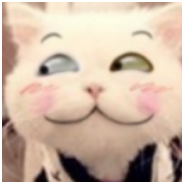


shuzfan的专栏

目录视图

个人资料



shuzfan

关注 发私信



访问：131616次
积分：2279
等级： **BLOG** > 5
排名：第14344名

原创：67篇 转载：7篇
译文：1篇 评论：265条

文章搜索

文章分类

- 计算机视觉 (2)
- MATLAB (3)
- 程序、数据资料 (12)
- 深度学习基础 (10)
- caffe (11)
- 人脸识别 (3)
- 神经网络压缩与加速 (7)
- 人脸检测(目标检测) (15)
- 人脸关键点检测 (1)
- 特征选择 (1)
- C++ (7)
- 排序算法 (7)

文章存档

- 2017年03月 (2)
- 2017年02月 (8)
- 2017年01月 (5)
- 2016年12月 (2)
- 2016年11月 (7)

展开

Eigen矩阵运算库使用记录

2016-08-30 10:42 1876人阅读 评论

分类： 程序、数据资料 (11)

版权声明：本文为博主原创文章，转载请注明出处

最近一直在做工程上的事情，比较多的使用了Eigen矩阵运算库。

简单说一下Eigen的特点：

- (1) 使用方便、无需预编译，调用开销小
- (2) 函数丰富，风格有点近似MATLAB，易上手；
- (3) 速度中规中矩，比OpenCV快，比MKL、openBLAS慢；

Eigen3.3版本链接 http://eigen.tuxfamily.org/index.php?title=Main_Page

注：绝大部分使用说明和示例都可以在官网上找到，所以有时候不需要纠结百度到的与实际不符，可以直接官网or谷歌

使用方法很简单：下载Eigen后解压，然后包含解压路径，最后只需要在程序里引用头文件

```
1 | #include <Eigen/Dense>
```

基本使用方法如下：

原址链接为<http://eigen.tuxfamily.org/dox/AsciiQuickReference.txt>

矩阵定义

```
1 | #include <Eigen/Dense>
2 |
3 | Matrix<double, 3, 3> A;           // Fixed rows and cols. Same as Matrix3d.
4 | Matrix<double, 3, Dynamic> B;     // Fixed rows, dynamic cols.
5 | Matrix<double, Dynamic, Dynamic> C; // Full dynamic. Same as MatrixXd.
6 | Matrix<double, 3, 3, RowMajor> E; // Row major; default is column-major.
7 | Matrix3f P, Q, R;                // 3x3 float matrix.
8 | Vector3f x, y, z;                 // 3x1 float matrix.
9 | RowVector3f a, b, c;              // 1x3 float matrix.
10 | VectorXd v;                       // Dynamic column vector of doubles
```

基本使用方法

```
1 | // Basic usage
2 | // Eigen           // Matlab           // comments
```

阅读排行		3	x.size()	// length(x)	// vector size
GoogLeNet系列解读		4	C.rows()	// size(C,1)	// number of rows
mxnet学习记录【1】		5	C.cols()	// size(C,2)	// number of columns
深度学习——Xavier初始化方法		6	x(i)	// x(i+1)	// Matlab is 1-based
caffe添加新层教程		7	C(i,j)	// C(i+1,j+1)	//
		8			
caffe层解读系列-softmax loss		9	A.resize(4, 4);	// Runtime error if assertions are on.	
人脸检测——DDFD		10	B.resize(4, 9);	// Runtime error if assertions are on.	
人脸检测——MTCNN		11	A.resize(3, 3);	// Ok; size didn't change.	
解读Batch Normalization		12	B.resize(3, 9);	// Ok; only dynamic cols changed.	
		13			
NDK各个版本链接		14	A << 1, 2, 3,	// Initialize A. The elements can also be	
VS编译生成MATLAB接口程序		15	4, 5, 6,	// matrices, which are stacked along cols	
		16	7, 8, 9;	// and then the rows are stacked.	
		17	B << A, A, A;	// B is three horizontally stacked A's.	
		18	A.fill(10);	// Fill A with all 10's.	
评论排行		特殊矩阵生成			
人脸检测——DDFD					
GoogLeNet系列解读		1	// Eigen	// Matlab	
mxnet学习记录【1】		2	MatrixXd::Identity(rows,cols)	// eye(rows,cols)	
caffe添加新层教程		3	C.setIdentity(rows,cols)	// C = eye(rows,cols)	
神经网络压缩：Deep Compr...		4	MatrixXd::Zero(rows,cols)	// zeros(rows,cols)	
Dlib + VS2013 人脸检测，无...		5	C.setZero(rows,cols)	// C = ones(rows,cols)	
人脸检测——MTCNN		6	MatrixXd::Ones(rows,cols)	// ones(rows,cols)	
caffe层解读系列-softmax loss		7	C.setOnes(rows,cols)	// C = ones(rows,cols)	
NMS——卷积网络改进实现		8	MatrixXd::Random(rows,cols)	// rand(rows,cols)*2-1	// Multivariate random
剪枝+再训练：稀疏化DeepID2		9	C.setRandom(rows,cols)	// C = rand(rows,cols)*2-1	
		10	VectorXd::LinSpaced(size,low,high)	// linspace(low,high,size)'	
		11	v.setLinSpaced(size,low,high)	// v = linspace(low,high,size)'	
最新评论		矩阵块操作			
快速多目标检测——YOLO9000					
shuzfan : @as_far_as:代码里面用了anchor		1	// Matrix slicing and blocks. All expressions listed here are read/write.		
人脸检测——MTCNN		2	// Templated size versions are faster. Note that Matlab is 1-based (a size N		
shuzfan : @LIXIAOLONGJIEQUAN:还没有，你可以参考另外一个同学用MXNET实现的，有训练过程		3	// vector is x(1)...x(N)).		
剪枝+再训练：稀疏化DeepID2		4	// Eigen	// Matlab	
shuzfan : @supe_king:置0的位置永远保持为0，其余位置还是可以继续学习的。如果不retrain，那么...		5	x.head(n)	// x(1:n)	
NMS——卷积网络改进实现		6	x.head<n>()	// x(1:n)	
shuzfan : @u011879633:当时是搜索ICLR的录用结果，没有发现这篇文章。更多的改进NMS的论文我也...		7	x.tail(n)	// x(end - n + 1: end)	
GoogLeNet系列解读		8	x.tail<n>()	// x(end - n + 1: end)	
shuzfan : @jiachen0212:都可以，比如我训练模型都喜欢训练24W、36W、48W次之类的，看着舒服T...		9	x.segment(i, n)	// x(i+1 : i+n)	
caffe添加新层教程		10	x.segment<n>(i)	// x(i+1 : i+n)	
shuzfan : @zj15939317693:可以直接调用，你可以参考下image_data_layer.cpp，里...		11	P.block(i, j, rows, cols)	// P(i+1 : i+rows, j+1 : j+cols)	
caffe添加新层教程		12	P.block<rows, cols>(i, j)	// P(i+1 : i+rows, j+1 : j+cols)	
Lonely20170107 : 您好，我想问一下opencv里的函数能在caffe中调用么？caffe里不是有opencv么？我有看...		13	P.row(i)	// P(i+1, :)	
caffe添加新层教程		14	P.col(j)	// P(:, j+1)	
Lonely20170107 : 您好，我想问一下opencv里的函数能在caffe中调用么？caffe里不是有opencv么？我有看...		15	P.leftCols<cols>()	// P(:, 1:cols)	
人脸检测——DDFD		16	P.leftCols(cols)	// P(:, 1:cols)	
hzd12368 : 博主，能不能发给我一份AFLW的原始数据库，去官网申请了很长时间都没申请到，谢谢。296756698...		17	P.middleCols<cols>(j)	// P(:, j+1:j+cols)	
GoogLeNet系列解读		18	P.middleCols(j, cols)	// P(:, j+1:j+cols)	
jiachen0212 : 博主你好，写得很好，入门特别好。有一个疑问是，Inception Module中的过度特征图个数32...		19	P.rightCols<cols>()	// P(:, end-cols+1:end)	
		20	P.rightCols(cols)	// P(:, end-cols+1:end)	
		21	P.topRows<rows>()	// P(1:rows, :)	
		22	P.topRows(rows)	// P(1:rows, :)	
		23	P.middleRows<rows>(i)	// P(i+1:i+rows, :)	
		24	P.middleRows(i, rows)	// P(i+1:i+rows, :)	
		25	P.bottomRows<rows>()	// P(end-rows+1:end, :)	
		26	P.bottomRows(rows)	// P(end-rows+1:end, :)	
		27	P.topLeftCorner(rows, cols)	// P(1:rows, 1:cols)	
		28	P.topRightCorner(rows, cols)	// P(1:rows, end-cols+1:end)	
		29	P.bottomLeftCorner(rows, cols)	// P(end-rows+1:end, 1:cols)	
		30	P.bottomRightCorner(rows, cols)	// P(end-rows+1:end, end-cols+1:end)	
		31	P.topLeftCorner<rows,cols>()	// P(1:rows, 1:cols)	
		32	P.topRightCorner<rows,cols>()	// P(1:rows, end-cols+1:end)	
		33	P.bottomLeftCorner<rows,cols>()	// P(end-rows+1:end, 1:cols)	
		34	P.bottomRightCorner<rows,cols>()	// P(end-rows+1:end, end-cols+1:end)	

快速多目标检测——YOLO9000		特殊矩阵生成			
shuzfan : @as_far_as:代码里面用了anchor					
人脸检测——MTCNN					
shuzfan : @LIXIAOLONGJIEQUAN:还没有，你可以参考另外一个同学用MXNET实现的，有训练过程		1	// Eigen	// Matlab	
剪枝+再训练：稀疏化DeepID2		2	MatrixXd::Identity(rows,cols)	// eye(rows,cols)	
shuzfan : @supe_king:置0的位置永远保持为0，其余位置还是可以继续学习的。如果不retrain，那么...		3	C.setIdentity(rows,cols)	// C = eye(rows,cols)	
NMS——卷积网络改进实现		4	MatrixXd::Zero(rows,cols)	// zeros(rows,cols)	
shuzfan : @u011879633:当时是搜索ICLR的录用结果，没有发现这篇文章。更多的改进NMS的论文我也...		5	C.setZero(rows,cols)	// C = ones(rows,cols)	
GoogLeNet系列解读		6	MatrixXd::Ones(rows,cols)	// ones(rows,cols)	
shuzfan : @jiachen0212:都可以，比如我训练模型都喜欢训练24W、36W、48W次之类的，看着舒服T...		7	C.setOnes(rows,cols)	// C = ones(rows,cols)	
caffe添加新层教程		8	MatrixXd::Random(rows,cols)	// rand(rows,cols)*2-1	// Multivariate random
shuzfan : @zj15939317693:可以直接调用，你可以参考下image_data_layer.cpp，里...		9	C.setRandom(rows,cols)	// C = rand(rows,cols)*2-1	
caffe添加新层教程		10	VectorXd::LinSpaced(size,low,high)	// linspace(low,high,size)'	
Lonely20170107 : 您好，我想问一下opencv里的函数能在caffe中调用么？caffe里不是有opencv么？我有看...		11	v.setLinSpaced(size,low,high)	// v = linspace(low,high,size)'	
caffe添加新层教程		矩阵块操作			
Lonely20170107 : 您好，我想问一下opencv里的函数能在caffe中调用么？caffe里不是有opencv么？我有看...					
人脸检测——DDFD		1	// Matrix slicing and blocks. All expressions listed here are read/write.		
hzd12368 : 博主，能不能发给我一份AFLW的原始数据库，去官网申请了很长时间都没申请到，谢谢。296756698...		2	// Templated size versions are faster. Note that Matlab is 1-based (a size N		
GoogLeNet系列解读		3	// vector is x(1)...x(N)).		
jiachen0212 : 博主你好，写得很好，入门特别好。有一个疑问是，Inception Module中的过度特征图个数32...		4	// Eigen	// Matlab	
		5	x.head(n)	// x(1:n)	
		6	x.head<n>()	// x(1:n)	
		7	x.tail(n)	// x(end - n + 1: end)	
		8	x.tail<n>()	// x(end - n + 1: end)	
		9	x.segment(i, n)	// x(i+1 : i+n)	
		10	x.segment<n>(i)	// x(i+1 : i+n)	
		11	P.block(i, j, rows, cols)	// P(i+1 : i+rows, j+1 : j+cols)	
		12	P.block<rows, cols>(i, j)	// P(i+1 : i+rows, j+1 : j+cols)	
		13	P.row(i)	// P(i+1, :)	
		14	P.col(j)	// P(:, j+1)	
		15	P.leftCols<cols>()	// P(:, 1:cols)	
		16	P.leftCols(cols)	// P(:, 1:cols)	
		17	P.middleCols<cols>(j)	// P(:, j+1:j+cols)	
		18	P.middleCols(j, cols)	// P(:, j+1:j+cols)	
		19	P.rightCols<cols>()	// P(:, end-cols+1:end)	
		20	P.rightCols(cols)	// P(:, end-cols+1:end)	
		21	P.topRows<rows>()	// P(1:rows, :)	
		22	P.topRows(rows)	// P(1:rows, :)	
		23	P.middleRows<rows>(i)	// P(i+1:i+rows, :)	
		24	P.middleRows(i, rows)	// P(i+1:i+rows, :)	
		25	P.bottomRows<rows>()	// P(end-rows+1:end, :)	
		26	P.bottomRows(rows)	// P(end-rows+1:end, :)	
		27	P.topLeftCorner(rows, cols)	// P(1:rows, 1:cols)	
		28	P.topRightCorner(rows, cols)	// P(1:rows, end-cols+1:end)	
		29	P.bottomLeftCorner(rows, cols)	// P(end-rows+1:end, 1:cols)	
		30	P.bottomRightCorner(rows, cols)	// P(end-rows+1:end, end-cols+1:end)	
		31	P.topLeftCorner<rows,cols>()	// P(1:rows, 1:cols)	
		32	P.topRightCorner<rows,cols>()	// P(1:rows, end-cols+1:end)	
		33	P.bottomLeftCorner<rows,cols>()	// P(end-rows+1:end, 1:cols)	
		34	P.bottomRightCorner<rows,cols>()	// P(end-rows+1:end, end-cols+1:end)	
		矩阵元素交换以及转置等			
		1	// Of particular note is Eigen's swap function which is highly optimized.		
		2	// Eigen	// Matlab	
		3	R.row(i) = P.col(j);	// R(i, :) = P(:, i)	
		4	R.col(j1).swap(mat1.col(j2));	// R(:, [j1 j2]) = R(:, [j2, j1])	
		5			

```

6 // Views, transpose, etc; all read-write except for .adjoint().
7 // Eigen // Matlab
8 R.adjoint() // R'
9 R.transpose() // R.' or conj(R')
10 R.diagonal() // diag(R)
11 x.asDiagonal() // diag(x)
12 R.transpose().colwise().reverse(); // rot90(R)
13 R.conjugate() // conj(R)

```

矩阵四则运算

```

1 // All the same as Matlab, but matlab doesn't have *= style operators.
2 // Matrix-vector. Matrix-matrix. Matrix-scalar.
3 y = M*x; R = P*Q; R = P*s;
4 a = b*M; R = P - Q; R = s*P;
5 a *= M; R = P + Q; R = P/s;
6 R *= Q; R = s*P;
7 R += Q; R *= s;
8 R -= Q; R /= s;

```

单个元素操作

```

1 // Vectorized operations on each element independently
2 // Eigen // Matlab
3 R = P.cwiseProduct(Q); // R = P .* Q
4 R = P.array() * s.array(); // R = P .* s
5 R = P.cwiseQuotient(Q); // R = P ./ Q
6 R = P.array() / Q.array(); // R = P ./ Q
7 R = P.array() + s.array(); // R = P + s
8 R = P.array() - s.array(); // R = P - s
9 R.array() += s; // R = R + s
10 R.array() -= s; // R = R - s
11 R.array() < Q.array(); // R < Q
12 R.array() <= Q.array(); // R <= Q
13 R.cwiseInverse(); // 1 ./ P
14 R.array().inverse(); // 1 ./ P
15 R.array().sin() // sin(P)
16 R.array().cos() // cos(P)
17 R.array().pow(s) // P.^ s
18 R.array().square() // P.^ 2
19 R.array().cube() // P.^ 3
20 R.cwiseSqrt() // sqrt(P)
21 R.array().sqrt() // sqrt(P)
22 R.array().exp() // exp(P)
23 R.array().log() // log(P)
24 R.cwiseMax(P) // max(R, P)
25 R.array().max(P.array()) // max(R, P)
26 R.cwiseMin(P) // min(R, P)
27 R.array().min(P.array()) // min(R, P)
28 R.cwiseAbs() // abs(P)
29 R.array().abs() // abs(P)
30 R.cwiseAbs2() // abs(P.^2)
31 R.array().abs2() // abs(P.^2)
32 (R.array() < s).select(P,Q); // (R < s ? P : Q)

```

矩阵缩减

```

1 // Reductions.
2 int r, c;
3 // Eigen // Matlab
4 R.minCoeff() // min(R(:))
5 R.maxCoeff() // max(R(:))
6 s = R.minCoeff(&r, &c) // [s, i] = min(R(:)); [r, c] = ind2sub(size(R), i);
7 s = R.maxCoeff(&r, &c) // [s, i] = max(R(:)); [r, c] = ind2sub(size(R), i);
8 R.sum() // sum(R(:))
9 R.colwise().sum() // sum(R)
10 R.rowwise().sum() // sum(R, 2) or sum(R')'
11 R.prod() // prod(R(:))
12 R.colwise().prod() // prod(R)
13 R.rowwise().prod() // prod(R, 2) or prod(R')'
14 R.trace() // trace(R)
15 R.all() // all(R(:))
16 R.colwise().all() // all(R)
17 R.rowwise().all() // all(R, 2)
18 R.any() // any(R(:))

```

```

19 R.colwise().any()           // any(R)
20 R.rowwise().any()           // any(R, 2)

```

矩阵点乘及归一化

```

1 // Dot products, norms, etc.
2 // Eigen                      // Matlab
3 x.norm()                      // norm(x). Note that norm(R) doesn't work in Eigen.
4 x.squaredNorm()               // dot(x, x) Note the equivalence is not true for complex
5 x.dot(y)                      // dot(x, y)
6 x.cross(y)                    // cross(x, y) Requires #include <Eigen/Geometry>

```

矩阵类型转换

```

1 //// Type conversion
2 // Eigen                      // Matlab
3 A.cast<double>();              // double(A)
4 A.cast<float>();              // single(A)
5 A.cast<int>();                // int32(A)
6 A.real();                     // real(A)
7 A.imag();                     // imag(A)
8 // if the original type equals destination type, no work is done
9
10 // Note that for most operations Eigen requires all operands to have the same type:
11 MatrixXf F = MatrixXf::Zero(3,3);
12 A += F;                       // illegal in Eigen. In Matlab A = A+F is allowed
13 A += F.cast<double>();        // F converted to double and then added (generally, conversion

```

内存映射创建矩阵

```

1 // Eigen can map existing memory into Eigen matrices.
2 float array[3];
3 Vector3f::Map(array).fill(10); // create a temporary Map over array and set
4 int data[4] = {1, 2, 3, 4};
5 Matrix2i mat2x2(data);          // copies data into mat2x2
6 Matrix2i::Map(data) = 2*mat2x2; // overwrite elements of data with 2*mat2x2
7 MatrixXi::Map(data, 2, 2) += mat2x2; // adds mat2x2 to elements of data (alternat

```

解方程

```

1 // Solve Ax = b. Result stored in x. Matlab: x = A \ b.
2 x = A.ldlt().solve(b); // A sym. p.s.d. #include <Eigen/Cholesky>
3 x = A.llt().solve(b); // A sym. p.d. #include <Eigen/Cholesky>
4 x = A.lu().solve(b); // Stable and fast. #include <Eigen/LU>
5 x = A.qr().solve(b); // No pivoting. #include <Eigen/QR>
6 x = A.svd().solve(b); // Stable, slowest. #include <Eigen/SVD>
7 // .ldlt() -> .matrixL() and .matrixD()
8 // .llt() -> .matrixL()
9 // .lu() -> .matrixL() and .matrixU()
10 // .qr() -> .matrixQ() and .matrixR()
11 // .svd() -> .matrixU(), .singularValues(), and .matrixV()

```

特征值

```

1 // Eigenvalue problems
2 // Eigen                      // Matlab
3 A.eigenvalues();              // eig(A);
4 EigenSolver<Matrix3d> eig(A); // [vec val] = eig(A)
5 eig.eigenvalues();            // diag(val)
6 eig.eigenvectors();           // vec
7 // For self-adjoint matrices use SelfAdjointEigenSolver<>

```

下面是我实际遇到的补充一下：

求广义逆矩阵

```

1 //Eigen中并没有求广义逆的函数，这里用SVD实现，数据类型大家可以按需修改为double
2 using Eigen::Dynamic;
3 using Eigen::Matrix;
4 using Eigen::RowMajor;
5 typedef Matrix<float, Dynamic, Dynamic, RowMajor> MatXf;
6

```

```
7 | MatXf pinv(MatXf x)
8 | {
9 |     JacobiSVD<MatXf> svd(x,ComputeFullU | ComputeFullV);
10 |     float pinvtoler=1.e-8; //tolerance
11 |     MatXf singularValues_inv = svd.singularValues();
12 |     for ( long i=0; i<x.cols(); ++i) {
13 |         if ( singularValues_inv(i) > pinvtoler )
14 |             singularValues_inv(i)=1.0/singularValues_inv(i);
15 |         else singularValues_inv(i)=0;
16 |     }
17 |     return svd.matrixV()*singularValues_inv.asDiagonal()*svd.matrixU().transpose();
18 | }
```

常用的和MATLAB类似的函数实现如下：

原址链接<http://igl.ethz.ch/projects/libigl/matlab-to-eigen.html>

MATLAB	Eigen
[Y, IX] = sort(Y, dim, mode)	igl::sort(X, dim, mode, Y, IX)
B(i:(i+w), j:(j+h)) = A(x:(x+w), y:(y+h))	B.block(i, j, w, h) = A.block(i, j, w, h)
max(A(:))	A.maxCoeff()
min(A(:))	A.minCoeff()
eye(w, h)	MatrixXd::Identity(w, h), MatrixXf::Identity(w, h), etc.
A(i:(i+w), j:(j+h)) = eye(w, h)	A.setIdentity()
[I, J, V] = find(X)	igl::find(X, I, J, V)
X(:, j) = X(:, j) + x	X.col(j).array() += x
Acol_sum = sum(A, 1)	Acol_sum = A.colwise().sum()
Arow_sum = sum(A, 2)	Arow_sum = A.rowwise().sum()
Adim_sum = sum(Asparse, dim)	igl::sum(Asparse, dim, Adim_sum)
D = diag(M)	igl::diag(M, D)
M = diag(D)	igl::diag(D, M)
[Y, I] = max(X, [], dim)	igl::mat_max(X, dim, Y, I)
Y = max(X, [], 1)	Y = X.colwise().maxCoeff()
Y = max(X, [], 2)	Y = X.rowwise().maxCoeff()
Y = min(X, [], 1)	Y = X.colwise().minCoeff()
Y = min(X, [], 2)	Y = X.rowwise().minCoeff()
C = A.*B	C = (A.array() * B.array()).matrix()
C = A.^b	C = A.array().pow(b).matrix()
A(B == 0) = C(B==0)	A = (B.array() == 0).select(C, A)
C = A + B'	SparseMatrixType BT = B.transpose() SparseMatrixType C = A+BT;
[L, p] = chol(A)	SparseLLT<SparseMatrixType> A_LLT(A.template triangularView<Lower>()) SparseMatrixType L = A_LLT.matrixL(); bool p = (L==0).eval().nonZeros()==0;
X = U\ (L\b)	X = b; L.template triangularView<Lower>().solveInPlace(X); U.template triangularView<Upper>().solveInPlace(X);
B = repmat(A, i, j)	igl::repmat(A, i, j, B)
I = low:step:hi	igl::colon(low, step, hi, I)
O = ones(m, n)	Matrix* O = Matrix*::Ones(m, n)
O = zeros(m, n)	Matrix* O = Matrix*::Zero(m, n)
B = A(I, J)	igl::slice(A, I, J, B)
B = A(I, :)	B = igl::slice(A, I, igl::colon(0, A.cols()-1))
B(I, J) = A	igl::slice_into(A, I, J, B)
B(I, :) = A	B = igl::slice_into(A, I, igl::colon(0, B.cols()-1))
M = mode(X, dim)	igl::mode(X, dim, M)
B = arrayfun(FUN, A)	B = A.unaryExpr(ptr_fun(FUN))
B = fliplr(A)	B = A.rowwise().reverse().eval()
B = flipud(A)	B = A.colwise().reverse().eval()
B = IM(A)	B = A.unaryExpr(bind1st(mem_fun(static_cast<VectorXi::Scalar>(&VectorXi::operator())), &IM)).eval());
A = sparse(I, J, V)	// build std::vector<Eigen::Triplet> IJV A.setFromTriplets(IJV);
A = min(A, c)	C.array() = A.array().min(c);
IP = I (P==0)	VectorXi IP = I; IP.conservativeResize(stable_partition(IP.data(), IP.size(), [&P](int i){return P(i)==0;}-IP.data()));
a = any(A(:))	bool a = any_of(A.data(), A.data()+A.size(), [](bool a){ return a;});