Matrix3D

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Mat3D Struct Reference

Contiguous 3D float matrix (tensor).

`#include <Matrix3D.h>`

Public Attributes

- size_t rows

  Number of rows (size of the i dimension).
- size_t cols

  Number of columns (size of the j dimension).
- size_t layers

  Number of layers (size of the k dimension).
- size_t stride

  Row stride in elements.
- float * elements

  Pointer to the contiguous array of elements.

3.1.1 Detailed Description

Contiguous 3D float matrix (tensor).

Elements are stored in a single contiguous allocation of rows * cols * layers floats using the layout: ((i * cols) + j) * layers + k

Definition at line 100 of file Matrix3D.h.

3.1.2 Member Data Documentation

3.1.2.1 cols

`size_t Mat3D::cols`

Number of columns (size of the j dimension).

Definition at line 108 of file Matrix3D.h.

Referenced by mat3D_alloc(), mat3D_copy(), mat3D_fill(), mat3D_identity_mat(), mat3D_mult(), mat3D_print(), mat3D_rand(), and mat3D_sum().

3.1.2.2 elements

`float* Mat3D::elements`

Pointer to the contiguous array of elements.

Definition at line 124 of file Matrix3D.h.

Referenced by mat3D_alloc(), and mat3D_free().

3.1.2.3 layers

`size_t Mat3D::layers`

Number of layers (size of the k dimension).

Definition at line 112 of file Matrix3D.h.

Referenced by mat3D_alloc(), mat3D_copy(), mat3D_fill(), mat3D_identity_mat(), mat3D_mult(), mat3D_print(), mat3D_rand(), and mat3D_sum().

3.1.2.4 rows

`size_t Mat3D::rows`

Number of rows (size of the i dimension).

Definition at line 104 of file Matrix3D.h.

Referenced by mat3D_alloc(), mat3D_copy(), mat3D_fill(), mat3D_identity_mat(), mat3D_mult(), mat3D_print(), mat3D_rand(), and mat3D_sum().

3.1.2.5 stride

`size_t Mat3D::stride`

Row stride in elements.

For this layout, stride == cols $*$ layers. Exposed for convenience and potential future use. Not required by the provided API.

Definition at line 120 of file Matrix3D.h.

Referenced by mat3D_alloc().

The documentation for this struct was generated from the following file:

- Matrix3D.h

Chapter 4

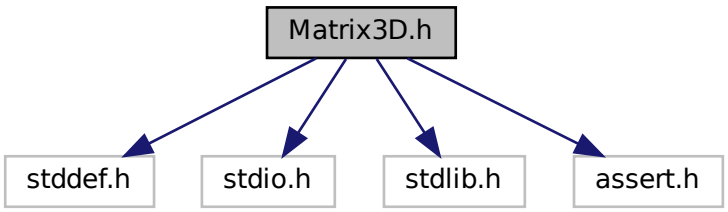
File Documentation

4.1 Matrix3D.h File Reference

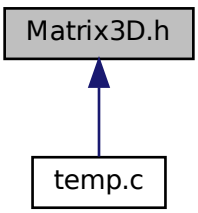
Single-header 3D matrix (tensor) utilities for contiguous float data.

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
```

Include dependency graph for Matrix3D.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Mat3D

  Contiguous 3D float matrix (tensor).

Macros

- #define MATRIX3D_MALLOC malloc

  Allocation function used by mat3D_alloc.
- #define MATRIX3D_ASSERT assert

  Assertion macro used for parameter and allocation checks.
- #define MAT3D_AT(m, i, j, k) (m).elements[((i)∗(m).cols + (j))∗(m).layers + (k)]

  Lvalue (Locator value) access to element at (i, j, k).
- #define MAT3D_PRINT(m) mat3D_print(m, #m, 0)

  Convenience macro to print a Mat3D with its variable name and no left padding.

Functions

- float mat3D_rand_float (void)

  Generate a pseudo-random float in [0, 1].
- Mat3D mat3D_alloc (size_t rows, size_t cols, size_t layers)

  Allocate a 3D matrix with the given shape.
- void mat3D_free (Mat3D m)

  Free the memory owned by a Mat3D.
- void mat3D_fill (Mat3D m, float x)

  Fill all elements with a constant value.
- void mat3D_rand (Mat3D m, float low, float high)

  Fill all elements with random values in [low, high].
- void mat3D_sum (Mat3D dst, Mat3D a)

  Element-wise addition: dst += a.
- void mat3D_mult (Mat3D m, float factor)

  In-place scaling by an integer factor: m ∗= factor.
- void mat3D_print (Mat3D m, const char ∗name, size_t padding)

  Print the matrix with a name and left padding.
- void mat3D_identity_mat (Mat3D m)

  Set m to the 3D identity tensor (Kronecker delta).
- void mat3D_copy (Mat3D res, Mat3D src)

  Copy all elements from src to res.

4.1.1 Detailed Description

Single-header 3D matrix (tensor) utilities for contiguous float data.

Matrix3D is a minimal, dependency-free C library for working with dense 3D arrays (tensors) of floats. It provides allocation, initialization, random filling, element-wise addition, scaling, identity construction, copying, and pretty-printing.

Memory layout is row-major with the following linearization: index(i, j, k) = ((i ∗ cols) + j) ∗ layers + k

- i in [0, rows)
- j in [0, cols)
- k in [0, layers)
- k is the fastest-varying dimension.

Usage:

- In exactly one translation unit, define MATRIX3D_IMPLEMENTATION before including this header to get the function definitions.
- In all other translation units, include this header without the define.

Example:

```
#define MATRIX3D_IMPLEMENTATION
#include "matrix3d.h"
int main(void) {
    Mat3D m = mat3D_alloc(2, 3, 4);
    mat3D_fill(m, 0.0f);
    MAT3D_AT(m, 1, 2, 3) = 42.0f;
    MAT3D_PRINT(m);
    mat3D_free(m);
    return 0;
}
```

Configuration:

- Define MATRIX3D_MALLOC to override the allocator used by mat3D_alloc.
- Define MATRIX3D_ASSERT to override assertions.

Thread-safety:

- All functions operate on caller-provided objects. There is no internal global state other than the use of C's rand() in mat3D_rand_float and mat3D_rand, which is not thread-safe. Seed with srand() as needed.

Error handling:

- By default, allocation failure triggers MATRIX3D_ASSERT. If you need graceful handling, override MATRIX3↩ D_ASSERT.

Note

This one-file library is heavily inspired by Tsoding's nn.h implementation of matrix creation and operation. you can find the source code in: https://github.com/tsoding/nn.h . featured in this video of his: https://youtu.be/L1TbWe8bVOc?list=PLpM-Dvs8t0VZPZKggcql-MmjaBdZKeDMw .

Definition in file Matrix3D.h.

4.1.2 Macro Definition Documentation

4.1.2.1 MAT3D_AT

```
#define MAT3D_AT(
                m,
                i,
                j,
                k ) (m).elements[((i)*(m).cols + (j))*(m).layers + (k)]
```

Lvalue (Locator value) access to element at (i, j, k).

Parameters

m	Mat3D lvalue or expression
i	Row index in [0, m.rows)
j	Column index in [0, m.cols)
k	Layer index in [0, m.layers)

Returns

An lvalue reference to the element at (i, j, k).

Warning

The macro evaluates its arguments multiple times. Avoid passing expressions with side effects.

Definition at line 142 of file Matrix3D.h.

4.1.2.2 MAT3D_PRINT

```
#define MAT3D_PRINT(
              m ) mat3D_print(m, #m, 0)
```

Convenience macro to print a Mat3D with its variable name and no left padding.

Parameters

m	Mat3D to print.

Definition at line 150 of file Matrix3D.h.

4.1.2.3 MATRIX3D_ASSERT

```
#define MATRIX3D_ASSERT assert
```

Assertion macro used for parameter and allocation checks.

Defaults to assert. You may override it to integrate with your project's error handling or to disable assertions in production builds.

Definition at line 89 of file Matrix3D.h.

4.1.2.4 MATRIX3D_MALLOC

```
#define MATRIX3D_MALLOC malloc
```

Allocation function used by mat3D_alloc.

Defaults to malloc. You may override this macro before including the header to provide a custom allocator (e.g., an arena or tracking allocator). The memory must be free()'able by mat3D_free (which calls free()) unless you also replace mat3D_free.

Definition at line 76 of file Matrix3D.h.

4.1.3 Function Documentation

4.1.3.1 mat3D_alloc()

```
Mat3D mat3D_alloc (
            size_t rows,
            size_t cols,
            size_t layers )
```

Allocate a 3D matrix with the given shape.

Parameters

rows	Number of rows (i dimension).
cols	Number of columns (j dimension).
layers	Number of layers (k dimension).

Returns

Mat3D with allocated elements and initialized metadata.

Precondition

rows $*$ cols $*$ layers $>$ 0. Behavior is undefined for zero-sized allocations due to assertion on NULL pointers.

Postcondition

m.elements is non-NULL on success (asserts otherwise).

Note

Elements are uninitialized.

See also

mat3D_free

Definition at line 199 of file Matrix3D.h.

References Mat3D::cols, Mat3D::elements, Mat3D::layers, MATRIX3D_ASSERT, MATRIX3D_MALLOC, Mat3D::rows, and Mat3D::stride.

Referenced by main().

4.1.3.2 mat3D_copy()

```
void mat3D_copy (
            Mat3D res,
            Mat3D src )
```

Copy all elements from src to res.

Parameters

res	Destination matrix.
src	Source matrix.

Precondition

res.rows == src.rows, res.cols == src.cols, res.layers == src.layers.

Definition at line 370 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, MATRIX3D_ASSERT, and Mat3D::rows.

4.1.3.3 mat3D_fill()

```
void mat3D_fill (
            Mat3D m,
            float x )
```

Fill all elements with a constant value.

Parameters

m	Destination matrix.
x	Value to assign to each element.

Definition at line 231 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, and Mat3D::rows.

Referenced by main().

4.1.3.4 mat3D_free()

```
void mat3D_free (
            Mat3D m )
```

Free the memory owned by a Mat3D.

Parameters

m	Mat3D to free.

Postcondition

m.elements becomes invalid after this call.

Note

Safe to call with m.elements == NULL (free(NULL) is a no-op).

Warning

Do not double-free the same Mat3D.

Definition at line 220 of file Matrix3D.h.

References Mat3D::elements.

Referenced by main().

4.1.3.5 mat3D_identity_mat()

```
void mat3D_identity_mat (
            Mat3D m )
```

Set m to the 3D identity tensor (Kronecker delta).

Parameters

m	Destination matrix.

Precondition

m.rows == m.cols == m.layers.

Sets element (i, j, k) to 1 when i == j == k, otherwise 0.

Definition at line 345 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, MATRIX3D_ASSERT, and Mat3D::rows.

Referenced by main().

4.1.3.6 mat3D_mult()

```
void mat3D_mult (
            Mat3D m,
            float factor )
```

In-place scaling by an integer factor: m ∗= factor.

Parameters

m	Matrix to scale.
factor	Scale factor (size_t). Elements are multiplied as floats.

Definition at line 290 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, and Mat3D::rows.

4.1.3.7 mat3D_print()

```
void mat3D_print (
            Mat3D m,
            const char * name,
            size_t padding )
```

Print the matrix with a name and left padding.

Parameters

m	Matrix to print.
name	A label to print before the data (e.g., the variable name).
padding	Number of spaces of left padding for each printed line.

The output is organized by layers (k), printing each 2D slice with row-wise lines.

Definition at line 312 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, and Mat3D::rows.

4.1.3.8 mat3D_rand()

```
void mat3D_rand (
            Mat3D m,
            float low,
            float high )
```

Fill all elements with random values in [low, high].

Parameters

m	Destination matrix.
low	Lower bound (inclusive).
high	Upper bound (exclusive when using typical rand()).

Note

Uses mat3D_rand_float and therefore rand(); not thread-safe.

Definition at line 251 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, mat3D_rand_float(), and Mat3D::rows.

4.1.3.9 mat3D_rand_float()

```
float mat3D_rand_float (
            void  )
```

Generate a pseudo-random float in [0, 1].

Uses C's rand() scaled to [0, 1]. The random stream is shared with the process; call srand() if you want a specific seed.

Returns

Random float in [0, 1].

Note

Not thread-safe due to rand().

Definition at line 179 of file Matrix3D.h.

Referenced by mat3D_rand().

4.1.3.10 mat3D_sum()

```
void mat3D_sum (
            Mat3D dst,
            Mat3D a )
```

Element-wise addition: dst += a.

Parameters

dst	Destination matrix; also the left operand.
a	Right operand; must have the same shape as dst.

Precondition

dst.rows == a.rows, dst.cols == a.cols, dst.layers == a.layers.

Definition at line 270 of file Matrix3D.h.

References Mat3D::cols, Mat3D::layers, MAT3D_AT, MATRIX3D_ASSERT, and Mat3D::rows.

Referenced by main().

4.2 Matrix3D.h

```
00001 /* */
00058 #ifndef MATRIX3D_H_
00059 #define MATRIX3D_H_
00060
00061 #include <stddef.h>
00062 #include <stdio.h>
00063 #include <stdlib.h>
00064
00075 #ifndef MATRIX3D_MALLOC
00076 #define MATRIX3D_MALLOC malloc
00077 #endif //MATRIX3D_MALLOC
00078
00087 #ifndef MATRIX3D_ASSERT
00088 #include <assert.h>
00089 #define MATRIX3D_ASSERT assert
00090 #endif //MATRIX3D_ASSERT
00091
00100 typedef struct {
00104     size_t rows;
00108     size_t cols;
00112     size_t layers;
00120     size_t stride;
00124     float *elements;
00125
00126 } Mat3D;
00127
00142 #define MAT3D_AT(m, i, j, k) (m).elements[((i)*(m).cols + (j))*(m).layers + (k)]
00150 #define MAT3D_PRINT(m) mat3D_print(m, #m, 0)
00151
00152 float mat3D_rand_float(void);
00153 Mat3D mat3D_alloc(size_t rows, size_t cols, size_t layers);
00154 void mat3D_free(Mat3D m);
00155 void mat3D_fill(Mat3D m, float x);
00156 void mat3D_rand(Mat3D m, float low, float high);
00157 void mat3D_sum(Mat3D dst, Mat3D a);
00158 void mat3D_mult(Mat3D m, float factor);
00159 void mat3D_print(Mat3D m, const char *name, size_t padding);
00160 void mat3D_identity_mat(Mat3D m);
00161 void mat3D_copy(Mat3D res, Mat3D src);
00162
00163 #endif // MATRIX3D_H_
00164
00165 #ifdef MATRIX3D_IMPLEMENTATION
00166 #undef MATRIX3D_IMPLEMENTATION
00167
00179 float mat3D_rand_float(void)
00180 {
00181     return (float) rand() / (float) RAND_MAX;
00182 }
00183
00199 Mat3D mat3D_alloc(size_t rows, size_t cols, size_t layers)
00200 {
00201     Mat3D m;
00202     m.rows = rows;
00203     m.cols = cols;
00204     m.layers = layers;
00205     m.stride = cols * layers;
```

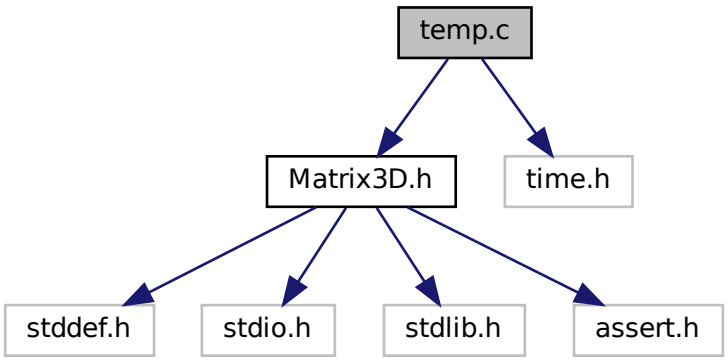
```
00206     m.elements = (float*)MATRIX3D_MALLOC(sizeof(*m.elements)*rows*cols*layers);
00207     MATRIX3D_ASSERT(m.elements != NULL);
00208     return m;
00209 }
00210
00220 void mat3D_free(Mat3D m)
00221 {
00222     free(m.elements);
00223 }
00224
00231 void mat3D_fill(Mat3D m, float x)
00232 {
00233     for (size_t i = 0; i < m.rows; ++i) {
00234         for (size_t j = 0; j < m.cols; ++j) {
00235             for (size_t k = 0; k < m.layers; ++k) {
00236                 MAT3D_AT(m, i, j, k) = x;
00237             }
00238         }
00239     }
00240 }
00241
00251 void mat3D_rand(Mat3D m, float low, float high)
00252 {
00253     for (size_t i = 0; i < m.rows; ++i) {
00254         for (size_t j = 0; j < m.cols; ++j) {
00255             for (size_t k = 0; k < m.layers; ++k) {
00256                 MAT3D_AT(m, i, j, k) = mat3D_rand_float()*(high - low) + low;
00257             }
00258         }
00259     }
00260 }
00261
00270 void mat3D_sum(Mat3D dst, Mat3D a)
00271 {
00272     MATRIX3D_ASSERT(dst.rows == a.rows);
00273     MATRIX3D_ASSERT(dst.cols == a.cols);
00274     MATRIX3D_ASSERT(dst.layers == a.layers);
00275     for (size_t i = 0; i < dst.rows; ++i) {
00276         for (size_t j = 0; j < dst.cols; ++j) {
00277             for (size_t k = 0; k < dst.layers; ++k) {
00278                 MAT3D_AT(dst, i, j, k) += MAT3D_AT(a, i, j, k);
00279             }
00280         }
00281     }
00282 }
00283
00290 void mat3D_mult(Mat3D m, float factor)
00291 {
00292     for (size_t i = 0; i < m.rows; ++i) {
00293         for (size_t j = 0; j < m.cols; ++j) {
00294             for (size_t k = 0; k < m.layers; ++k) {
00295                 MAT3D_AT(m, i, j, k) *= factor;
00296             }
00297         }
00298     }
00299 }
00300
00312 void mat3D_print(Mat3D m, const char *name, size_t padding)
00313 {
00314     printf("%*s%s = [\n", (int) padding, "", name);
00315     for (size_t k = 0; k < m.layers; ++k) {
00316         printf("%*s    k=%zu\n", (int) padding, "", k);
00317         for (size_t i = 0; i < m.rows; ++i) {
00318             printf("%*s    ", (int) padding, "");
00319             for (size_t j = 0; j < m.cols; ++j) {
00320                 printf("%f ", MAT3D_AT(m, i, j, k));
00321             }
00322             printf("\n");
00323         }
00324
00325         printf("%*s    ", (int) padding, "");
00326         for (size_t j = 0; j < m.cols; ++j) {
00327             printf("---------");
00328         }
00329         printf("\n");
00330     }
00331
00332     printf("%*s]\n", (int) padding, "");
00333 }
00334
00345 void mat3D_identity_mat(Mat3D m)
00346 {
00347     MATRIX3D_ASSERT(m.cols == m.rows && m.rows == m.layers);
00348     for (size_t i = 0; i < m.rows; ++i) {
00349         for (size_t j = 0; j < m.cols; ++j) {
00350             for (size_t k = 0; k < m.layers; ++k) {
00351                 if (i == j && j == k) {
```

```
00352                     MAT3D_AT(m, i, j, k) = 1;
00353                 }
00354                 else {
00355                     MAT3D_AT(m, i, j, k) = 0;
00356                 }
00357             }
00358         }
00359     }
00360 }
00361
00370 void mat3D_copy(Mat3D res, Mat3D src)
00371 {
00372     MATRIX3D_ASSERT(res.cols == src.cols);
00373     MATRIX3D_ASSERT(res.rows == src.rows);
00374     MATRIX3D_ASSERT(res.layers == src.layers);
00375
00376     for (size_t i = 0; i < res.rows; ++i) {
00377         for (size_t j = 0; j < res.cols; ++j) {
00378             for (size_t k = 0; k < res.layers; ++k) {
00379                 MAT3D_AT(res, i, j, k) = MAT3D_AT(src, i, j, k);
00380             }
00381         }
00382     }
00383 }
00384
00385 #endif // MATRIX3D_IMPLEMENTATION
```

4.3 temp.c File Reference

```
#include "Matrix3D.h"
#include "time.h"
```

Include dependency graph for temp.c:



Macros

- #define MATRIX3D_IMPLEMENTATION

Functions

- int main (void)

4.3.1 Macro Definition Documentation

4.3.1.1 MATRIX3D_IMPLEMENTATION

```
#define MATRIX3D_IMPLEMENTATION
```

Definition at line 1 of file temp.c.

4.3.2 Function Documentation

4.3.2.1 main()

```
int main (
            void  )
```

Definition at line 5 of file temp.c.

References mat3D_alloc(), MAT3D_AT, mat3D_fill(), mat3D_free(), mat3D_identity_mat(), MAT3D_PRINT, and mat3D_sum().

4.4 temp.c

```
00001 #define MATRIX3D_IMPLEMENTATION
00002 #include "Matrix3D.h"
00003 #include "time.h"
00004
00005 int main(void)
00006 {
00007     Mat3D m1 = mat3D_alloc(3, 3, 3);
00008     Mat3D m2 = mat3D_alloc(3, 3, 3);
00009
00010     mat3D_fill(m1, 0);
00011     mat3D_fill(m2, 1);
00012
00013     MAT3D_AT(m1, 1, 1, 2) = 1;
00014
00015     mat3D_sum(m1, m2);
00016     mat3D_identity_mat(m2);
00017
00018     MAT3D_AT(m2, 2, 2, 0) = 1123;
00019
00020     MAT3D_PRINT(m1);
00021     MAT3D_PRINT(m2);
00022
00023     mat3D_free(m1);
00024     mat3D_free(m2);
00025     return 0;
00026 }
```

Index