mtx — C++ class for 3D matrices

v2.1(β) / 07.03.2022

Mihai-Tiberiu Dima, Maria Dima
Meshcheryakov Lab. Information Tek, JINR, Russia

Header: `#include "mtx.hh"`

Libraries: `libmtx.a` (static) `libmtx.so` (dynamic)

linux-2.6.32-504.12.2.el6.x86_64
gcc-4.4.7 20120313

Variables (public):

- `x11 … x33 = scalar` ..... polymorphic elements

Constructors (public)

- `mtx<double> a` ..... initialisation to zero, or a
- `mtx<int> a(1,0,…)` ..... initialisation to given value
- `auto a = b` ..... initialisation to b (mtx, or scalar)
- `auto a(b)` ..... initialisation to b (mtx, or scalar)

Assign (public)

- `mtx<int> a; a = b` .....initialisation to b (mtx, or scalar)

Cast operators

- `scalar(z)` ..... conversion mtx<scalar'> → scalar
- `mtx<scalar>(z)` ..... conversion mtx<scalar'> → mtx<scalar>

Negative (friend)

- `auto a = -b` .....initialisation to -b (const&, or &&)

Conjugate (friend)

- `auto a = ~b` .....initialisation to b* (const&, or &&)

Algebraic operators (friends)

- `auto a = b+c` .....b, c are (const&, or &&), (mtx, or scalar)
- `auto a = b-c` .....b, c are (const&, or &&), (mtx, or scalar)
- `auto a = b*c` .....b, c are (const&, or &&), (mtx, or scalar)
- `auto a = b/c` .....b, c are (const&, or &&), (mtx, or scalar)
- `auto a += b` .....b is (const&, or &&), (mtx, or scalar)
- `auto a -= b` .....b is (const&, or &&), (mtx, or scalar)
- `auto a *= b` .....b is (const&, or &&), (mtx, or scalar)
- `auto a /= b` .....b is (const&, or &&), (mtx, or scalar)

- `auto a = M|v` .....mtx M applied to vec |v >
- `auto a = v|M` .....mtx M^+ applied to vec < v|
- `auto a = (a|M|b)` ..... mtx element M_{ab}
- `auto a = (P|Q)` ..... Frobenius scalar-prod of mtx P and Q

Functions (friend)

- `fabs(z)` ..... Frobenius norm
- `log(z)` ..... log, minimal phase convention
- `exp(z)` ..... (U)exp(M_{diag})(U^+)
- `eigen(z)` ..... mtx<cpx<scalar>> w/ normed eigen-v's as columns

Print (friend)

- `cout << z << endl << endl;` ..... note 2 endl !
- `cout << boolalpha << z << endl;` ..... print scalar type appended

Usage examples

- `auto z = exp(Lz*PI/4)` ..... PI/4 rotation around $|e_z>$

Description

The ***mtx*** class is a very slim (2 variables, constructors, cast operators) templated C++ class. The huge number of non-class operators (27+1205) are `friend`, saving an extra variable (`this`) in the call, for somewhat higher runtime expediency. A deeper reason is due to templated coding, each operator function needing ca. 7 implementations, in order to accomodate *quasi-polymorphism*.

Quasi-polymorphism means the package mimics polymorphism for the usual scalar types used in science and engineering. Statements such as:

```
auto z = double(1) + mtx<int>(3,0,1,0,2,1,4,1,0) ;
```

benefit of the templated function type-calculator to determine the output type as `mtx<double>`.

The class overloads `fabs` to calculate the norm – and has eigen to output a cpx<scalar> matrix w/ normed eigen-vec's as columns. Log and exp also available.

The class comes with all instantiation combinations for `int, float, double, long double` – and `cpx<int>, cpx<float>, cpx<double>, cpx<long double>`.

The ***makefile*** is banale, however with full pfledged functionality: `make libs`, `make test`, `make run`, `make clean`.

The class comes with 4 examples and 1 application example.