Homework #1, EECS 598-006, W20. Due **Thu. Jan. 16**, by 4:00PM

Notes

- **Autograder instructions:**
  Coding problems will be graded via an automated system that verifies the correctness of your code by evaluating it on test cases. To submit your code, attach your file to an email (from your `@umich.edu` account) and send it to: `mailto:eecs556@autograder.eecs.umich.edu`. The system will send an email response, often within a few seconds, saying whether your code passes or fails its (often randomly generated) test cases.
  If your code fails: Edit and resubmit your code via the same process. You have unlimited retries.
  It is much more intelligent to first write your own test cases rather than trying to use the autograder as a debugger!
  Once your code passes: You are done! You'll receive full credit for the problem. The system already saved an electronic copy for our records. The "pass" email also contains a confirmation code. If we accidentally fail to give you credit for the problem, forward to us the confirmation code and we'll record your credit.
  Typically you will not submit anything to gradescope for autograder problems. In some cases we will also grade code based on *efficiency*, and such problems will instruct you specifically to also submit code to gradescope.
  The autograder is using JULIA 1.3.1. If you test your code with JULIA 0.7 and you get deprecation warning messages, then your code is unlikely to work with JULIA 1.3. Test with 1.3 yourself before submitting to autograder.

Problems

The following problems are all designed to review concepts from EECS 551 that will be used in this course. Many of them are closely related (but not identical) to homework problems in EECS 551.

It is permitted for you to use your own solutions to homework problems in EECS 551/505 to help solve these problems. It is an honor code violation to refer to instructor-provided solutions from any course when solving any HW problem in this class.

---

1. [13]

Approximating a matrix by another matrix of lower rank is an important operation in many signal processing problems.

(a) [10] Write a JULIA function `rank1` that computes the rank-1 approximation to its input matrix that is "best" in the sense of minimizing the Frobenius norm of the difference between the original matrix and its approximation. (The last character there is the digit 1 not the letter l.)
Your file should be named `rank1.jl` and should contain the following function:

```julia
"""
    B = rank1(A)

In:
* `A` `M x N` matrix

Out:
* `B` best rank-1 approximation to A in Frob. norm sense
"""
function rank1(A::AbstractMatrix)
```

Submit your solution to `mailto:eecs556@autograder.eecs.umich.edu`.

(b) [3] Is your best approximation unique? Discuss.

---

2. [3] Show that the **Schatten p-norm** is **unitarily invariant** for any $p > 0$ even though it is only truly a **norm** for $p \geq 1$.

3. [9] A function $f(\cdot) : \mathbb{F}^N \mapsto \mathbb{F}^M$ is **Lipschitz continuous** with respect to norms $\|\cdot\|_X$ and $\|\cdot\|_Y$ iff there exists $L < \infty$ such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{z})\|_Y \leq L \|\boldsymbol{x} - \boldsymbol{z}\|_X, \quad \forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{F}^N.$$

The usual choice of norm is the **Euclidean norm** (when otherwise unspecified).

(a) [3] Suppose that $f(\boldsymbol{x})$ is Lipschitz continuous with **Lipschitz constant** L_f and $g(\boldsymbol{x})$ is Lipschitz continuous with Lipschitz constant L_g. Find a Lipschitz constant for the function $h(\boldsymbol{x}) = f(\boldsymbol{x}) + \beta g(\boldsymbol{x})$.
For this part, consider general norms $\|\cdot\|_X$ and $\|\cdot\|_Y$, and allow β to be any complex number.

(b) [3] Is your preceding answer the best Lipschitz constant for h in general? Explain why or why not.

(c) [3] Consider the regularized least-squares cost function $\Psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \frac{1}{2}\|\boldsymbol{T}\boldsymbol{x}\|_2^2$, where $\beta > 0$. Find a Lipschitz constant for the gradient of $\Psi(\boldsymbol{x})$, expressed in terms of the **singular values** of $\boldsymbol{A}$ and $\boldsymbol{T}$.
For this part, use the usual Euclidean norms.

4. [25] An operation that appears in numerous optimization problems is the **proximal operator** associated with a **convex** function $f : \mathbb{F}^N \mapsto \mathbb{R}$, defined as

$$\operatorname{prox}_f(\boldsymbol{v}) \triangleq \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \frac{1}{2}\|\boldsymbol{v} - \boldsymbol{x}\|_2^2 + f(\boldsymbol{x}).$$

Consider $\mathbb{F} = \mathbb{C}$ throughout this problem.

(a) [3] When $f(\cdot)$ is a **convex** function (but not necessarily strictly convex), determine whether the "arg min" above is **unique**. (Prove or provide a simple counter-example.)

(b) [3] Determine analytically the proximal operator for the 2-norm: *i.e.*, $f(\boldsymbol{z}) = \beta \frac{1}{2}\|\boldsymbol{z}\|_2^2$ for $\beta \geq 0$.

(c) [3] Even though proximal operators are usually defined for convex functions, we also sometimes use them for non-convex functions. Determine the proximal operator for the (nonconvex) 0-norm $f(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_0$.

(d) [3] Determine analytically the proximal operator for the (convex) **Fair potential**, *i.e.*,

$$f(\boldsymbol{z}) = \beta \boldsymbol{1}_N' \psi.(\boldsymbol{z}) = \beta \sum_{n=1}^N \psi(z_n), \qquad \psi(z) = \delta^2 \left(|z|/\delta - \log(1 + |z|/\delta)\right).$$

Hint. First consider the case where y is real and positive, then generalize to $\mathbb{C}^N$.

(e) [10] Write a JULIA function `shrink_fair` that has inputs $\boldsymbol{y}$, β and δ and returns the corresponding proximal operator.
Your file should be named `shrink_fair.jl` and should contain the following function:

```
"""
    xh = shrink_fair(v, reg::Real, delta::Real)

Compute minimizer over `x` of `1/2 |v - x|^2 + reg fair(x,delta)`
where `fair(x,delta) = delta^2 (|x/delta| - log(1 + |x/delta|))`
Do not use any "for" loop in your solution!

In
* `v` scalar, vector, or array of input values
* `reg` regularization parameter
* `delta` regularization parameter

Out
* `xh` solution to minimization problem for each element of `v`
(same size as `v`)
"""
function shrink_fair(v, reg::Real, delta::Real)
```

Submit your solution to `mailto:eecs556@autograder.eecs.umich.edu`.
Hint. Your solution will need a `.` in it. Try to write a clean, readable solution that has only one such `.` in it.

(f) [3] Make a single plot that shows the proximal operators for both the 0-norm and for the Fair potential for $\beta = 8$ and $\delta = 2$, for `v = LinRange(-30,30,501)`. Label your axes and provide an appropriate legend.

---

5. [24] Prove each of the following statements, or provide a counter-example. Throughout assume that $f(\boldsymbol{x})$ is **convex** on $\mathbb{F}^N$.
Optional: in cases where the answer is False, try to find a small modification of the statement that would make it true.
These properties of convex functions are used extensively in optimization.
(a) [0] Optional. If $g(\boldsymbol{x})$ is convex, then $h(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + g(\boldsymbol{x})$ is convex. (sum)
(b) [0] Optional. If $g(\boldsymbol{x})$ is **strictly convex**, then $h(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + g(\boldsymbol{x})$ is strictly convex. (sum)
(c) [3] If $b \in \mathbb{R}$ and $\boldsymbol{v} \in \mathbb{F}^N$, then $f(\boldsymbol{x}) + \text{real}\{\boldsymbol{v}'\boldsymbol{x}\} + b$ is convex. (affine)
(d) [3] If $\alpha \in \mathbb{R}$ then $\alpha f(\boldsymbol{x})$ is convex. (scaling)
(e) [3] If $\boldsymbol{A} \in \mathbb{F}^{M\times N}$ then $g(\boldsymbol{y}) \triangleq f(\boldsymbol{A}'\boldsymbol{y})$ is convex on $\mathbb{F}^M$.
(f) [3] If g is strictly convex on $\mathbb{F}^N$ and if $\boldsymbol{A} \in \mathbb{F}^{N\times N}$, then $h(\boldsymbol{x}) \triangleq g(\boldsymbol{A}\boldsymbol{x})$ is strictly convex on $\mathbb{F}^N$.
(g) [3] Any **norm** $\|\cdot\|$ for $M \times N$ matrices is convex on $\mathbb{F}^{M\times N}$.
(h) [3] If $\psi : \mathbb{F} \to \mathbb{R}$ is convex, then the **additively separable** function $R(\boldsymbol{x}) \triangleq \boldsymbol{1}_N' \, \psi_.(\boldsymbol{x})$ is convex on $\mathbb{F}^N$.
(i) [3] Both $\|\boldsymbol{x}\|_2^2$ and $\|\boldsymbol{x}\|_2$ are **strictly convex** on $\mathbb{F}^N$.
(j) [3] If $g : \mathbb{R} \mapsto \mathbb{R}$ is convex and non-decreasing, then $h(\boldsymbol{x}) = g(f(\boldsymbol{x}))$ is convex. (composition)

---

6. [12] The $N \times N$ **finite-difference** matrix with **periodic boundary conditions** looks like

$$\boldsymbol{D}_N = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \dots & \\ 0 & \dots & 0 & -1 & 1 \\ 1 & 0 & \dots & 0 & -1 \end{bmatrix}.$$

(a) [3] Determine the **eigenvalues** of $\boldsymbol{D}_N$. Hint. This matrix is **circulant**.
(b) [3] Determine the **spectral radius** $\rho(\boldsymbol{D}_N)$ when N is even.
(c) [3] For 2D images of size $M \times N$, the finite difference matrix of interest is

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{I}_M \otimes \boldsymbol{D}_N \\ \boldsymbol{D}_M \otimes \boldsymbol{I}_N \end{bmatrix}$$

and the corresponding **Gram matrix** is

$$\boldsymbol{T}'\boldsymbol{T} = (\boldsymbol{I}_M \otimes (\boldsymbol{D}_N'\boldsymbol{D}_N)) + ((\boldsymbol{D}_M'\boldsymbol{D}_M) \otimes \boldsymbol{I}_N) = (\boldsymbol{D}_M'\boldsymbol{D}_M) \oplus (\boldsymbol{D}_N'\boldsymbol{D}_N),$$

where $\otimes$ denotes the **Kronecker product** and $\oplus$ denotes the **Kronecker sum**.
Determine the eigenvalues of this Gram matrix.
(d) [3] Determine the **spectral norm** $\|\boldsymbol{T}'\boldsymbol{T}\|_2$ of this Gram matrix when N is even. Hint. $\boldsymbol{T}'\boldsymbol{T}$ is **Hermitian symmetric**.

7. [13]

For a $M \times N$ matrix $\boldsymbol{A}$ having rank r, the following three SVD types are of interest.

- The **compact SVD** $\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r' = \sum_{k=1}^{r} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k'$ where $\boldsymbol{U}_r$ is $M \times r$, $\boldsymbol{\Sigma}_r$ is $r \times r$, and $\boldsymbol{V}_r$ is $N \times r$.
- The **economy SVD** $\boldsymbol{A} = \boldsymbol{U}_K \boldsymbol{\Sigma}_K \boldsymbol{V}_K' = \sum_{k=1}^{K} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k'$ where $K = \min(M, N)$.
  In particular, if $\boldsymbol{A}$ is tall, *i.e.*, $M \geq N$, then $\boldsymbol{A} = \boldsymbol{U}_N \boldsymbol{\Sigma}_N \boldsymbol{V}'$.
- The **full SVD** $\boldsymbol{A} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}'$ where $\boldsymbol{U}$ is $M \times M$, $\boldsymbol{\Sigma}$ is $M \times N$, and $\boldsymbol{V}$ is $N \times N$.

The built-in JULIA `svd` command does not quite provide the matrix form of any of these three options, but it is fairly easy to use the output of `svd` to generate the pieces $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$ needed.

(a) [10] Write a JULIA function `svdgen` that computes the user-specified SVD of its input matrix. An optional second argument specifies the type of interest, using one the three symbols `:compact`, `:economy` or `:full`.
The output should be three *matrices* of appropriate sizes that the user can multiply together to regenerate $\boldsymbol{A}$.
Do not call the `rank` function because that in turn would call `svd` again, which would be inefficient.
Your file should be named `svdgen.jl` and should contain the following function:

```
"""
    U,Sigma,V = svdgen(A, how::Symbol = :economy)

In
* `A` `M x N` matrix
* `how` one of `:compact` `:economy` or `:full`

Out
The sizes of `U, Sigma, V` depend on which type of SVD,
but in every case `U*Sigma*V'` should be the same as `A`
to within numerical precision.
"""
function svdgen(A, how::Symbol = :economy)
```

Submit your solution to mailto:eecs556@autograder.eecs.umich.edu.

(b) [3] Also submit a screen shot of your solution to gradescope so that graders can check it for efficiency.

8. [3] Consider the **constrained optimization** problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{C}} \Psi(\boldsymbol{x}),$$

where $\Psi(\boldsymbol{x})$ is a **convex** function whose gradient is L-Lipschitz smooth, and $\mathcal{C} \subset \mathbb{R}^N$ is a convex set.

The **gradient projection** algorithm for this problem is

$$\boldsymbol{x}_{k+1} = \mathcal{P}_{\mathcal{C}} \left(\boldsymbol{x}_k - \frac{1}{L} \nabla \Psi(\boldsymbol{x}_k) \right),$$

where $\mathcal{P}_{\mathcal{C}}(\cdot)$ denotes the **projection** of its argument onto $\mathcal{C}$.

Consider the **box constraint** or **bound constraint** where $\mathcal{C} = \left\{ \boldsymbol{x} \in \mathbb{R}^N \,:\, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \right\}$.

Write a one-line JULIA function that computes the projection onto this set $\mathcal{C}$.

9. [16] One important data-driven regularization method involves learning sparsifying transforms. One step needed for learning such transforms involves solving the following optimization problem:

$$\hat{\boldsymbol{Q}} = \underset{\boldsymbol{Q} \in \mathcal{V}_K(\mathbb{F}^M)}{\arg\min} \sum_{n=1}^{N} \|\boldsymbol{Q}\boldsymbol{x}_n - \boldsymbol{z}_n\|_2^2 ,$$

where $\mathcal{V}_K(\mathbb{F}^M)$ denotes the **Stiefel manifold** of $M \times K$ matrices having orthonormal columns, $\{\boldsymbol{x}_n\}$ with $\boldsymbol{x}_n \in \mathbb{F}^K$ denotes extracted signal or image patches and $\{\boldsymbol{z}_n\}$ with $\boldsymbol{z}_n \in \mathbb{F}^M$ denotes sparse coefficients.
Assume $N \geq M \geq K$ throughout.

(a) [3] Write an expression for $\hat{\boldsymbol{Q}}$ in terms of an SVD involving the matrices $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_N \end{bmatrix}$ and $\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}_1 & \dots & \boldsymbol{z}_N \end{bmatrix}$.
Hint: this is a generalized Procrustes problem considered in the F18 EECS 551 course notes.

(b) [3] Is the solution $\hat{\boldsymbol{Q}}$ unique? Explain why or why not.
Hint. Consider separately some cases depending on the rank of a certain matrix involved in computing the solution.

(c) [10] Write a JULIA function `stief1` that has inputs $\boldsymbol{X}$, and $\boldsymbol{Z}$ and returns the corresponding $\hat{\boldsymbol{Q}}$.
Note that the last character is the digit 1 not the letter l.
Your file should be named `stief1.jl` and should contain the following function:

```
"""
    Q = stief1(X, Z)

In:
* `X` ``K \\times N`` matrix
* `Z` ``M \\times N`` matrix, where ``M \\geq K``

Out:
* `Q` ``M \\times K`` matrix with orthonormal columns,
i.e., a matrix in the Stiefel manifold ``V_K(C^M),``
that solves
``\\arg \\min_Q \\sum_{n=1}^N \\|Q X[:,n] - Z[:,n]\\|_2^2 .``

This is a generalized version of the Procrustes problem.
"""
function stief1(X, Z)
```

Submit your solution to mailto:eecs556@autograder.eecs.umich.edu.

Non-graded problems

The following problems also review important concepts from EECS 551 but are essentially identical to homework problems assigned in a previous EECS 551 term, so they will not be graded here. We will provide solutions for self check and future reference. Exam problems and future HW problems may use material from these problems.

---

10. [0] This problem constructs a simple and practical **diagonal majorizer** for use in iterative algorithms.

A $n \times n$ square matrix $\boldsymbol{H}$ is called **diagonally dominant** iff $|h_{ii}| \geq \sum_{j \neq i} |h_{ij}|$ for $i = 1, \ldots, n$.

(a) Use the **Gershgorin disk theorem** to prove that if $\boldsymbol{H}$ is a (Hermitian) symmetric matrix that is diagonally dominant with $h_{ii} \geq 0$ for all i, then $\boldsymbol{H} \succeq \boldsymbol{0}$.

(b) Prove that if $\boldsymbol{B}$ is a $n \times n$ Hermitian symmetric matrix, then $\boldsymbol{D} \triangleq \mathsf{diag}\{|\boldsymbol{B}| \boldsymbol{1}_n\} \succeq \boldsymbol{B}$ where $|\boldsymbol{B}|$ denotes the matrix consisting of the absolute values of the elements of $\boldsymbol{B}$, *i.e.*, `abs.(B)` and $\boldsymbol{1}_n$ is `ones(n)` in JULIA.

(c) Prove that if $\boldsymbol{B} = \boldsymbol{A}'\boldsymbol{A}$, then $\boldsymbol{B} \preceq \|\boldsymbol{A}\|_2^2 \boldsymbol{I}_n$. Assume $\boldsymbol{A}$ is $m \times n$ here and below.

(d) Compare $\|\boldsymbol{A}\|_2^2 \boldsymbol{I}$ and $\boldsymbol{D}$; is either of these a tighter majorizer for $\boldsymbol{B} = \boldsymbol{A}'\boldsymbol{A}$?
(If $\boldsymbol{B} \preceq \boldsymbol{D}_1$ and $\boldsymbol{B} \preceq \boldsymbol{D}_2$, then we say $\boldsymbol{D}_1$ is the tighter majorizer for $\boldsymbol{B}$ if $\boldsymbol{D}_1 \preceq \boldsymbol{D}_2$.)

(e) Prove that if $\boldsymbol{B} = \boldsymbol{A}'\boldsymbol{A}$, then $\boldsymbol{B} \preceq \boldsymbol{E} \triangleq \mathsf{diag}\left\{|\boldsymbol{A}|' \, |\boldsymbol{A}| \, \boldsymbol{1}_n\right\}$. Hint: use triangle inequality.

(f) Prove that $\boldsymbol{D} \preceq \boldsymbol{E}$ so $\boldsymbol{D}$ is a tighter majorizer than $\boldsymbol{E}$.

(g) Prove that $\boldsymbol{A}'\boldsymbol{A} \preceq \|\boldsymbol{A}\|_1 \|\boldsymbol{A}\|_\infty \boldsymbol{I}_n$.

---

11. [0]

(a) Show that if $\boldsymbol{W}$ is a $N \times N$ **positive definite matrix**, then the weighted Euclidean norm $\|\boldsymbol{x}\|_{\boldsymbol{W}} = \sqrt{\boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}}$ is a valid **norm** on $\mathbb{F}^N$.

(b) Now show the converse, *i.e.*, if $\|\boldsymbol{x}\|_{\boldsymbol{W}} = \sqrt{\boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}}$ is a valid **norm** on $\mathbb{F}^N$, and $\boldsymbol{W}$ is (Hermitian) symmetric, then $\boldsymbol{W} \succ \boldsymbol{0}$.

---

12. [0] This problem examines import properties of **positive semidefinite** and **positive definite** matrices.

Recall that $\boldsymbol{B}'\boldsymbol{B} \succeq \boldsymbol{0}$ for any matrix $\boldsymbol{B}$.

(a) Show that $\boldsymbol{B}'\boldsymbol{B} \succ \boldsymbol{0}$ if $\boldsymbol{B}$ has **full column rank**.

(b) Show that $\boldsymbol{A} \succ \boldsymbol{0}$ implies $\boldsymbol{A}$ is invertible.

(c) Show $\boldsymbol{A} \succeq \boldsymbol{0}$ and $\boldsymbol{B} \succeq \boldsymbol{0} \Longrightarrow \boldsymbol{A} + \boldsymbol{B} \succeq \boldsymbol{0}$.

(d) $\boldsymbol{A} \succ \boldsymbol{0}$ and $\boldsymbol{B} \succeq \boldsymbol{0} \Longrightarrow \boldsymbol{A} + \boldsymbol{B} \succ \boldsymbol{0}$.

(e) Show $\boldsymbol{A}'\boldsymbol{A} + \boldsymbol{B}'\boldsymbol{B}$ is invertible if $\boldsymbol{B}$ has full column rank
(This property is relevant to **regularized least-squares** problems.)

(f) Show $\boldsymbol{A}'\boldsymbol{A} + \boldsymbol{B}'\boldsymbol{B}$ is invertible if $\mathcal{N}(\boldsymbol{A}) \cap \mathcal{N}(\boldsymbol{B}) = \{\boldsymbol{0}\}$, i.e., if $\boldsymbol{A}$ and $\boldsymbol{B}$ have disjoint **null spaces** other than $\boldsymbol{0}$.

(g) Show that $\boldsymbol{A} \succ \boldsymbol{0},\ \boldsymbol{B} \succ \boldsymbol{0} \Longrightarrow \boldsymbol{B}\boldsymbol{A}\boldsymbol{B} \succ \boldsymbol{0}$.

---

13. [0]

(a) Show that for arbitrary (possibly complex-valued) matrices of compatible sizes:

$$\mathsf{vec}\left(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T\right) = (\boldsymbol{B} \otimes \boldsymbol{A}) \, \mathsf{vec}(\boldsymbol{X}), \tag{1}$$

where $\mathsf{vec}(.)$ is the operator that stacks the columns of the input matrix into a vector, and $\otimes$ denotes the **Kronecker product**. Note that here we really mean transpose $\boldsymbol{B}^T$ even if $\boldsymbol{B}$ is complex valued!

(b) One application of (1) is computing a 2D **discrete Fourier transform** (DFT). The (1D) DFT of a vector $\boldsymbol{x} \in \mathbb{C}^N$ is the vector $\boldsymbol{v} \in \mathbb{C}^N$ with entries

$$v_k = \sum_{n=1}^{N} x_n \exp(-\imath 2\pi (k-1)(n-1)/N), \quad k = 1, \ldots, N.$$

(Here we are using the linear algebra (and JULIA) convention where the first array index is 1, whereas DSP books usually use $0, \ldots, N-1$.) We can represent the above computation in matrix-vector form as $\boldsymbol{v} = \boldsymbol{F}_N \boldsymbol{x}$, where $\boldsymbol{F}_N$ is the $N \times N$ **DFT matrix** with entries

$$(\boldsymbol{F}_N)_{kn} = \exp(-\imath 2\pi (k-1)(n-1)/N).$$

If needed (though rarely is it needed), we can generate $\boldsymbol{F}_N$ in JULIA as follows.

```
using FFTW
using LinearAlgebra
F = fft(Matrix(I, N, N), 1)
```

One can verify that $\boldsymbol{F}_N' \boldsymbol{F}_N = \boldsymbol{F}_N \boldsymbol{F}_N' = N \boldsymbol{I}_N$, so $\boldsymbol{F}_N^{-1} = \frac{1}{N} \boldsymbol{F}_N'$ and the (1D) **inverse DFT** of $\boldsymbol{v}$ can be computed as $\boldsymbol{x} = \frac{1}{N} \boldsymbol{F}_N' \boldsymbol{v}$.

Now in two dimensions, we compute the **2D DFT**, call it $\boldsymbol{V}$, of the $M \times N$ matrix $\boldsymbol{X}$, by computing the 1D DFT of each column of $\boldsymbol{X}$ followed by the 1D DFT of each row of the result (or vice versa).
Explain why the following expression computes the 2D DFT of $\boldsymbol{X}$:

$$\boldsymbol{V} = \boldsymbol{F}_M \boldsymbol{X} \boldsymbol{F}_N^T.$$

(c) Use (1) to write $\mathsf{vec}(\boldsymbol{V})$ as the product of a matrix and $\mathsf{vec}(\boldsymbol{X})$.

(d) Show that

$$\boldsymbol{X} = \frac{1}{MN} \boldsymbol{F}_M' \boldsymbol{V} \overline{\boldsymbol{F}_N}$$

computes the **2D inverse DFT** of $\boldsymbol{V}$, where $\overline{\boldsymbol{Y}}$ denotes (element-wise) complex conjugate.

(e) Use (1) to write $\mathsf{vec}(\boldsymbol{X})$ as the product of a matrix and $\mathsf{vec}(\boldsymbol{V})$.