Accelerate / cblas_sgemm(_:_:_:_:_:_:_:_:_:_:_:_:_:_:)

Function

cblas_sgemm(_:_:_:_:_:_:_:_:_:_:_:_:_:_:)

Multiplies two matrices (single-precision).

iOS 16.4+ | iPadOS 16.4+ | Mac Catalyst 16.4+ | macOS 13.3+ | tvOS 16.4+ | visionOS 1.0+ | watchOS 9.4+

```
func cblas_sgemm(
    _ ORDER: CBLAS_ORDER,
    _ TRANSA: CBLAS_TRANSPOSE,
    _ TRANSB: CBLAS_TRANSPOSE,
    _ M: __LAPACK_int,
    _ N: __LAPACK_int,
    _ K: __LAPACK_int,
    _ ALPHA: Float,
    _ A: UnsafePointer<Float>?,
    _ LDA: __LAPACK_int,
    _ B: UnsafePointer<Float>?,
    _ LDB: __LAPACK_int,
    _ BETA: Float,
    _ C: UnsafeMutablePointer<Float>?,
    _ LDC: __LAPACK_int
)
```

Parameters

`ORDER`

Specifies row-major (C) or column-major (Fortran) data ordering.

`TRANSA`

Specifies whether to transpose matrix A.

`TRANSB`

Specifies whether to transpose matrix `B`.

`M`

Number of rows in matrices `A` and `C`.

`N`

Number of columns in matrices `B` and `C`.

`K`

Number of columns in matrix `A`; number of rows in matrix `B`.

`ALPHA`

Scaling factor for the product of matrices A and B.

`A`

Matrix A.

`LDA`

The size of the first dimension of matrix `A`; if you are passing a matrix `A[m][n]`, the value should be `m`.

`B`

Matrix B.

`LDB`

The size of the first dimension of matrix `B`; if you are passing a matrix `B[m][n]`, the value should be `m`.

`BETA`

Scaling factor for matrix C.

`C`

Matrix C.

`LDC`

The size of the first dimension of matrix `C`; if you are passing a matrix `C[m][n]`, the value should be `m`.

Discussion

This function multiplies `A * B` and multiplies the resulting matrix by `alpha`. It then multiplies matrix `C` by `beta`. It stores the sum of these two products in matrix `C`.

Thus, it calculates either

$C \leftarrow \alpha AB + \beta C$

or

$C \leftarrow \alpha BA + \beta C$

with optional use of transposed forms of A, B, or both.

> **Important**
>
> Apple provides the BLAS and LAPACK libraries under the Accelerate framework to be in line with LAPACK 3.9.1. Starting with iOS 19, iPadOS 19, macOS 16, tvOS 19, watchOS 19, and visionOS 3, the libraries are in line with LAPACK 3.12.0. These new interfaces provide additional functionality, as well as a new ILP64 interface. To use the new interfaces, define `ACCELERATE_NEW_LAPACK` before including the Accelerate or vecLib headers. For ILP64 interfaces, also define `ACCELERATE_LAPACK_ILP64`. For Swift projects, specify `ACCELERATE_NEW_LAPACK=1` and `ACCELERATE_LAPACK_ILP64=1` as preprocessor macros in Xcode build settings.

See Also

Single-precision float matrix functions

```
func cblas_sasum(__LAPACK_int, UnsafePointer<Float>?, __LAPACK_int) -> Float
```

Computes the sum of the absolute values of elements in a vector (single-precision).

```
func cblas_saxpy(__LAPACK_int, Float, UnsafePointer<Float>?, __LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Computes a constant times a vector plus a vector (single-precision).

```
func cblas_scopy(__LAPACK_int, UnsafePointer<Float>?, __LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Copies a vector to another vector (single-precision).

```
func cblas_sgbmv(CBLAS_ORDER, CBLAS_TRANSPOSE, __LAPACK_int, __LAPACK_int, __LAPACK_int, __LAPACK_int, Float, UnsafePointer<Float>?, __LAPACK_int, UnsafePointer<Float>?, __LAPACK_int, Float, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Scales a general band matrix, then multiplies by a vector, then adds a vector (single precision).

```
func cblas_sgemv(CBLAS_ORDER, CBLAS_TRANSPOSE, __LAPACK_int, __LAPACK_int, Float, UnsafePointer<Float>?, __LAPACK_int, UnsafePointer<Float>?, __LAPACK_int, Float, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Multiplies a single-precision matrix by a vector.

```
func cblas_sger(CBLAS_ORDER, __LAPACK_int, __LAPACK_int, Float, UnsafePointer<Float>?, __LAPACK_int, UnsafePointer<Float>?, __LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Multiplies vector X by the transpose of vector Y, then adds matrix A (single precison).

```
func cblas_snrm2(__LAPACK_int, UnsafePointer<Float>?, __LAPACK_int) -> Float
```

Computes the L2 norm (Euclidian length) of a vector (single precision).

```
func cblas_srot(__LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int, Float, Float)
```

Applies a Givens rotation matrix to a pair of vectors.

```
func cblas_srotg(UnsafeMutablePointer<Float>, UnsafeMutablePointer<Float>, UnsafeMutablePointer<Float>, UnsafeMutablePointer<Float>)
```

Constructs a Givens rotation matrix.

```
func cblas_srotm(__LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int, UnsafeMutablePointer<Float>?, __LAPACK_int, UnsafePointer<Float>)
```

Applies a modified Givens transformation (single precision).

```
func cblas_srotmg(UnsafeMutablePointer<Float>, UnsafeMutablePointer<Float>, UnsafeMutablePointer<Float>, Float, UnsafeMutablePointer<Float>)
```

Generates a modified Givens rotation matrix.

```
func cblas_ssbmv(CBLAS_ORDER, CBLAS_UPLO, __LAPACK_int, __LAPACK_int, Float, UnsafePointer<Float>?, __LAPACK_int, UnsafePointer<Float>?, __LAPACK_int, Float, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Scales a symmetric band matrix, then multiplies by a vector, then adds a vector (single-precision).

```
func cblas_sscal(__LAPACK_int, Float, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Multiplies each element of a vector by a constant (single-precision).

```
func cblas_sspmv(CBLAS_ORDER, CBLAS_UPLO, __LAPACK_int, Float, UnsafePointer<Float>?, UnsafePointer<Float>?, __LAPACK_int, Float, UnsafeMutablePointer<Float>?, __LAPACK_int)
```

Scales a packed symmetric matrix, then multiplies by a vector, then scales and adds another vector (single precision).

```
func cblas_sspr(CBLAS_ORDER, CBLAS_UPLO, __LAPACK_int, Float, UnsafePointer<Float>?, __LAPACK_int, UnsafeMutablePointer<Float>?)
```

Rank one update: adds a packed symmetric matrix to the product of a scaling factor, a vector, and its transpose (single precision).