Teaching Multivariable Math in DASC 2594

John Tipton

The University of Arkansas

2022/06/01 (updated: 2022-06-02)

Materials

- Available on gitHub at https://github.com/jtipton25/dasc-2594-teaching-2022
- Lecture notes at https://jtipton25.github.io/multivariable-math/
- R package available at https://github.com/jtipton25/dasc2594

Review HWs and Labs

- Access for materials
- Open up examples from OneDrive
    - Class Scripts
    - Exams
    - Homeworks (Gradescope)
    - Lecture Notes
    - Labs (Gradescope)
    - Exercises and Quizzes (In development)

Guiding principles

- Teach the fundamental multivariable math needed for data science
- Focus on the higher order math and concepts, leave the computation to the computer
- Focus on the key ideas needed for data science from linear algebra and (multivariable) calculus III
    - Matrix and vector operations
    - Solving systems of equations
    - Understanding rank and invertibility
    - Matrix factorizations
    - Gradients and Geometry
- Omit topics in traditional multivariable calculus courses
    - curl and divergence
    - line integrals
    - multiple integration over irregular boundaries
    - vector fields
    - Green's and Stokes' Theorem

Fundamental concepts

- Matrix operations (multiplication, addition)
- Solving systems of linear equations
  - Applying this to least squares problems (linear regression)
- Demonstrating concepts like rank through data-science applications
  - Fitting models with basis expansions
- Applying calculus to optimization/gradient descent problems

Learning Evaluation

- Goal: Make this a challenging but applicable course
- Homeworks (traditional paper and pencil problems)
- Labs (application using programming)
- Exams (In class and take home)

Continuous Improvement

- Students actually wanted more, smaller exams
- Homeworks and labs were challenging, make these more approachable
- Onboarding students (freshman) with that are Cal III ready but don't have programming experience
- Developing mini-problems and quizzes for more, but lower stakes, practice
    - e.g., how many pivot columns, what is the rank, multiply these matrices, etc.

Getting started with matrices

Examples (Elementary Operations)

```
library(dasc2594)
A <- matrix(1:9, 3, 3)
print(array_to_latex(A))
```

```
## [1] "\\begin{pmatrix} 1 & 4 & 7 \\\\ 2 & 5 & 8 \\\\ 3 & 6 & 9 \\end{pmatrix}"
```

Can display the matrix (in Rmarkdown) with

```
$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$
```

to get

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

```
# using dasc2594
rref(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0   -1
## [2,]    0    1    2
## [3,]    0    0    0
```

Examples (Elementary Operations)

- Add 2 times row 2 to row 3 of `A` and save as `B`

```
B <- A
B[3, ] <- 2 * A[2, ] + A[3, ]
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    7   16   25
```

- Normalize row 2 of `A` and save as `C`

```
C <- A
C[2, ] <- C[2, ] / C[2, 1]
C
```

```
##      [,1] [,2] [,3]
## [1,]    1  4.0    7
## [2,]    1  2.5    4
## [3,]    3  6.0    9
```

Examples (Elementary Operations)

- Swap columns 2 and 3 in the matrix `A` and save as `B`

```
B <- A[, c(1, 3, 2)]
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    7    4
## [2,]    2    8    5
## [3,]    3    9    6
```

- Normalize row 2 of `A` and save as `C`

```
C <- A
C[2, ] <- C[2, ] / C[2, 1]
C
```

```
##      [,1] [,2] [,3]
## [1,]    1  4.0    7
## [2,]    1  2.5    4
## [3,]    3  6.0    9
```

Examples (Elementary Operations)

- After some practice (1-2 labs) have them write

```
# zero out first column in second row
A[2, ] <- A[2, ] - 2 * A[1, ]
# zero out first column in third row
A[3, ] <- A[3, ] - 3 * A[1, ]
# normalize leading column in second row
A[2, ] <- A[2, ] / (-3)
# normalize leading column in third row
A[3, ] <- A[3, ] / (-6)
# zero out second column in first row
A[1, ] <- A[1, ] - 4 * A[3, ]
# zero out second column in third row
A[3, ] <- A[3, ] - A[2, ]
```

Examples (Elementary Operations)

- Have them do row operations, then they write functions that do the elementary row operations
- Add 2 times row 2 to row 3 of `A` and save as `B`

```
A <- matrix(1:9, 3, 3)
add_rows <- function(A, row_1, row_2, a) {
  A[row_1, ] <- A[row_1, ] + a * A[row_2, ]
  return(A)
}

add_rows(A, row_1 = 3, row_2 = 2, a = 2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    7   16   25
```

Examples (Elementary Operations)

- Can add in error checking (for more practice and students more experienced programming)

```
add_rows <- function(A, row_1, row_2, a) {
  if ((row_1 < 1) | row_1 > nrow(A)) {
    stop("row_1 must be a valid row")
  }
  A[row_1, ] <- A[row_1, ] + a * A[row_2, ]
  return(A)
}
add_rows(A, row_1 = 4, row_2 = 2, a = 2)
```

```
## Error in add_rows(A, row_1 = 4, row_2 = 2, a = 2): row_1 must be a valid row
```

Examples (RREF)

- Using the matlib R package

```
library(matlib)
```

```
## Warning: package 'matlib' was built under R version 4.1.2
```

```
gaussianElimination(A, verbose=TRUE)
```

```
##
## Initial matrix:
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## row: 1
##
##  exchange rows 1 and 3
##      [,1] [,2] [,3]
## [1,]    3    6    9
## [2,]    2    5    8
## [3,]    1    4    7
##
##  multiply row 1 by 0.3333333
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    2    5    8
## [3,]    1    4    7
##
```

Examples

- By "hand"

```
# zero out first column in second row
A[2, ] <- A[2, ] - 2 * A[1, ]
# zero out first column in third row
A[3, ] <- A[3, ] - 3 * A[1, ]
# normalize leading column in second row
A[2, ] <- A[2, ] / (-3)
# normalize leading column in third row
A[3, ] <- A[3, ] / (-6)
# zero out second column in first row
A[1, ] <- A[1, ] - 4 * A[3, ]
# zero out second column in third row
A[3, ] <- A[3, ] - A[2, ]
```

- Can use functions that do the elementary row operations, then use functions from `dasc2594` or `matlib` libraries.
- I really don't care that they can do RREF by hand.

Examples (Matrix multiplication)

- Multiply using rows and columns

```
A <- matrix(1:9, 3, 3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
r1 <- A[1, ]
r2 <- A[2, ]
r3 <- A[3, ]
```

```
B <- matrix(1:3, 3, 3)
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    2    2    2
## [3,]    3    3    3
```

```
c1 <- B[, 1]
c2 <- B[, 2]
c3 <- B[, 3]
```

Examples (Matrix multiplication)

- Multiply using rows and columns

```
# initialize a matrix
AB <- matrix(0, 3, 3)
AB[1, 1] <- r1 %*% c1 # or A[1, 1] <- sum(r1 * c1)
AB[1, 2] <- r1 %*% c2 # or A[1, 2] <- sum(r1 * c2)
AB[1, 3] <- r1 %*% c3 # or A[1, 3] <- sum(r1 * c3)
AB[2, 1] <- r2 %*% c1 # or A[2, 1] <- sum(r2 * c1)
AB[2, 2] <- r2 %*% c2 # or A[2, 2] <- sum(r2 * c2)
AB[2, 3] <- r2 %*% c3 # or A[2, 3] <- sum(r2 * c3)
AB[3, 1] <- r3 %*% c1 # or A[3, 1] <- sum(r3 * c1)
AB[3, 2] <- r3 %*% c2 # or A[3, 2] <- sum(r3 * c2)
AB[3, 3] <- r3 %*% c3 # or A[3, 3] <- sum(r3 * c3)

AB
```

```
##      [,1] [,2] [,3]
## [1,]   30   30   30
## [2,]   36   36   36
## [3,]   42   42   42
```

```
A %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   30   30   30
## [2,]   36   36   36
## [3,]   42   42   42
```

Examples (Matrix multiplication)

- Multiply using for loop

```
AB <- matrix(0, nrow(A), ncol(B))
for (i in 1:nrow(A)) {
  for (j in 1:ncol(B)) {
    AB[i, j] <- sum(A[i, ] * B[, j])
  }
}

all.equal(AB, A %*% B)
```

```
## [1] TRUE
```

Example (Matrix multiplication complexity)

- Some basic introduction to $O(\cdot)$ notation and benchmarking code

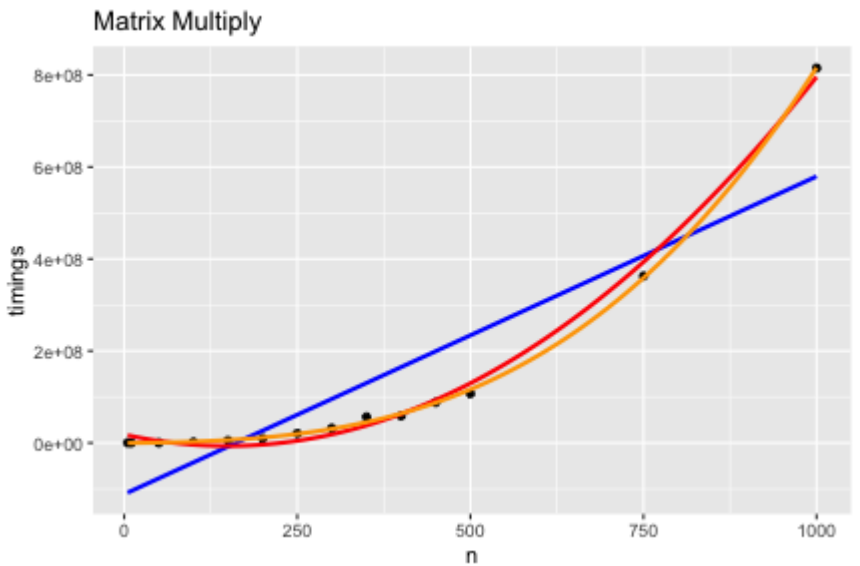
```
library(microbenchmark)
# takes a few seconds
n <- c(5, 10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 750, 1000)
multiply_matrices <- function(n) {
  # generate n x n matrices
  A <- matrix(rnorm(n^2), n, n)
  B <- matrix(rnorm(n^2), n, n)
  A %*% B
}

# timings
timings <- rep(0, length(n))
for (i in 1:length(n)) {
  # create the benchmarking object with 20 replicates
  bm <- microbenchmark(multiply_matrices(n[i]), times = 20)
  # calculate the time
  timings[i] <- mean(bm$time)
}
```

Example (Matrix multiplication complexity)

- Matrix multiplication is $O(n^3)$ for multiplication of $n \times n$ matrices

```
data.frame(timings, n) %>%
  ggplot(aes(x = n, y = timings)) +
  geom_point() +
  stat_smooth(method = "lm", color = "blue", se = FALSE) +
  stat_smooth(method = "lm", color = "red", formula = y ~ poly(x, 2), se = FALSE) +
  stat_smooth(method = "lm", color = "orange", formula = y ~ poly(x, 3), se = FALSE) +
  ggtitle("Matrix Multiply")
```



Solving systems of equations

Using `R`, plot the following systems of equations. Based on these plots, are the equations consistent? If the equations are consistent, is the solution unique?

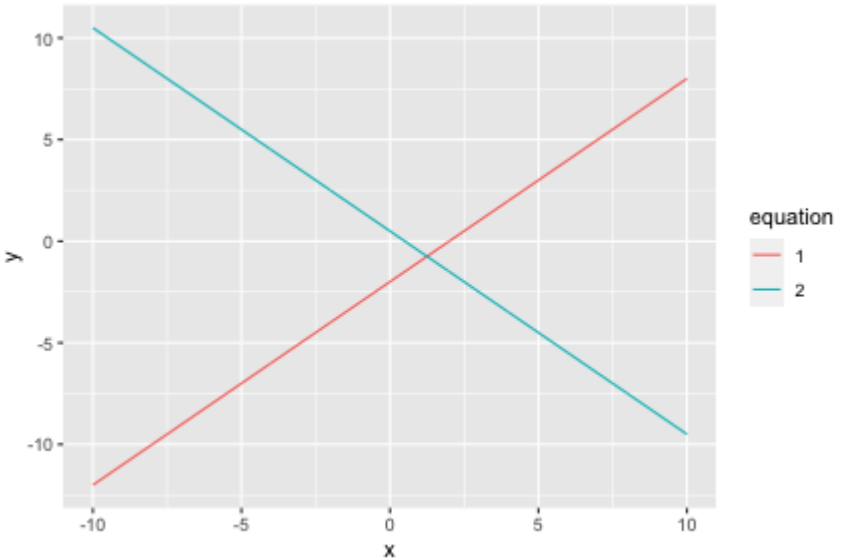
Hint: For each equation, define a grid for `x` using a sequence of numbers. Then, evaluate the equation for `y` for each value of `x`. Put these into a `data.frame` and plot using `ggplot2`.

$$\begin{aligned} x - \quad y &= 2 \\ 2x + 2y &= 1 \end{aligned}$$

Solving systems of equations (Solution)

```
x <- seq(-10, 10, length.out = 1000)
y1 <- x - 2
y2 <- 0.5 - x
dat <- data.frame(x = c(x, x), y = c(y1, y2),
                  equation = factor(rep(c(1, 2), each = 1000)))

ggplot(dat, aes(x = x, y = y, color = equation)) +
    geom_line()
```



Because the lines intersect at a single point, the system of equations is consistent with a unique solution.

Solving systems of equations

For the following matrix equations, are the system of equations consistent? If so, is the solution unique? For each matrix equations, how many pivot columns are in the matrix $\mathbf{A}$?

$$\mathbf{A} = \begin{pmatrix} 3 & -2 & 3 \\ -2 & 4 & 1 \\ 3 & 2 & 7 \\ 1 & 6 & 8 \end{pmatrix} \text{ and } \mathbf{b} = \begin{pmatrix} 2 \\ -1 \\ 4 \\ 3 \end{pmatrix}$$

Solving systems of equations (Solution)

```
A <- matrix(c(3, -2, 3, 1, -2, 4, 2, 6, 3, 1, 7, 8), 4, 4)
b <- c(2, -1, 3, 4)

dasc2594::rref(cbind(A, b))
```

```
##              b
## [1,] 1 0 0 1 0
## [2,] 0 1 0 0 0
## [3,] 0 0 1 0 0
## [4,] 0 0 0 0 1
```

```
dasc2594::rref(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    1
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    0
```

The system of equations is inconsistent because there is a pivot in the last column. There are three pivot columns in the matrix $\mathbf{A}$.