INTRODUCTION TO FOR AND WHILE LOOPS IN MATLAB

For loops and while loops allow the computer to run through a series of commands, repeatedly. In the case of a for loop, the commands are executed a fixed number of times, whereas in a while loop the commands are executed until some specified condition is met. In both, the variables can change values from one iteration (= cycle through the commands of the loop) to the next. Here is the basic structure of each type of loop:

for loop:	while loop:

`for n = vector`
`   …MATLAB Commands…`
`end`
	`while <<condition>>`
	`   …MATLAB Commands…`
	`end`

In the for loop, `n` is the **counter**, and the `…MATLAB Commands…`, constituting the **body** of the loop get executed (in order) each time the counter runs through a different element of `vector`, a list of numbers. Common choices for `vector` are things like 1:10 (which in MATLAB corresponds to the vector [1 2 3 4 5 6 7 8 9 10]. The counter must be a variable and it does not have to be called `n` (we could instead call it `counter`, `count`, `month`, or whatever is convenient). The counter may, or may not appear in the body of the loop. Notice that both for and while loops require and end after the body of the loops (to complete the loop). Here is a simple example of a for loop.

EXAMPLE 1: Write a for loop to compute the sum of the squares of all integers from 2 to 20:

$$2^2+3^2+4^2+\cdots+20^2.$$

SOLUTION: We will first initialize a variable `Sum` for the cumulative sum throughout the loop (its initial value is zero):

line	MATLAB Code
1	`Sum = 0; %initialize sum`
2	`for n = 2:20`
3	`      Sum = Sum + n^2;`
4	`end`
5	`Sum %This will display the final sum`

Note that we suppressed the cumulative sums (in the body of the loop) from outputting using a semicolon (since we only cared to see the final answer). The way this loop operates, is first the counter `n` starts off at 2 (`n` = 2), the body then resets the cumulative sum from its initial value (`Sum` = 0) to this value + n^2, or 0 + 2^2 = 4. Next, `n` gets bumped to its next value in the vector 2:20 (`n` = 3), the body now updates the cumulative sum to be its previously stored value (4) plus $n^2 = 3^3 = 9$. So `Sum` now gets reset to be 13. Next `n` = 4, and this all continues until `n` has reached its final value of the vector (20) and the body is executed for the last time.[1] When this loop is entered into the command window, the only output one sees is: →Sum = 2869

Writing loops and more general programs to solve a given problem sometimes takes several attempts and some debugging, even for experienced programmers, but especially for beginners. Here are a few good general suggestions for writing a program to solve a problem. First, you must totally understand the problem and should do some "small" cases (or smaller versions of it) by hand. Then write your program/loop for such a small case in a way that will make it easy to generalize. For example, the next exercise for the reader asks to write a for loop to compute the sum $1^2+3^2+5^2+\cdots+501^2$. This sum is too large to do by hand, so instead try writing a loop that computes the smaller corresponding sum $1^2+3^2+5^2+7^2$, and check that it gives the correct answer (that you can get by hand). If it does, your loop should be set up so you can make a minimal change (say changing 7 to 501) so that it can next do the original "big" sum. If it fails to get the correct answer (for the "smaller" sum), there is no way it will work correctly for the "larger" sum, so you have to debug your program. Sometimes the error will be a basic

[1] Any time you are having difficulty understanding how a loop operates, a good idea is to (temporarily) remove any output suppressing semicolons in the body (and maybe even put in some extra commands to display additional data). The additional output can often serve as a check and help one to detect any bugs.

syntax error where the loop does not even run or give any output. MATLAB will often give you useful feedback to correct such problems. If the loop executes, but gives a wrong answer, start by taking out all semicolons in your loop so you can run the loop again and view more intermediate calculations to help you find which step the problem is occurring. Carefully examine this intermediate output data, and follow each step with hand calculations to try to find the error. If you need more feedback, you can (temporarily, for diagnostic purposes) add extra variables (without semicolons) in the loop so that their values can be displayed. Perseverance is very important here. The experience that one gains by going through this trial, error, and debugging process (sometimes several times for a given program/loop) is very valuable in learning how to write programs. Once you have your program finally working correctly for the smaller problem, remember to put back all of the semicolons you removed (and take out any extra display variables) so that when you run the program on the bigger problem, you won't be flooded with a bunch of unnecessary data.

EXERCISE FOR THE READER 1: Write a for loop to compute the sum of all of the odd integers from 1 to 501 (inclusive):

$$1^2 + 3^2 + 5^2 + \cdots + 501^2.$$

Our next example comes from finance, it is mathematically quite similar to the first one, but it will lead us to introduce a different way to track the data through the creation of a vector.[2]

EXAMPLE 2: (a) Suppose that $1000.00 is left to sit in a bank account that pays 8% interest per year, compounded annually. What is the account balance after 30 years?
(b) Create a plot of the annual balance from year 1 through year 30.

SOLUTION: Part (a): Note that in going from any year to the next, the Balance will change by having 8% of it added to itself, i.e., new Balance = current Balance + (0.08)(current Balance) = (1.08)(current Balance).

```
line  MATLAB Code
1     Balance = 1000; %initialize Balance
2     for year = 1:30
3           Balance = (1.08)*Balance;
4     end
5     Balance %This will display the final Balance
```

Here is the output when these commands are entered: →Balance = 10062.66
Part (b): To create a plot of the progression of balances over the 30 years, we need to create a vector of all of these annual balances. This can be accomplished by simply modifying line 3 of the above loop as follows:

```
3           BalanceVec(year) = (1.08)*Balance;  Balance = BalanceVec(year);
```

This new line does two things: first it places each year's balance in the corresponding slot of a vector `BalanceVec` that is being built up as the loop progresses. Also, the yearly `Balance` (a number rather than a vector) gets updated so the next iteration of the loop will work correctly. Line 5 can be deleted for this part. Once the loop with this modified line 3 is entered, the desired plot can be created with the command: `plot(1:30, BalanceVec)` See the Figure.

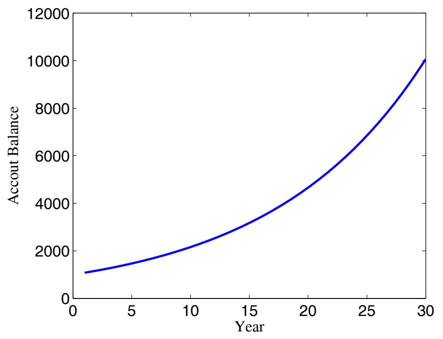

FIGURE: Progression of account balances for Example 2 (*x*- and *y*-labels have been added).

EXERCISE FOR THE READER 2: (a) Suppose that at the beginning of every year, $1000 is deposited into a retirement annuity that pays 8% interest per year. What is the account balance after 30 years? (Assume that a final deposit is made at the end of the 30th year = beginning of

[2] For finance problems, it is a good idea to work in `format bank` (in MATLAB). The numbers will display nicely as dollars and cents.

the 31st year.)
(b) Create a plot of the annual balance from year 1 through year 30.

In a while loop, a `<<condition>>` is entered next to the while operator. The loop continues to get iterated as long as the condition tests true. Such a `<<condition>>` usually depends on the variables in the body and so can change with the iterations (as it should). The `<<condition>>` can be made up using some of the equality or inequality operators (`==, >, <, >=, >=, ~=`), as well as any of MATLAB's full set of logical operators that are discussed in Chapter 4 (for OR, AND, etc.). In this introduction, we only focus on the first type of operators. All of the variables appearing in `<<condition>>` must be initialized before the loop is entered (otherwise MATLAB will not recognize them and will give an error message). For loops are simpler than while loops; the latter are useful when one does not know initially how many iterations of the body of the loop are needed, as in the next example.

EXAMPLE 3: Suppose, starting at his 25th birthday, Michael deposits \$5000 at the beginning of every year into a retirement annuity that pays 9% interest per year, compounded annually He wants to retire when his annuity first reaches or exceeds \$1 million. In how many years will he be able to retire with this plan?

SOLUTION:

MATLAB Code
```
Balance = 5000; %initialize Balance
year = 0; %initialize year counter
while Balance < 1000000
      Balance = (1.09)*Balance + 5000;
      year = year + 1; %update year counter
end
year, Balance %This will display the year and the
corresponding balance that first broke $1 million.
```

When this code is entered, the following output results: year = 34.00, Balance = 1078553.77
Thus, Michael will be able to retire at age 59, with a nest egg of \$1,078,553.77.

EXERCISE FOR THE READER 3: Suppose that the squares of all positive odd integers, $1^2+3^2+5^2+7^2+\cdots$ were to be added up until this sum first equaled or exceeds 5 million. How many terms would be added? What is the final sum?

EXERCISES:

In Exercises 1-4, for each part, write a for loop that will compute (as its only output) each of the following finite sums. Run it and give the answer (= the output). Use `format rat` so that all of the digits of these integer sums will be displayed.

1. (a) $1+2+3+4+\cdots+9999$. (b) $500+501+502+\cdots+1500$.
   (c) $500^2+501^2+502^2+\cdots+1500^2$. (d) $500+501^2+502+503^2\cdots+1500$.

   Suggestion: For part (d), the exponent changes at each iteration. Notice that it toggles between 1 and 2. One way to accomplish this would be to include an assignment line like: `exponent = 3 – exponent` in the body of the loop.

2. (a) $1+2+3+4+\cdots+20000$. (b) $50+51+52+\cdots+5050$.
   (c) $50^3+51^3+52^3+\cdots+2000^3$. (d) $50+51^3+52+53^3+\cdots+2000$.

3. (a) $1+3+5+7+\cdots+9999$. (b) $1^2+3^2+5^2+7^2+\cdots+9999^2$.
   (c) $1^2-3^2+5^2-7^2+\cdots-9999^2$. (d) $1+3^2+5+7^2+9+11^2+\cdots+9999^2$.

4. (a) $2+4+6+\cdots+9998$. (b) $2^2+4^2+6^2+\cdots+9998^2$.
   (c) $2^2-4^2+6^2-\cdots+9998^2$. (d) $2^2+4+6^2+8+\cdots+9998^2$.

5. (*Compound Interest*) Redo part (b) of Example 2 with each of the following annual interest rates: r = 6%, r = 10%, and r = 12%. Plot all three curves in the same window, along with the 8% interest rate curve of the Example.

6. (*Compound Interest versus Simple Interest*) (a) Suppose that your family has just discovered a savings account that was started by an ancient ancestor 240 years ago. The account was started with $10 and paid 6.5% interest compounded annually. What is the account balance today?
(b) Suppose instead that your ancestor had invested the $10 in an account that paid 8% simple interest. This means that each year, the interest earned was 8% of the principal (i.e., 8% of $10, or 80¢). Without using any loop (just basic arithmetic) what would the account balance of this account be today–240 years later?

7. (*Retirement Annuities*) (a) Suppose that Jackie sets up a retirement annuity that pays r = 7.5% interest, compounded annually. Starting when she turns 30 years old, Jackie deposits $3500 each year, and plans to continue this until her 65th birthday (when she makes her last deposit and plans to retire). How much will Jackie's annuity be worth then?
(b) Repeat part (a) with r changed to 9%.
(c) Repeat part (a) with r changed to 12%.

8. (*Retirement Annuities*) (a) Suppose that Wendell starts a 401(k) annuity that pays r = 10% interest, compounded annually. Starting when he turns 27 years old, Wendell deposits $2500 each year, and plans to continue this until his 70th birthday (when he makes his last deposit and plans to retire). How much will Wendell's annuity be worth then?
(b) Repeat part (a) with r changed to 8%.
(c) Repeat part (a) if Wendell starts instead when he turns 33.

9. (*Retirement Annuities*) (a) Suppose that Randall sets up a 401(k) retirement annuity that pays r = 6% interest, compounded annually. When he turns 35 years old, Randall deposits $3500, and each subsequent year he plans to deposit 5% more than the previous year (due to planned raises and better money management). He continues this until his 65th birthday (when he makes his last deposit and plans to retire). How much will Randall's annuity be worth then?
(b) Repeat part (a) if the annual deposits are doubled.
(c) Repeat part (a) if the annual interest rate is doubled.
(d) Repeat part (a) if he starts the annuity ten years earlier (when he turns 25), but still continues until his 65th birthday.

10. (*Retirement Annuities*) (a) Suppose that Evelyn sets up a retirement annuity that pays r = 7.5% interest, compounded annually. Starting when she turns 30 years old, Evelyn deposits $3500 each year, and each subsequent year she deposits an additional $100 (so the second year she puts in $3600, the third, $3700, etc.). Evelyn continues this until her 65th birthday (when she makes her last deposit and plans to retire). How much will Evelyn's annuity be worth then?
(b) Repeat part (a) if instead she deposits an additional $200 per year. So $3500 the first year, $3700 the second, $3900 the third, and so on.
(c) Repeat part (a) if the annual interest rate is changed to 9%.
(d) Repeat part (a) if she starts the annuity ten years later (when she turns 40), but still continues until his 65th birthday.

In Exercises 11-14, for each part, write a while loop that will add up the terms of the given sequence until the sum first exceeds or equals the number M that is given. You loop should have (exactly) two outputs: both the number of terms that were added up, and the sum of these terms. Run it and give the answers (= the output). Use `format rat` so that all of the digits of these integer answers will be displayed.

11. (a) $1+2+3+4+\cdots;\ M=10,000.$ (b) $500+501+502+\cdots;\ M=1,000,000.$
(c) $500^2+501^2+502^2+\cdots;\ M=1,000,000.$ (d) $500+501^2+502+503^2\cdots;\ M=1,000,000.$

Suggestion: For part (d), the exponent changes at each iteration. Notice that it toggles between 1 and 2. One way to accomplish this would be to include an assignment line like: `exponent = 3 – exponent` in the body of the loop.

12. (a) $1+2+3+4+\cdots;\ M=2,000,000.$ (b) $50+51+52+\cdots;\ M=2,000,000.$
(c) $50^3+51^3+52^3+\cdots;\ M=2,000,000,000.$ (d) $50+51^3+52+53^3+\cdots;\ M=2,000,000,000.$

13. (a) $1+3+5+7+\cdots;\ M=12,345.$ (b) $1^2+3^2+5^2+7^2+\cdots;\ M=5,000,000.$
(c) $1^2-3^2+5^2-7^2+\cdots;\ M=50,000.$ (d) $1+3^2+5+7^2+9+11^2+\cdots;\ M=25,000,000.$

14. (a) $2+4+6+\cdots;\ M=20,000.$ (b) $2^2+4^2+6^2+\cdots;\ M=1,000,000.$
(c) $2^2-4^2+6^2-\cdots;\ M=2000.$ (d) $2^2+4+6^2+8+10^2\cdots;\ M=100,000,000.$

15. (*Compound Interest*) (a) Suppose that $1000 dollars is invested in a long term CD that pays 6% interest compounded

annually. Write a while loop to determine the number of years that it would take the money to double (to equal or exceed $2000), and the resulting balance after this number of years.
(b) Redo part (a) if the amount invested is changed to $1 million.
(c) Redo part (a) with the annual interest rate being changed to 8%.
(d) Redo part (a) with the annual interest rate being changed to 4%.

16. (*Compound Interest*) (a) Suppose that $1000 dollars is invested in a long term CD that pays 7% interest compounded annually. Write a while loop to determine the number of years that it would take the money to triple (to equal or exceed $3000), and the resulting balance after this number of years.
(b) Redo part (a) if the amount invested is changed to $1 million.
(c) Redo part (a) with the annual interest rate being changed to 9%.
(d) Redo part (a) with the annual interest rate being changed to 5%.

17. (*Retirement Annuities*) For each part below, redo Example 3 with the indicated changes (the changes indicated for each part apply only to that part).
(a) Suppose that Michael instead decides to retire when his annuity reaches or exceeds $500,000.
(b) Suppose that the interest rate is 10% (rather than 9%).
(c) Suppose that rather than making the same deposit every year, because of regular raises and better money management, Michael's annual deposits grow by 6% each year. So the first year he puts in $5000, the second year $5300, the third year $5618, etc.

18. (*Retirement Annuities*) Suppose that Jocelyn sets up a 401(k) that pays r = 10% interest, compounded annually. Starting when she turns 36 years old, Jocelyn deposits $10,000 each year, and plans to continue this until her annuity meets or exceeds $1 million. How many years will it take for this to happen?
(a) Suppose instead that Jocelyn instead decides to retire when his annuity reaches or exceeds $1.5 million.
(b) Suppose instead that the interest rate is 12% (rather than 10%).
(c) Suppose that rather than making the same deposit every year, because of regular raises and better money management, Jocelyn's annual deposits grow by $1000 each year. So the first year she puts in $10,000, the second year $11,000, the third year $12,000, etc.

19. (*Paying off a Loan*) (a) Suppose that Gomez takes out a loan for $22,000 to buy a new SUV (after trading in his car). The bank charges 6% annual interest on the unpaid balance, compounded monthly (that's 0.5% interest each month). If Gomez pays $300 at the end of each month, how many months will it take him to pay off his loan? What is the amount of his last payment? His last payment will just cover the unpaid balance so will likely be less than $300.
(b) If, in writing a while loop to solve this problem you accidentally typed 220000 (rather than 22000); i.e., you added an extra zero to the loan amount. Explain what would happen. If you don't know what would happen, try it out.

20. (*Paying off a Loan*) (a) Suppose that the Jones family borrows $420,000 to buy a house. The bank charges 6% annual interest on the unpaid balance, compounded monthly. If the Jones's pay $3000 at the end of each month, how long will it take them to pay of the loan, and what will the amount of their last payment be? Their last payment will just cover the unpaid balance so will likely be less than $3000.
(b) If, in entering a while loop to solve this problem you accidentally typed 4200000 (rather than 420000); i.e., you added an extra zero to the loan amount. Explain what would happen. If you don't know what would happen, try it out.

21. (*Paying off a Loan*) (a) Repeat Exercise 20(a), but suppose instead of making the same payment every month, the Jones's increase their payments by $20 each month (through better financial planning).
(b) Repeat Exercise 20(a), but suppose instead of making the same payment every month, the Jones's increase their payments by 0.5% each month (through better financial planning).

+++

SOLUTIONS TO EXERCISES FOR THE READER:

EFR 1: In the following loop, the vector after the for operator includes exactly the odd integers in the desired range (since the gaps between successive odd integers is 2):

```
Sum = 0;
for n = 1:2:501
    Sum = Sum + n^2;
end
Sum
```

→Sum = 21084251

EFR 2: We write a single loop that will provide us with the data we need. As with all finance problems, we work in "format bank":

```
format bank
Balance = 1000;
for i=1:30
      BalanceVec(i) = 1.08*Balance + 1000; Balance = BalanceVec(i);
```

```
end
```

Part (a): `Balance` →Balance = \$123,345.87 (I added the \$ sign and comma.)
Part (b): The required plot can now be created with the command: `plot(1:30,BalanceVec)`

EFR 3: For larger integer problems (but where the number of digits is less than about 15; see Chapter 5) it is a good idea to work in "`format rat`" (stands for rational format: every output gets approximated by fractions and integers only—no decimals).

```
format rat
Sum = 0;  Counter = 0;
while Sum < 5000000
      Counter = Counter + 1; %bump up the counter
      Term = (2*Counter - 1)^2;    %this is the term that will be added to Sum
      Sum = Sum + Term;
end
```

`>> Counter` →Counter =156 (This is the number of terms that were added.)
`>> Sum` →Sum = 5061836 (This is the corresponding cumulative sum.)

The reader should notice that as the `Counter` runs though the positive integers: 1, 2, 3, 4, $\cdots$, the odd number that is being squared to be added to the cumulative Sum, `2*Counter – 1`, runs through the odd positive integers: 1, 3, 5, 7, $\cdots$. A variation of the above program would be to initialize this odd number and bump it up by two (at each iteration) in the body of the loop:

```
Sum = 0;  Counter = 0;  OddNumber = 1;
while Sum < 5000000
      Counter = Counter + 1; %bump up the counter
      Term = OddNumber^2;    %this is the term that will be added to Sum
      Sum = Sum + Term;
      OddNumber = OddNumber + 2;
end
```