# MATH 2138: LINEAR ALGEBRA

## Linear Algebra in Computer Science Proposal

**Course:** Math 2138 – Linear Algebra

**Group Members:** Braulio, Alessandra Veronica, Malicsi, Ivan Louie, Padillo, Catherine Mae, Valdepeñas, Sarah Marie

**Date Submitted:** October 31, 2025

**Proposal Title 1:** Hill Cypher Encryption and Decryption using Matrices

## I. Introduction and CS Context

Considering the present constantly evolving cybersecurity threat landscape, cryptography is a crucial aspect that has become critical to protecting sensitive information. Cryptography, derived from the Greek words "kryptos" meaning concealed and "grapho" meaning to write, is both an art and a science, easily transferring from ancient methods to the digital age. In 1929, Lester S. Hill developed the Hill cipher, which is a notable development in traditional cryptography. In order to encrypt and decode communications, it makes use of the concepts of linear algebra, particularly matrix operations. When compared to monoalphabetic ciphers, this cipher is especially secure because it can encrypt many letters at once (*Decrypting the Hill cipher: Learn with a 3×3 matrix inverse, n.d*).

Through the application of algebraic structures in practical computing settings like data encoding and cybersecurity, the Hill Cipher demonstrates the essential links between mathematics and computer science. The cipher's reliance on modular arithmetic, determinants, and matrix operations makes it a great example of how computational algorithms do encryption well (Darji, 2024). Furthermore, because block ciphers and symmetric-key systems both use comparable linear transformations, an understanding of the Hill Cipher makes it easier to understand contemporary cryptographic concepts.

The goal of this project is to use Python's linear algebra techniques to model and implement the encryption and decryption process of the Hill Cipher. In doing so, it illustrates how the encoding and decoding of secure signals are directly made possible by mathematical operations like matrix multiplication and modular inverses. This project's main objective is to create a Python-based application that can validate key matrices for invertibility under modular arithmetic, perform secure message encryption and decryption using the Hill Cipher algorithm, and demonstrate how the mathematical underpinnings of data encryption are derived from linear algebraic concepts, specifically matrix operations and modular inverses.

To confirm that the encryption and decryption procedures were done correctly, the project's output will contain both the encrypted ciphertext and the decoded plaintext. To further clarify how linear algebra operations are used within the Hill Cipher algorithm to convert plaintext into ciphertext and back, it will also optionally show intermediate results that demonstrate the matrix-vector multiplication phases.

## II. Mathematical Background

The cipher operates within the modular arithmetic system $\mathbb{Z}_m$, where *m* typically equals 26 to represent the English alphabet. Each plaintext message is divided into fixed-size blocks that can be expressed as vectors. A square key matrix $K \in \mathbb{Z}_m^{n\text{x}n}$ performs a linear transformation on these vectors using matrix multiplication, resulting in the ciphertext. For decryption to be valid, $K$ must be invertible under modulo *m*, meaning gcd(det(K), m) = 1. This condition ensures that the determinant of $K$ has a multiplicative inverse, allowing computation of $K^{-1}$ for decryption (Kolman & Hill, 2013).

The basic ideas of linear independence and basis in vector spaces are directly related to the invertibility of a matrix in modular arithmetic and the nonzero condition of its determinant (Kolman and Hill, 2013). These characteristics, when applied to the Hill Cipher, ensure that each distinct block of plaintext matches a distinct block of ciphertext, preserving the one-to-one mapping necessary for safe communication.

Additionally, the Hill Cipher is a prime example of how cryptographic transformations can be modeled using systems of linear equations. Cryptanalysts can create a modular set of equations to solve for the unknown key matrix given many plaintext–ciphertext combinations $K$. Therefore, the implementation and analysis of the technique depend on the ability to solve linear systems over finite fields (Vijayaraghavan et al., 2018).

## III. Mathematical Formulation

The Hill Cipher encrypts plaintext using an invertible matrix as a key. Each letter is assigned a numerical value ($A = 0, B = 2,..., Z = 25$). The plaintext is divided into vectors that match the key's dimension, then multiplied by the key matrix modulo 26:

$$C = (K\,x\,P)\;mod\;26$$

Where:

- $K$ = key matrix (must have a nonzero determinant modulo 26)
- $P$ = plaintext vector
- $C$ = ciphertext vector

To decrypt, the inverse of the key matrix $modulo\;26$ is used:

$$P = (K^{-1}\,x\,C)\;mod\;26$$

If $det\,(K)$ is coprime with 26, then $K$ has a valid modular inverse.

The invertibility condition $gcd(det(k),\,m) = 1$ ensures that **K$^{-1}$** exists in $\mathbb{Z}_m^{nxn}$. Kolman and Hill (2013) describe this as a direct consequence of the properties of determinants and matrix inverses—foundational linear algebra concepts that the Hill Cipher leverages in a modular arithmetic setting.

## IV. Implementation Plan

### Data Representation

Vectors and matrices will be represented using **NumPy arrays**, as this library provides efficient handling of numerical operations and modular arithmetic. Each letter in the plaintext will be mapped to an integer value between 0 and 25 ($A = 0,\; B = 1,\; ...,\; Z = 25$). The plaintext will then be divided into fixed-size blocks corresponding to the dimensions of the key matrix $K \in \mathbb{Z}_m^{nxn}$, where $m = 26.X$

### Algorithmic Steps (Pseudocode)

1. **Input Key and Plaintext**
    - Convert the key (string) into a numeric matrix K.
    - Convert the plaintext into numeric vectors.
2. **Encryption**
    - For each block vector P, compute the ciphertext vector C using C = (K×P) mod 26.

- Convert *C* back to letters to form the ciphertext string.

3. **Decryption**
   - Compute the modular inverse of K, denoted $K^{-1}$, ensuring gcd(det(K), 26) = 1.
   - Each ciphertext block C, compute the plaintext vector P using $P=(K^{-1}\times C) \bmod 26$.
   - Convert P back to letters to reconstruct the original message.

```
None
INPUT: Plaintext string P, Key matrix K (n x n)
OUTPUT: Ciphertext string C (and optionally decrypted plaintext)

1. Convert each letter in P to a number (A=0, B=1, ..., Z=25)
2. Pad the numeric list so its length is divisible by n
3. For each block of n numbers:
    a. Multiply block vector by key matrix K modulo 26 → encrypted block
    b. Append encrypted block numbers to ciphertext list
4. Convert ciphertext numbers back to letters → C
5. (Optional) Decrypt:
    a. Compute modular inverse of key matrix K modulo 26 → K_inv
    b. For each block of ciphertext numbers:
        i. Multiply block by K_inv modulo 26 → decrypted block
    c. Convert decrypted numbers back to letters → decrypted plaintext
```

The program is expected to accept a square key matrix, entered either as a string or a list of numbers, along with a plaintext string containing letters only. The output will include the encrypted ciphertext string, with the decrypted plaintext optionally displayed for verification purposes. To ensure correctness, the implementation will be tested using known key–plaintext pairs from standard Hill Cipher examples in the literature (Kolman & Hill, 2013; Stallings, 2017). Testing will confirm that decryption accurately reverses encryption for all cases, that non-invertible matrices where $gcd(det(K), 26) \neq 1$ are properly rejected, and that the computed inverse matrix $K^{-1}$ matches manually verified modular inverses, thereby ensuring mathematical accuracy.

## V. References

Darji, D. (2022, May 29). *Linear algebra-hill cipher*. Medium. https://medium.com/@darjidevendra1/linear-algebra-hill-cipher-fbae5b1fa6f1

*Decrypting the Hill cipher: Learn with a 3×3 matrix inverse*. ExamCollection. (n.d.). https://www.examcollection.com/blog/decrypting-the-hill-cipher-learn-with-a-3x3-matrix-inverse/

Kolman, B., & Hill, D. R. (2013). *Elementary Linear Algebra* (9th ed.). Pearson Education.

Patel, P. (2025, September 4).*The crucial role of cryptography in cybersecurity*. eInfochips.https://www.einfochips.com/blog/the-crucial-role-of-cryptography-in-cybersecurity

Vijayaraghavan, N., Narasimhan, S., & Baskar, M. (2018). A study on the analysis of Hill's cipher in cryptography. *International Journal of Mathematics Trends and Technology, 54*(7), 519–522.

**Proposal Title 2: Image Enhancement Using Matrices**

## I. Introduction and CS Context

Raw digital photos may have low contrast, uneven illumination, noise, and other quality problems in a variety of applications, ranging from smartphone photography to medical imaging. The usefulness of computer vision, diagnostics, and general imaging workflows is improved by enhancing such pictures to make key elements more visible (Lozano-Vázquez et al., 2022; Joshi et al., 2025). The goal of this research is to use linear-algebraic matrix operations to model and implement picture improvement. To improve the quality of a picture, we will specifically consider it as a matrix (or set of matrices for color channels) and perform decompositions and transformations (such as scaling, filtering, and singular value decomposition) (Jain, 2024).

Given that image processing incorporates ideas from linear algebra, data structures, algorithms, and signal processing, this subject is extremely pertinent to computer science. For real-time systems, such mobile cameras and medical scanners, where usability is directly impacted by speed and accuracy, effective matrix-based solutions are essential. This project aims to create a Python-based program that can translate digital images into one or more matrices, apply matrix operations to enhance the quality of the images, such as filtering, contrast scaling, and singular value decomposition (SVD)-based enhancement, and output the improved images along with metrics that compare them to the original. The improved image file for either grayscale or RGB inputs, numerical indicators of quality improvement like contrast enhancement and signal-to-noise ratio, and optional intermediate results showing matrix transformations or SVD components to show the underlying linear algebra operations are all expected outputs of the program.

## II. Mathematical Background

In **Vector** and **Matrix Representation**, a digital grayscale image $m\,x\,n$ can be represented as matrix $I \in \mathbb{R}^{mxn}$. For color image, three channel matrices $I_R$, $I_G$, $I_B$ are similarly defined.

**Scaling and Linear Transformation:** To adjust contrast or brightness, one can apply:

$$I = \alpha I + \beta$$

where α (gain) and β offset) are scalars, and 1 is the all-ones matrix. This is a linear combination of the matrix and a constant matrix.

**Filtering via Convolution as Matrix Multiplication:** A smoothing or sharpening filter can be implemented by multiplying or convolving with a kernel, which in matrix form is often represented via Toeplitz or circulant matrices. Filtering is thus a linear transformation of the image vector space.

**Singular Value Decomposition (SVD) and Eigenanalysis:** The SVD of an image matrix,

$$I = U \Sigma V^T$$

decomposes it into orthogonal (or unitary) matrices U,V and a diagonal matrix Σ of singular values. By modifying or truncating Σ, one can enhance prominent features or reduce noise. The concept of eigenvalues and eigenvectors is similar: they capture fundamental modes of the image matrix which can be used to filter, denoise or enhance.

**Inner Products and Orthogonality:** Inner products measure similarity between image vectorized patches or between singular vectors; orthogonality ensures that the basis vectors carry independent information—useful when we extract dominant singular vectors for enhancement.

**Linear Systems and Optimization:** Some enhancement techniques solve linear systems (e.g., deblurring via $Ax = b$, or optimize contrast subject to constraints. Matrix inversion or pseudo-inverse solutions are used, providing direct ties to linear algebra.

## IV. Mathematical Formulation

In this project, the input image is represented as a matrix $I \in \mathbb{R}^{mxn}$ for grayscale images, with contrast gain $\alpha \in \mathbb{R}$ and brightness offset $\beta \in \mathbb{R}$ applied to adjust pixel intensities. A filter kernel $K \in \mathbb{R}^{kxk}$ is used for smoothing or sharpening operations, which can be implemented as a convolution $I_2 = K * I_1$ or equivalently in matrix form as $vec(I_2) = M\, vec(I_1)$, where $M$ is the transformation matrix corresponding to the convolution. For SVD-based enhancement, the input matrix $I$ is decomposed as $I = U \Sigma V^T$, where $U$ and $V$ are orthogonal matrices and $\Sigma$ contains singular values. By keeping the top-r singular values, we reconstruct the enhanced image as $I_{e}$▯▯ $= U\Sigma r V^T$ which either reduces noise by zeroing out smaller singular values or emphasizes features by amplifying larger ones.

All enhancement processes, such as contrast adjustment, filtering, and SVD decomposition, can be handled as linear algebra operations when an image is represented as a matrix. Effective matrix multiplication, vectorization, and readily available Python libraries like NumPy and SciPy are necessary for algorithmic implementation. The selection of singular values and the choice of $r$ are related to feature extraction and dimensionality reduction, which are essential methods in computer vision and image processing (Namdev & Lal, 2023).

## V. Implementation Plan

In this project, images will be read using Python libraries such as OpenCV or Pillow and represented as NumPy arrays. Grayscale images will be stored as two-dimensional arrays $I \in \mathbb{R}^{mxn}$, while RGB images will be represented by three separate channel matrices $I_R$, $I_G$, $I_B$. The filter kernel K will be a small two-dimensional NumPy array of size , and SVD components U, Σ, and V will be computed using numpy.linalg.svd.

```
None
Input: image_file, parameters α, β, kernel K, truncation rank r
Read image → I
If RGB: split into R,G,B channels
Compute I1 = α * I + β * ones
Compute I2 = convolve(I1, K)
Compute U, Sigma, Vt = svd(I2)
Modify Sigma → Sigma_r (keep top-r singular values)
Reconstruct I_enh = U @ Sigma_r @ Vt
Clip I_enh values to valid range (e.g., 0-255)
Convert back to image format and save/output
Compute and display metrics: e.g., contrast improvement, PSNR, SSIM
```

The program will accept as input an image file (such as .png or .jpg), numeric parameters including contrast scaling (α), brightness offset (β), and SVD truncation rank (r), along with an optional kernel file or a default kernel. The output will consist of an enhanced image file in the same format as the input, accompanied by a printed report containing numerical metrics such as contrast ratio, PSNR, and SSIM, and optionally, a side-by-side comparison of the original and enhanced images. To verify correctness and effectiveness, standard test images such as “Lens” and “Cameraman” will be used with known parameter settings, allowing both visual inspection and quantitative evaluation of metric improvements. Contrast enhancement will be confirmed by comparing histograms of unmodified versus enhanced images. The SVD truncation will be validated by ensuring that the Frobenius norm $|| I_2 - I_{e}$⍰⍰$||F$ decreases or behaves as expected when varying rrr. Edge cases, including extremely dark or bright images, non-square images, and unusual kernel sizes (such as 1×1 or large k), will be tested to ensure robustness. Additionally, performance tests will be conducted by measuring execution time for increasingly large images to confirm the program's efficiency for practical use.

## V. References

Jain, R. (2024, February 14). *Image transformation with linear algebra*. Medium. https://medium.com/@riya.jain.2026/image-transformation-with-linear-algebra-e2f2e8e7cdf3

Joshi, S. V., Saoji, S., Patil, S. V., Dhabliya, D., & Gandhi, Y. (2025). Application of linear algebra in image processing for medical electronics. *Communications on Applied Nonlinear Analysis, 32*(1). https://internationalpubls.com/index.php/cana/article/view/1634

Lozano-Vázquez, V., Miura, J., Rosales-Silva, A. J., Luviano-Juárez, A., & Mújica-Vargas, D. (2022). Analysis of different image enhancement and feature extraction methods. *Mathematics, 10*(14), 2407. https://doi.org/10.3390/math10142407

Namdev, A., & Lal, N. (2023). Image enhancement based on histogram equalization with linear perception neural network method. *International Journal of Advanced Computer Technology*. https://ijact.org/index.php/ijact/article/view/139

**Proposal Title 3:** Music Genre Recognition using Matrix Decomposition

## I. Introduction and CS Context

Music genre recognition is a process that automatically identifies the category of a song, such as pop, rock, jazz, or classical, based on its audio features. It plays a major role in applications such as music recommendation systems, playlist organization, and automated tagging on streaming platforms (Müller, 2015). This project explores how matrix decomposition, a fundamental concept in linear algebra, can be used to extract meaningful patterns from complex audio data and improve classification accuracy.

This problem is relevant to computer science because it integrates machine learning, signal processing, and mathematical modeling. Through matrix decomposition techniques, large and complex audio datasets can be simplified into smaller, interpretable components that reveal the underlying structure of musical signals (Klein, 2013). By combining computational and mathematical approaches, this project demonstrates how core linear algebra principles can contribute to data-driven solutions in artificial intelligence.

The goal of this project is to develop a Python-based system that applies matrix decomposition methods to classify music by genre. The system will focus on identifying unique audio features that differentiate one genre from another by using Singular Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF) to extract the dominant components of each audio signal (Lee & Seung, 1999).

The expected output of the project is a working prototype that can take an input audio file and predict its genre based on the extracted and decomposed feature data. The system aims to demonstrate how linear algebra can be used not only as a mathematical tool but also as a foundation for real-world computer science applications in audio analysis.

## II. Mathematical Background

The application of matrix decomposition in music genre recognition relies on fundamental concepts and principles of linear algebra, particularly those involving vectors, matrices, and eigenvalue analysis. These mathematical tools enable the representation, transformation, and reduction of complex audio data into simpler, interpretable forms that highlight unique features of each genre.

A vector is a mathematical object that represents magnitude and direction. Each vector may represent a frame of sound frequencies or amplitude at a given time in the context of audio signal processing. Linear algebra makes it possible for the vectors to be combined and manipulated through operations such as linear combinations, norms, and projections.

A linear combination of vectors is defined as

$$v = c_1 v_1 + c_2 v_2 + \cdots + c_k v_k$$

which expresses a new vector as a weighted sum of existing ones. This concept is crucial in audio analysis because each segment of a sound can be considered a combination of frequency basis vectors.

The norm of a vector, which measures its magnitude or energy, is defined as:

$$|v| = \sqrt{v\frac{2}{1} + v\frac{2}{2}} + \ldots + v\frac{2}{n}$$

and is used to normalize audio signals so that each sample contributes proportionally to the overall analysis.

The projection of one vector onto another is expressed as:

$$proj_u\left(v\right) = \frac{v \cdot u}{|u|^2} u$$

which quantifies the similarity between two signals, making it useful for comparing audio feature vectors.

A matrix is a structured arrangement of numerical values organized into rows and columns, which provides a concise representation of linear transformations. Each audio signal can be represented as a spectrogram matrix, where rows correspond to frequency bands and columns represent time frames. Mathematically, this can be written as:

$$A = \left[a_{i_j}\right],\ A \,\epsilon\, \mathbb{R}^{\mathbb{m}\mathbb{x}\mathbb{m}}$$

Multiplying a matrix by a vector, as in

$$Ax$$

performs a linear transformation that maps the input space (time domain) into another space (frequency domain). These transformations perform similarly to scaling, rotation, or reflection, which correspond to emphasizing or suppressing specific audio features. This property allows matrix decomposition methods such as Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) to identify the dominant structures within the spectrogram data.

The concepts of eigenvalues and eigenvectors are important in PCA and other dimensionality reduction methods. An eigenvector of a matrix

$$\mathrm{Ax} = \lambda\mathrm{x}$$

where $\lambda$ is the eigenvalue. The eigenvalue represents the amount of variance or “energy” preserved along the direction of its eigenvector. Eigenvectors represent distinctive frequency patterns that identify the key variations among different genres, whereas eigenvalues indicate the magnitude or significance of these patterns (Kolman & Hill, 2014). Preserving only the top eigenvalues enables the system to reduce dimensionality, filtering out noise while maintaining essential musical characteristics.

Another important concept is the inner product (or dot product), which measures the similarity between two vectors:

$$u \cdot v = \sum_{i=1}^{n} u_i v_i$$

Two vectors are orthogonal if their inner product equals zero, implying that they carry independent information. In SVD, the matrices $U\, and\, V$ contain orthogonal vectors that represent independent frequency and temporal patterns, respectively. This orthogonality ensures that the extracted features are not redundant, enhancing the performance of classification algorithms (Kolman & Hill, 2014).

Finally, the decomposition process involves solving linear systems and performing optimization to minimize reconstruction error. A linear system can be expressed as

$$Ax = b$$

where $A$ represents a data matrix and x\mathbf{x}x contains the coefficients or features to be determined. In SVD, the decomposition

$$A = U\Sigma V^T$$

approximates the original matrix $A$ using a reduced number of singular values. This low-rank approximation minimizes the error

$$||A - \tilde{A}||_F$$

known as the Frobenius norm, which measures the difference between the original and reconstructed spectrograms (Kolman & Hill, 2014). With this process, the algorithm effectively identifies and preserves the latent structure of audio data, which allows the system to classify genres based on mathematical patterns rather than subjective assessment.

Linear Algebra concepts such as vector operations, matrix transformations, eigenvalue analysis, orthogonality, and linear systems collectively serve as a mathematical foundation for music genre recognition. With these concepts and principles, the essential components of complex audio signal data can be extracted, compressed, and classified, illustrating how mathematical abstractions translate into intelligent audio understanding and genre prediction.

## III. Mathematical Formulation

This project applies matrix decomposition techniques—specifically Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)—to extract dominant musical features from audio data. Let the spectrogram of an audio signal be represented by a matrix $A \in \mathbb{R}^{mxn}$ where each row corresponds to a frequency band and each column corresponds to a time frame. Using SVD, the matrix can be factorized as $A = U \Sigma V^T$, where U is an orthogonal matrix containing the left singular vectors that represent frequency-based features, Σ is a diagonal matrix whose singular values capture the energy or significance of each component, and V is an orthogonal matrix containing the right singular vectors that describe temporal patterns in the music. By retaining only the top k singular values, we obtain a rank-k approximation $A_k = U_k \Sigma_k V_k^T$ which preserves the essential structure of the audio signal while filtering out noise and redundant information.

The same decomposition can be understood in PCA terms as projecting the high-dimensional spectrogram data into a lower-dimensional subspace that is spanned by the principle components, or the data with the biggest eigenvalues. This captures the greatest amount of variance in the data. By reducing its dimensionality, the raw audio signal is converted into a condensed feature representation that can be used for genre recognition and classification. The technique demonstrates how linear algebra facilitates effective feature extraction and pattern recognition in computational music analysis by showcasing the mathematical capabilities of eigenvalues, orthogonal transformations, and vector spaces in revealing latent structures inside big datasets.

## IV. Implementation Plan

The proposed implementation will be developed in Python, utilizing libraries such as **NumPy**, **Librosa**, and **scikit-learn** for matrix computation, audio processing, and machine learning. The process begins with loading and preprocessing audio files by converting each into a **spectrogram matrix** using Librosa’s stft() (short-time Fourier transform) function, which transforms the time-domain signal into its time–frequency representation (Müller, 2015). These spectrograms are stored as NumPy arrays to enable efficient mathematical manipulation and decomposition. Matrix decomposition techniques—specifically **Singular Value Decomposition (SVD)** and **Non-negative Matrix Factorization (NMF)**—will then be applied to extract the most significant frequency–time patterns that characterize each genre.

```
None
Input: audio_file (.wav or .mp3)
Output: predicted_genre_label

BEGIN
  1. Load audio file using Librosa
     signal ← librosa.load(audio_file)

  2. Compute spectrogram matrix
     A ← abs(librosa.stft(signal))

  3. Normalize spectrogram
     A ← A / max(A)

  4. Apply matrix decomposition
     IF method = "SVD" THEN
        [U, Σ, Vt] ← svd(A)
        A_reduced ← U[:, 1:k] × Σ[1:k, 1:k] × Vt[1:k, :]
     ELSE IF method = "NMF" THEN
        [W, H] ← nmf(A, components = k)
        A_reduced ← W × H
     END IF

  5. Extract feature vectors from A_reduced
     features ← flatten(A_reduced)
```

```
    6. Train classifier (KNN or SVM)
       model.fit(features_train, labels_train)

    7. Predict genre for new audio input
       predicted_genre_label ← model.predict(features_test)

    8. Display results
       PRINT predicted_genre_label
       PLOT original and reconstructed spectrograms
END
```

The extracted feature matrices will form a **feature dataset** used to train a machine learning classifier such as **K-Nearest Neighbors (KNN)** or **Support Vector Machine (SVM)** to automatically recognize and predict the genre of unseen songs. The program's input will consist of an audio file in .wav or .mp3 format, and the output will be the **predicted genre label**, optionally accompanied by a visualization of the spectrogram and decomposition results.

To evaluate the model's effectiveness, a benchmark dataset such as the **GTZAN Genre Collection** will be used for both training and testing (GTZAN Dataset, n.d.). Evaluation metrics including **accuracy**, **precision**, **recall**, and **F1-score** will be computed to assess classification performance. Additionally, experiments will compare the results of SVD and NMF to determine which decomposition technique offers the best trade-off between accuracy and computational efficiency. This approach demonstrates how mathematical decomposition models can be directly implemented in Python to extract meaningful musical patterns and apply them to real-world genre recognition tasks.

## V. References

GTZAN Genre Collection Dataset. (n.d.). Retrieved from http://marsyas.info/downloads/datasets.html

Klein, P. N. (2013). *Coding the Matrix: Linear Algebra through Computer Science Applications.* Newtonian Press.

Kolman, B., & Hill, D. R. (2013). *Elementary Linear Algebra* (9th ed.). Pearson Education.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 788–791.

Müller, M. (2015). *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications.* Springer.