

線形代数ライブラリ

Eigen入門

情報数理研究室B4
根岸徹也

資料

資料

- [https://github.com/TetsuyaNegishi/
Eigen Seminar 2015 11 04](https://github.com/TetsuyaNegishi/Eigen_Seminar_2015_11_04)

スライド有
ソースコードなど上げていくつもり

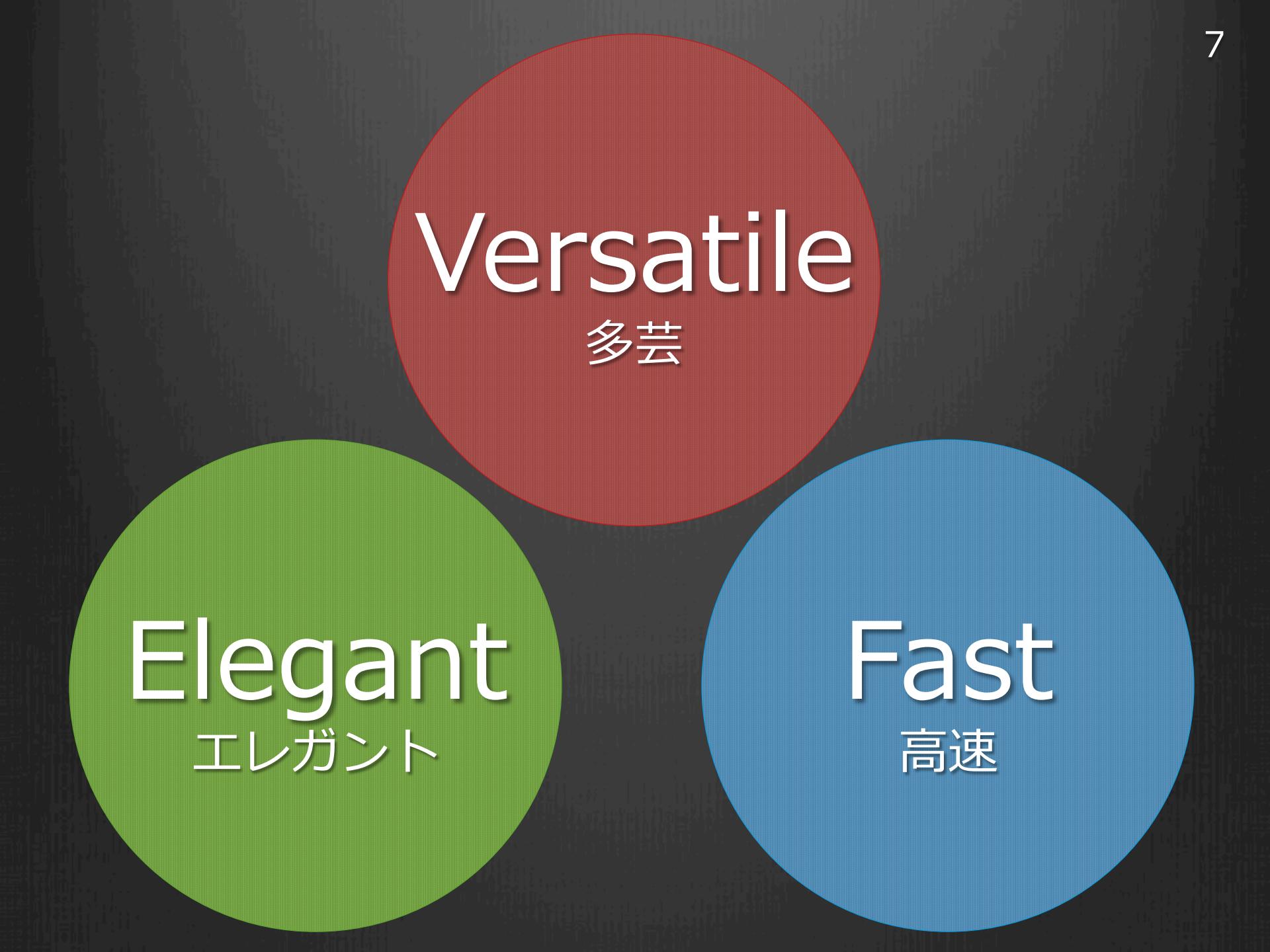
Eigenと(よ)?

Eigenとは？



- 線形代数ライブラリ
- 言語：C++

根観



Versatile

多芸

Elegant

エレガント

Fast

高速

Versatile?

行列サイズ	小規模・大規模
行列成分	密・疎
メモリ確保	動的・静的
線形代数	コレスキー・QR・LU SVD・固有値解法…
幾何学	オイラー角・等距離写像 超平面・投射写像…
その他	フーリエ・多項式ソルバ…

行列サイズ

小規模・大規模

行列成分

密・疎

メモリ確保

動的・静的

すべてナポート

幾何学

オイラー角・等距離写像
超平面・投射写像…

その他

フーリエ・多項式ソルバ…

Fast?



Benchmark

The following benchmark results have been generated using a (heavily) modified version of the Benchmark for Templated Libraries ([BTL](#)) from Laurent Plagne. Our modified version can be found in the mercurial repository under eigen/bench/btl. We did our best to make the best use of each library, however, any hints on making a lib working better are welcome. All libs have been configured to use **dynamic-size column-major matrices** and only **one thread**. [Try it yourself.](#)

Higher is better. By MFLOPS we mean millions of (effective) arithmetic operations per second. The reason why the values are typically low for small sizes, is that in this benchmark we deal with dynamic-size matrices which are relatively inefficient for small sizes. The reason why some libraries/benchmarks show a decline for large sizes, is that for such large matrices issues of CPU cache friendliness become predominant.

Previous benchmarks:

- [August 2008](#): Eigen 2, includes Eigen w/o vectorization, MKL, Goto, Atlas, ublas, mtl4, blitz, and gmm++.
- [March 2009](#): Early version of eigen3, includes Eigen w/o vectorization, MKL, Goto, Atlas, and ACML.

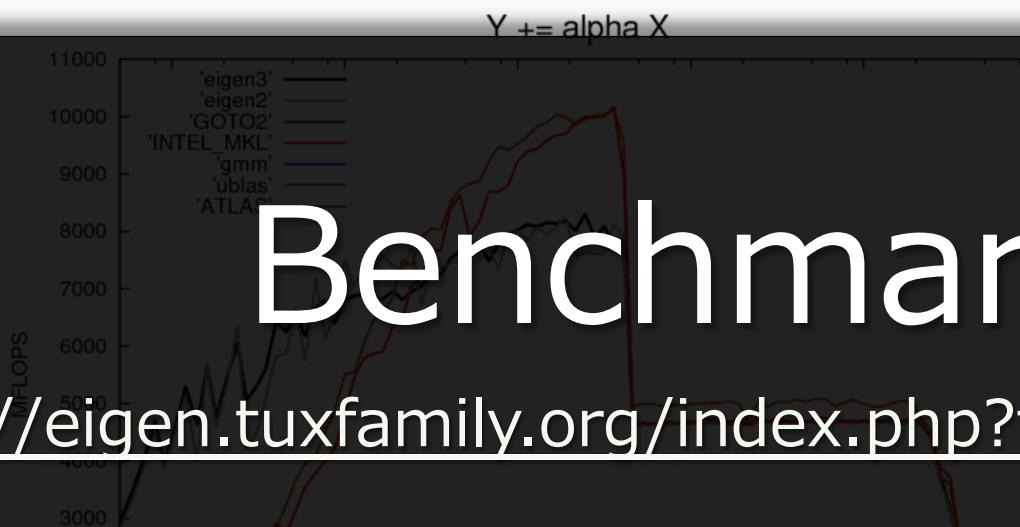
Here is the list of the libraries included in the following benchmarks:

- **eigen3**: ourselves, with the default options (SSE2 vectorization enabled).
- **eigen2**: the previous stable version of Eigen, with the default options (SSE2 vectorization enabled).
- **INTEL_MKL**: The [Intel Math Kernel Library](#), which includes a BLAS/LAPACK (11.0). Closed-source.
- **ACML**: The AMD's core math library, which includes a BLAS/LAPACK (4.2.0). Closed-source.
- **GOTO**: The [GOTO BLAS](#) library (2-1.13). This library have been compiled by hand specifically for the penryn architecture.
- **ATLAS**: The [math-atlas](#) BLAS library (3.8.3). This library has been compiled by hand specifically for the penryn architecture.

23 March 2011

Configuration

- model name : Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz (x86_64)
- compiler: c++ (SUSE Linux) 4.5.0 20100604 [gcc-4_5-branch revision 160292]



Elegant?

Elegant?

```
Matrix3d m // 行列定義  
m << 1, 2, 3, // 要素代入  
    4, 5, 6,  
    7, 8, 9;  
// 標準出力  
std::cout << m << std::endl;
```

Elegant?

```
Matrix3d m, n;  
x = m + n;           // 加算  
y = m - n;           // 減算  
z = m * n;           // 乗算  
m.col(a);            // a行目を取得  
m.col(a).sum();      // a行目の要素合計
```

Elegant?

Matrix3f m, n;

x = m

y = m - n; // 減算

z = n * m; // 乗算

m.col(a); // a行目を取得

m.col(a).sum(); // a行目の要素合計

直感的で

分かりやすい

環境設定



Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

Main page
Forum
Bugs & feature requests
FAQ
Contributing
Benchmark
Publications
Recent changes

Documentation
Eigen 3
Dev branch
Tools



Contents [hide]

- 1 Overview
- 2 Documentation
- 3 Requirements
- 4 License
- 5 Compiler support
- 6 Get support
- 7 Bug reports
- 8 Mailing list
- 9 IRC Channel
- 10 Contributing to Eigen
- 11 Projects using Eigen
- 12 Credits

Announcements

Eigen 3.2.6 released! (01.10.2015) [\[details\]](#)

Eigen 3.3-alpha1 released!

(04.09.2015) [\[details\]](#)

Eigen 3.2.5 released! (16.06.2015) [\[details\]](#)

Eigen 3.2.4 released! (21.01.2015) [\[details\]](#)

Eigen 3.2.3 released! (16.12.2014) [\[details\]](#)

Get it

The latest stable release is Eigen 3.2.6

The latest development release is Eigen Changelog.

The unstable source code from the dev

To check out the Eigen repository using

hg clone <https://bitbucket.org/eigen/eigen>

Looking for the outdated Eigen2 version

[other downloads] [browse the sour

1, 公式webからファイルを

ダウンロードして解凍

2. ファイルをインクルード

```
#include<Eigen/Dense> //密行列用  
#include<Eigen/Sparse> //疎行列用  
using namespace Eigen;
```

※密行列のみ,疎行列のみ使用ならばDense,Sparseのみインクルードで可

※3行目のusing～は必須ではないが以降の例では宣言したものとする

3,解凍したファイルとリンクしてコンパイル

```
$ g++ -I ~/Eigen test.cpp
```

基本的な 使い方

変数宣言

```
3×3      double  
Matrix3d m;  
  
↑  
同じ意味  
Matrix<double, 3, 3>
```

変数宣言

```
Vector3d v0;           // 3次元ベクトル  
Matrix<double,3,1> v1 // v0と同義  
// 疎行列  
SparseMatrix<double> sp(100,100);  
// メモリの動的確保  
MatrixXd mx0;  
Matrix<double,Dynamic,1> v2;
```

入力・出力

```
Matrix2i m;  
// 要素代入(dynamicでない場合のみ)  
m << 1, 2,  
    3, 4;  
m(0, 0) = 3; // 1行1列目に代入  
// 標準出力  
std::cout << m << "\n";
```

初期化

```
MatrixXd m;  
m.setZero(4,4); // 要素がすべて0  
m.setIdentity(3,3); // 単位行列  
m.setOnes(4,5); // 要素がすべて1  
m.setRandom(1,4); // ランダム
```

その他操作

```
m.rows();      // 行数  
m.mean();     // 要素の平均  
// m(3,3)から2×2の行列を取得  
m.block(3,3,2,2);  
// 3×3に行列サイズ変更  
m.resize(3,3);
```

線形方程式 固有値ノリレバ

ソルバの使い方①

- 1,ソルバメソッドを呼ぶ
- 2,問題の解を返すメソッドを呼ぶ

線形方程式の解
m.householderQr().solve(b)
ソルバメソッド
右辺ベクトル

ソルバの使い方②

1, ソルバ変数を作成

2, 問題の解を返すメソッドを呼ぶ

EigenSolver<Matrix3d> eig(A)

eig.eigenvalues(); // 固有値

eig.eigenvectors(); // 固有ベクトル

線形方程式ソルバ

コレスキー

密 : LDLT, LLT
疎 : SimplicialLDLT, ...

LU分解

密 : FullPivLU, ...
疎 : SparseLU

QR分解

密 : HouseholderQR, ...
疎 : SparseQR

SVD

密 : JacobiSVD
疎 : 無し

固有値ソルバ

標準

EigenSolver

一般

GeneralizedEigenSolver

その他

ComplexEigenSolver, …

参考文献

- 公式リファレンス
<http://eigen.tuxfamily.org/dox/>
- Eigen-C++で使える線形代数ライブラリ
<http://blog.livedoor.jp/teknishi/archives/8623876.html>
- 行列ライブラリEigenメモ
<http://mikaka.org/~kana/dl/pdf/pdf-eigennote.pdf>