument, expressed in radians between 0 and pi. |
| [ACOSD](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfacosd.htm) | Returns the arc cosine of the argument, expressed in degrees between 0 and 180. |
| [ALOG](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog.htm) | Returns natural log of the argument. |
| [ALOG10](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog10.htm) | Returns common log (base 10) of the argument. |
| [ASIN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfasin.htm) | Returns the arc sine of the argument, expressed in radians between ±pi/2. |
| [ASIND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfasind.htm) | Returns the arc sine of the argument, expressed in degrees between ±90°. |
| [ATAN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatan.htm) | Returns the arc tangent of the argument, expressed in radians between ±pi/2. |
| [ATAND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatand.htm) | Returns the arc tangent of the argument, expressed in degrees between ±90°. |
| [ATAN2](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatan2.htm) | Returns the arc tangent of the second argument divided by the first argument, expressed in radians between ±pi. |
| [ATAN2D](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatan2d.htm) | Returns the arc tangent of the second argument divided by the first argument, expressed in degrees between ±180°. |
| [CCOS](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcos.htm) | Returns complex cosine of the argument. |
| [CDCOS](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcos.htm) | Returns the double-precision complex cosine of the argument. |
| [CDEXP](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfexp.htm) | Returns double-precision complex exponential value of the argument. |
| [CDLOG](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog.htm) | Returns the double-precision complex natural log of the argument. |
| [CDSIN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsin.htm) | Returns the double-precision complex sine of the argument. |
| [CDSQRT](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsqrt.htm) | Returns the double-precision complex square root of the argument. |
| [CEXP](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfexp.htm) | Returns the complex exponential value of the argument. |
| [CLOG](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog.htm) | Returns the complex natural log of the argument. |
| [COS](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcos.htm) | Returns the cosine of the argument, which is in radians. |
| [COSD](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcosd.htm) | **COSD**(*x*). Returns the cosine of the argument, which is in degrees. |
| [COSH](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcosh.htm) | Returns the hyperbolic cosine of the argument. |
| [COTAN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcotan.htm) | Returns the cotangent of the argument, which is in radians. |
| [COTAND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcotand.htm) | Returns the cotangent of the argument, which is in degrees. |
| [CSIN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsin.htm) | Returns the complex sine of the argument. |
| [CSQRT](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsqrt.htm) | Returns the complex square root of the argument. |
| [DACOS](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfacos.htm) | Returns the double-precision arc cosine of the argument radians between 0 and pi. |
| [DACOSD](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfacosd.htm) | Returns the arc cosine of the argument in degrees between 0 and 180*.* |
| [DASIN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfasin.htm) | Returns the double-precision arc sine of the argument in radians between ±pi/2. |
| [DASIND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfasind.htm) | Returns the double-precision arc sine of the argument degrees between ±90°. |
| [DATAN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatan.htm) | Returns the double-precision arc tangent of the argument radians between ±pi/2. |
| [DATAND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatand.htm) | Returns the double-precision arc tangent of the argument degrees between ±90°. |
| [DATAN2](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatan2.htm) | Returns the double-precision arc tangent of the second argument divided by the first argument radians between ±pi. |
| [DATAN2D](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfatan2d.htm) | Returns the double-precision arc tangent of the second argument divided by the first argument degrees between ±180°. |
| [DCOS](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcos.htm) | Returns the double-precision cosine of the argument radians. |
| [DCOSD](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcosd.htm) | Returns the double-precision cosine of the argument degrees. |
| [DCOSH](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcosh.htm) | Returns the double-precision hyperbolic cosine of the argument. |
| [DCOTAN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfcotan.htm) | Returns the double-precision cotangent of the argument. |
| [DEXP](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfexp.htm) | Returns the double-precision exponential value of the argument. |
| [DLOG](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog.htm) | Returns the double-precision natural log of the argument. |
| [DLOG10](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog10.htm) | Returns the double-precision common log (base 10) of the argument. |
| [DSIN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsin.htm) | Returns the double-precision sin of the argument radians. |
| [DSIND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsind.htm) | Returns the double-precision sin of the argument degrees. |
| [DSINH](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsinh.htm) | Returns the double-precision hyperbolic sine of the argument. |
| [DSQRT](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsqrt.htm) | Returns the double-precision square root of the argument. |
| [DTAN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rftan.htm) | Returns the double-precision tangent of the argument radians. |
| [DTAND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rftand.htm) | Returns the double-precision tangent of the argument degrees. |
| [DTANH](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rftanh.htm) | Returns the double-precision hyperbolic tangent of the argument. |
| [EXP](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfexp.htm) | Returns the exponential value of the argument. |
| [LOG](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog.htm) | Returns the natural log of the argument. |
| [LOG10](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rflog10.htm) | Returns the common log (base 10) of the argument. |
| [SIN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsin.htm) | Returns the sine of the argument, which is in radians. |
| [SIND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsind.htm) | Returns the sine of the argument, which is in degrees. |
| [SINH](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsinh.htm) | Returns the hyperbolic sine of the argument. |
| [SQRT](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rfsqrt.htm) | Returns the square root of the argument. |
| [TAN](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rftan.htm) | Returns the tangent of the argument, which is in radians. |
| [TAND](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rftand.htm) | Returns the tangent of the argument, which is in degrees. |
| [TANH](ms-its:C:\Program%20Files\Intel\Compiler\Fortran\9.1\Docs\lref_for.chm::/rftanh.htm) | Returns the hyperbolic tangent of the argument. |