

Leverrier-Faddeev Algorithm

Math 323 Honors Option
Vincent Purcell

Introduction

The academic paper *Systems of Linear Differential Equations by Laplace Transform* by Heinrich Guggenheimer details the mathematical construct that when texts present scalar differential equations the preferred method to solve them is Laplace Transforms, but when texts present a system of scalar differential equations then throw away Laplace Transforms. Instead the texts present the method of Eigenvalues and Eigenvectors of the coefficient matrix. In the paper by Guggenheimer he presents the method of Leverrier-Faddeev coupled with Laplace transforms as a method to solve these systems of differential equations.

What I have done in this report is implemented the Leverrier-Faddeev algorithm as a MATLAB function and wrote a script that tests the function on three different matrices, a 2x2 matrix, a 3x3 matrix, a 4x4 matrix, and a 7x7 magic matrix. The algorithm is able to produce the characteristic polynomial for the system of equations and also produce the inverse of the matrix at no extra computational power. I will show by hand the steps required to produce the characteristic polynomial of the 3x3 and 4x4 matrices using the eigenvalue/eigenvector and Leverrier-Faddeev methods. The published MATLAB code is found at the bottom of this report and the 3x3 and 4x4 matrices are *Example2* and *Example3*, respectively, in the code. The 3x3 and 4x4 matrices in question are the following:

$$A_1 = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} \quad A_2 = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix}$$

Eigenvalue and Eigenvector Methods

In this section I will simply be solving for the characteristic polynomial for a 3x3 matrix and 4x4 matrix from the examples in the MATLAB code. Finding the characteristic polynomial for any matrix larger than a 3x3 will require a lot of math which is why Guggenheimer proposed the method of Leverrier-Faddeev as an alternate. To find the characteristic polynomial the following is done:

$$B_1 = \det |A_1 - \lambda I| \quad (1)$$

$$B_1 = \det \left[\begin{pmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right] \quad (2)$$

$$B_1 = \det \begin{bmatrix} 3-\lambda & -5 & 5 \\ 2 & -10-\lambda & 7 \\ -1 & 20 & 11-\lambda \end{bmatrix} \quad (3)$$

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \Rightarrow |A| = a \cdot \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \cdot \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \cdot \begin{vmatrix} d & e \\ g & h \end{vmatrix} \quad (4)$$

$$|B_1| = (3-\lambda) \cdot \begin{vmatrix} -10-\lambda & 7 \\ 20 & 11-\lambda \end{vmatrix} - (-5) \cdot \begin{vmatrix} 2 & 7 \\ -1 & 11-\lambda \end{vmatrix} + 5 \cdot \begin{vmatrix} 2 & -10-\lambda \\ -1 & 20 \end{vmatrix} \quad (5)$$

$$|B_1| = (3-\lambda)(-\lambda + \lambda^2 - 250) + 5(29 - 2\lambda) + 5(30 - \lambda) \quad (6)$$

$$|B_1| = -3\lambda + 3\lambda^2 - 750 + \lambda^2 - \lambda^3 + 250\lambda + 145 - 10\lambda + 150 - 5\lambda \quad (7)$$

$$|B_1| = -\lambda^3 + 4\lambda^2 + 232\lambda - 455 \quad (8)$$

$$\text{Characteristic Polynomial} = \lambda^3 - 4\lambda^2 - 232\lambda + 455 \quad (9)$$

Overall the method of finding the characteristic polynomial for the 3x3 matrix using the method above was not too difficult, but any matrix larger there is an exponential growth in the amount of times you must find the determinant of matrices. I will find the characteristic polynomial for a 4x4 matrix to show the increase of math needed which leaves more room for error.

$$B_2 = \det |A_2 - \lambda I| \quad (1)$$

$$B_2 = \det \left[\begin{pmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \right] \quad (2)$$

$$B_2 = \det \begin{bmatrix} 2-\lambda & 5 & 6 & 7 \\ 6 & 7-\lambda & -10 & 6 \\ 2 & -4 & 2-\lambda & -1 \\ -2 & -2 & 20 & 5-\lambda \end{bmatrix} \quad (3)$$

$$|A| = \begin{vmatrix} a & \cdots & i \\ \vdots & \ddots & \vdots \\ j & \cdots & n \end{vmatrix} = (-1)^{i+1} \cdot a_{i,1} \cdot \Delta_{i,1} + (-1)^{i+2} \cdot a_{i,2} \cdot \Delta_{i,2} + \cdots + (-1)^{i+n} \cdot a_{i,n} \cdot \Delta_{i,n} \quad (4)$$

$$|B_2| = (-1)^{1+1} \cdot (2-\lambda) \cdot \Delta_{1,1} + (-1)^{1+2} \cdot 5 \cdot \Delta_{1,2} + (-1)^{1+3} \cdot 6 \cdot \Delta_{1,3} + (-1)^{1+4} \cdot 7 \cdot \Delta_{1,4} \quad (5)$$

Now the smaller 3x3 Determinants must be found ($\Delta_{1,1}, \Delta_{1,2}, \dots$)

$$\Delta_{1,1} = \begin{vmatrix} 7-\lambda & -10 & 6 \\ -4 & 2-\lambda & -1 \\ -2 & 20 & 5-\lambda \end{vmatrix} = (7-\lambda) \cdot \begin{vmatrix} 2-\lambda & -1 \\ 20 & 5-\lambda \end{vmatrix} - (-10) \cdot \begin{vmatrix} -4 & -1 \\ -2 & 5-\lambda \end{vmatrix} + 6 \cdot \begin{vmatrix} -4 & 2-\lambda \\ -2 & 20 \end{vmatrix} \quad (6)$$

$$\Delta_{1,1} = (7-\lambda)(10-2\lambda-5\lambda+\lambda^2+20) + 10(-20+4\lambda-2) + 6(-80+4-2\lambda) \quad (7)$$

$$\Delta_{1,1} = -\lambda^3 + 14\lambda^2 - 51\lambda - 466 \quad (8)$$

$$\Delta_{1,2} = \begin{vmatrix} 6 & -10 & 6 \\ 2 & 2-\lambda & -1 \\ -2 & 20 & 5-\lambda \end{vmatrix} = 6 \cdot \begin{vmatrix} 2-\lambda & -1 \\ 20 & 5-\lambda \end{vmatrix} - (-10) \cdot \begin{vmatrix} 2 & -1 \\ -2 & 5-\lambda \end{vmatrix} + 6 \cdot \begin{vmatrix} 2 & 2-\lambda \\ -2 & 20 \end{vmatrix} \quad (9)$$

$$\Delta_{1,2} = 6(30-7\lambda+\lambda^2) + 10(10-2\lambda-2) + 6(40+4-2\lambda) \quad (10)$$

$$\Delta_{1,2} = 6\lambda^2 - 74\lambda + 524 \quad (11)$$

$$\Delta_{1,3} = \begin{vmatrix} 6 & 7-\lambda & 6 \\ 2 & -4 & -1 \\ -2 & -2 & 5-\lambda \end{vmatrix} = 6 \cdot \begin{vmatrix} -4 & -1 \\ -2 & 5-\lambda \end{vmatrix} - (7-\lambda) \cdot \begin{vmatrix} 2 & -1 \\ -2 & 5-\lambda \end{vmatrix} + 6 \cdot \begin{vmatrix} 2 & -4 \\ -2 & -2 \end{vmatrix} \quad (12)$$

$$\Delta_{1,3} = 6(4\lambda-22) - (7-\lambda)(-2\lambda+8) + 6(-12) \quad (13)$$

$$\Delta_{1,3} = -2\lambda^2 + 46\lambda - 260 \quad (14)$$

$$\Delta_{1,4} = \begin{vmatrix} 6 & 7-\lambda & -10 \\ 2 & -4 & 2-\lambda \\ -2 & -2 & 20 \end{vmatrix} = 6 \cdot \begin{vmatrix} -4 & 2-\lambda \\ -2 & 20 \end{vmatrix} - (7-\lambda) \cdot \begin{vmatrix} 2 & 2-\lambda \\ -2 & 20 \end{vmatrix} + (-10) \cdot \begin{vmatrix} 2 & -4 \\ -2 & -2 \end{vmatrix} \quad (15)$$

$$\Delta_{1,4} = 6(-2\lambda-76) - (7-\lambda)(-2\lambda+44) - 10(-12) \quad (16)$$

$$\Delta_{1,4} = -2\lambda^2 + 46\lambda - 644 \quad (17)$$

Now back to the original 4x4 determinant equation

$$|B_2| = (2-\lambda) \cdot \Delta_{1,1} - 5 \cdot \Delta_{1,2} + 6 \cdot \Delta_{1,3} - 7 \cdot \Delta_{1,4} \quad (18)$$

$$|B_2| = (2-\lambda) \cdot (-\lambda^3 + 14\lambda^2 - 51\lambda - 466) - 5 \cdot (6\lambda^2 - 74\lambda + 524) + 6 \cdot (-2\lambda^2 + 46\lambda - 260) - 7 \cdot (-2\lambda^2 + 46\lambda - 644) \quad (19)$$

$$\text{Characteristic Polynomial} = \lambda^4 - 16\lambda^3 + 51\lambda^2 + 688\lambda - 604 \quad (20)$$

The amount of math required to find the characteristic polynomial of a 4x4 matrix is a lot, as seen above, with the more popular method taught in Linear Algebra texts. I did not include some simplification steps above, so the amount of math required is 20+ steps. The math required leaves too much room for algebraic mistakes if done by hand.

Leverrier-Faddeev Algorithm

The Leverrier-Faddeev Algorithm requires much less steps so it will leave less room for mistakes if done by hand and should be a much quicker method. The algorithm is the following steps:

<i>input = matrix A</i>	<i>* Note *</i>
<i>n = rows in matrix A</i>	<i>Input matrix must be a square matrix</i>
<i>Save copy of Matrix A</i>	
<i>LeverrierFaddev Matrix = $LF_{mat} = A$</i>	
<i>polynomial coefficients = $[1, 1, \dots, 1]_{n+1}$</i>	<i>Chararacteristic Polynomial coefficients are set to an 'n + 1' long vector</i>
<i>for i = 2:n</i>	<i>This 'for' loop does the bulk of the LF Algorithm. The coefficient for the current iteration is calculated and then that coefficient is used to update the LF matrix for the following iteration</i>
$coeff_i = -\frac{trace(LF_{mat})}{(i-1)}$	
$LF_{mat} = A * (LF_{mat} + coeff_i * I_{n \times n})$	<i>* Notes *</i>
<i>end</i>	<i>'trace' refers to sum of the diagonal of a matrix, and 'I' is a nxn identity matrix</i>
$coeff_{n+1} = -\frac{trace(LF_{mat})}{n}$	<i>The last coefficient is calculated after the for loop</i>

The algorithm is quite simple and requires only 2 calculations per row/column of an n-by-n matrix, the coefficient for each iteration and recalculating the L-F matrix for the next iteration. So, for example a 4x4 matrix should only take 4 calculations. The difference with the Leverrier-Faddeev method is that it requires no algebra and only matrix multiplication and addition is required. What that means is that if you have a scientific calculator available then you can easily do this calculation for large matrices in significantly less time than the method of Eigenvalues/Eigenvectors. I did the calculations by hand for the 3x3 and 4x4 matrices using a scientific calculator for the matrix multiplication. Those calculations are below.

Characteristic Polynomial of 3x3 Matrix using Leverrier-Faddeev Algorithm

$$A = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix}; \text{coefficient vec} = [1, 1, 1, 1]; n = \text{rows of } A = 3; i = \text{iteration of Algorithm}$$

(1)
Initial Conditions

$$\text{sum of diagonal of } A = \text{trace}(A) = 3 - 10 + 11 = 4$$

(2)

$$\text{coeff}_2 = -\frac{\text{trace}(A)}{i-1} = -\frac{4}{2-1} = -4$$

(3)

$$\text{coefficient vec} = [1, -4, 1, 1]$$

(4)

$$LF_{mat} = A * (A + \text{coeff}_2 * I_{3 \times 3})$$

(5)

$$LF_{mat} = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} * \left(\begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} - 4 \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)$$

(6)

$$LF_{mat} = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} * \left(\begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} + \begin{bmatrix} -4 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -4 \end{bmatrix} \right)$$

(7)

$$LF_{mat} = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} * \begin{bmatrix} -1 & -5 & 5 \\ 2 & -14 & 7 \\ -1 & 20 & 7 \end{bmatrix}$$

(8)

$$LF_{mat} = \begin{bmatrix} -18 & 155 & 15 \\ -29 & 270 & -11 \\ 30 & -55 & 212 \end{bmatrix}$$

(9)

$$\text{coeff}_3 = -\frac{\text{trace}(LF_{mat})}{i-1} = -\frac{-18 + 270 + 212}{3-1} = -\frac{464}{2} = -232$$

(10)

$$\text{coefficient vec} = [1, -4, -232, 1]$$

(11)

$$LF_{mat} = A * (LF_{mat} + \text{coeff}_3 * I_{3 \times 3})$$

(12)

$$LF_{mat} = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} * \left(\begin{bmatrix} -18 & 155 & 15 \\ -29 & 270 & -11 \\ 30 & -55 & 212 \end{bmatrix} - 232 \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)$$

(13)

$$LF_{mat} = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} * \left(\begin{bmatrix} -18 & 155 & 15 \\ -29 & 270 & -11 \\ 30 & -55 & 212 \end{bmatrix} + \begin{bmatrix} -232 & 0 & 0 \\ 0 & -232 & 0 \\ 0 & 0 & -232 \end{bmatrix} \right)$$

(14)

$$LF_{mat} = \begin{bmatrix} 3 & -5 & 5 \\ 2 & -10 & 7 \\ -1 & 20 & 11 \end{bmatrix} * \begin{bmatrix} -250 & 155 & 15 \\ -29 & 38 & -11 \\ 30 & -55 & -20 \end{bmatrix}$$

(15)

$$LF_{mat} = \begin{bmatrix} -455 & 0 & 0 \\ 0 & -455 & 0 \\ 0 & 0 & -455 \end{bmatrix} \quad (16)$$

$$coeff_4 = -\frac{trace(LF_{mat})}{n} = -\frac{3 * -455}{3} = 455 \quad (17)$$

$$coefficient\ vec = [1, -4, -232, 455] \quad (18)$$

$$Characteristic\ Polynomial = \lambda^3 - 4\lambda^2 - 232\lambda + 455 \quad (19)$$

Characteristic Polynomial of 4x4 Matrix using Leverrier-Faddeev Algorithm

The above solution of the 3x3 matrix is not really 19 steps of calculations. I expanded some steps to increase its readability. It is actually only two iterations of the L-F Algorithm. It is two matrix equations and then three simple calculations to find the coefficients of the polynomial. The tradeoff isn't much between using the L-F Algorithm and the Eigenvalue/Eigenvector method for the 3x3 matrix but there is certainly a difference between the two for calculating the polynomial for the 4x4 matrix.

$$A = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix}; \quad coefficient\ vec = [1, 1, 1, 1]; \quad n = rows\ of\ A = 4; \quad (1) \quad \text{Initial Conditions}$$

$$sum\ of\ diagonal\ of\ A = trace(A) = 2 + 7 + 2 + 5 = 16 \quad (2)$$

$$coeff_2 = -\frac{trace(A)}{i-1} = -\frac{16}{2-1} = -16 \quad (3)$$

$$coefficient\ vec = [1, -16, 1, 1] \quad (4)$$

$$LF_{mat} = A * (A + coeff_2 * I_{4x4}) \quad (5)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \left(\begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} - 16 \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \quad (6)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \left(\begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} + \begin{bmatrix} -16 & 0 & 0 & 0 \\ 0 & -16 & 0 & 0 \\ 0 & 0 & -16 & 0 \\ 0 & 0 & 0 & -16 \end{bmatrix} \right) \quad (7)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \begin{bmatrix} -14 & 5 & 6 & 7 \\ 6 & -9 & -10 & 6 \\ 2 & -4 & -14 & -1 \\ -2 & -2 & 20 & -11 \end{bmatrix} \quad (8)$$

$$LF_{mat} = \begin{bmatrix} 0 & -73 & 18 & -39 \\ -74 & -5 & 226 & 28 \\ -46 & 40 & 4 & -1 \\ 46 & -82 & -172 & -101 \end{bmatrix} \quad (9)$$

$$coeff_3 = -\frac{trace(LF_{mat})}{i-1} = -\frac{0-5+4-101}{3-1} = \frac{102}{2} = 51 \quad (10)$$

$$coefficient\ vec = [1, -16, 51, 1, 1] \quad (11)$$

$$LF_{mat} = A * (LF_{mat} + coeff_3 * I_{4 \times 4}) \quad (12)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \left(\begin{bmatrix} 0 & -73 & 18 & -39 \\ -74 & -5 & 226 & 28 \\ -46 & 40 & 4 & -1 \\ 46 & -82 & -172 & -101 \end{bmatrix} + 51 \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \quad (13)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \left(\begin{bmatrix} 0 & -73 & 18 & -39 \\ -74 & -5 & 226 & 28 \\ -46 & 40 & 4 & -1 \\ 46 & -82 & -172 & -101 \end{bmatrix} + \begin{bmatrix} 51 & 0 & 0 & 0 \\ 0 & 51 & 0 & 0 \\ 0 & 0 & 51 & 0 \\ 0 & 0 & 0 & 51 \end{bmatrix} \right) \quad (14)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \begin{bmatrix} 51 & -73 & 18 & -39 \\ -74 & 46 & 226 & 28 \\ -46 & 40 & 55 & -1 \\ 46 & -82 & -172 & -50 \end{bmatrix} \quad (15)$$

$$LF_{mat} = \begin{bmatrix} -222 & -250 & 292 & -294 \\ 524 & -1008 & 108 & -328 \\ 260 & -168 & -586 & -142 \\ -644 & 444 & -248 & -248 \end{bmatrix} \quad (16)$$

$$coeff_4 = -\frac{trace(LF_{mat})}{i-1} = -\frac{-222-1008-586-248}{4-1} = \frac{2064}{3} = 688 \quad (17)$$

$$coefficient\ vec = [1, -16, 51, 688, 1] \quad (18)$$

$$LF_{mat} = A * (LF_{mat} + coeff_4 * I_{4 \times 4}) \quad (19)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \left(\begin{bmatrix} -222 & -250 & 292 & -294 \\ 524 & -1008 & 108 & -328 \\ 260 & -168 & -586 & -142 \\ -644 & 444 & -248 & -248 \end{bmatrix} + 688 \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \quad (20)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \left(\begin{bmatrix} -222 & -250 & 292 & -294 \\ 524 & -1008 & 108 & -328 \\ 260 & -168 & -586 & -142 \\ -644 & 444 & -248 & -248 \end{bmatrix} + \begin{bmatrix} 688 & 0 & 0 & 0 \\ 0 & 688 & 0 & 0 \\ 0 & 0 & 688 & 0 \\ 0 & 0 & 0 & 688 \end{bmatrix} \right) \quad (21)$$

$$LF_{mat} = \begin{bmatrix} 2 & 5 & 6 & 7 \\ 6 & 7 & -10 & 6 \\ 2 & -4 & 2 & -1 \\ -2 & -2 & 20 & 5 \end{bmatrix} * \begin{bmatrix} 466 & -250 & 292 & -294 \\ 524 & -320 & 108 & -328 \\ 260 & -168 & 102 & -142 \\ -644 & 444 & -248 & 440 \end{bmatrix} \quad (22)$$

$$LF_{mat} = \begin{bmatrix} 604 & 0 & 0 & 0 \\ 0 & 604 & 0 & 0 \\ 0 & 0 & 604 & 0 \\ 0 & 0 & 0 & 604 \end{bmatrix} \quad (23)$$

$$coeff_5 = -\frac{trace(LF_{mat})}{n} = -\frac{4 * 604}{4} = -604 \quad (24)$$

$$coefficient\ vec = [1, -16, 51, 688, -604] \quad (25)$$

$$Characteristic\ Polynomial = \lambda^4 - 16\lambda^3 + 51\lambda^2 + 688\lambda - 604 \quad (26)$$

Conclusion

Overall, I prefer the Leverrier-Faddeev Algorithm for calculating the characteristic polynomial of a system of scalar differential equations. There is no algebra required, only matrix arithmetic. If a scientific calculator is available, you can calculate the characteristic polynomial for a 4x4 matrix in only a few minutes while that is simply not possible with the Eigenvalue/Eigenvector method that is often taught in classes. It is definitely an interesting and useful algorithm that I will keep in my back pocket for a situation that requires me to calculate the characteristic polynomial of a large matrix.

Code Implementation

Another benefit of this algorithm is that it is easily implemented in code. I was able to make a function for the algorithm that was less than 10 lines and only had a big O complexity of O(N-1). What this means is that for a say a 4x4 matrix it will have a complexity of O(4-1) since the for loop only runs 3 times for it to calculate the polynomial. If you were to implement the Eigenvalue/Eigenvector method in MATLAB you would have to use recursion with is almost an exponential growth in complexity as the matrix size increases which means large matrices would take several minutes to compute while the L-F algorithm will compute almost instantly.

I have implemented the Leverrier-Faddeev Algorithm in MATLAB and ran four examples through the function. The published MATLAB code is located below. It details examples of computing the characteristic polynomial and inverse of 2x2, 3x3, 4x4, and 7x7 matrices