ME 500
Numerical Methods in Mechanical Engineering
Assignment 3

Brandon Lampe

October 14, 2015

Abstract

The focus for this assignment was on linear algebra and the linear algebraic problem. All code for calculations has been appended at the end of this document.

1 Summary of relevant theory

1.1 Mathematical Vectors

Following terms are specific to mathematical vectors, which implies these vectors are not related to a physical basis (unlike physical vectors).

column vector $\{v\}$, a column of ordered terms with components $v_1, v_2, ..., v_n$.

row vector $< v >$, a row of ordered terms with components $v_1, v_2, ..., v_n$.

size n, the number of components in the vector; also referred to as the dimension of the vector

transpose ^{T}, an operation that swaps column and row components:

$$\{v\}^T = < v > \quad \text{and} \quad < v >^T = \{v\}$$

inner product results in a scalar and is only defined between vectors (or vector spaces) of the same dimension n, and is only defined if the vectors are of the same size

$$< v > \{x\} = \langle v, x\rangle = \sum_{i=1}^{n} v_i x_i$$

analogous to the dot product, where:

$$\mathbf{v} \cdot \mathbf{x} = v_i x_j \mathbf{e}_i \cdot \mathbf{e}_j = v_i x_i \implies \text{physical vectors}$$

magnitude $|v|$, a nonnegative scalar value of a vector defined as the square root of the inner product of a vector with itself; analogous to the L_2 norm

$$|v| = \sqrt{< v > \{v\}}$$

norm $\|x\|$, a nonnegative scalar measure of a vector that can be zero only if every component of the vector is zero

$$\|x\| = |x| \implies 1D \quad \text{Absolute Value norm}$$

$$\|x\|_{L_1} = \left(\sum_{i=1}^{n} |x_i|\right)^{1/1} \implies \text{sum of positive values, Taxicab or Manhattan norms}$$

$$\|x\|_{L_2} = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2} = \sqrt{< x > \{x\}} \implies \text{square root of the sum of squares, magnitude or Euclidean norm}$$

unit vector $\{\hat{v}\}$, a vector having magnitude of unity, which is the vector divided by its magnitude

$$\{\hat{v}\} = \frac{\{v\}}{|v|}$$

angle between vectors θ, defined as

$$cos(\theta_{uv}) =< \hat{u} > \{\hat{v}\} \implies \theta_{uv} = cos^{-1}(< \hat{u} > \{\hat{v}\})$$

real vector space a set of vectors together with eight rules for vector addition and multiplication by real numbers. The vector space is denoted as R^n where n indicates the number of components, and the sets of vectors that make up the space must be given; rules include

1. association of addition: $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
2. commutativity of addition: $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
3. identity element of addition: $\mathbf{v} + \mathbf{0} = \mathbf{v} \quad \forall \quad \mathbf{v}$
4. inverse elements of addition: $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
5. compatibility of scalar multiplication with field multiplication: $a(b\mathbf{v}) = (ab)\mathbf{v}$
6. identity element: $1\mathbf{v} = \mathbf{v}$
7. distributivity of scalar multiplication with respect to vector addition: $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$
8. distributivity of scalar multiplication with respect to field addition: $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$

Within all vector spaces, two operations are possible that allow us to take *linear combinations* of the vectors. These operations result in vectors that are in the same vector space:

- we can add any two vectors
- we can multiply all vectors by scalars

notation for a vector space R^n, consists of *all column vectors with n components.* R is used to denote the space because the components are real numbers. The number of components in the vector space is denoted by n. If the dimension of R^n is m, then the set of given vectors also define a subspace R^n_m, where m is the number of linearly independent vectors in the vector space and $m \leq n$.

Example: say we have a 3×4 matrix that defines a vector space that is R^3. A column $\{c\}$ of the matrix is a linear combination of two other columns e.g., $\{c\} = \{a\} + \{b\}$. $\{c\}$ defines a subspace that is R^1, and that subspace is a one-dimensional line that lies on the two-dimensional plane defined by the columns $\{a\}$ and $\{b\}$.

span if the vector subspace $\{c\}$ can be expressed in terms of a linear combination of vectors in the vector space $\{a\}^i$, then $\{a\}^i$ is said to span the subspace $\{c\}$. This concept holds in higher dimensions.

dimension of a vector space is the number of linearly independent vectors given in the definition of the vector space, this is different than the size or dimension of a single vector. The dimension of a vector space may be determined via the Gram-Schmidt procedure. Vectors in a vector space are linearly independent if

$$\sum_{i=1}^{m} \alpha_i \{v\}^i = \{0\} \implies \alpha = 0 \text{ for } i = 1...m$$

linear independence of a vector a set of vectors is not linearly independent one of the vectors in the set can be defined as a linear combination of other vectors in the set. If no vector in the set can be written in this way, then the vectors are linearly independent.

basis of a vector space unless otherwise state: $\mathbf{e_i}$ or $[I]$, is the frame of reference for the vector space. That is, the components of vector $\{x\}$ are implicitly the components with respect to the coordinate basis.

$$[I] = \delta_{ij}$$

projection the vector projection of $\{a\}$ onto $\{b\}$ results in vector $\{c\}$ having the components of $\{a\}$ that are parallel to $\{b\}$. $\{c\}$ is formed by multiplying the inner product between $\{a\}$ and the unit vector $\{\hat{b}\}$ by $\{\hat{b}\}$

$$\{c\} = (< a > \{\hat{b}\})\{\hat{b}\}$$

orthonormal basis the set of vectors defining the basis are orthonormal if

1. vectors are normal (magnitude of unity): $|\{v\}| = 1$ or $< v >^i \{v\}^i = 1$
2. vectors in the space are orthogonal: $< v >^i \{v\}^j = 0 \quad \forall \quad i \neq j$
3. vectors are orthonormal if: $< v >^i \{v\}^j = \delta_{ij} \quad \forall \quad i$ and j

An orthonormal basis ($< e >$) is particularly convenient if the components of a vector with respect to that basis $\{x\}^e$ are desired, the components of that vector are obtained via the inner product with each of the base vectors

$$x_i^e = < e >^i \{x\}$$

Gram-Schmidt procedure a method of obtaining an orthonormal set of vectors $\{Q\}^i$ that span the same vector space of a given set of vectors $\{A\}^i$. In essence, the $\{Q\}^1$ calculated from the unit vector of $\{A\}^1$ and each additional $\{Q\}^k$ results from subtracting out the sum of previously calculated values of $\{Q\}^k$

$$\{Q\}^{*k} = \{A\}^k - \sum_{j=1}^{k-1} < \{Q\}^{jT} > \{A\}^k$$

$$\{Q\}^k = \frac{\{Q\}^{*k}}{|\{Q\}^{*k}|}$$

below is a snippet of code from the subroutine `GS` that performs the Gram-Schmidt procedure on the input matrix $[A]$:

```
def GS(A):
    Q = np.zeros((nrow, ncol))
    neg_terms = np.zeros(nrow)
    cnt = 0
    vspace = 0

    for i in xrange(ncol):
        for j in xrange(cnt):
            neg_terms = neg_terms - Q[:,j].dot(A[:,cnt]) * Q[:,j]
        Q_star = A[:,cnt] + neg_terms
        Q[:,i] = Q_star / np.sqrt(Q_star.dot(Q_star))
        # increment/clear terms
        cnt = cnt + 1
        neg_terms = np.zeros(nrow)
    return Q
```

1.2 Matrices

Following terms are specific to matrices.

matrix $[A]_{m\times n}$, is an ordered array of column or row vectors with m rows and n columns. Additionally, a matrix can be an ordered array of components (scalars). $R^{m\times n}$ denotes the vector space of all real $m \times n$ matrices, and the dimension of the vector space is the number of independent matrices used to fine the space. Note: dimensions of the matrix and dimensions of the vector space are different items.

ordered array of column vectors

$$[A] = \begin{bmatrix} \{A\}^1 & \{A\}^2 & \{A\}^3 & \dots & \{A\}^n \end{bmatrix}$$

ordered array of row vectors

$$[A] = \begin{bmatrix} < A >^1 \\ < A >^2 \\ < A >^3 \\ \vdots \\ < A >^m \end{bmatrix}$$

ordered array of scalars

$$[A] = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{m1} & \dots\dots\dots & & A_{mn} \end{bmatrix}$$

transpose $[A]^T$, rows and columns are exchanged. In indicinal form: $A^T_{ij} = A_{ji}$

matrix product $[A]_{m\times n}[B]_{p\times q} = [C]_{m\times q}$, is only defined if $n = p$

$$C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj} = A_{i1}B_{1j} + A_{i2}B_{2j} + \ldots + A_{in}B_{nj}$$

transpose of a matrix product transpose of the product of two matrices equals the product of the transpose of the two matrices in the reverse order

$$\big[[A][B]\big]^T = [B]^T[A]^T$$

multiplication with a vector $< x > [A]$ or $[A]\{x\}$, results in a vector having the same length $\{x\}$, analogous to a dot product between a vector and a tensor.

$$< x > [A] = [A]\{x\} = \sum_{j=1}^{n} a_{ij}x_j$$

outer product of vectors Suppose $\{u\} \in R^m$ and $\{v\} \in R^n$, then the outer product of this two vectors is the matrix $[A] \in R^{m\times n}$

$$[A] = \{u\} < v >= \begin{bmatrix} u_1v_1 & u_1v_2 & \dots & u_1v_n \\ u_2v_1 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ u_mv_1 & \dots\dots\dots & & u_mv_n \end{bmatrix}$$

partitioned matrix results from partitioning a general matrix into an array of sub matrices and is useful because sub matrices follow the same rules as general matrices

$$[A] = \begin{bmatrix} [A]_{11} & \vdots & [A]_{12} \\ \cdots & \cdots & \cdots \\ [A]_{21} & \vdots & [A]_{22} \end{bmatrix}$$

diagonal matrix for a matrix with i rows and j columns, has potentially nonzero components along the main diagonal of the matrix ($i = j$) and all other components ($i \neq j$) are zero

lower triangular for a matrix with i rows and j columns, has potentially nonzero components along and below the main diagonal of the matrix ($i \geq j$) and all other components ($i < j$) are zero

upper triangular for a matrix with i rows and j columns, has potentially nonzero components along and above the main diagonal of the matrix ($i \leq j$) and all other components ($i > j$) are zero

range vector space formed by the columns of a matrix, also known as the column space, which is a subspace of R^m (the whole space)

rank r, is the number of independent vectors in a given vector space, which may be obtained via the Gram-Schmidt procedure.

$$\text{for } [A]_{m\times n}\ r \leq n$$

nullspace a vector space of $[A]$ formed from the solution of $[A]\{x\} = 0$. The *nullspace* of a matrix consists of all vectors $\{x\}$

inverse of a product of matrices the inverse of a product of matrices equals the product of the inverse of the two matrices in reverse order

$$\big[[A][B]\big]^{-1} = [B]^{-1}[A]^{-1}$$

transpose of the inverse transpose of the inverse of a matrix is equal to the inverse of the transpose of the matrix

$$\big[[A]^T\big]^{-1} = \big[[A]^{-1}\big]^T$$

$$\big[[A][B]\big]^{-T} = [A]^{-T}[B]^{-T}$$

orthogonal matrix composed of orthonormal columns and rows

$$[Q][Q]^T = [I] \implies [Q]^T = [Q]^{-1}$$
$$det[Q] = \pm 1$$

positive definite $< x > [A]\{x\} > 0\ \forall\ \{x\}$, additionally, a matrix is said to be positive definite if

- its eigenvalues are all positive.
- matrix is nonsingular (an inverse exists)
- its determinant is positive
- all diagonal components must be positive, $A_{ii} > 0\ \forall\ i$

if a matris is symmetric-positive definite, then

$$|A_{ij}| \leq \frac{1}{2}\left(A_{ii} + A_{jj}\right)$$
$$|A_{ij}| \leq \sqrt{A_{ii} + A_{jj}}$$

elementary $[E]$, is a matrix that differs from the identity matrix by one single elementary row operation. By pre or post multiplying a matrix by $[E]$ you can affect either a rows or columns

$$[E][A] \rightarrow \text{elementary row operation}$$
$$[A][E] \rightarrow \text{elementary column operation}$$

determinant

minor M_{ij}, the determinant obtained by deleting the i^{th} row and j^{th} column of a matrix.

cofactor the number obtained by $M_{ij}^C = (-1)^{i+j} M_{ij}$

Inverse via cofactor $[A]^{-1} = \frac{[M]^a}{det[A]}$ where $[M]^a$ (the adjoint matrix) is the transpose of $[M]^c$

determinant of a matrix product determinant of a product of matrices equals the product of the determinants of the matrices

$$det\left([A][B]\right) = det[A]det[B]$$

determinant of the identity matrix $det[I] = 1$

singular a square matrix is not invertible, and a square matrix is not invertible iff its determinant is zero

trace sum of diagonal terms

magnitude Frobenius norm, which is a scalar measure of a matrix

$$|[A]| = \left(tr\left[[A][A]^T\right]\right)$$
$$|[I]| = \sqrt{n}$$

Linear Algebraic problem $[A]\{x\} = \{b\}$, where given $[A]$ and $\{b\}$ we wish to obtain the solution $\{x\}$

QR algorithm a method to obtain an approximate solution to the linear algebraic problem, where $Q \implies$ an orthogonal matrix and $R \implies$ an upper (right) triangular matrix. The algorithm consists of decomposing $[A]$ into the product between orthogonal and upper-diagonal matrices ($[Q][R] = [A]$) by first calculate $[Q]$ via the Gram-Schmidt procedure then calculating $[R]$ such that it is an upper-diagonal matrix. Then with the equation $[R]\{x\} = \{\hat{b}\}$, solve for the uknown $\{x\}$ via back substitution. An outline of this method is:

1. perform Gram-Schmidt on $[A]$ to obtain the orthogonal matrix $[Q]$
2. knowing that $[A] = [Q][R]$, solve for the upper-diagonal matrix $[R] \implies [R] = [Q]^T[A]$
3. calculate the transformed vector $\{\hat{b}\}$ such that $\{\hat{b}\} = [Q]^T\{b\}$
4. use the back-substitution routine to solve $[R]\{x\} = \{\hat{b}\}$

A code snippet for the calculation of $[R]$ is provided below, where the function `QR` calls `GS` to obtain $[Q]$ then solves for $[R]$:

```
def QR(A):
    nrow = A.shape[0]
    ncol = A.shape[1]

    Q = np.zeros((nrow, ncol))
    R = np.zeros((nrow, ncol))
    neg_terms = np.zeros(nrow)
    diag_vect = np.zeros(nrow)
    diag_vect_sum = np.zeros(nrow)
    diag_norm = 0
    Q, vector_space = GS(A)

    for i in xrange(nrow):
```

```python
        for j in xrange(i+1, ncol, 1):
            R[i,j] = Q[:,i].dot(A[:,j])
        for k in xrange(0, i):
            diag_vect_sum = diag_vect_sum + Q[:,k].dot(A[:,i])*Q[:,k]
        diag_vect = A[:,i] - diag_vect_sum
        diag_norm = np.sqrt(diag_vect.dot(diag_vect))
        R[i,i] = diag_norm
        diag_vect_sum = np.zeros(nrow) # zero the summation
    return Q, R
```

A code snippet for my back-substitution routine `BackSub` is included below:

```python
def BackSub(R, b):
    nrow = R.shape[0]
    ncol = R.shape[1]
    cnt = 0
    x = np.zeros(nrow)

    # ipdb.set_trace()

    for i in reversed(xrange(nrow)):
        num_star = 0
        for j in np.arange(nrow - cnt, nrow,1):
            # this loop is skipped on first i loop
            num_star = num_star - R[i,j]* x[j]
        cnt = cnt + 1
        num = b[i] + num_star
        den = R[i,i]
        x[i] = num / den
    return x
```

A code snippet that `QR_solve` is provided below, this program acts as a wrapper for `GS`, `QR`, and `BackSub` to solve for $\{x\}$:

```python
def QR_solve(A, b, opt = 0):
    Q_orth = np.zeros((A.shape))
    R_ud = np.zeros((A.shape))

    Q_orth, R_ud = QR(A) # performs Gram-Schmidt procedure and obtains [Q] and [R]
    b_dim = len(b.shape)
    nrow = b.shape[0]

    if b_dim > 1: # if b is a 2D array
        ncol = b.shape[1]
        x = np.zeros((nrow, ncol))
        for i in xrange(ncol):
            b_hat = np.transpose(Q_orth).dot(b[:,i])
            x[:,i] = BackSub(R_ud, b_hat)
    else: # if be is a vector
        x = np.zeros(nrow)
        b_hat = np.transpose(Q_orth).dot(b)
        x = BackSub(R_ud, b_hat)

    if opt == 0:
        return x
    if opt != 0:
        return x, R_ud, Q_orth
```

2 Construct matrices and show that you obtain the shown results

```
[A]=      [ 0.543   0.278   0.425]
          [ 0.845   0.005   0.122]
          [ 0.671   0.826   0.137]
          [ 0.575   0.891   0.209]

[B]=      [ 0.185   0.108]
          [ 0.22    0.979]
          [ 0.812   0.172]
```

below is a code snippet for the problem when solved via the inner product between rows of $[A]$ and colums of $[B]$:

```
for i in xrange(nrow):
    for j in xrange(ncol):
        C_a[i,j] = A[i,:].dot(B[:,j])
```

below is a code snippet for the problem when solved via the outer product between columns of $[A]$ and rows of $[B]$:

```
for i in xrange(nloop_outer):
    C_b = C_b + np.outer(A[:,i],B[i,:])
```

both of the above methods resulted in the same output matrix:

```
[C]=      [ 0.506   0.404]
          [ 0.256   0.117]
          [ 0.417   0.904]
          [ 0.472   0.971]
```

3 Perform Gram-Schmidt

Matrix $[A]$ is composed of four vectors, where rows 1, 3, and 4 are independent and row 2 is linearly dependent on row 1:

```
[A]=     [[ 8   5   5   7]
          [16  10  10  14]
          [ 0   7   5   6]
          [ 1   7   0   4]]
```

The Gram-Schmidt procedure was programed into the subroutine `GS` and is shown in the attached code listing under problem 3. The vector space based on the results of `GS` is 3, and the resulting matrix is not orthonormal because the vectors of $[A]$ are not independent. This was checked and $[Q][Q]^T \neq [I]$.

The vector $< e_1 >$ was calculated with respect to the orthogonal basis such as:

$$< e_1 >=< v_1 > [Q]$$

where

```
<v1> =   [ 8   5   5   7]

[Q] =    [[ 0.447  -0.017   0.018   0.    ]
          [ 0.893  -0.034   0.037   0.    ]
          [ 0.      0.74    0.673   0.124]
          [ 0.056   0.672  -0.739   0.992]]

<e1> =   [ 8.428   8.099  -1.475   7.566]
```

4 QR algorithm

4.1 Apply Gram-Schmidt procedure to obtain $[Q]$

Below are the results of running `GS` on the matrix $[A]$:

```
[A] =   [[8 0 1 9]
         [5 7 7 7]
         [5 5 0 5]
         [7 6 4 3]]

[Q] =   [[ 0.627 -0.737  0.147  0.207]
         [ 0.392  0.57   0.582  0.428]
         [ 0.392  0.275 -0.8    0.362]
         [ 0.548  0.238 -0.012 -0.802]]
```

If $[Q]$ is orthonormal, then $[Q][Q]^T = [I]$. The results of $[Q][Q]^T$ were:

```
[  1.000e+00,  -2.220e-16,   1.665e-16,   4.718e-16],
[ -2.220e-16,   1.000e+00,  -2.776e-16,   1.110e-16],
[  1.665e-16,  -2.776e-16,   1.000e+00,  -3.331e-16],
[  4.718e-16,   1.110e-16,  -3.331e-16,   1.000e+00]
```

which is equivalent to $[I]$.

4.2 find $[R]$

The matrix $[R]$ was obtained using the `QR` subroutine, and the result was:

```
[R] =   [[ 12.767   7.989   5.561  11.984]
         [  0.      6.795   4.205  -0.551]
         [  0.      0.      4.171   1.36 ]
         [  0.      0.      0.      4.27 ]]
```

$[R]$ was calculated to be upper triangular.

4.3 Solve $[A]\{x\} = \{b\}$ using the QR algorithm

The back-substitution routine `BackSub` was written and the subroutine `QR_solve` was written to act as a wrapper for the the `GS`, `QR`, and `BackSub` routines. `QR_solve` allows for a solution to the linear algebraic problem with only one function call. The solution obtained using `QR_solve` was:

```
<x> = [ 0.427  0.42  -0.192 -0.247]
```

4.4 Calculate a scalar measure of error by manufacturing a solution

The chosen exact solution $\{x\}^{ex}$, chosen matrix $[A]$, and calculated $\{b\}$ were:

```
<x_ex> = [3 9 4 6]

[A] =   [[8 0 1 9]
         [5 7 7 7]
         [5 5 0 5]
         [7 6 4 3]]

<b> = [ 82 148  90 109]
```

The `QR_solve` subroutine was then used to solve the linear algebraic problem $[A]\{x\} = \{b\}$, which resulted in an approximate solution of:

```
<x_ap> = [ 3.  9.  4.  6.]
```

Where the exact solution contains integers, the approximate solution is now composed of floating point values. Differences between the exact and approximate solutions are not immediately evident. A scalar measure of error (ϵ) was computed such that:

$$\{x\}^{diff} = \{x\}^{ex} - \{x\}^{ap}$$
$$\epsilon = \frac{\|\{x\}^{diff}\|}{\|\{x\}^{ex}\|} = 2.755 \times 10^{-15}$$

The error was also calculated based on a residual (r), such as:

$$\{r\}^{vect} = \{b\} - [A]\{x\}^{ap}$$
$$r = \frac{\|\{r\}^{vect}\|}{\|\{b\}\|} = 3.411 \times 10^{-16}$$

4.5 Iterative improvement of $\{x\}^{ap}$

One iteration was performed by calculating an increment vector $\{\delta\}$ to modify my approximate solution by, where $\{\delta\}$ was solved for using `QR_solve` such that:

$$[A]\{\delta\} = \{x\}^{ap}$$
$$\{x\}^{imp} = \{x\}^{ap} + \{\delta$$
$$\{x\}^{diff} = \{x\}^{ex} - \{x\}^{imp}$$
$$\epsilon = \frac{\|\{x\}^{diff}\|}{\|\{x\}^{ex}\|} = 5.689 \times 10^{-16}$$

and the improved residual was:

$$\{r\}^{vect} = \{b\} - [A]\{x\}^{imp}$$
$$r = \frac{\|\{r\}^{vect}\|}{\|\{b\}\|} = 6.446 \times 10^{-17}$$

Both measures of error decreased upon iterating the approximate solution.

4.6 Calculate the inverse of $[A]$

The subroutine `Inv` was written to calculate the inverse of a matrix by wrapping around the `QR_solve` subroutine by calculating the solution to:

$$[A]\{x\}^i = \{I\}^i$$

where $\{x\}^i$ and $\{I\}^i$ are i^{th} column vectors of the inverse of $[A]$ and the identity matrix $[I]$, respectfully. This calculation resulted in an approximate inverse for $[A]$ being:

```
[A_inv]=          [ 0.068  -0.126  -0.068   0.204]
                  [-0.117   0.026   0.183  -0.016]
                  [ 0.019   0.107  -0.219   0.058]
                  [ 0.049   0.1     0.085  -0.188]]
```

Error based on the Frobenious norm was then calculated such that:

$$[I]^{ap} = [A][A]^{-1}$$
$$[I]^{diff} = [I]^{ex} - [I]^{ap}$$
$$\epsilon = \frac{\|[I]^{diff}\|}{\|[I]^{ex}\|} = 1.685 \times 10^{-15}$$

My result is pretty good based on my calculated error, nearly at the accuracy of my PC.