Complexity of Singular Value Decomposition (SVD)

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `fullmatrix` format

Operation : SVD of M

	Storage	Time (Seconds)
$n = 256$	$\frac{1}{2}$ MB	

Complexity of Singular Value Decomposition (SVD)

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `fullmatrix` format

Operation : SVD of M

	Storage	Time (Seconds)
$n = 256$	$\frac{1}{2}$ MB	0.0
$n = 512$	2 MB	

Complexity of Singular Value Decomposition (SVD)

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `fullmatrix` format

Operation : SVD of M

	Storage	Time (Seconds)
$n = 256$	$\frac{1}{2}$ MB	0.0
$n = 512$	2 MB	0.3
$n = 1024$	8 MB	

Complexity of Singular Value Decomposition (SVD)

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `fullmatrix` format

Operation : SVD of M

	Storage	Time (Seconds)
$n = 256$	$\frac{1}{2}$ MB	0.0
$n = 512$	2 MB	0.3
$n = 1024$	8 MB	3.3
$n = 2048$	32 MB	

Complexity of Singular Value Decomposition (SVD)

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `fullmatrix` format

Operation : SVD of M

	Storage	Time (Seconds)
$n = 256$	$\frac{1}{2}$ MB	0.0
$n = 512$	2 MB	0.3
$n = 1024$	8 MB	3.3
$n = 2048$	32 MB	40.0
$n = 4096$	128 MB	354.0
$n = 8192$	512 MB	$\sim$ 48 m
$n = 16384$	2 GB	$\sim$ 7 h
$n = 32768$	8 GB	$\sim 2\frac{1}{2}$ days
$n = 65536$	32 GB	$\sim$ 20 days
$n = 131072$	128 GB	$\sim$ 5 months
$n = 262144$	512 GB	$\sim$ 4 years

Complexity of cSVD in `rkmatrix` format

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `rkmatrix` format,
$n = 262144$, rank k

Operation : SVD of M

$n = 262144$	Storage	Time (Seconds)
$k = 4$	16 MB	

COMPLEXITY OF CSVD IN `rkmatrix` FORMAT

INPUT : Matrix $M \in \mathbb{R}^{n \times n}$ in `rkmatrix` format,
$n = 262144$, rank k

OPERATION : SVD of M

$n = 262144$	Storage	Time (Seconds)
$k = 4$	16 MB	0.08
$k = 8$	32 MB	

Complexity of cSVD in `rkmatrix` format

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `rkmatrix` format,
$n = 262144$, rank k

Operation : SVD of M

$n = 262144$	Storage	Time (Seconds)
$k = 4$	16 MB	0.08
$k = 8$	32 MB	0.21
$k = 16$	64 MB	

Complexity of cSVD in `rkmatrix` format

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `rkmatrix` format,
$n = 262144$, rank k

Operation : SVD of M

$n = 262144$	Storage	Time (Seconds)
$k = 4$	16 MB	0.08
$k = 8$	32 MB	0.21
$k = 16$	64 MB	0.60
$k = 32$	128 MB	

Complexity of cSVD in `rkmatrix` format

Input : Matrix $M \in \mathbb{R}^{n \times n}$ in `rkmatrix` format,
$n = 262144$, rank k

Operation : SVD of M

$n = 262144$	Storage	Time (Seconds)
$k = 4$	16 MB	0.08
$k = 8$	32 MB	0.21
$k = 16$	64 MB	0.60
$k = 32$	128 MB	2.10
$k = 64$	256 MB	

COMPLEXITY OF CSVD IN `rkmatrix` FORMAT

INPUT : Matrix $M \in \mathbb{R}^{n \times n}$ in `rkmatrix` format,
$n = 262144$, rank k

OPERATION : SVD of M

$n = 262144$	Storage	Time (Seconds)
$k = 4$	16 MB	0.08
$k = 8$	32 MB	0.21
$k = 16$	64 MB	0.60
$k = 32$	128 MB	2.10
$k = 64$	256 MB	7.64

ALGORITHM FOR CSVD IN rkmatrix FORMAT

```
void csvd_rkmatrix(prkmatrix r, double *u, double *s, double *v,
                   int *k){




}
```

ALGORITHM FOR CSVD IN rkmatrix FORMAT

```
void csvd_rkmatrix(prkmatrix r, double *u, double *s, double *v,
                   int *k){
   double *Qa, *Ra, *Qb, *Rb, *Z, Us, Vs;
   int kt = r->kt, rows = r->rows, cols = r->cols;
   ... allocate ...









   ... deallocate ...
}
```

ALGORITHM FOR CSVD IN rkmatrix FORMAT

```
void csvd_rkmatrix(prkmatrix r, double *u, double *s, double *v,
                   int *k){
   double *Qa, *Ra, *Qb, *Rb, *Z, Us, Vs;
   int kt = r->kt, rows = r->rows, cols = r->cols;
   ... allocate ...

   qr_factorisation(r->a, rows, kt, Qa, Ra);
   qr_factorisation(r->b, cols, kt, Qb, Rb);









   ... deallocate ...
}
```

ALGORITHM FOR CSVD IN rkmatrix FORMAT

```
void csvd_rkmatrix(prkmatrix r, double *u, double *s, double *v,
                   int *k){
   double *Qa, *Ra, *Qb, *Rb, *Z, Us, Vs;
   int kt = r->kt, rows = r->rows, cols = r->cols;
   ... allocate ...

   qr_factorisation(r->a, rows, kt, Qa, Ra);
   qr_factorisation(r->b, cols, kt, Qb, Rb);

   multrans2_lapack(kt, kt, kt, Ra, Rb, Z);
   csvd_factorisation(Z, Us, s, Vs, kt, k);



   ... deallocate ...
}
```

ALGORITHM FOR CSVD IN `rkmatrix` FORMAT

```
void csvd_rkmatrix(prkmatrix r, double *u, double *s, double *v,
                   int *k){
   double *Qa, *Ra, *Qb, *Rb, *Z, Us, Vs;
   int kt = r->kt, rows = r->rows, cols = r->cols;
   ... allocate ...

   qr_factorisation(r->a, rows, kt, Qa, Ra);
   qr_factorisation(r->b, cols, kt, Qb, Rb);

   multrans2_lapack(kt, kt, kt, Ra, Rb, Z);
   csvd_factorisation(Z, Us, s, Vs, kt, k);

   mul_lapack(rows, kt, *k, Qa, Us, u);
   mul_lapack(cols, kt, *k, Qb, Vs, v);

   ... deallocate ...
}
```

(A2) Crossapproximation, full Pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

for $\nu = 1, \ldots, k$ **do**

end for

(A2) Crossapproximation, full Pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

for $\nu = 1, \ldots, k$ **do**

Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_{(i,j)} |M_{i,j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

end for

(A2) Crossapproximation, full Pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

for $\nu = 1, \ldots, k$ **do**

Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_{(i,j)} |M_{i,j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

if $\delta = 0$ **then**

Stop with rank $\nu - 1$, exact representation $R_{\nu-1} = M$.

else

end if

end for

(A2) Crossapproximation, full Pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

for $\nu = 1, \ldots, k$ **do**

Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_{(i,j)} |M_{i,j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

if $\delta = 0$ **then**

Stop with rank $\nu - 1$, exact representation $R_{\nu-1} = M$.

else

Compute entries of a^ν, b^ν:

$$(a^\nu)_i := M_{i, j_\nu}, \quad i \in \mathcal{I}, \qquad (b^\nu)_j := M_{i_\nu, j}/\delta, \quad j \in \mathcal{J}.$$

end if

end for

(A2) Crossapproximation, full Pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

for $\nu = 1, \ldots, k$ **do**

Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_{(i,j)} |M_{i,j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

if $\delta = 0$ **then**

Stop with rank $\nu - 1$, exact representation $R_{\nu-1} = M$.

else

Compute entries of a^ν, b^ν:

$$(a^\nu)_i := M_{i,j_\nu}, \quad i \in \mathcal{I}, \qquad (b^\nu)_j := M_{i_\nu,j}/\delta, \quad j \in \mathcal{J}.$$

Subtract new rank 1 approximation

$$M_{i,j} := M_{i,j} - (a^\nu)_i (b^\nu)_j, \qquad i \in \mathcal{I}, j \in \mathcal{J}.$$

end if

end for

(A3) CROSSAPPROXIMATION, PARTIAL PIVOTING

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

Wähle ein i_1.
for $\nu = 1, \ldots, k$ **do**

end for

(A3) Crossapproximation, partial pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k

Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

Wähle ein i_1.

for $\nu = 1, \ldots, k$ **do**

Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_j |M_{i_\nu, j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

end for

(A3) Crossapproximation, partial pivoting

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

Wähle ein i_1.
for $\nu = 1, \ldots, k$ **do**
  Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_j |M_{i_\nu, j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

  if $\delta = 0$ **then**
    Stop with rank $\nu - 1$, hopefully $R_{\nu-1} = M$?
  else

  end if
end for

(A3) CROSSAPPROXIMATION, PARTIAL PIVOTING

Input: Matrix $M \in \mathbb{R}^{n \times m}$, rank k Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

Wähle ein i_1.
for $\nu = 1, \ldots, k$ **do**
  Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_j |M_{i_\nu, j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

  if $\delta = 0$ **then**
    Stop with rank $\nu - 1$, hopefully $R_{\nu-1} = M$?
  else
    Compute entries of a^ν, b^ν:

$$(a^\nu)_i := M_{i, j_\nu} - \sum_{\mu=1}^{\nu-1} (a^\mu)_i (b^\mu)_{j_\nu}, \quad i \in \mathcal{I},$$

$$(b^\nu)_j := \frac{1}{\delta} \left(M_{i_\nu, j} - \sum_{\mu=1}^{\nu-1} (a^\mu)_{i_\nu} (b^\mu)_j \right), \quad j \in \mathcal{J}.$$

  end if
end for

(A3) CROSSAPPROXIMATION, PARTIAL PIVOTING

Input: Matrix $M \in \mathbb{R}^{n\times m}$, rank k Output: $R_k = \sum_{\nu=1}^{k} a^\nu (b^\nu)^T \approx M$

Wähle ein i_1.
for $\nu = 1, \ldots, k$ **do**
  Determine Index of largest absolute entry

$$(i_\nu, j_\nu) := \operatorname{argmax}_j |M_{i_\nu, j}|, \qquad \delta := M_{i_\nu, j_\nu}.$$

  if $\delta = 0$ **then**
    Stop with rank $\nu - 1$, hopefully $R_{\nu-1} = M$?
  else
    Compute entries of a^ν, b^ν:

$$(a^\nu)_i := M_{i,j_\nu} - \sum_{\mu=1}^{\nu-1} (a^\mu)_i (b^\mu)_{j_\nu}, \quad i \in \mathcal{I},$$

$$(b^\nu)_j := \frac{1}{\delta}\left(M_{i_\nu, j} - \sum_{\mu=1}^{\nu-1} (a^\mu)_{i_\nu} (b^\mu)_j\right), \quad j \in \mathcal{J}.$$

    New Pivot index $i_{\nu+1} = \operatorname{argmax}_{i \neq i_\nu} |a_i^\nu|$ (or some other heuristic)
  end if
end for