Riemann Sums II[1]
MATLAB exercise for MATH263B

While working through this MATLAB assignment, keep a notepad handy to write down the answers to the problems listed in the text. Be sure to include the problem number with each answer. These answers may be collected and graded as part of a future quiz.

How can we easily compute in MATLAB sums of many terms, such as $\sum_{i=1}^{100} i^2$? The best way to do this is by using *vectors*. A vector is an ordered list of numbers, such as $[3, 7.6, 4]$. The number 3 is called the *first term* of the vector, the number 7.6 is the *second term* of the vector, and so on. Vectors can be assigned as values to MATLAB variables as follows:

```
>> v = [3, 7.6, 4]
```

Now we can instruct MATLAB to compute the sum $3 + 7.6 + 4$ by entering:

```
>> sum([3, 7.6, 4])
```

or

```
>> sum(v)
```

If we want MATLAB to compute $3^2 + 7.6^2 + 4^2$, we can enter:

```
>> sum([3^2, 7.6^2, 4^2])
```

Recall that an easy way to enter this command is to use the $\uparrow$ key to recover the line `>> sum([3, 7.6, 4])` and then to add the missing `^2`'s. You will want to use this trick a number of times in this assignment.

A simpler way of getting the same answer is to enter

```
>> sum(v.^2)
```

Note the period in front of the `^`-symbol. This period instructs MATLAB to apply exponentiation to each term of the vector `v` separately. To see that it cannot be omitted, enter

```
>> sum(v^2)
```

You will get an error message.

Now let us return to the problem of having MATLAB compute $\sum_{i=1}^{100} i^2$. The idea is to create a vector $w = [1, 2, 3, \ldots, 100]$ and then instruct MATLAB to compute `sum(w.^2)`. Here is how we can create the vector with little effort:

```
>> w = 1:1:100;
```

The first number to the right of the = sign tells MATLAB that the first term of the vector w is 1, the second number tells MATLAB what that the terms of w increase in increments of 1, and the last number tells MATLAB that the last term of the vector w is 100. The semicolon at the end teels MATLAB to suppress output; you may want to try and see what happens if you leave out the semicolon. Now we can compute $\sum_{i=1}^{100} i^2$ by entering

```
>> sum(w.^2)
```

Problem 1 *How would you compute* $\sum_{i=3}^{167} \sin i$ *in* MATLAB*? Record both the commands you type and* MATLAB*'s answer.* Hint: MATLAB*'s command for computing* $\sin x$ *is simply* `sin(x)`.

Now we are ready to explore Riemann sums with MATLAB. In order to compute a Riemann sum, we need the following ingredients:

- A function $f(x)$ that is defined on an interval $[a, b]$.
- A partition P of $[a, b]$ into consecutive subintervals $[x_{i-1}, x_i]$, where i ranges from 1 to a number n.
- Sample points x_i^* for $i = 1, \ldots, n$, where x_i^* is in the interval $[x_{i-1}, x_i]$.

The Riemann sum defined by the above items is the number $R = \sum_{i=1}^{n} f(x_i^*)\Delta x_i$, where $\Delta x_i = x_i - x_{i-1}$ is the length of the i-th interval.

Let us start with a simple example. Let $f(x) = x^2$, let $a = 0$, $b = 4$, $n = 8$, let x_i^* be the left endpoint of the i-th interval, and assume the intervals $[x_{i-1}, x_i]$ all have the same length. Then $\Delta x_i = \frac{b-a}{n} = \frac{4-0}{8} = 0.5$ for all i. Thus the Riemann sum we are looking for will be given by the formula: $\sum_{i=1}^{8} (x_i^*)^2 \cdot 0.5$.

The left endpoint of the first interval is 0; the left endpoint of the last interval is $2 - \Delta x_n = 3.5$, and the left endpoints increase in increments of 0.5. Thus we can define the vector of x_i^*'s as follows:

```
>> xs = 0:0.5:3.5
```

This time you may want to omit the semicolon and look at the vector you just defined. Now you can instruct MATLAB to compute the Riemann sum $\sum_{i=1}^{8} (x_i^*)^2 \cdot 0.5$ as follows:

```
>> sum(xs.^2*0.5)
```

You should get the answer 17.5000. Now let us see what happens if we increase the number of intervals. Let us try $n = 20$. While the left endpoint of the first interval remains at 0, the length of each interval will be $\Delta x_i = \frac{4}{20} = 0.2$. Thus the left endpoint of the last interval will be $4 - 0.2 = 3.8$, and the left endpoints increase in increments of 0.2. We can instruct MATLAB to calculate the corresponding Riemann sum by entering

```
>> xs = 0:0.2:3.8;
>> sum(xs.^2*0.2)
```

Problem 2 *Compute the Riemann sum for the above example that corresponds to* $n = 40$*. Record both the commands you enter and* MATLAB*'s answer.*

It would be interesting to see what happens if n gets larger and larger, but the typing of commands may get a little tedious. So let us reflect upon what we have been doing, and then find an easier way of instructing MATLAB to do the same thing. First we defined a vector `xs` whose first term was 0, whose terms increased in increments of $\Delta x_i = \frac{b-a}{n} = \frac{4}{n}$, and whose last term was $b - \Delta x_i = 4 - \frac{4}{n}$. Then we instructed MATLAB to calculate `sum(xs.^2*4/n)`. This could also be accomplished by entering a single command. For example, for $n = 8$ we could have entered

```
>> sum([0:4/8:4-4/8].^2*4/8)
```

and for $n = 20$ we could have entered

```
>> sum([0:4/20:4-4/20].^2*4/20)
```

This gives the same results as before. Note that the above commands can be obtained by substituting the appropriate value of n into the expression `sum([0:4/n:4-4/n].^2*4/n)`. Thus these commands are a *function* of the number n of intervals in the partition. To create an inline function that will do the same computations, enter

```
>> g = inline('sum([0:4/n:4-4/n].^2*(4/n))')
```

If you leave out the single quotation marks, MATLAB will give you an error message.

Now you can compute the Riemann sum for $n = 8$ and $n = 20$ simply by entering

```
>> g(8)
>> g(20)
```

You should get the same results as before.

Problem 3 *Experiment with larger and larger values of n. What happens? Does $g(n)$ appear to approach a fixed number? What appears to be its value? How large needs n to get so that $g(n)$ differs from the apparent limit of the Riemann sums by no more than 0.001?*

Now let us redo the problem for the same function, but with the sample points x_i^* taken as the midpoint of the i-th interval. If there are n intervals, the midpoint of the first interval will be $\frac{\Delta x_1}{2} = \frac{b-a}{2n} = \frac{4}{2n} = \frac{2}{n}$. The midpoint of the last interval will be $b - \frac{\Delta x_n}{2} = 4 - \frac{2}{n}$. The sequence of midpoints will increase in increments of $\Delta x_i = \frac{4}{n}$. Thus the corresponding Riemann sums are represented by a new inline function. Be sure to use a new function name here, because re-using the letter `g` would destroy the old definition of `g`, but you will need to use the function `g` again in a later part of this assignment.

```
>> h = inline('sum([2/n:4/n:4-2/n].^2*(4/n))')
```

Problem 4 *Experiment with computing $h(n)$ for larger and larger values of n. What happens? Does $h(n)$ appear to approach the same number that $g(n)$ approaches? If so, which functions approaches this limit faster? How large needs n to get so that $h(n)$ differs from the apparent limit of the Riemann sums by no more than 0.001? Try to guess the reason why $g(n)$ and $h(n)$ approach the limit at different speeds and explain it.*

Problem 5 *Suppose you want to to calculate Riemann sums for the above example, but with making x_i^* the right endpoint of the i-th interval. How would you define the corresponding inline function?*

Now let us investigate Riemann sums for a different function. Let $\ell(x) = \ln x$, let $a = 1$, $b = 2$, and let the sample points x_i^* be the left endpoints of the partition of the interval $[a, b]$ into n subintervals of length $\frac{1}{n}$ each. MATLAB's command for the ln-function is `log(x)`. If `x` is a vector, then this command instructs MATLAB to calculate logarithms of each term separately. Thus our Riemann sum will be given by the inline function

```
>> r = inline('sum(log([1:1/n:2-1/n])*(1/n))')
```

Problem 6 *Experiment with larger and larger values of n. What happens? Does $r(n)$ appear to approach a fixed number? What appears to be its value? How large needs n to get so that $r(n)$ differs from the apparent limit of the Riemann sums by no more than 0.001?*

Now let us work with the same function $f(x) = \ln x$ and take again left endpoints as our sample points x_i^*, but let us change the interval to $[20, 21]$ (thus $a = 20$, $b = 21$).

Problem 7 *Define an inline function $s(n)$ for computing the corresponding Riemann sums. Experiment with calculating $s(n)$ for larger and larger values of n. Does $s(n)$ appear to approach a fixed number? What appears to be its? How large needs n to get so that $s(n)$ differs from the apparent limit of the Riemann sums by no more than 0.001? Try to guess the reason why one of the functions $r(n)$, $s(n)$ appears to approach its limit faster than the other one and explain it.* Hint: *It may help you to look at a graph of $\ln x$ on the interval $[1, 21]$. To quickly produce one in* MATLAB*, use the* `ezplot`*-function that was introduced in the previous assignment.*

For convergence of Riemann sums, it is not necessary that all intervals in the partition have equal length. Sometimes partitions with unequal length work even better. However, such partitions are more difficult to handle in MATLAB. For example, consider our first function $f(x) = x^2$ on the interval $[0, 4]$. Let us partition this interval into n subintervals $[x_{i-1}, x_i]$ so that $x_i = \sqrt{\frac{16i}{n}}$, and let $x_i^* = \sqrt{\frac{16(i-1)}{n}}$ be the left endpoint of the i-th interval, where $i = 1, \ldots, n$. Then the length of the i-th interval is $\Delta_i = \sqrt{\frac{16i}{n}} - \sqrt{\frac{16(i-1)}{n}} = \frac{4}{\sqrt{n}}(\sqrt{i} - \sqrt{i-1})$, and these lengths are no longer equal. Thus the distances between the sample points x_i^* are no longer equal, and we cannot conveniently define the vector of x_i^*'s by specifying an increment. One way to overcome the problem is to specify the vector of indices $[1, 2, 3, \ldots, n]$ and define everything else in terms of this vector. Notice that $f(x_i^*) = (\sqrt{\frac{16(i-1)}{n}})^2 = \frac{16(i-1)}{n}$. This leads to the following inline function

```
>> t = inline('sum((16/n)*[0:1:n-1]*sqrt(16/n).*(sqrt([1:1:n])-
sqrt([0:1:n-1])))')
```

Be sure to enter this function exactly as above. In particular, don't forget to enter the symbols on the second line of this definition as it appears in the handout, and make sure all your brackets and quotation marks are in the right place. The whole formula should be entered as one line in the Command Window. Note the period in front of the last multiplication sign. It instructs MATLAB to multiply two vectors term by term.

Problem 8 *Experiment with computing both $t(n)$ and $g(n)$ for larger and larger values of n. Describe the behaviour of these two functions.*