1. Overview

The book is divided into four chapters and a larger appendix to address the goal of different error functions with different accuracies, calculation types and performance capabilities.

The first chapter briefly explains the necessary basics/definitions and defines the requirements for hardware and software. For the calculations, the compiler and the data type definition (integer, real and logical) are agreed first. This is followed by the definition of constants and calculation limits. The module definition is then presented, followed by the tables for the various error functions and explanations of the menu-driven test program. The test program and the table values for various error functions are printed in full in the appendix for reasons of clarity.

The second chapter first presents some mathematical definitions and specifications for the “Error function, y=erf(x)”. This is followed by the different implementations, whereby the structure is always divided into high and low precision implemented functions. Functions that work by means of series expansions and continued functions are then presented. A runtime and performance summary can be found at the end of the chapter.

The third chapter is structured similarly to the second chapter, but for the “Complementary Error Function, y=erfc(x)”. The subchapter on series expansions focuses in particular on a diverse representation for asymptotic series expansions.

In the fourth chapter on the "Scaled Complementary Error Function, y=erfcx(x)", in addition to the standard structure of the previous chapters, the focus is on multi-precision algorithms with and without lookup tables. These are the routines according to Johnson/Wuttke, Zaghloul and a function attributed to the author. The last two authors mentioned achieve an accuracy of up to 32-digits with their multi-precision functions with or without the aid of a lookup table. For the functions with lookup tables, the tables have been moved to the appendix for reasons of clarity.

1.1 Basic Definitions

The “Error Function” or “Gaussian Error Function, y=erf(x), the “Complementary Error Function, y=erfc(x)” and thus the “Scaled Complementary Error Function, y=erfcx(x)” acquire their important meaning through their close relationship to the standard normal distribution, historically Glaisher [1,1871] and with a historical development of the error function at Marsaglia [2,2004].
For example, the error function is used in statistics and probability theory to calculate the probability that a measurement error falls within a certain range for normally distributed data. The error function also plays an important role in other areas of technology and science. In heat transfer, it is of crucial importance in the solution of transient diffusion problems, especially in heat conduction in the transient state, and in mass transfer it is used in the calculation of concentration distributions in transient diffusion processes. It is used in digital communication to determine the bit error rate of digital communication systems. In finance, it is used in option pricing models. Howard [3,2022] provides a good overview with the references given there.

The following definitions and notations are used in this book:

The ***Error Function, y = erf(x)*** is twice the integral of the Gaussian distribution with a mean zero and a variance of half $(1/2)$. The normalizing factor $2/\sqrt{\pi}$ ensures that $\mathrm{erf}(x = \infty) = 1$

$$y = \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt \tag{1.1.1}$$

or

$$y = \mathrm{erf}(x) = 1 - \mathrm{erfc}(x) \tag{1.1.2}$$

The ***Complementary Error Function, y = erfc(x)*** is defined as

$$y = \mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\, dt \tag{1.1.3}$$

or

$$y = \mathrm{erfc}(x) = 1 - \mathrm{erf}(x) \tag{1.1.4}$$

The ***Scaled Complementary Error Function, y = erfcx(x)*** is defined as

$$y = \mathrm{erfcx}(x) = e^{x^2} \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\, dt \tag{1.1.5}$$

or

$$y = \mathrm{erfcx}(x) = e^{x^2}\mathrm{erfc}(x) = \exp(x^2)\,\mathrm{erfc}(x) \tag{1.1.6}$$

The ***inverses of the different Error Functions*** use of the following nomenclatures:

$$x = \mathrm{erfinv}(y) = \mathrm{erf}^{-1}(y) \tag{1.1.7}$$

$$x = \mathrm{erfcinv}(y) = \mathrm{erfc}^{-1}(y) \tag{1.1.8}$$

$$x = \mathrm{erfcxinv}(y) = \mathrm{erfcx}^{-1}(y) \tag{1.1.9}$$

1.2 System and Development Environment

The hardware system (notebook) and the development environmental components consist of:

Hardware: Dell Inspiron 17 5000 Series -5770
The notebook contains an 8th generation Intel Core i7-8550U processor (8 MB Cache, up to 4.0 GHz) with 8 GB, DDR4, 2400MHz.

Operating System: Windows 10 pro (64 bit)
Microsoft Windows 10 is part of the Windows family of operating systems with graphical user interfaces from the Microsoft Corporation.
Here, Windows 10 Version 22H2 for x64-based systems (KB5046613), dated 12. Nov. 2024.

Editor: Notepad++, version v8.7.1 (64 bit).
Notepad++ is a free source code editor that supports several programming languages for Windows. Its use is governed by GNU General Public License. ASCII and various Unicode (e.g. UTF-8) encodings are supported as character sets. For many programming languages, syntax and structure are highlighted using typographical means or code folding. For a free download see https://notepad-plus-plus.org
Here, the coding UTF-8 is used only.

Compiler 1: GFortran (GNU Fortran) version 6.3.0-1 [4,2024], part of GCC (GNU Compiler Collection). The GNU Fortran compiler supports the Fortran 77, 90 and 95 standards completely, parts of the Fortran 2003 and Fortran 2008 standards, and several vendor extensions.
Here, http://gcc.gnu.org/ or https://gcc.gnu.org/onlinedocs/gcc-4.8.5/gfortran/About-GNU-Fortran.html#About-GNU-Fortran

Compiler 2: SILVERFROST FORTRAN FTN95, version 9.00, released since 13. Nov. 2023 [5,2024]. The Silverfrost Fortran FTN95 for Windows is a compiler for the Fortran programming language for computers running Microsoft Windows. A lot of features from the later Fortran 2003 and 2008 standards are available with FTN95.
There is a free-of-charge personal edition, which generates programs which briefly display a banner, see https://www.silverfrost.com/32/ftn95/ftn95_personal_edition.aspx.
Here, Silverfrost Fortran is used from the Command Line only.

Remark:
All compilations and linking processes of the source/object files always via a system batch file.

1.3 Fortran Data Type Definitions (module kinds)

The module “KINDS” is located in two separate files:

“201-def-kinds-GFortran.f95”	(GFortran [4,2024] declarations)	or
“201-def-kinds-Silverfrost.f95”	(Silverfrost FTN95 [5,2024] declarations)	

and defines type of compiler, optional ISO use and the three needed data types, which are INTEGER, REAL and LOGICAL.

INTEGER and LOGICAL are adjusted to 4 bytes for both compilers.

REAL can be selected between

4 bytes	single precision, sp, 6 digits precision, selection = 1
8 bytes	double precision, dp, 15 digits precision, selection = 2
10 bytes	long double (extended) precision, ep, 18 digits precision, selection = 3, Silverfrost FTN95 only
16 bytes	quadruple precision, qp, 33 digits precision, selection = 3, GFortran only

GFortran uses the ISO Environment Instruction: “use ISO_FORTRAN_ENV” with selection (“only”) of INT8, INT16, INT32, INT64, REAL32, REAL64, REAL128

However, all "Error Function" routines are programmed independently of a "real" type declaration and could be used separately at any time.

Changing a definition, e.g. a "real" definition, requires a recompilation of the program.

Source code „0201-def-Kinds-GFortran.f95“:

```
0001 !
0002 !   kinds
0003 !    Definition for the data types for GFortran
0004 !   Reference:
0005 !    GFortran (GNU Fortran) version 6.3.0-1, part of GCC (GNU Compiler
0006 !    Collection). http://gcc.gnu.org/ or https://gcc.gnu.org/onlinedocs/
0007 !    gcc-4.8.5/gfortran/About-GNU-Fortran.html#About-GNU-Fortran
0008 !   Remarks:
0009 !    Changing one of the default precision requires a recompilation of
0010 !    the complete program, including all linked routines.
0011 !   Implementation, adjustments and/or extensions by Thomas Hoering,
0012 !     last modification: 31. January 2025
0013 !
0014     module kinds
0015 !
0016 !-------------------------------------------------------------------------
0017 !
0018 !   ISO Environment (do not reduce elements of the environment)
0019 !
0020     use ISO_FORTRAN_ENV, only: INT8, INT16, INT32,  INT64,          &
0021                                             REAL32, REAL64, REAL128
0022 !
0023 !-------------------------------------------------------------------------
0024 !
```

```
0025     implicit none
0026 !
0027 !-------------------------------------------------------------------------------
0028 !
0029 !   COMPILER definition:
0030 !
0031 !   [do not change this parameter. In case you want to use Silverfrost ftn95,
0032 !    use the file "201-def-kinds-Silverfrost.f95"                              ]
0033 !
0034     integer, parameter       :: compiler  =  1        ! 1 := GFortran
0035 !                                                       2 := Silverfrost
0036 !
0037 !-------------------------------------------------------------------------------
0038 !
0039 !   INTEGER definition:
0040 !
0041 !   Select one of the integer kinds: iki = 1, 2, 3 or 4;
0042 !   preselection: kind=3 = INT32. No necessity to change the selection.
0043 !
0044 !   kind ISO   digits radix range bit byte Huge                 former design.
0045 !     1  INT8    7      2     2    8   1   127                  integer*1
0046 !     2  INT16  15      2     4   16   2   32767                integer*2
0047 !     3  INT32  31      2     9   32   4   2147483647           integer*4
0048 !     4  INT64  63      2    18   64   8   9223372036854775807  integer*8
0049 !
0050     integer, parameter       :: iki       =  3   ! kind=3 = INT32
0051 !
0052     integer, parameter       :: ikx(1:4)  = (/ INT8, INT16, INT32, INT64 /)
0053 !
0054     integer, parameter       :: ik        = ikx( iki )
0055 !
0056 !-------------------------------------------------------------------------------
0057 !
0058 !   REAL definition:
0059 !
0060 !   Select one of the real kinds: rki = 1, 2 or 3;
0061 !   preselection: kind=2 = REAL64. Change the preselection in line 88.
0062 !
0063 !   kind kind ISO     storage digits precision range byte other designation
0064 !     1   sp  REAL32    32      24      6        37    4  real*4
0065 !     2   dp  REAL64    64      53     15       307    8  real*8/double prec.
0066 !    n.a. ep  n.a.      80      64     18      4931   10  new, extended prec.
0067 !     3   qp  REAL128  128     113     33      4931   16  quadruple precision
0068 !
0069 !   special values
0070 !   kind kind ISO     Tiny               Huge                minexp  maxexp
0071 !     1   sp  REAL32  1.1754943...E-0038 3.4028234...E+0038    -125     128
0072 !     2   dp  REAL64  2.2250738...E-0308 1.7976931...E+0308   -1021    1024
0073 !    n.a. ep  n.a.    3.3621031...E-4932 1.1897314...E+4932  -16381   16384
0074 !     3   qp  REAL128 3.3621031...E-4932 1.1897314...E+4932  -16381   16384
0075 !
0076 !   kind kind ISO     Epsilon                 √(Epsilon)
0077 !     1   sp  REAL32  2^-023=1.1920928...E-07 √(2^-023)=     =3.4526697...E-04
0078 !     2   dp  REAL64  2^-052=2.2204460...E-16 √(2^-052)=2^-26=1.4901161...E-08
0079 !    n.a. ep  n.a.    2^-063=1.0842021...E-19 √(2^-063)=     =3.2927225...E-10
0080 !     3   qp  REAL128 2^-112=1.9259299...E-34 √(2^-112)=2^-56=1.3877787...E-17
0081 !
0082 !   kind kind ISO     SPACING             RRSPACING
0083 !     1   sp  REAL32  1.1920928...E-0007  8.3886080...E+0006
0084 !     2   dp  REAL64  2.2204460...E-0016  4.5035996...E+0015
0085 !    n.a. ep  n.a.    1.0842021...E-0019  9.2233720...E+0018
0086 !     3   qp  REAL128 1.9259299...E-0034  5.1922968...E+0033
0087 !
0088     integer(ik), parameter   :: rki       =  3   ! kind=3 = REAL128
0089 !
0090     integer(ik), parameter   :: rkx(1:3)  = (/ REAL32, REAL64, REAL128 /)
0091 !
0092     integer(ik), parameter   :: rk        = rkx( rki )
```

```
0093 !
0094 !
0095 !   REAL VALUES definitions from Fortran 2008 (ISO_FORTRAN_ENV)
0096 !
0097 !   sp2008   =  4,   Fortran 2008 value for REAL32,   sp, Single Precision
0098 !   dp2008   =  8,   Fortran 2008 value for REAL64,   dp, Double Precision
0099 !   qp2008   = 16,   Fortran 2008 value for REAL128,  qp, Quadruple Precision
0100 !   rk2008   = cv2008(rki) allocates real value from current Fortran 2008
0101 !
0102     integer(ik), parameter   :: sp2008    =  4_ik
0103     integer(ik), parameter   :: dp2008    =  8_ik
0104     integer(ik), parameter   :: qp2008    = 16_ik
0105     integer(ik), parameter   :: cv2008(3) = (/ sp2008, dp2008, qp2008 /)
0106     integer(ik), parameter   :: rk2008    = cv2008( rki )  ! rk = 1, 2 or 3
0107 !
0108 !-------------------------------------------------------------------------------
0109 !
0110 !   LOGICAL definition:
0111 !
0112 !   Select one of the logical kinds: 1, 2 or 3;
0113 !   preselection: kind=3 = 4 bytes. No necessity to change the selection
0114 !
0115 !   kind   bit   byte   former designation
0116 !     1     8     1     logical*1
0117 !     2    16     2     logical*2
0118 !     3    32     4     logical*4
0119 !
0120     integer(ik), parameter   :: lki       =  3   ! kind=3 = 4 bytes selected
0121 !
0122     integer(ik), parameter   :: lkx(1:3)  = (/ 1_ik, 2_ik, 4_ik /)
0123 !
0124     integer(ik), parameter   :: lk        = lkx( lki )
0125 !
0126 !-------------------------------------------------------------------------------
0127 !
0128     end module kinds
0129 !
```

Source code „0201-def-Kinds-Silverfrost.f95“:

```
0001 !
0002 !   kinds
0003 !    Definition for the data types for Silverfrost FTN95
0004 !   Reference:
0005 !    SILVERFROST FORTRAN FTN95, version 9.00, released since 13. Nov. 2023.
0006 !    https://www.silverfrost.com/32/ftn95/ftn95_personal_edition.aspx and
0007 !    for the basic data types
0008 !    https://www.silverfrost.com/ftn95-help/mixlan/basicdatatypes.aspx
0009 !   Remarks:
0010 !    Changing one of the default precision requires a recompilation of
0011 !    the complete program, including all linked routines.
0012 !   Implementation, adjustments and/or extensions by Thomas Hoering,
0013 !     last modification: 31. January 2025
0014 !
0015     module kinds
0016 !
0017 !-------------------------------------------------------------------------------
0018 !
0019     implicit none
0020 !
0021 !-------------------------------------------------------------------------------
0022 !
0023 !   COMPILER definition:
0024 !
0025 !   [do not change this parameter. In case you want to use GFortran,
0026 !    use the file "201-def-kinds-GFortran.f95"                          ]
```

```
0027 !
0028     integer, parameter       :: compiler  =  2        ! 1 := GFortran
0029 !                                                      2 := Silverfrost
0030 !
0031 !------------------------------------------------------------------------------
0032 !
0033 !   INTEGER definition:
0034 !
0035 !   Select one of the integer kinds: iki = 1, 2 or 3;
0036 !   preselection: kind = 3. No necessity to change the selection
0037 !
0038 !   kind ISO   digits radix range bit byte Huge                 former design.
0039 !     1  INT8    7      2     2    8   1   127                  integer*1
0040 !     2  INT16  15      2     4   16   2   32767                integer*2
0041 !     3  INT32  31      2     9   32   4   2147483647           integer*4
0042 !     4  INT64  63      2    18   64   8   9223372036854775807 not available
0043 !
0044     integer, parameter       :: iki       =  3   ! kind = 3 / 4 bytes selected
0045 !
0046     integer, parameter       :: ikx(1:3)  = (/ 1, 2, 3 /)
0047 !
0048     integer, parameter       :: ik        = ikx( iki )
0049 !
0050 !------------------------------------------------------------------------------
0051 !
0052 !   REAL definition:
0053 !
0054 !   Select one of the real kinds: rki = 1, 2 or 3;
0055 !   preselection: kind=2 = real*8. Change the preselection in line 82.
0056 !
0057 !   kind kind ISO     storage digits precision range byte other designation
0058 !     1   sp  REAL32    32      24      6        37    4  real*4
0059 !     2   dp  REAL64    64      53     15       307    8  real*8/double prec.
0060 !     3   ep  n.a.      80      64     18      4931   10  extended precision
0061 !    n.a. qp  REAL128  128     113     33      4931   16  not available
0062 !
0063 !   special values
0064 !   kind kind ISO     Tiny               Huge               minexp maxexp
0065 !     1   sp  REAL32  1.1754943...E-0038 3.4028234...E+0038   -125    128
0066 !     2   dp  REAL64  2.2250738...E-0308 1.7976931...E+0308  -1021   1024
0067 !     3   ep  n.a.    3.3621031...E-4932 1.1897314...E+4932 -16381  16384
0068 !    n.a. qp  REAL128 3.3621031...E-4932 1.1897314...E+4932 -16381  16384
0069 !
0070 !   kind kind ISO     Epsilon                 √(Epsilon)
0071 !     1   sp  REAL32  2^-023=1.1920928...E-07 √(2^-023)=     =3.4526697...E-04
0072 !     2   dp  REAL64  2^-052=2.2204460...E-16 √(2^-052)=2^-26=1.4901161...E-08
0073 !     3   ep  n.a.    2^-063=1.0842021...E-19 √(2^-063)=     =3.2927225...E-10
0074 !    n.a. qp  REAL128 2^-112=1.9259299...E-34 √(2^-112)=2^-56=1.3877787...E-17
0075 !
0076 !   kind kind ISO     SPACING             RRSPACING
0077 !     1   sp  REAL32  1.1920928...E-0007  8.3886080...E+0006
0078 !     2   dp  REAL64  2.2204460...E-0016  4.5035996...E+0015
0079 !     3   ep  n.a.    1.0842021...E-0019  9.2233720...E+0018
0080 !     3   qp  REAL128 1.9259299...E-0034  5.1922968...E+0033
0081 !
0082     integer(ik), parameter   :: rki       =  2   ! kind = 2 = double precision
0083 !
0084     integer(ik), parameter   :: rkx(1:3)  = (/ 1_ik, 2_ik, 3_ik /)
0085 !
0086     integer(ik), parameter   :: rk        = rkx( rki )
0087 !
0088 !
0089 !   REAL VALUES definitions from Fortran 2008 (ISO_FORTRAN_ENV)
0090 !
0091 !   sp2008   =  4,   Fortran 2008 value for REAL32,   sp, Single Precision
0092 !   dp2008   =  8,   Fortran 2008 value for REAL64,   dp, Double Precision
0093 !   qp2008   = 16,   Fortran 2008 value for REAL128,  qp, Quadruple Precision
0094 !   rk2008   = cv2008(rki) allocates real value from current Fortran 2008
```

```
0095 !
0096     integer(ik), parameter   :: sp2008    =  4_ik
0097     integer(ik), parameter   :: dp2008    =  8_ik
0098     integer(ik), parameter   :: qp2008    = 16_ik   ! here: extended precision
0099     integer(ik), parameter   :: cv2008(3) = (/ sp2008, dp2008, qp2008 /)
0100     integer(ik), parameter   :: rk2008    = cv2008( rki )  ! rk = 1, 2 or 3
0101 !
0102 !------------------------------------------------------------------------------
0103 !
0104 !   LOGICAL definition:
0105 !
0106 !   Select one of the logical kinds: lki = 1, 2 or 3;
0107 !   preselection: kind=3 = 4 bytes. No necessity to change the selection
0108 !
0109 !   kind   bit   byte   former designation
0110 !     1     8     1     logical*1
0111 !     2    16     2     logical*2
0112 !     3    32     4     logical*4
0113 !
0114     integer(ik), parameter   :: lki       =  3   ! kind=3 = 4 bytes selected
0115 !
0116     integer(ik), parameter   :: lkx(1:3)  = (/ 1_ik, 2_ik, 3_ik /)
0117 !
0118     integer(ik), parameter   :: lk        = lkx( lki )
0119 !
0120 !------------------------------------------------------------------------------
0121 !
0122     end module kinds
0123 !
```

1.4 Constants and Parameters (module const)

The module “CONST” is located in the separate file named “202-def-Constants.f95” and defines various constants, parameters and limits. The constant numbers, mathematical and machine-dependent constants are all self-explanatory. In addition, several calculation limits are defined, and the display of the error output is specified. Further literature for study can be found at Cody [6,1969], SLATEC Common Mathematical Library [11,1993], and/or Zaghloul [13,2024]. Here the boundaries were defined as follows:

a) ***eps***, the precision and/or termination accuracy for iterative algorithms and/or formulas

eps := 0.10 * (EPSILON / 2.00)

b) ***eMax***, the maximum value, where "EXP(eMax*eMax)" does not generate an overflow

eMax := SQRT(LOG(1.00 / TINY / 48.00))

c) ***sqrteps***, the limit for the y = erf(x), erfc(x) or erfcx(x)
where y = erf ($|x| \le$ sqrteps) := 2.00/SQRT(π) * x,
y = erfc ($|x| \le$ sqrteps) := 1.00 - 2.00/SQRT(π) * x,
y = erfcx($|x| \le$ sqrteps) := 1.00 - 2.00/SQRT(π) * x,

sqrteps := SQRT(EPSILON)

d) ***limiterf***, the limit for the y = erf(x), where y = erf($|x| \ge$ limiterf) := +1.00

limiterf := SQRT(-LOG(SQRT(π) * EPSILON / 2.00))

e) ***limiterfc***, the limit for the y = erfc(x), where y = erfc($x \le$ limiterfc) := +2.00

limiterfc := -SQRT(-LOG(SQRT(π) * EPSILON / 2.00))

f) ***limiterfcH***, the limit for y = erfc(x), where y = erfc($x \le$ limiterfcH) := +0.00, to avoid an underflow, x should be less than limiterfcH

limiterfcH := SQRT(-LOG(SQRT(π) * TINY))

g) ***limiterfcxL***, the lower limit for y = erfcx(x), where y = erfcx($x \le$ limiterfcxL) := +∞, to avoid an underflow, x must be greater than limiterfcxL

limiterfcxL := -SQRT(LOG(HUGE / 2.00))

h) ***limiterfcxL1***, the limit for y = erfcx(x), where y = erfcx($x \le$ limiterfcxL1) := $2*\exp(x^2)$

limiterfcxL1 := -SQRT(-LOG(SQRT(π) * EPSILON / 2.00))

i) ***limiterfcxBig***, the limit for y = erfcx(x), where y = erfcx($x \le$ limiterfcxBig) := 1.00 / (SQRT(π) * (x + 0.5 / x))

limiterfcxBig := SQRT(1.00 / EPSILON)

j) ***limiterfcxH***, the upper limit for y = erfcx(x), where y = erfcx(x ≥ limiterfcxH) := 0.00

limiterfcxH := 1.00 / (SQRT(π) * TINY)

k) ***typerr***, the parameter for the error type representation in the output modules. The definitions of the error type with preselection “3” are:

typerr := 1,	deviation	:=	true value - measured value
typerr := 2,	absolute error	:=	ABS(true value - measured value)
typerr := 3,	relative error	:=	absolute error / true value

Source code „202-def-Constants.f95“:

```
0001 !
0002 !   const
0003 !     Definitions of constant numbers, mathematical constants/parameters,
0004 !     limits of error functions and error type usage
0005 !   Remarks:
0006 !     "(kind = ik)" and "(kind = rk)" are definied in "kinds.f95"
0007 !   Implementation, adjustments and/or extensions by Thomas Hoering,
0008 !     last modification: 31. January 2025
0009 !
0010     module const
0011 !
0012     use kinds, only : ik, rk
0013 !
0014     implicit none
0015 !
0016 !-------------------------------------------------------------------------------
0017 !
0018 !   constant numbers
0019 !
0020     real   ( kind = rk ), parameter    :: zero          =  0.00E+00_rk
0021     real   ( kind = rk ), parameter    :: half          =  0.50E+00_rk
0022     real   ( kind = rk ), parameter    :: one           =  1.00E+00_rk
0023     real   ( kind = rk ), parameter    :: two           =  2.00E+00_rk
0024     real   ( kind = rk ), parameter    :: three         =  3.00E+00_rk
0025     real   ( kind = rk ), parameter    :: four          =  4.00E+00_rk
0026     real   ( kind = rk ), parameter    :: five          =  5.00E+00_rk
0027     real   ( kind = rk ), parameter    :: six           =  6.00E+00_rk
0028     real   ( kind = rk ), parameter    :: seven         =  7.00E+00_rk
0029     real   ( kind = rk ), parameter    :: eight         =  8.00E+00_rk
0030     real   ( kind = rk ), parameter    :: nine          =  9.00E+00_rk
0031     real   ( kind = rk ), parameter    :: ten           = 10.00E+00_rk
0032     real   ( kind = rk ), parameter    :: twelve        = 12.00E+00_rk
0033     real   ( kind = rk ), parameter    :: sixteen       = 16.00E+00_rk
0034     real   ( kind = rk ), parameter    :: fortyeight    = 48.00E+00_rk
0035 !
0036 !-------------------------------------------------------------------------------
0037 !
0038 !   mathematical constants and parameters (up to 40 digits precision)
0039 !
0040     real   ( kind = rk ), parameter    :: pi            = &
0041     3.14159265358979323846264338327950288419 72E+00_rk   !         pi
0042 !
0043     real   ( kind = rk ), parameter    :: twodivpi      = &
0044     0.63661977236758134307553505349005744814 78E+00_rk   !       2/pi
0045 !
0046     real   ( kind = rk ), parameter    :: sqrtpi        = &
0047     1.77245385090551602729816748334114518279 75E+00_rk   !    SQRT(pi)
0048 !
```