École polytechnique fédérale de Lausanne

MATH-453. Computational linear algebra

Spring 2022

Blendenpik: Randomized least squares

bruno rodriguez carrillo

Sciper 326180

Professor:
Daniel Kresner

June 5th, 2022

1. The goal of the Blendenpik algorithm is to solve the following problem:

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|, \tag{1.1}$$

such a system is highly overdetermined where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $m \geq n$. Problem (1.1) is equivalent to approximate a solution to the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

As shown in [3], let $\mathcal{E}$ denote the set of solutions to Eq.(1.1) and let $\mathbf{r}_\mathbf{x} = \mathbf{b} - \mathbf{A}\mathbf{x}$ be the residual for a specific $\mathbf{x}$, that is

$$\mathcal{E} = \left\{ \mathbf{x} \in \mathbb{R}^n \text{ with } \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 \to \min \right\}.$$

Then it can be shown that:

$$\mathbf{x} \in \mathcal{E} \iff \mathbf{A}^T\mathbf{r}_\mathbf{x} = 0 \Rightarrow \mathbf{r}_\mathbf{x} \perp \mathcal{R}(\mathbf{A}),$$

with $\mathcal{R}(\mathbf{A})$ the space spanned by the columns of $\mathbf{A}$. From this we can obtain the so-called normal equations:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}. \tag{1.2}$$

Furthermore, we can look at Blendenpik as an algorithm to construct a proper preconditioner $\mathbf{P}$. Let us assume we count on the reduced QR factorization $\mathbf{A} = \mathbf{Q}\mathbf{R}$. Choosing $\mathbf{P} = \mathbf{R}^{-1}$, we could solve $\mathbf{A}\mathbf{x} = \mathbf{b}$, which corresponds to a minimization problem of the form (1.3).

$$\text{Solve } \min_{\mathbf{y}} \left\|\mathbf{A}\mathbf{R}^{-1}\mathbf{y} - \mathbf{b}\right\|_2, \text{then solve } \mathbf{R}\mathbf{x} = \mathbf{y}. \tag{1.3}$$

That is, using $\mathbf{P} = \mathbf{R}^{-1}$, we obtain $\mathbf{A}\mathbf{R}^{-1} = \mathbf{Q}$, and then $\kappa(\mathbf{Q}) = 1$, a perfect preconditioner. However, constructing such a $\mathbf{P}$ can be too expensive for large matrices. As a consequence, Blendenpik proposes constructing a cheaper $\mathbf{P}$ from a reduced QR factorization of a few rows of $\mathbf{A}$. In other words, we select a few rows from $\mathbf{A}$ by applying a uniform row-sampling operator $\mathcal{S}$ on $\mathbf{A}$, and then we compute its reduced QR decomposition, i.e $\mathcal{S}\mathbf{A} = \mathbf{Q}_s\mathbf{R}_s$. By doing this, we hope that the preconditioned system $\mathbf{A}\mathbf{R}_s^{-1}$ has almost orthonormal columns, $\kappa(\mathbf{A}\mathbf{R}_s^{-1}) \approx 1$. Throughout this report, when we mention QR decomposition, we refer to the reduced version.

Now, we can summarize the main ideas of [1].

(a) Blendenpik is a careful-engineering least-squares solver

(b) Blendenpik is slower than LAPACK on tiny matrices, nearly square ones, and on some sparse matrices. On large matrices, Blendenpik is about four times faster than LAPACK

(c) We can use a small sample of rows of $\mathbf{A}$ and solve (1.1) using such a sample as long as the size of sample is sufficiently large. In general, under certain conditions on a matrix $\mathbf{A}$, a uniform sample of $\Omega(n\log(m)\log(n\log(m)))$ rows leads to a residual that is within a factor of $1+\epsilon$ of the optimal with high probability for $\epsilon > 0$

(d) A uniform sample of the rows of $\mathbf{A}$ works well when the *coherence* of $\mathbf{A}$ is small - coherence is defined in point 2). The coherence of matrices with random independent uniform entries tends to be small and we use a randomized row-mixing, pre-processing phase to reduce the coherence. With high probability, from a uniform sample of the rows of the row-mixed matrix $\mathbf{A}$, we obtain a good pre-conditioner

(e) The quality of uniform sampling pre-conditioners depends on how much the solution to (1.1) depends on specific rows. Coherence is the key concept for measuring the dependence of the solution on specific rows of the original matrix $\mathbf{A}$

(f) The condition number of the matrix $\kappa(\mathbf{A}\mathbf{R}_s^{-1})$ does not depend on $\kappa(\mathbf{A})$ as shown in point 3.

(g) Convergence of LSQR (least-squares) depends on the distributions of the singular values of $\mathbf{A}\mathbf{R}_s^{-1}$, not just on the extreme singular values

(h) This algorithm can fail to produce an effective pre-conditioner when 1) the pre-conditioner is rank deficient or highly ill conditioned, and 2) the condition number $\kappa\left(\mathbf{A}\mathbf{R}_s^{-1}\right)$ is high which results in LSQR converging slowly but the solution does not fail)

(i) Three different unitary transformation for row mixing: WHT, DCT and DHT; these improve coherence in different ways. For semi-coherent -intuitively, a value of coherence between the minimal and maximal value of coherence- and coherent -maximal coherence- matrices there is no significant difference between these different mixing methods

(j) Row-mixing steps reduce the coherence and improve the outcome of random sampling. After a single row-sampling step, the coherence is within $O(\log m)$ factor of the optimal with high probability. However, a single row-mixing phase may not remove the coherence completely. As a result, extra row-mixing steps will reduce coherence further, which means that LSQR converges faster. If $\mathbf{A}$ is completely incoherent, there is no need of row mixing

(k) Convergence rate is slower on coherent matrices than on incoherent -minimal coherence- and semi-coherent ones. On incoherent and semi-coherent matrices the number of iterations grows very slowly. In coherent ones it grows faster

(l) The behavior of Blendenpik depends on the seed unitary transformation, on the number of row-mixing steps and on the sample size

(m) The most expensive phase is the LSQR phase. Each LSQR iteration takes $O(mn)$ time and the number of iterations grows slowly. Asymptotic running time: row-mixing phase is $O(mn \log m)$ and for QR computation it is $O(n^3)$. A considerable speedup can be achieved by relaxing the converge threshold of the LSQR. The row mixing phase takes about 15% of the overall solver time

As a result, the proposed algorithm can be summarized as: row-mixing step of matrix $\mathbf{A}$, uniform sampling of rows of $\mathbf{A}$, and solution to (1.3) using matrix $\mathbf{R}_s$ as a preconditioner.

2. We now define formally the concept of *coherence*. Given an $m \times n$ matrix $\mathbf{U}$ whose columns form an orthonormal basis for the column space of an $m \times n$ full rank matrix $\mathbf{A}$, the coherence of $\mathbf{A}$, denoted by $\mu(\mathbf{A})$, is defined as:

$$\mu(\mathbf{A}) := \max \|\mathbf{U}_{i,\star}\|_2^2. \tag{1.4}$$

That is, $\mu(\mathbf{A})$ represents the row of $\mathbf{U}$ with the maximum 2-norm. As stated in [1], $n/m \leq \mu(A) \leq 1$. Intuitively, when $\mu(\mathbf{A})$ is close to n/m, that means that the solution depends homogeneously on each row and then we could take a small uniform sample of rows of $\mathbf{A}$; otherwise, the row corresponding to the maximum coherence must be in the uniform sample since the solution depends greatly on such a row; that is, it is a canonical vector. When $\mu(A) = n/m$, $n/m \leq \mu(A) \leq 1$, and $\mu(A) = 1$, we have an incoherent, semi-coherent and coherent matrix, respectively.

One way to show that $n/m \leq \mu(\mathbf{A}) \leq 1$ is as follows. Let us consider $\mathbf{U}$, as given in (1.4), has row vectors $\boldsymbol{u}_i \in \mathbb{R}^n$ for $i = 1, .., m$. Besides, we have that $\|\mathbf{U}\|_F^2 = n$. Thus, we can bound $\|\mathbf{U}\|_F^2$ as follows:

$$\|\mathbf{U}\|_F^2 \leq m \cdot \max \|\mathbf{U}_{i,\star}\|_2^2 = m \cdot \mu(\mathbf{A}),$$

where $\|\mathbf{U}_{i,\star}\|_2^2$ is the row vector $\boldsymbol{u}_i$ with the largest 2-norm. Thus, we have:

$$\frac{n}{m} \leq \mu(\mathbf{A}) \leq 1,$$

with the last inequality coming from the fact that $\mathbf{U}$ is orthonormal.

Let us analyze why $\mu(\mathbf{A})$ plays a crucial role when it comes to row sampling. Uniform row sampling fails for a general matrix when $\mathbf{A}$ has a column j that is zero except for $A_{ij} \neq 0$, any subset of the matrix $\mathbf{A}$ that does not include row j is rank deficient. If $m >> n$, we would require to sample close to m rows in order to increase the probability of finding row j in such a subset. When the sample size is too small, the pre-conditioner $\mathbf{R}_s$ is rank deficient and LSQR fails. Thus, the size of a uniform sample must be too large.

For this reason, uniform sampling works well only when the coherence of the matrix $\mathbf{A}$ is small. The smaller $\mu(\mathbf{A})$, the fewer rows of $\mathbf{A}$ we need in a sample.

An $m \times n$ matrix $\mathbf{A}$ such that $\mu(\mathbf{A}) = 1$ can be constructed by setting to 0 all the $j - th$ column of $\mathbf{A}$ expect at row i. For example if we have $\mathbf{A} \in \mathbb{R}^{m \times n}$, in Matlab, this can be achieved by:

```
m = 1000; n = 50;
A = rand(m,n);
A(:, j) = 0;
A(i, j) = rand();
[Q, R] = qr(A, 0);
coherence = max(sum(Q.^2, 2));
```

where $1 \leq i \leq m$ and $1 \leq j \leq n$. From the above Matlab routine, we have $\mu(\mathbf{A}) = 1.00$, as expected.

Furthermore, a Matlab code to compute 1000 random matrices is attached with this report. We use the Matlab function rand(1000, 50) to create such matrices. As expected, we have that the averaged cohrence is close to n/m; namely, 0.072; this follows what theory predicts: with high probability, random matrices have small coherence.

3. We now prove Theorem 1.1.1, Theorem 3.2 in [1], as stated below.

Theorem 1.1.1. *Let $\mathbf{A}$ be an $m \times n$ full-rank matrix, and let $\mathcal{S}$ be a random sampling operator that samples $r \geq n$ rows from $\mathbf{A}$ uniformly. Let $\tau = C\sqrt{\frac{m\mu(\mathbf{A}) \log r}{r}}$, where C*

is some constant defined in the proof. Assume that $\delta^{-1}\tau < 1$. *With probability of at least* $1-\delta$, *the sampled matrix* $\mathcal{S}\mathbf{A}$ *is full-rank, and if* $\mathcal{S}\mathbf{A} = \mathbf{QR}$ *is a reduced QR factorization of* $\mathcal{S}\mathbf{A}$, *we have*

$$\kappa\left(\mathbf{AR}^{-1}\right) \leq \sqrt{\frac{1+\delta^{-1}\tau}{1-\delta^{-1}\tau}} \tag{1.1}$$

.

Furthermore, to prove Theorem 1.1.1, we need Theorem 7 in [2] and Theorem in [4] below.

Theorem 1.1.2 (Theorem 1 in [4])**.** *Suppose* l, m *and* n *are positive integers such that* $m \geq l \geq n$. *Suppose further that* $\mathbf{A}$ *is an* $m \times n$ *full-rank matrix, and that the SVD of* $\mathbf{A}$ *is*

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times n}\mathbf{\Sigma}_{n\times n}\mathbf{V}^*_{n\times n}. \tag{1.1}$$

Suppose in addition that $\mathbf{T}$ *is an* $l \times m$ *matrix such that the* $l \times n$ *matrix* $\mathbf{TU}$ *has full rank. Then there exist an* $n \times n$ *matrix* $\mathbf{P}$, *and an* $l \times n$ *matrix* $\mathbf{Q}$ *whose columns are orthonormal, such that*

$$\mathbf{T}_{l\times m}\mathbf{A}_{m\times n} = \mathbf{Q}_{l\times n}\mathbf{P}_{n\times n}. \tag{1.2}$$

Futhermore, if $\mathbf{P}$ *is any* $n \times n$ *matrix, and* $\mathbf{Q}$ *is any* $l \times n$ *matrix whose columns are orthonormal, such that* $\mathbf{P}$ *and* $\mathbf{Q}$ *satisfy* (1.1), *then* $\kappa\left(\mathbf{AP}^{-1}\right) = \kappa\left(\mathbf{TU}\right)$.

Theorem 1.1.3 (Theorem 7 in [2])**.** *Suppose* $\mathbf{A} \in \mathrm{R}^{m\times n}$, *and* $\mathbf{B} \in \mathrm{R}^{n\times p}$, *and* $c \leq n$. *Construct matrices* $\mathbf{C}$ *and* $\mathbf{R}$ *with Algorithm 6, using the EXPECTED(c) algorithm. If the sampling probabilities* $\left\{p_i\right\}_{i=1}^{n}$ *used by the algorithm are of the form*

$$p_i \geq \beta\frac{\left|\mathbf{A}^{(i)}\right|_2\left|\mathbf{B}^{(i)}\right|_2}{\sum_{j=1}^{n}\left|\mathbf{A}^{(j)}\right|_2\left|\mathbf{B}^{(j)}\right|_2} \text{ or } p_i \geq \beta\frac{\left|\mathbf{A}^{(i)}\right|_2}{\left\|\mathbf{A}\right\|_F^2}. \tag{1.1}$$

Then

$$\mathrm{E}\left[\left\|\mathbf{AB}-\mathbf{CR}\right\|_F\right] \leq \sqrt{\frac{1}{\beta c}}\left\|\mathbf{A}\right\|_F\left\|\mathbf{B}\right\|_F. \tag{1.2}$$

If, in addition, $\mathbf{A} = \mathbf{B}^T$, *then*

$$\mathrm{E}\left[\left\|\mathbf{AA}^T-\mathbf{CC}^T\right\|_2\right] \leq C\sqrt{\frac{\log c}{\beta c}}\left\|\mathbf{A}\right\|_F\left\|\mathbf{A}\right\|_2, \tag{1.3}$$

where C *is a constant,* $\beta \in (0,1]$, $\sum_{i=1}^{n} p_i = 1$, *and* $\left|\mathbf{A}^{(i)}\right|$ *is the norm of the* i*-th column of* $\mathbf{A}$. *Refer to [2] for information on the Algorithm 6 and EXPECTED(c) algorithm.*

Proof. According to theorem (1.1.1), we sample $r \geq n$ rows from $\mathbf{A}$, which clearly means that $n \leq r \leq m$.

According to *Algorithm 6* in [2], we can select r columns of any $m \times n$ matrix $\mathbf{A}$ by constructing an $n \times r$ matrix $\mathbf{S}$ and an $r \times r$ diagonal matrix $\mathbf{D}$ by using the $EXACTLY(r)$

or the $EXPECTED(r)$ algorithms [2]. By doing this, the matrix $\mathbf{A}$ can be approximated by an $m \times r$ matrix $\mathbf{C} = \mathbf{ASD}$.

Let us assume that we select $r \leq n$ columns from $\mathbf{A}$ instead of rows; this will become clear in next steps. Then, we uniformly sample r columns from $\mathbf{A}$, which means that the probability of picking the i-th column of $\mathbf{A}$ is $p_i = \frac{1}{n}$ for $i = 1, \ldots, n$. Using the *EXACTLY(r)* algorithm, $\mathbf{D}$ is computed as $D_{t,t} = \dfrac{1}{\sqrt{r p_{i_t}}} = \sqrt{\dfrac{n}{r}}$ for $t = 1, \ldots, r$; that is, $\mathbf{D} = \sqrt{\dfrac{n}{r}} \cdot \mathbf{I}_r$, where $\mathbf{I}_r$ is an $r \times r$ identity matrix. We also notice that $\mathbf{S}$ is responsible for picking r columns from $\mathbf{A}$. This means that $\mathbf{C}$ can be viewed as a matrix whose columns are actual columns of $\mathbf{A}$ re-scaled by a factor $\sqrt{\dfrac{n}{r}}$. We define the sampling operator $\mathcal{S} := \mathbf{SD}$.

In the other hand, we know that $\mathbf{A}$ is full rank, thus we compute its QR decomposition as $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q} \in \mathrm{R}^{m \times n}$ is orthogonal and $\mathbf{R} \in \mathrm{R}^{n \times n}$ is upper-triangular. It is quite important to realize that if we use (1.3) in theorem (1.1.3) by choosing $\mathbf{A} = \mathbf{Q}^T$, sampling columns, as previously discussed, is equivalent to sampling rows, which is what we are interested in. This means that we can bound $\mathrm{E}\left[\left\|\mathbf{Q}^T\mathbf{Q} - (\mathcal{S}\mathbf{Q}^T)^T\left(\mathcal{S}\mathbf{Q}^T\right)\right\|_2\right]$ using (1.3), where the sampled rows can be written as

$$\left(\mathcal{S}\mathbf{Q}^T\right)^T = \sqrt{\frac{m}{r}} \cdot \begin{bmatrix} — & \mathbf{r}_1^T & — \\ — & \mathbf{r}_2^T & — \\ — & \vdots & — \\ — & \mathbf{r}_r^T & — \end{bmatrix}, \tag{1.4}$$

where $\mathbf{r}_i \in \mathrm{R}^n$ is the i-th row of $\mathbf{Q}$, and $i = 1, \ldots, r$.

On the other hand, it can be proven that for any matrix $\mathbf{A}$, its condition number $\kappa(\mathbf{A})$ satisfies (1.5), [5].

$$\kappa\left(\mathbf{A}^T\mathbf{A}\right) = \left\|\mathbf{A}^T\mathbf{A}\right\|_2 \left\|\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\right\|_2 = \left(\kappa(\mathbf{A})\right)^2. \tag{1.5}$$

All previous discussion leads to the following expression using (1.3)

$$\mathrm{E}\left[\left\|\mathbf{Q}^T\mathbf{Q} - (\mathcal{S}\mathbf{Q}^T)^T(\mathcal{S}\mathbf{Q}^T)\right\|_2\right] \leq C\sqrt{\frac{\log r}{\beta r}}\left\|\mathbf{Q}^T\right\|_F\left\|\mathbf{Q}^T\right\|_2. \tag{1.6}$$

We can easily see that $\left\|\mathbf{Q}^T\right\|_F^2 = \mathrm{trace}\left(\mathbf{I}_{n\times n}\right) = n$ and that $\left\|\mathbf{Q}^T\right\|_2 = \sqrt{\lambda_{\max}\left(\mathbf{Q}\mathbf{Q}^T\right)} = 1$. In addition, if we take $\beta = \dfrac{n}{m\mu(\mathbf{A})}$, we fulfill all conditions on β stated in Theorem 1.1.3. Consequently, inserting all these values into (1.6) we obtain (1.7).

$$\mathrm{E}\left[\left\|\mathbf{Q}^T\mathbf{Q} - (\mathcal{S}\mathbf{Q}^T)^T(\mathcal{S}\mathbf{Q}^T)\right\|_2\right] \leq C\sqrt{\frac{m\mu(\mathbf{A})\log r}{r}}. \tag{1.7}$$

We now define $\tau := C\sqrt{\dfrac{m\mu(\mathbf{A})\log c}{c}}$. Afterwards, we use Markov's inequality as follows:

$$\mathrm{P}\left(\left\|\mathbf{I}_{n\times n}-(\mathcal{S}\mathbf{Q}^T)^T(\mathcal{S}\mathbf{Q}^T)\right\|_2\geq\delta^{-1}\tau\right)\leq\frac{\mathrm{E}\left[\mathbf{I}_{n\times n}-(\mathcal{S}\mathbf{Q}^T)^T(\mathcal{S}\mathbf{Q}^T)\right\|_2\right]}{\delta^{-1}\tau}\leq\delta,$$

then with probability $1-\delta$,

$$\left\|\mathbf{I}_{n\times n}-(\mathcal{S}\mathbf{Q}^T)^T(\mathcal{S}\mathbf{Q}^T)\right\|_2<\delta^{-1}\tau<1, \tag{1.8}$$

and therefore $\mathcal{S}\mathbf{Q}^T$ is full rank.

Now, we state that the desired bound on $\kappa(\mathcal{S}\mathbf{Q}^T)$ follows from a Rayleigh quotient argument. Let us consider the matrix $\mathbf{M}:=(\mathcal{S}\mathbf{Q}^T)^T(\mathcal{S}\mathbf{Q}^T)$. We note that since $\mathbf{M}$ is symmetric, we can obtain its spectrum using the Rayleight quotient $R(\mathbf{M},\mathbf{x})$ for $\mathbf{x}\neq 0$. That is:

$$R(\mathbf{M},\mathbf{x})=\frac{\mathbf{x}^T\mathbf{M}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}=\frac{\mathbf{x}^T\mathbf{x}-\mathbf{x}^T\mathbf{x}+\mathbf{x}^TM\mathbf{x}}{\mathbf{x}^T\mathbf{x}}=\frac{\mathbf{x}^T\mathbf{x}}{\mathbf{x}^T\mathbf{x}}+\frac{\mathbf{x}^T(\mathbf{M}-\mathbf{I}_{n\times n})\mathbf{x}}{\mathbf{x}^T\mathbf{x}}=1-\eta. \tag{1.9}$$

where $\eta=R(\mathbf{I}_{n\times n}-\mathbf{M},\mathbf{x})$ and we should mention that the matrix $\mathbf{I}_{n\times n}-\mathbf{M}$ is also symmetric. Thus, by (1.8), it follows that $\eta<\delta^{-1}\tau$. As a result, the eigenvalues of $\mathbf{M}$ lie between $1-\delta^{-1}\tau$ and $1+\delta^{-1}\tau$. Consequently, the desired bound (1.10) follows from (1.5) using $\left(\kappa\left(\mathcal{S}\mathbf{Q}^T\right)\right)^2$, and from leveraging theorem (1.1.2).

$$\kappa(\mathbf{A}\mathbf{R}^{-1})=\kappa(\mathcal{S}\mathbf{Q}^T)\leq\sqrt{\frac{1+\delta^{-1}\tau}{1-\delta^{-1}\tau}}. \tag{1.10}$$

□

4. A simple implementation of Blendenpik is attached with this report.

   Let us detail how the operator $\mathcal{S}$ is computed. First, we generate a vector of size $\tilde{m}$ with random uniform entries in $[0,1]$ and $\tilde{m}=\lceil m/1000\rceil\times 1000$. We call this vector $\mathbf{u}$. Then if $\tilde{m}>m$, we pad $\mathbf{A}$ with a matrix of size $(\tilde{m}-m)\times n$ with zero entries to create a new matrix $\mathbf{A}$ of size $\tilde{m}\times n$; if $\tilde{m}=m$, $\mathbf{M}=\mathbf{A}$.

   A row of $\mathbf{A}$ is sampled if $u_i<\gamma n/\tilde{m}$, with $u_i\in\mathbf{u}$ and $1\leq i\leq\tilde{m}$. That is if $u_i<\gamma n/\tilde{m}$, the $ii-th$ entry in $\mathcal{S}$ is 1 and 0 otherwise, and $\mathcal{S}_{ij}=0, i\neq j$.

   By doing this, we are left with those rows that are sufficiently important to be kept in the sample. The expected number of sampled rows is γn, but this number can be higher or smaller. When we compute $\mathcal{S}\mathbf{A}$, we are left with a matrix that has fewer rows than the original one $\mathbf{A}$.

5. We construct two ill-conditioned matrices $\mathbf{A}$ using the following commands on Matlab.

   Incoherent matrix

```
U = orth(rand(20000, 400));
S = diag(linspace(1, 1e5, 400));
V = orth(rand(400));
A = U*S*V';
b = rand(20000, 1);
```

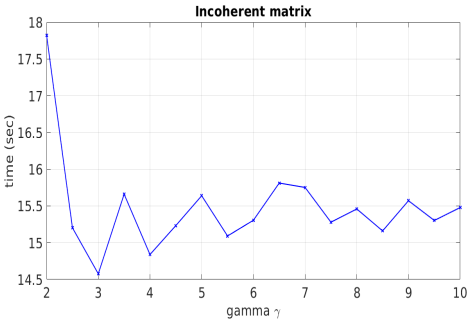
For one run of the algorithm, $\mu(\mathbf{A}) = 0.02362$, as expected since $\mathbf{A}$ is incoherent and $\mu(A)$ should be close to $n/m = 0.020$, and $\kappa(\mathbf{A}) = 10^5$ using cond command.

Coherent matrix

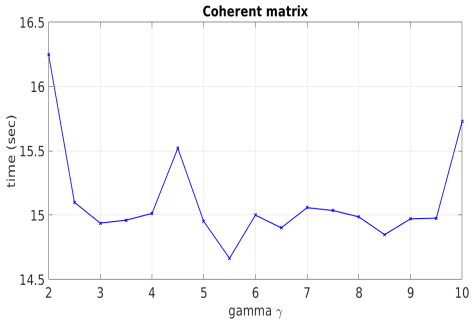
```
A = [diag(linspace(1, 1e5, 400)); zeros(19600, 400)];
A = A + 1e-8*ones(20000, 400);
```

For one run of the algorithm, $\mu(\mathbf{A}) = 1.0$, as expected since $\mathbf{A}$ is coherent and $\mu(\mathbf{A})$ should be close to 1, and $\kappa(\mathbf{A}) = 10^5$ using cond command.

In both cases, to numerically compute the minimal value of γ, we use sample sizes γn, where γ ranges from 2 to 10 by taking 0.5 steps. To construct $\mathbf{A}$, we do not fix the random seed and for each value of γ, we create a new matrix $\mathbf{A}$ of the form given above. Furthermore, we measure how much time it takes for our code to compute the solution; then we have the following plots for a single run, figure 1.1.



(a) Minimal γ for incoherent $\mathbf{A}$, LSQR.



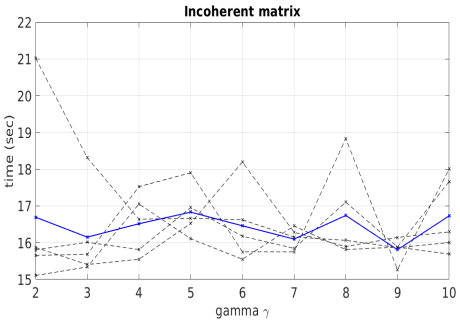
(b) Minimal γ for coherent $\mathbf{A}$, LSQR.

Figure 1.1: Minimal γ computation for LSQR.

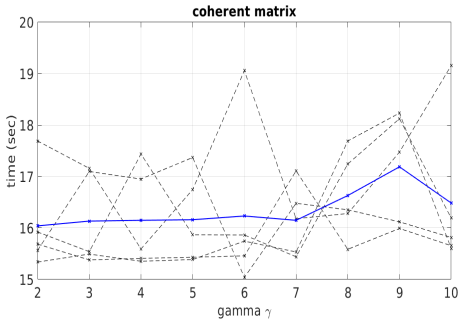
Consequently, we will use $\gamma = 5$ for our coherent and incoherent matrices given above. We use these values for next question too. Note that these plots correspond to what theory predicts. Indeed, the more coherent a matrix is, the larger the uniform sample is. It is also important to say that the time required to compute the solution to (1.1) is proportional to the number of iterations of LSQR/MINRES.

It is relevant to observe that the values for γ in both cases vary from one run to another due to randomness. That is, in some cases we have γ distinct to the values shown above. To have a more reliable approximation to γ, we could run our code multiple times and then compute the average values for γ. According to the Strong Law of Large Numbers, as the number of samples tends to infinity, as long as these are independent and identically distributed random samples, the computed value is closer to the true mean value. Nevertheless, this is computationally expensive in our case due to the size of our matrices.

As manner of example, we take five samples for each of the matrices shown above and plot the mean values of time as function of γ, which leads to figure 1.2.



(a) Mean γ for incoherent $\mathbf{A}$, LSQR.



(b) Mean γ for coherent $\mathbf{A}$, LSQR.

Figure 1.2: Mean γ computation for LSQR.

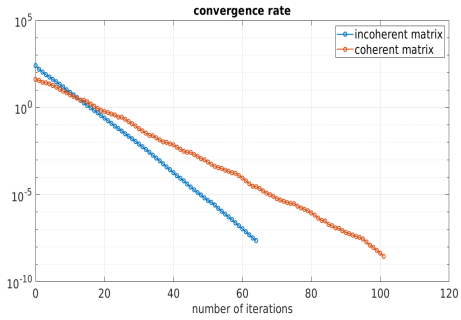
The mean value corresponds to the solid line in both figures and the dashed lines correspond to each of the five independent runs. Such a numerical experiment helps support our choice of γ for each of the given matrices. For figure 1.2, we use LSQR only.

6. For all following plots, we use the same parameters of question e) and the tolerance is set to 10^{-10}.

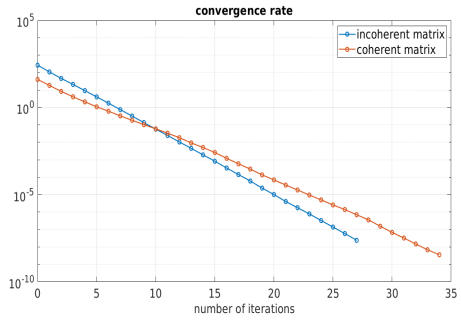
   Let us recall that we aim at solving (1.1). One of the reason Blendenpik is successful is due to the fact that $\kappa\left(\mathbf{A}\mathbf{R}_s^{-1}\right)$ is well-conditioned with high probability, then we can make use of such the preconditioner $\mathbf{R}_s^{-1}$ as previously discussed. Consequently, we can write the normal equations (1.2) as:

$$\left(\mathbf{A}\mathbf{R}_s^{-1}\right)^T\left(\mathbf{A}\mathbf{R}_s^{-1}\right)\mathbf{y} = \left(\mathbf{A}\mathbf{R}_s^{-1}\right)^T\mathbf{b}. \tag{1.11}$$

   Finally, we solve $\mathbf{R}_s\mathbf{x} = \mathbf{y}$, which corresponds to the solution to (1.1). We use the same preconditoner for both LSQR and MINRES. We do point out that we solve (1.11) in Matlab by passing the preconditoned system (1.11). As a result, we have figure 1.3 where we show how the convergence behaves for LSQR and MINRES, respectively.



(a) Convergence LSQR.



(b) Convergence MINRES.

Figure 1.3: Convergence behaviour for LSQR and MINRES.

Instead of plotting $\|\left(\mathbf{A}\mathbf{R}_s^{-1}\right)^T\mathbf{r}^{(i)}\|_2/\|\mathbf{A}\mathbf{R}_s^{-1}\|_F\|\mathbf{r}^{(i)}\|_2$ as in [1], we compute the numerator only since it is the quantity we have access to in Matlab; that is, on the y axis of figure 1.3

we plot in $\log_{10}$ scale $\|\left(\mathbf{A}\mathbf{R}_s^{-1}\right)^T \mathbf{r}^{(i)}\|_2$ and on the x axis we plot the number of iteration for both matrices described in the point 5.

As observed, when we use MINRES, we require fewer iteration to converge to the solution to (1.1) for a fixed tolerance. We observe that MINRES requires fewer than 50 % of LSQR iterations to achieve the given tolerance. Finally, as suggested in [1], we could get rid of the row-mixing step for the incoherent matrix and still obtain a similar convergence rate.

Bibliography

[1] Haim Avron, Petar Maymounkov, and Sivan Toledo. "Blendenpik: Supercharging LAPACK's least-squares solver". In: *SIAM Journal on Scientific Computing* 32.3 (2010), pp. 1217–1236.

[2] Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. "Relative-error CUR matrix decompositions". In: *SIAM Journal on Matrix Analysis and Applications* 30.2 (2008), pp. 844–881.

[3] Walter Gander, Martin J Gander, and Felix Kwok. *Scientific computing-An introduction using Maple and MATLAB*. Vol. 11. Springer Science & Business, 2014.

[4] Vladimir Rokhlin and Mark Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression". In: *Proceedings of the National Academy of Sciences* 105.36 (2008), pp. 13212–13217.

[5] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.