Verkade: LU-Factorisation on Sparse Matrices

Micky Faas

October 10, 2018

1 Introduction

Verkade is an implementation of LU-factorisation on sparse matrices using the simple Gaussian elimination method with partial pivotting. Partially pivotting lends itself very well for sparse matrices as only row or column permutations are required but not both. Therefore the matrix data can be stored either by row or by column. Verkade stores matrices by row in *compressed row storage* (CRS)[1].

To load matrix data, the *matrix market* format[2] is used. The various test matrices described in this document are obtained from Suite Sparse[3].

2 Implementation

2.1 Memory Management

References

[1] Compressed Row Storage. `http://www.netlib.org/linalg/html_templates/node91.html`

[2] The Matrix Market fileformat. `https://math.nist.gov/MatrixMarket/formats.html`

[3] SuiteSparse, Tim Davis et al. `https://sparse.tamu.edu/`

Matrix	Original size (KiB)	Size after LUP (KiB)	Bytes lost (KiB)
mcfe	190	684	27
c-21	251	38405	3582
flowmeter5	526	9241	1010
epb1	742	24062	3188
meg4	365	5830	204
cell1	272	3707	18
nopoly	553	120446	4666
mhd4800b	215	261	5
ex10	428	1421	27
aft01	980	13385	726

Table 1: Memory footprint and loss due to fragmentation.

Matrix	Elements	Density	Elements after	Density after	Fill-in ratio
mcfe	24382	.042	87396	.15	3.58
c-21	32157	.0026	4915965	.4	152.87
flowmeter5	67391	.0007	1182851	.013	17.56
epb1	95053	.00044	3079993	.014	32.40
meg4	46842	.0014	746284	.022	15.93
cell1	34855	.0007	474506	.0095	13.61
nopoly	70842	.00061	15417152	.13	217.63
mhd4800b	27520	.0012	33490	.0015	1.22
ex10	54840	.0094	183234	.032	3.41
aft01	125567	.0019	1713317	.025	13.64

Table 2: