Math Vector Library
Reference Manual (Fortran)
DD-00002-110

Jan Adelsbach

January 26, 2024

Contents

1 About this Guide

1.1 Legal Information

1.2 Feedback and Contact

For feedback on this document, please use the following email address:
techpubs@adelsbach-research.eu

Please include the page number or a link to the page.

For general contact details, please visit https://adelsbach-research.eu/contact.

1.3 Introduction

This manual describes the *Application Programming Interface* (API) of the *Math Vector Library* for the Fortran programming language families.

1.4 Audience for This Guide

The audience of this guide is assumed to be Fortran programmers who understand the basic concepts of at least one of the aforementioned programming languages.

1.5 How to Use This Guide

This guide first describes some general programming details of the library and then documents each function individually.

The documentation for each function applies both the *single* and *double* precision versions. The former can be differentiated by a suffix letter `f`.

1.6 Conventions Used in This Guide

x
: Normal math typesetting represents a normal variable.

$\mathbf{x}$
: Bold math typesetting represents a vector.

`Mono`
: Monospace typesetting represents C function names, variables or data types.

2 Overview

2.1 Introduction

The *Math Vector Library* is a high-performance function library with vectorized versions of standard mathematical functions. The functions can operate both on dense and strided vectors, the latter can be supplied individually for result and operand vectors. Stride only executes the function on every n-th element leaving the elements in between untouched.

This manual describes the *Application Programming Interface* (API) of the mathematics vector functions.

2.2 Thread Safety

All routines in the library are completely thread-safe, as long as the data supplied in arguments is exclusive to the current thread.

2.3 SIMD/SPMD Unit Usage

This library makes excessive use of *Single Instruction Multiple Data* (SIMD) or *Single program Multiple Data* (SPMD) style extensions of the respective processor platform. It thereby abides by the standard system calling conventions when utilizing such.

2.4 Performance Characteristics

All subroutines in this library have a performance characteristic of $O(n)$. The routines may have different execution profiles depending upon the arguments supplied.

3 Utility

3.1 `mvecver` - Version query

```
subroutine mvecver(major, minor);
```

Queries the version of the library and stores the *major* and *minor* version numbers in the respective arguments.

3.1.1 Parameters

`MAJOR` - **INTEGER** *EXIT:* The major version number of the library.

`MINOR` - **INTEGER** *EXIT:* The minor version number of the library

4 Rounding

4.1 `vfloor` - Vector round down

```
subroutine vfloor (n, y, incy, x, incx);
subroutine vfloorf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y** this function rounds the elements of **x** to the nearest integral part less or equal than |**x**| and stores the result in **y**.

4.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.2 vceil - Vector round up

```
subroutine vceil (n, y, incy, x, incx);
subroutine vceilf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y** this function rounds the elements of **x** to the nearest integral part greater or equal than |**x**| and stores the result in **y**.

4.2.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.3 vtrunc - Vector truncate

```
subroutine vtrunc (n, y, incy, x, incx);
subroutine vtruncf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y** this function truncates the elements of **x** to the nearest integral part lower or equal than |**x**| and stores the result in **y**.

4.3.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.4 vround - Vector rounding

```
subroutine vround (n, y, incy, x, incx);
subroutine vroundf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y** this function rounds the elements of **x** to the nearest integral part and stores the result in **y**.

4.4.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

5 Roots

5.1 `vsqrt` - Vector square root $\sqrt{\mathbf{x}}$

```
subroutine vsqrt (n, y, incy, x, incx);
subroutine vsqrtf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sqrt{\mathbf{x}}$$

5.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

5.2 vrsqrt - Vector reciprocal square root $1/\sqrt{\mathbf{x}}$

```
subroutine vrsqrt (n, y, incy, x, incx);
subroutine vrsqrtf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \frac{1}{\sqrt{\mathbf{x}}}$$

5.2.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

5.3 vcbrt - Vector cube root $\sqrt[3]{\mathbf{x}}$

```
subroutine vcbrt (n, y, incy, x, incx);
subroutine vcbrtf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sqrt[3]{\mathbf{x}}$$

5.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

5.4 vrcbrt - Vector reciprocal cube root $1/\sqrt[3]{\mathbf{x}}$

```
subroutine vrcbrt (n, y, incy, x, incx);
subroutine vrcbrtf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \frac{1}{\sqrt[3]{\mathbf{x}}}$$

5.4.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6 Trigonometric Functions

6.1 `vsin` - Vector sine sin(**x**)

```
subroutine vsin (n, y, incy, x, incx);
subroutine vsinf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sin(\mathbf{x})$$

6.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.2 vcos - Vector cosine cos(**x**)

```
subroutine vcos (n, y, incy, x, incx);
subroutine vcosf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cos(\mathbf{x})$$

6.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.3 vtan - Vector tangent tan(**x**)

```
subroutine vtan (n, y, incy, x, incx);
subroutine vtanf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tan(\mathbf{x})$$

6.3.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.4 vasin - Vector arcsine $\sin^{-1}(\mathbf{x})$

```
subroutine vasin (n, y, incy, x, incx);
subroutine vasinf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sin^{-1}(\mathbf{x})$$

6.4.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.5 vacos - Vector arccosine $\cos^{-1}(\mathbf{x})$

```
subroutine vacos (n, y, incy, x, incx);
subroutine vacosf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cos^{-1}(\mathbf{x})$$

6.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.6 vatan - Vector arctangent $\tan^{-1}(\mathbf{x})$

```
subroutine vatan (n, y, incy, x, incx);
subroutine vatanf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tan^{-1}(\mathbf{x})$$

6.6.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.7 vatan2 - Vector arctangent $\tan^{-1}(\mathbf{x}/\mathbf{y})$

```
subroutine vatan2 (n, z, incz, x, incx, y, incy);
subroutine vatan2f(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes:

$$\mathbf{z} = \tan^{-1}(\mathbf{x}/\mathbf{y})$$

6.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

6.8 vsind - Vector sine sin(**x**) (degrees)

```
subroutine vsind (n, y, incy, x, incx);
subroutine vsindf(n, y, incy, x, incx);
```

Given an input vector **x** with angles in degrees (°) and a result vector **y**, this function computes:

$$\mathbf{y} = \sin(\mathbf{x})$$

6.8.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.9 vcosd - Vector cosine cos(**x**) (degrees)

```
subroutine vcosd (n, y, incy, x, incx);
subroutine vcosdf(n, y, incy, x, incx);
```

Given an input vector **x** with angles in degrees (°) and a result vector **y**, this function computes:

$$\mathbf{y} = \cos(\mathbf{x})$$

6.9.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.10 vtand - Vector tangent tan(**x**) (degrees)

```
subroutine vtand (n, y, incy, x, incx);
subroutine vtandf(n, y, incy, x, incx);
```

Given an input vector **x** with angles in degrees (°) and a result vector **y**, this function computes:

$$\mathbf{y} = \tan(\mathbf{x})$$

6.10.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.11 vasind - Vector arcsine $\sin^{-1}(\mathbf{x})$ (degrees)

```
subroutine vasind (n, y, incy, x, incx);
subroutine vasindf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sin^{-1}(\mathbf{x})$$

6.11.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.12 vacosd - Vector arccosine $\cos^{-1}(\mathbf{x})$ (degrees)

```
subroutine vacosd (n, y, incy, x, incx);
subroutine vacosdf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cos^{-1}(\mathbf{x})$$

6.12.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.13 vatand - Vector arctangent $\tan^{-1}(\mathbf{x})$ (degrees)

```
subroutine vatand (n, y, incy, x, incx);
subroutine vatandf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tan^{-1}(\mathbf{x})$$

6.13.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.14 vsinpi - Vector sine sin(πx)

```
subroutine vsinpi (n, y, incy, x, incx);
subroutine vsinpif(n, y, incy, x, incx);
```

Given an input vector **x** with angles in degrees (°) and a result vector **y**, this function computes:

$$\mathbf{y} = \sin(\pi\mathbf{x})$$

6.14.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.15 vcospi - Vector cosine cos(πx)

```
subroutine vcospi (n, y, incy, x, incx);
subroutine vcospif(n, y, incy, x, incx);
```

Given an input vector **x** with angles in degrees (°) and a result vector **y**, this function computes:

$$\mathbf{y} = \cos(\pi\mathbf{x})$$

6.15.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.16 vtanpi - Vector tangent tan(πx)

```
subroutine vtanpi (n, y, incy, x, incx);
subroutine vtanpif(n, y, incy, x, incx);
```

Given an input vector **x** with angles in degrees (°) and a result vector **y**, this function computes:

$$\mathbf{y} = \tan(\pi\mathbf{x})$$

6.16.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.17 vasinpi - Vector arcsine $\sin^{-1}(\mathbf{x})/\pi$

```
subroutine vasinpi (n, y, incy, x, incx);
subroutine vasinpif(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sin^{-1}(\mathbf{x})/\pi$$

6.17.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.18 vacospi - Vector arccosine $\cos^{-1}(\mathbf{x})/\pi$

```
subroutine vacospi (n, y, incy, x, incx);
subroutine vacospif(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cos^{-1}(\mathbf{x})/\pi$$

6.18.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

6.19 vatanpi - Vector arctangent $\tan^{-1}(\mathbf{x})/\pi$

```
subroutine vatanpi (n, y, incy, x, incx);
subroutine vatanpif(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tan^{-1}(\mathbf{x})/\pi$$

6.19.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

7 Hypergeometric Functions

7.1 `vsinh` - Vector hypergeometric sine sinh(**x**)

```
subroutine vsinh (n, y, incy, x, incx);
subroutine vsinhf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sinh(\mathbf{x})$$

7.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

7.2 vcosh - Vector hypergeometric cosine cosh(**x**)

```
subroutine vcosh (n, y, incy, x, incx);
subroutine vcoshf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cosh(\mathbf{x})$$

7.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

7.3 vtanh - Vector hypergeometric tangent tanh(**x**)

```
subroutine vtanh (n, y, incy, x, incx);
subroutine vtanhf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tanh(\mathbf{x})$$

7.3.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

7.4 vasinh - Vector hypergeometric arcsine $\sinh^{-1}(\mathbf{x})$

```
subroutine vasinh (n, y, incy, x, incx);
subroutine vasinhf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sinh^{-1}(\mathbf{x})$$

7.4.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

7.5 vacosh - Vector hypergeometric arccosine $\cosh^{-1}(\mathbf{x})$

```
subroutine vacosh (n, y, incy, x, incx);
subroutine vacoshf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cosh^{-1}(\mathbf{x})$$

7.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

7.6 vatanh - Vector hypergeometric arctangent $\tanh^{-1}(\mathbf{x})$

```
subroutine vatanh (n, y, incy, x, incx);
subroutine vatanhf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tanh^{-1}(\mathbf{x})$$

7.6.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8 Exponentials and Logarithms

8.1 `vexp` - Vector exponential $e^{\mathbf{x}}$

```
subroutine vexp (n, y, incy, x, incx);
subroutine vexpf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = e^{\mathbf{x}}$$

8.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.2 vexpm1 - Vector exponential $e^{\mathbf{x}}-1$

```
subroutine vexpm1 (n, y, incy, x, incx);
subroutine vexpm1f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = e^{\mathbf{x}} - 1$$

8.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.3 vexp2 - Vector binary exponential $2^{\mathbf{x}}$

```
subroutine vexp2 (n, y, incy, x, incx);
subroutine vexp2f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = 2^{\mathbf{x}}$$

8.3.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.4 vlog - Vector logarithm log(**x**)

```
subroutine vlog (n, y, incy, x, incx);
subroutine vlogf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \log(\mathbf{x})$$

8.4.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.5 vlog2 - Vector binary logarithm $\log_2(\mathbf{x})$

```
subroutine vlog2 (n, y, incy, x, incx);
subroutine vlog2f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \log_2(\mathbf{x})$$

8.5.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.6 vlog10 - Vector base-10 logarithm $\log_{10}(\mathbf{x})$

```
subroutine vlog10 (n, y, incy, x, incx);
subroutine vlog10f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \log_{10}(\mathbf{x})$$

8.6.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.7 vlog1p - Vector logarithm log(**x** + 1)

```
subroutine vlog1p (n, y, incy, x, incx);
subroutine vlog1pf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \log(\mathbf{x} + 1)$$

8.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

8.8 vpow - Vector power $\mathbf{x}^{\mathbf{y}}$

```
subroutine vpow (n, z, incz, x, incx, y, incy);
subroutine vpowf(n, z, incz, x, incx, y, incy);
```

Given an input vectors **x** and **y** as well as a result vector **z**, this function computes:

$$\mathbf{z} = \mathbf{x}^{\mathbf{y}}$$

8.8.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z **- ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ **- INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X **- ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y **- ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

8.9 vpows - Vector power scalar exponent$\mathbf{x}^y$

```
subroutine vpows (n, z, incz, x, incx, y);
subroutine vpowsf(n, z, incz, x, incx, y);
```

Given an input vectors **x** and a scalar value y as well as a result vector **z**, this function computes:

$$\mathbf{z} = \mathbf{x}^y$$

8.9.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **REAL** *ENTRY:* Exponentiation value y.

8.10 vmod - Vector modulus mod(**x**,**y**)

```
subroutine vmod (n, z, incz, x, incx, y, incy);
subroutine vmodf(n, z, incz, x, incx, y, incy);
```

Given an input vectors **x** and **y** as well as a result vector **z**, this function computes:

$$\mathbf{z} = \mathrm{mod}(\mathbf{x},\mathbf{y})$$

8.10.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

9 Special Functions

9.1 `verf` - Vector error function erf(**x**)

```
subroutine verf (n, y, incy, x, incx);
subroutine verff(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \mathrm{erf}(\mathbf{x})$$

9.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.2 verfc - Vector complementary error function erfc(**x**)

```
subroutine verfc (n, y, incy, x, incx);
subroutine verfcf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \text{erfc}(\mathbf{x})$$

9.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.3 vbesj0 - Vector Bessel Function $J_0(\mathbf{x})$

```
subroutine vbesj0 (n, y, incy, x, incx);
subroutine vbesj0f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = J_0(\mathbf{x})$$

9.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.4 vbesy0 - Vector Bessel Function $Y_0(\mathbf{x})$

```
subroutine vbesy0 (n, y, incy, x, incx);
subroutine vbesy0f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = Y_0(\mathbf{x})$$

9.4.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.5 vbesj1 - Vector Bessel Function $J_1(\mathbf{x})$

```
subroutine vbesj1 (n, y, incy, x, incx);
subroutine vbesj1f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = J_1(\mathbf{x})$$

9.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.6 vbesy1 - Vector Bessel Function $Y_1(\mathbf{x})$

```
subroutine vbesy1 (n, y, incy, x, incx);
subroutine vbesy1f(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = Y_1(\mathbf{x})$$

9.6.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.7 vbesjn - Vector Bessel Function $J_n(\mathbf{x})$

```
subroutine vbesjn (n, y, incy, k, x, incx);
subroutine vbesjnf(n, y, incy, k, x, incx);
```

Given an input vector $\mathbf{x}$, an order k and a result vector $\mathbf{y}$, this function computes:

$$\mathbf{y} = J_k(\mathbf{x})$$

9.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of $\mathbf{x}$ and $\mathbf{y}$.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector $\mathbf{y}$.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array $\mathbf{x}$.

INCY - **INTEGER** *ENTRY:* Stride for the vector $\mathbf{y}$.
CONSTRAINT: incy > 0.

K - **INTEGER** *ENTRY:* Order k.

X - **ARRAY OF REAL** *ENTRY:* Input vector $\mathbf{x}$.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array $\mathbf{y}$.

INCX - **INTEGER** *ENTRY:* Stride for the vector $\mathbf{x}$.
CONSTRAINT: incx > 0.

9.8 vbesyn - Vector Bessel Function $Y_n(\mathbf{x})$

```
subroutine vbesyn (n, y, incy, k, x, incx);
subroutine vbesynf(n, y, incy, k, x, incx);
```

Given an input vector **x**, an order k and a result vector **y**, this function computes:

$$\mathbf{y} = Y_k(\mathbf{x})$$

9.8.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

K - **INTEGER** *ENTRY:* Order k.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

9.9 vlgamma - Vector Log-Gamma $\log\Gamma(\mathbf{x})$

```
subroutine vlgamma (n, y, incy, x, incx);
subroutine vlgammaf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \log\Gamma(\mathbf{x})$$

9.9.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

10 Other Functions

10.1 `vabs` - Vector absolute value |**x**|

```
subroutine vabs (n, y, incy, x, incx);
subroutine vabsf(n, y, incy, x, incx);
```

Given an input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = |\mathbf{x}|$$

10.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

10.2 vhypot - Vector euclidean distance $\sqrt{\mathbf{x}^2+\mathbf{y}^2}$

```
subroutine vhypot (n, z, incz, x, incx, y, incy);
subroutine vhypotf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes:

$$\mathbf{z} = \sqrt{\mathbf{x}^2+\mathbf{y}^2}$$

10.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

10.3 vrem - Vector remainder

```
subroutine vrem (n, z, incz, x, incx, y, incy);
subroutine vremf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes the remainder of dividing x by y and stores the result in z.

10.3.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

`Z` - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

`INCZ` - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

`Y` - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

11 Arithmetic Functions

11.1 `vadd` - Vector addition $\mathbf{x} + \mathbf{y}$

```
subroutine vadd (n, z, incz, x, incx, y, incy);
subroutine vaddf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x} + \mathbf{y}$$

11.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

`Z` - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

`INCZ` - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

`Y` - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

11.2 vsadd - Vector scalar addition $\mathbf{x}+\alpha$

```
subroutine vsadd (n, y, incy, x, incx, a);
subroutine vsaddf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \mathbf{x} + \alpha$$

11.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

A - **REAL** *ENTRY:* Scalar constant α.

11.3 vsub - Vector subtraction x − y

```
subroutine vsub (n, z, incz, x, incx, y, incy);
subroutine vsubf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x} - \mathbf{y}$$

11.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

11.4 vssub - Vector scalar subtraction $\mathbf{x} - \alpha$

```
subroutine vssub (n, y, incy, x, incx, a);
subroutine vssubf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \mathbf{x} - \alpha$$

11.4.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

A - **REAL** *ENTRY:* Scalar constant α.

11.5 vmul - Vector multiplication xy

```
subroutine vmul (n, z, incz, x, incx, y, incy);
subroutine vmulf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{xy}$$

11.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

11.6 vsmul - Vector scalar multiplication $\alpha\mathbf{x}$

```
subroutine vsmul (n, y, incy, x, incx, a);
subroutine vsmulf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \alpha\mathbf{x}$$

11.6.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

A - **REAL** *ENTRY:* Scalar constant α.

11.7 vdiv - Vector division x/y

```
subroutine vdiv (n, z, incz, x, incx, y, incy);
subroutine vdivf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x}/\mathbf{y}$$

11.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF REAL** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF REAL** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

11.8 vsdiv - Vector scalar division **x**/α

```
subroutine vsdiv (n, y, incy, x, incx, a);
subroutine vsdivf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \mathbf{x}/\alpha$$

11.8.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

A - **REAL** *ENTRY:* Scalar constant α.
CONSTRAINT: $\alpha \neq 0$.

11.9 vrecp - Vector reciprocal 1/**x**

```
subroutine vrecp (n, z, incz, x, incx, y, incy);
subroutine vrecpf(n, z, incz, x, incx, y, incy);
```

Given input vector **x** and a result vector **y**, this function computes

$$\mathbf{y} = 1/\mathbf{x}$$

11.9.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12 Complex Numbers

12.1 `vcreal` - Vector complex real compoment Re(**x**)

```
subroutine vcreal (n, y, incy, x, incx);
subroutine vcrealf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a real result vector **y**, this function computes:

$$\mathbf{y} = \mathrm{Re}(\mathbf{x})$$

Where $\mathbf{x} \in \mathbb{C}$ and $\mathbf{y} \in \mathbb{R}$.

12.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.2 vcimag - Vector complex imaginary compoment Im(**x**)

```
subroutine vcimag (n, y, incy, x, incx);
subroutine vcimagf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a real result vector **y**, this function computes:

$$\mathbf{y} = \mathrm{Im}(\mathbf{x})$$

Where $\mathbf{x} \in \mathbb{C}$ and $\mathbf{y} \in \mathbb{R}$.

12.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.3 vcabs - Vector complex absolute value |**x**|

```
subroutine vcabs (n, y, incy, x, incx);
subroutine vcabsf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = |\mathbf{x}|$$

Where $\mathbf{x} \in \mathbb{C}$ and $\mathbf{y} \in \mathbb{R}$.

12.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.4 vcarg - Vector complex argument arg(**x**)

```
subroutine vcarg (n, y, incy, x, incx);
subroutine vcargf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \arg(\mathbf{x})$$

Where $\mathbf{x} \in \mathbb{C}$ and $\mathbf{y} \in \mathbb{R}$.

12.4.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.5 vconj - Vector complex conjugate $\overline{\mathbf{x}}$

```
subroutine vconj (n, y, incy, x, incx);
subroutine vconjf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \overline{\mathbf{x}}$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

12.5.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.6 vcproj - Vector complex Riemann sphere projection proj(**x**)

```
subroutine vcproj (n, y, incy, x, incx);
subroutine vcprojf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \text{proj}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

12.6.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.7 vcexp - Vector complex exponentiation exp(**x**)

```
subroutine vcexp (n, y, incy, x, incx);
subroutine vcexpf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \exp(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

12.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.8 vclog - Vector complex logarithm log(**x**)

```
subroutine vclog (n, y, incy, x, incx);
subroutine vclogf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \log(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

12.8.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.9 vcsqrt - Vector complex square root $\sqrt{\mathbf{x}}$

```
subroutine vcsqrt (n, y, incy, x, incx);
subroutine vcsqrtf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sqrt{\mathbf{x}}$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

12.9.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

12.10 vcpow - Vector complex power $\mathbf{x}^{\mathbf{y}}$

```
subroutine vcpow (n, z, incz, x, incx, y, incy);
subroutine vcpowf(n, z, incz, x, incx, y, incy);
```

Given an input vectors **x** and **y** as well as a result vector **z**, this function computes:

$$\mathbf{z} = \mathbf{x}^{\mathbf{y}}$$

Where $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{C}$.

12.10.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z **- ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ **- INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X **- ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y **- ARRAY OF COMPLEX** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

12.11 vcpows - Vector complex power scalar exponent $\mathbf{x}^y$

```
subroutine vcpows (n, z, incz, x, incx, y);
subroutine vcpowsf(n, z, incz, x, incx, y);
```

Given an input vectors **x** and a scalar value y as well as a result vector **z**, this function computes:

$$\mathbf{z} = \mathbf{x}^y$$

12.11.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

z - **ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **COMPLEX** *ENTRY:* Exponentiation value y.

13 Complex Trigonometric Functions

13.1 `vcsin` - Vector complex sine $\sin(\mathbf{x})$

```
subroutine vcsin (n, y, incy, x, incx);
subroutine vcsinf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sin(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.2 vccos - Vector complex cosine cos(**x**)

```
subroutine vccos (n, y, incy, x, incx);
subroutine vccosf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cos(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.2.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.3 vctan - Vector complex tangent tan(**x**)

```
subroutine vctan (n, y, incy, x, incx);
subroutine vctanf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tan(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.4 vcasin - Vector complex arcsine $\sin^{-1}(\mathbf{x})$

```
subroutine vcasin (n, y, incy, x, incx);
subroutine vcasinf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sin^{-1}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.4.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.5 vcacos - Vector complex arccosine $\cos^{-1}(\mathbf{x})$

```
subroutine vcacos (n, y, incy, x, incx);
subroutine vcacosf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cos^{-1}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.6 vcatan - Vector complex arctangent $\tan^{-1}(\mathbf{x})$

```
subroutine vcatan (n, y, incy, x, incx);
subroutine vcatanf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tan^{-1}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.6.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.7 vcsinh - Vector complex hyperbolic sine sinh(**x**)

```
subroutine vcsinh (n, y, incy, x, incx);
subroutine vcsinhf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sinh(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.8 vccosh - Vector complex hyperbolic cosine cosh(**x**)

```
subroutine vcosh (n, y, incy, x, incx);
subroutine vcoshf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \cosh(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.8.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.9 vctanh - Vector complex hyperbolic tangent tanh(**x**)

```
subroutine vctanh (n, y, incy, x, incx);
subroutine vctanhf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tanh(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.9.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.10 vcasinh - Vector complex hyperbolic arcsine $\sinh^{-1}(\mathbf{x})$

```
subroutine vcasinh (n, y, incy, x, incx);
subroutine vcasinhf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \sinh^{-1}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.10.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

13.11 vcacosh - Vector complex hyperbolic arccosine $\cosh^{-1}(\mathbf{x})$

```
subroutine vcacosh (n, y, incy, x, incx);
subroutine vcacoshf(n, y, incy, x, incx);
```

Given a complex input vector $\mathbf{x}$ and a result vector $\mathbf{y}$, this function computes:

$$\mathbf{y} = \cosh^{-1}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.11.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of $\mathbf{x}$ and $\mathbf{y}$.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector $\mathbf{y}$.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array $\mathbf{x}$.

INCY - **INTEGER** *ENTRY:* Stride for the vector $\mathbf{y}$.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector $\mathbf{x}$.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array $\mathbf{y}$.

INCX - **INTEGER** *ENTRY:* Stride for the vector $\mathbf{x}$.
CONSTRAINT: incx > 0.

13.12 vcatanh - Vector complex hyperbolic arctangent $\tanh^{-1}(\mathbf{x})$

```
subroutine vcatanh (n, y, incy, x, incx);
subroutine vcatanhf(n, y, incy, x, incx);
```

Given a complex input vector **x** and a result vector **y**, this function computes:

$$\mathbf{y} = \tanh^{-1}(\mathbf{x})$$

Where $\mathbf{x}, \mathbf{y} \in \mathbb{C}$.

13.12.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

14 Complex Arithmetic

14.1 `vcadd` - Vector complex addition $\mathbf{x} + \mathbf{y}$

```
subroutine vcadd (n, z, incz, x, incx, y, incy);
subroutine vcaddf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x} + \mathbf{y}$$

14.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

`Z` - **ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

`INCZ` - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

`Y` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

14.2 vcsadd - Vector complex scalar subtraction $\mathbf{x}+\alpha$

```
subroutine vcsadd (n, y, incy, x, incx, a);
subroutine vcsaddf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \mathbf{x} + \alpha$$

14.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

A - **COMPLEX** *ENTRY:* Scalar constant α.

14.3 vcsub - Vector complex subtraction $\mathbf{x} - \mathbf{y}$

```
subroutine vcsub (n, z, incz, x, incx, y, incy);
subroutine vcsubf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x} - \mathbf{y}$$

14.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF COMPLEX** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

14.4 vcssub - Vector complex scalar subtraction $\mathbf{x} - \alpha$

```
subroutine vcssub (n, y, incy, x, incx, a);
subroutine vcssubf(n, y, incy, x, incx, a);
```

Given an input vector $\mathbf{x}$ and a scalar constant α and a result vector $\mathbf{y}$, this function computes:

$$\mathbf{y} = \mathbf{x} - \alpha$$

14.4.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of $\mathbf{x}$ and $\mathbf{y}$.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector $\mathbf{y}$.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array $\mathbf{x}$.

INCY - **INTEGER** *ENTRY:* Stride for the vector $\mathbf{y}$.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector $\mathbf{x}$.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array $\mathbf{y}$.

INCX - **INTEGER** *ENTRY:* Stride for the vector $\mathbf{x}$.
CONSTRAINT: incx > 0.

A - **COMPLEX** *ENTRY:* Scalar constant α.

14.5 vcmul - Vector complex multiplication xy

```
subroutine vcmul (n, z, incz, x, incx, y, incy);
subroutine vcmulf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{xy}$$

14.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF COMPLEX** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

14.6 vcsmul - Vector complex scalar multiplication $\alpha\mathbf{x}$

```
subroutine vcsmul (n, y, incy, x, incx, a);
subroutine vcsmulf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \alpha\mathbf{x}$$

14.6.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

`A` - **COMPLEX** *ENTRY:* Scalar constant α.

14.7 vcmulc - Vector complex conjugate multiplication $\mathbf{x}\bar{\mathbf{y}}$

```
subroutine vcmulc (n, z, incz, x, incx, y, incy);
subroutine vcmulcf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x}\bar{\mathbf{y}}$$

14.7.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF COMPLEX** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

14.8 vcdiv - Vector complex division x/y

```
subroutine vcdiv (n, z, incz, x, incx, y, incy);
subroutine vcdivf(n, z, incz, x, incx, y, incy);
```

Given input vectors **x** and **y** and a result vector **z**, this function computes

$$\mathbf{z} = \mathbf{x}/\mathbf{y}$$

14.8.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x**, **y** and **z**.
CONSTRAINT: $n \geq 1$.

Z - **ARRAY OF COMPLEX** *EXIT:* Result vector **z**.
CONSTRAINT: Must contain $n \times$ incz elements.
CONSTRAINT: Must not overlap with array **x** or **y**.

INCZ - **INTEGER** *ENTRY:* Stride for the vector **z**.
CONSTRAINT: incz > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **z** or **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

Y - **ARRAY OF COMPLEX** *ENTRY:* Input vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **z** or **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

14.9 vcsdiv - Vector complex scalar division **x**/α

```
subroutine vcsdiv (n, y, incy, x, incx, a);
subroutine vcsdivf(n, y, incy, x, incx, a);
```

Given an input vector **x** and a scalar constant α and a result vector **y**, this function computes:

$$\mathbf{y} = \mathbf{x}/\alpha$$

14.9.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF COMPLEX** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF COMPLEX** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

A - **COMPLEX** *ENTRY:* Scalar constant α.

15 Acknowledgements

Parts of this product include or are derived from code under the following licences:

```
/*
 * Copyright (c) 2003, Steven G. Kargl
 * Copyright (c) 2005, 2011 David Schultz <das@FreeBSD.ORG>
 * Copyright (c) 2005 Bruce D. Evans and Steven G. Kargl
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

/*
 * Copyright (c) 2008 Stephen L. Moshier <steve@moshier.net>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */

------------------------------------------------------------------------
Copyright © 2005-2020 Rich Felker, et al.
Copyright © 2013 Szabolcs Nagy

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
```