## 1 Using this document

I would like this to become a tutorial on the linear algebra of binary trees and its applications. In order for it to be interactive, you might find it useful to ask questions now and then.

A particularly useful way to do this in LaTeX is to use the "marginpar" command. Just insert a question in the margin at the appropriate place and as I periodically review/update the document, I'll add more text to answer the question, or find an appropriate link you can go to to learn a little more.

Ask a question here.

## 2 Prerequisites

This document will help familiarize you with the linear algebra used in creating fractal binary trees. I want to focus on trees, so here are some things from linear algebra I'll assume you are already familiar with:

1. Representation of matrices as linear transformations – such as rotations, shears, reflections, scaling axes, projections, etc. Here is a brief summary I used for my digital art course.
2. Singular and nonsingular matrices.
3. Finding eigenvalues and eigenvectors of a matrix; algebraic and geometric multiplicities of eigenvalues; defective matrices; writing a matrix as $A = PDP^{-1}$, where $D$ is a diagonal matrix of eigenvalues, and $P$ is a matrix whose columns are the corresponding eigenvectors (sometimes called the spectral decomposition).
4. Trace and determinant of a matrix.
5. Range, kernel, and rank of a matrix.
6. Basic matrix algebra.

## 3 Representation of matrices

There are many ways to classify $2 \times 2$ matrices, but in general, writing a matrix in the form $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is not particularly helpful. As we will see in later examples, it is useful to use two criteria for our classification: whether the matrix is invertible or singular, or whether it is defective or nondefective.

(IN): Invertible and nondefective. In this case, the matrix is diagonalizable, and has the form

$$[\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda_1\lambda_2 \neq 0, \tag{1}$$

where the eigenvalue $\lambda_1$ has eigenvector $\mathbf{v}_1$, and $\lambda_2$ has eigenvector $\mathbf{v}_2$. The notation "$[\mathbf{v}_1 : \mathbf{v}_2]$" means that matrix whose columns are the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Note that in this (and all subsequent cases), $\mathbf{v}_1$ and $\mathbf{v}_2$ must be linearly independent so that the matrix $[\mathbf{v}_1 : \mathbf{v}_2]$ is invertible.

Note that it is possible in this case that $\lambda_1 = \lambda_2$. In this case, the matrix is just a multiple of the identity, and takes the simpler form

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_1 \end{bmatrix} = \lambda_1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

It is also possible that $\lambda_1$ and $\lambda_2$ are complex. In this case, they are complex conjugates of each other, and the eigenvectors are also complex conjugates of each other. The simplest example of this case is the rotation matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

which has eigenvalues of $\cos\theta \pm i\sin\theta$ and corresponding eigenvectors $(\pm i, 1)$. In general, if arbitrary complex conjugates $\lambda_1, \lambda_2$ and conjugates $\mathbf{v}_1, \mathbf{v}_2$ are chosen, the matrix given by (1) will have real entries.

(ID): Invertible and defective. In this case, the matrix has the form

$$[\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda \neq 0. \tag{2}$$

This is an example of a shear. The unit square is transformed into a parallelogram whose base is the same as its height (both $\lambda$). Most matrices representing shears are not defective; this is a special case. To see why this case occurs, you can look up"Jordan normal form."

(SN): Singular, nondefective, and nonzero.

$$[\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} \lambda & 0 \\ 0 & 0 \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda \neq 0. \tag{3}$$

This is simply (1) with $\lambda_2 = 0$. The reason this case is singled out is that it is often necessary to solve matrix equations that have the form

$$AB = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = [0],$$

where we use the notation [0] for the matrix of all 0's for convenience. If one of $A$ or $B$ is invertible, then the other must be [0], which usually gives a degenerate binary tree. So in solving matrix equations, it is usually necessary to consider the singular case separately.

(SD): Singular, defective, and nonzero. In this case, the matrix has the form (2) with $\lambda = 0$ :

$$[\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda \neq 0 \tag{4}$$

This is a particular interesting case. Such a matrix $M$ satisfies $M^2 = [0]$ but $M \neq [0]$. These matrices may also be classified by the condition $\operatorname{Ker} M = \operatorname{Rng} M$. They are significant because they "prune" trees in the sense that if such a transformation is applied twice in a row, the branch ends (since $M^2 = [0]$).

(S0): This is just the zero matrix, [0]. This case is singled out, since if one of the transformations is [0], the tree is essentially a unary tree.

Summary of the different types of matrices:

$$\text{(IN)}: \quad [\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda_1\lambda_2 \neq 0. \tag{1}$$

$$\text{(ID)}: \quad [\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda \neq 0. \tag{2}$$

$$\text{(SN)}: \quad [\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} \lambda & 0 \\ 0 & 0 \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}, \quad \lambda \neq 0. \tag{3}$$

$$\text{(SD)}: \quad [\mathbf{v}_1 : \mathbf{v}_2] \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} [\mathbf{v}_1 : \mathbf{v}_2]^{-1}. \tag{4}$$

$$\text{(S0)}: \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{5}$$

## 4 Definition of a binary tree

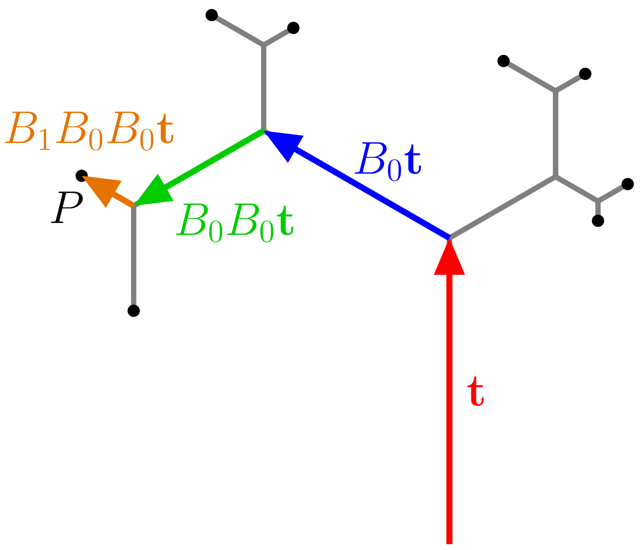
Here, we discuss various useful notations and definitions for describing binary trees. Without a rigorous definition, any mathematical analysis is not really possible.

Let $B_0$ and $B_1$ be the affine transformations which represent left and right branching; let $\mathbf{t}$ be the trunk of the tree. In most discussions of binary trees, $B_0$ and $B_1$ are scaled rotations (which always commute, incidentally, making much of the linear algebra much easier), and the trunk is usually $(0, 1)$. These restrictions are rather confining; part of the wide range of trees possible results from allowing the most general transformations.

$B_0\mathbf{t}$ and $B_1\mathbf{t}$ are first-level branches, $B_1B_0\mathbf{t}$ is a second-level branch, etc. For example, the instructions 001 determine the node $P$, as illustrated below:

$$(I + B_0 + B_0B_0 + B_1B_0B_0)\mathbf{t}. \tag{6}$$

Care must be taken about the order of the transformations, since in general, they need not commute.



Now, we abstract a definition from this specific example.

Define $\Sigma = \{0, 1, 2, ..., m-1\}$; $m$ will represent the number of branchings at each node of a tree. In the above example, $m = 2$.

Define $\Sigma^n$ to be the set of all strings of length $n$; elements of $\Sigma^n$ will represent sequences of instructions for traversing a binary tree. We denote the empty string by $\Lambda$, so that $\Sigma^0 = \{\Lambda\}$. For $s \in \Sigma^n$, we use $|s| = n$ to denote the length of $s$. We say that $\Sigma^\omega$ is the set of all infinite strings with letters from $\Sigma$, and put

$$\Sigma^* = \Sigma^\omega \cup \bigcup_{n=0}^{\infty} \Sigma^n. \tag{7}$$

Let $s_i$ denote the $i^{\text{th}}$ letter in $s$, and $s_{[i:j]}$ denote that substring of $s$ from the $i^{\text{th}}$ to the $j^{\text{th}}$ letter when $i \leq j$. When $i > j$, we put $s_{[i:j]} = \Lambda$. Also, we write $\overleftarrow{s}$ for the string consisting of the letters of $s$ written in reverse order. This is fairly standard notation for strings used in many computer science applications.

Now let $\mathcal{B}$ be a finite set of $m$ affine transformations, $\{B_0, \dots B_{m-1}\}$. These transformations describe how to create the branches in an $m$-ary tree. When all the transformations are invertible, we define $\mathcal{B}^{-1}$ to be the set of inverse transformations, $\{B_0^{-1}, \ldots, B_{m-1}^{-1}\}$.

Now we look at (6). We first need to define the individual terms of the sum. To this end, we define, for a string $s$ of length $n$,

$$\eta_{\mathcal{B}}(s) = \prod_{i=1}^{n} B_{s_i} = B_{s_n} \ldots B_{s_1}. \tag{8}$$

Note that $\eta_{\mathcal{B}}(\Lambda) = I$, and that the order of transformations in the product is important as the affine transformations may not commute. As an example, $\eta_{\mathcal{B}}(001) = B_1 B_0 B_0$. Important here is that $\eta_{\mathcal{B}}$ defines a *transformation*, not a *vector*. This is because different trunk vectors may actually produce different trees even though the same $B_0$ and $B_1$ are used. Again, we strive for the greatest generality here.

To add all the branches, we define, for $s \in \Sigma^n$,

$$\tau_{\mathcal{B}}(s) = \sum_{i=0}^{n} \eta_{\mathcal{B}}(s_{[1:i]}). \tag{9}$$

For the above example, we have $s = 001$ and

$$\begin{aligned} \tau_{\mathcal{B}}(001) &= \eta_{\mathcal{B}}(\Lambda) + \eta_{\mathcal{B}}(0) + \eta_{\mathcal{B}}(00) + \eta_{\mathcal{B}}(001) \\ &= I + B_0 + B_0 B_0 + B_1 B_0 B_0. \end{aligned}$$

For $n \geq 0$, we define

$$\mathcal{C}_{\mathcal{B}}^{n} = \{\tau_{\mathcal{B}}(s) \mid s \in \Sigma^n\}. \tag{10}$$

When a trunk (initial branch) $\mathbf{t}$ of a tree is given, we call

$$\mathcal{C}_{\mathcal{B}}^{n}\mathbf{t} \tag{11}$$

the **canopy** of depth $n$, and call elements of $\mathcal{C}_{\mathcal{B}}^{n}\mathbf{t}$ **nodes** of depth $n$. Defining a canopy in this way is important for us, since the number of nodes of depth $n$ often depends on the choice of trunk $\mathbf{t}$.

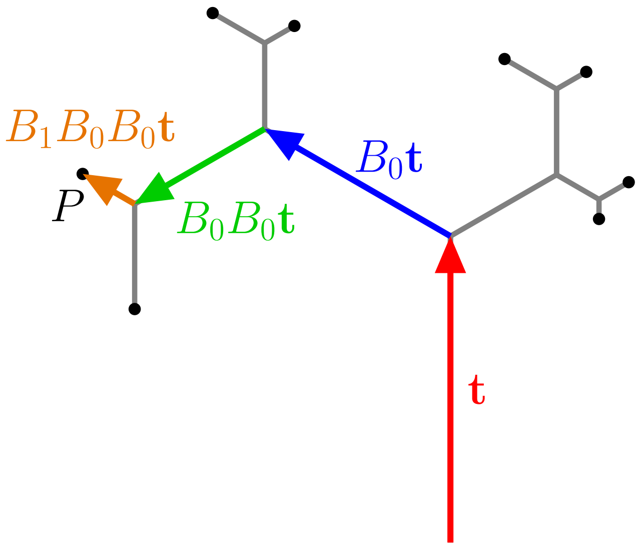
Of current interest is the sequence

$$|\mathcal{C}_{\mathcal{B}}^{1}\mathbf{t}|, |\mathcal{C}_{\mathcal{B}}^{2}\mathbf{t}|, |\mathcal{C}_{\mathcal{B}}^{3}\mathbf{t}|, |\mathcal{C}_{\mathcal{B}}^{4}\mathbf{t}|, |\mathcal{C}_{\mathcal{B}}^{5}\mathbf{t}|, \ldots, \tag{12}$$

which we call the sequence of **node numbers.** In other words, how many new nodes are created each time we iterate the binary tree algorithm? What we're finding is that an amazingly diverse range of sequences is possible, and moreover, that *proving* the validity of any particular sequence is usually quite involved.

## 5 Definition of a Merkle tree

Here, we discuss various useful notations and definitions for describing merkle trees.



Now, we abstract a definition from this specific example.

Define $\Sigma = \{0, 1, 2, ..., m - 1\}$; $m$ will represent the number of branchings at each node of a tree. In the above example, $m = 2$.