LU 분해 구현 및 분석

구현 목록

scipy.linalg.lu	DoolittleLU	CroutLU
LU	LUPP	LUCP
LDLT	Cholesky	

시간 분석

테스트 환경

Python	3.7.4
scipy	1.3.1
numpy	1.17.1
Architecture	amd64
CPU	Intel i5-4570 3.20GHz
RAM	12GB
OS	Windows 10 Home 1903

$$\text{평균 실행 시간} = \frac{\text{1,000회 실행 시간}}{1,000}\ (\mu s)$$

PASS 조건:

$A = \{a_{i,j}\}$

$B = \{b_{i,j}\} = LU = PLU = PLUQ = LDL^T = LL^T$

$\forall (i,j), \quad |a_{i,j} - b_{i,j}| < 10^{-8}$

테스트 결과

	Low rank Vandermonde	Non-square(2, 3)	Non-square(3, 2)
scipy.linalg.lu	34	28	32
DoolittleLU	36	20	26
CroutLU	35	27	21
LU	18	10	16
LUPP	48	37	45
LUCP	61	52	56

	Single element	Symmetric	Positive definite
scipy.linalg.lu	8	32	33
DoolittleLU	4	37	38
CroutLU	4	36	37
LU	5	18	18
LUPP	22	45	49
LUCP	33	61	61
LDLT	11	25	25
Cholesky	.	.	19

	Bad condition of naive LU	Bad condition of LUPP	Big(50, 50)
scipy.linalg.lu	33	33	5559
DoolittleLU	FAIL	37	37020
CroutLU	FAIL	38	36227
LU	FAIL	20	49356
LUPP	47	FAIL	49310
LUCP	64	61	49550

	High rank(16) Vandermonde	High rank(17) Vandermonde	Singular
scipy.linalg.lu	FAIL	FAIL	37
DoolittleLU	1264	FAIL	FAIL
CroutLU	1259	FAIL	FAIL
LU	FAIL	FAIL	FAIL
LUPP	FAIL	FAIL	FAIL
LUCP	FAIL	FAIL	FAIL

결과 분석

- naive LU가 기본적으로 가장 빠르다.
- scipy.linalg의 lu는 singular 행렬도 LU 분해에 성공한다.
- 행렬이 커지면 scipy.linalg의 lu는 다른 구현보다 빠르다.
    - scipy의 LU 분해는 Sivan Toledo's recursive LU의 iterative 구현을 사용한 것으로 추측된다. (M, N) 행렬을 Partial Pivoting을 적용하여 LU 분해를 한다. LAPACK라이브러리의 일부로 포트란으로 구현되어있다.

```
if check_finite:
    a1 = asarray_chkfinite(a)
else:
    a1 = asarray(a)
if len(a1.shape) != 2:
    raise ValueError('expected matrix')
overwrite_a = overwrite_a or (_datacopied(a1, a))
flu, = get_flinalg_funcs(('lu',), (a1,))
p, l, u, info = flu(a1, permute_l=permute_l, overwrite_a=overwrite_a)
if info < 0:
    raise ValueError('illegal value in %d-th argument of '
                                        'internal lu.getrf' % -info)
if permute_l:
    return l, u
return p, l, u
```

DGETRF

DGETRF VARIANT: iterative version of Sivan Toledo's recursive LU algorithm

DGETRF VARIANT: left-looking Level 3 BLAS version of the algorithm.

Download DGETRF + dependencies [TGZ] [ZIP] [TXT]

Purpose:

```
DGETRF computes an LU factorization of a general M-by-N matrix A
using partial pivoting with row interchanges.

The factorization has the form
   A = P * L * U
where P is a permutation matrix, L is lower triangular with unit
diagonal elements (lower trapezoidal if m > n), and U is upper
triangular (upper trapezoidal if m < n).

This is the right-looking Level 3 BLAS version of the algorithm.
```

- 큰 행렬에서 naive LU 보다 doolittle, crout LU가 조금 더 빠른 것으로 나타난다.
    - 왜 더 빠른지 모르겠다.
- non-square 행렬에서는 doolittle과 crout의 구조적 차이로 행이 적을 때는 doolittle이, 열이 적을 때는 crout가 빠르다
- 놀랍게도 doolittle LU와 crout LU는 16 rank의 방데르몽드 행렬 테스트를 통과했다.
- 행렬이 positive definite 행렬이면 Cholesky 분해가 naive LU만큼 빠르다. 정확성에 대해서는 큰 positive definite 행렬을 구하지 못해서 테스트 하지 못했다.