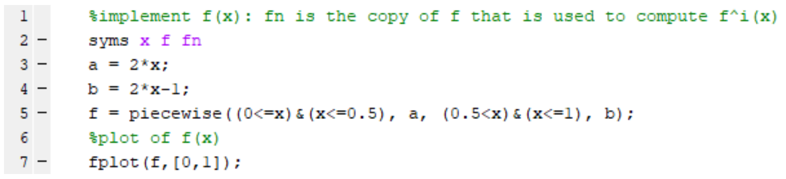
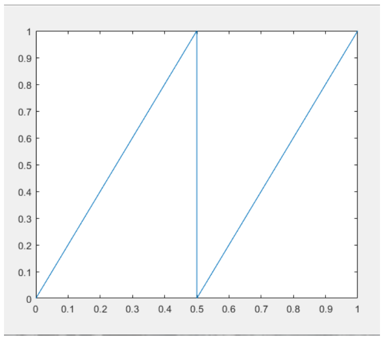
Question 1:

Lines 1-7 of the code (below) show the implementation and plot the graph (Fig. 1) of the piecewise function.



```
1      %implement f(x): fn is the copy of f that is used to compute f^i(x)
2  -   syms x f fn
3  -   a = 2*x;
4  -   b = 2*x-1;
5  -   f = piecewise((0<=x)&(x<=0.5), a,  (0.5<x)&(x<=1), b);
6      %plot of f(x)
7  -   fplot(f,[0,1]);
```



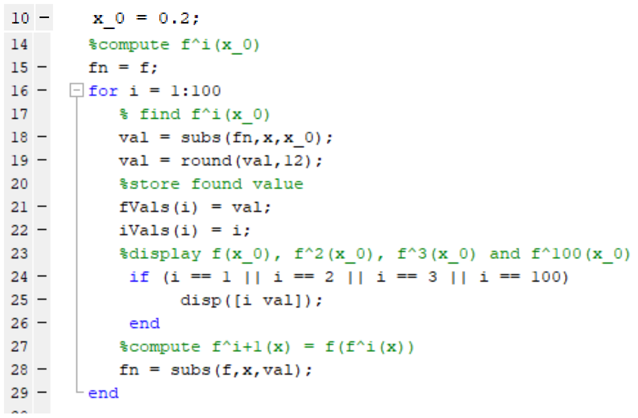
Observations: The vertical line at 0.5 should not exist. It is likely an artifact of *fplot()* not dealing with the discontinuity of the function.

Challenges: The main challenge of this question was to figure out how to implement a piecewise function and use the symbolic variables, which was overcome by reading the piecewise documentation.

Fig 1. plot of f(x)

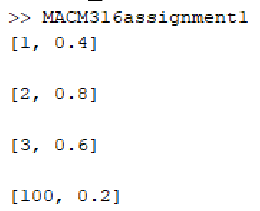
Question 2:

Lines 10, 14-29 implement a for loop which is used to calculate and display f(x_0), f^2(x_0), f^3(x_0) and f^100(x_0).



```
10 -    x_0 = 0.2;
14     %compute f^i(x_0)
15 -   fn = f;
16 -  for i = 1:100
17        % find f^i(x_0)
18 -      val = subs(fn,x,x_0);
19 -      val = round(val,12);
20        %store found value
21 -      fVals(i) = val;
22 -      iVals(i) = i;
23        %display f(x_0), f^2(x_0), f^3(x_0) and f^100(x_0)
24 -       if (i == 1 || i == 2 || i == 3 || i == 100)
25 -            disp([i val]);
26 -       end
27        %compute f^i+1(x) = f(f^i(x))
28 -      fn = subs(f,x,val);
29 -   end
```

With x_0 declared as 0.2 (in line 10) the code outputs the following values:



```
>> MACM316assignment1
[1, 0.4]

[2, 0.8]

[3, 0.6]

[100, 0.2]
```

The first number in each set of brackets is the number of composite functions i (ex i=2 is equal to f(f(x_0))) and the second is the computed value.

Observations: The output of the calculation are the expected values with x_0 = 0.2 (as seen in the example given in Question 3) and changing x_0 to a different value will also change the output. The for loop above finds the answer to every i between 1 to 100, but only displays 1, 2, 3, and 100 as per the question requirements.

Challenges: Initially I had implemented the for loop to iteratively find the composite function without a value, output the function, then calculated the value at x_0. This would run quickly for 6 iterations, then dramatically slowed down and took several minutes to compute the function f^7(x); this makes sense since it had to calculate many separate functions for an increasing number of intervals of decreasing size. Since the function itself was not required as an output I solved this performance problem by simply calculating the output f(x_0) as the variable *val* (line 18), then using that number as the input of the function *f* (line 28).

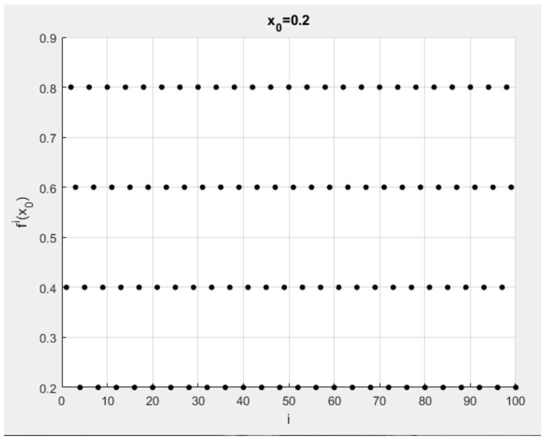
Question 3:

Still using x_0 = 0.2 (which made verification easy as it was used an example) the for-loop from question 2 (above) and lines 11, 12, and 31-37 output the graph in Fig. 2.

```
11 -    fVals = zeros(1,100);
12 -    iVals = zeros(1,100);
```

```
31      %plot sequence
32 -    scatter(iVals,fVals, 20, 'black','filled');
33 -    hold on
34 -    grid on
35 -    xlabel('i');
36 -    ylabel('f^i(x_0)')
37 -    title('x_0 = 0.2');%change to reflect x_0
```

fVals (the array declared in line 11) is filled with the numbers calculated at each iteration (line 21), which correspond to iVals (declared in line 12) the iteration number 1 to 100 (line 22); fVals corresponds to the y axis and iVals to the x axis of the scatterplot graph (lines 32-37).



Observations: This shows the value of x_0 repeating in a repeating, predictable pattern. Using another value x_0= 0.6 gave a different, but still repeating, pattern.

Challenges: No challenges encountered during this question.

Fig 2. Plot of sequence with x_0 = 0.2

Question 4:

The code for this question is the same as the code for Question 3, except since the value of x_0 is changed to either 1/pi or 0.2000000001, lines 10 and lines 37 are updated to reflect the change. The values computed are rounded to 12 decimal places (line 19) in order to remove error propagation that was occurring from around f^55(x_0) to f^100(x_0), where the values are erroneously appearing as 1 (Fig. 3 and Fig. 5 show before rounding, Fig. 4 and Fig 6. are after rounding). Since the round-off error occurs because the machine is choosing some small value to round up (< 10^-12), forcing Matlab to round the value at 12 digits prevents the computed round-off error from occurring.

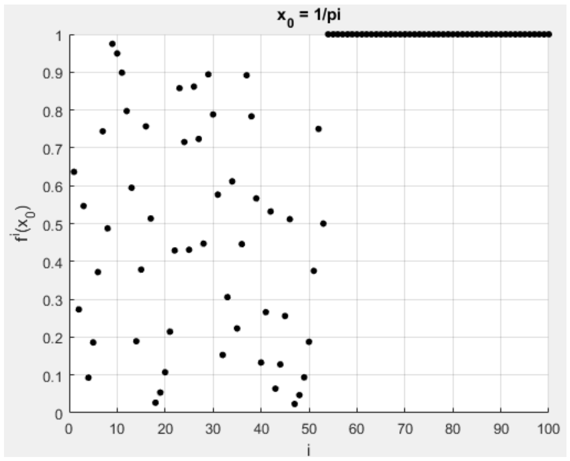


Fig. 3 Plot of sequence with x_0 = 1/pi

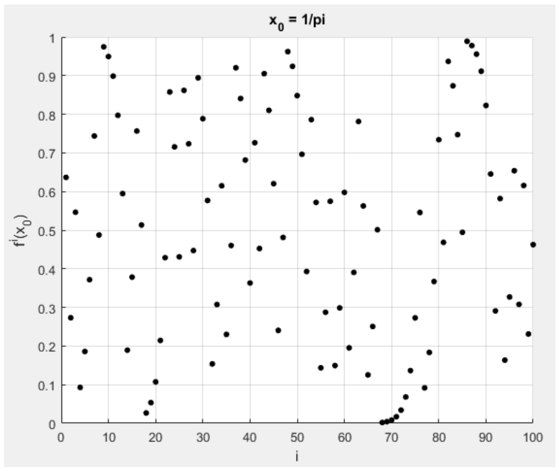


Fig 4. Plot of sequence with x_0 = 1/pi, rounded

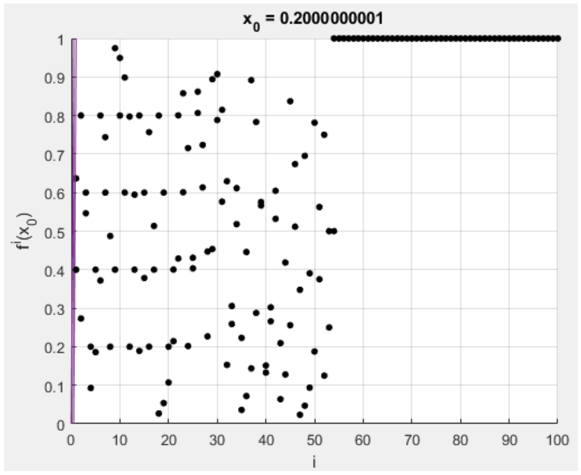


Fig. 5 Plot of sequence with x_0 = 0.2000000001

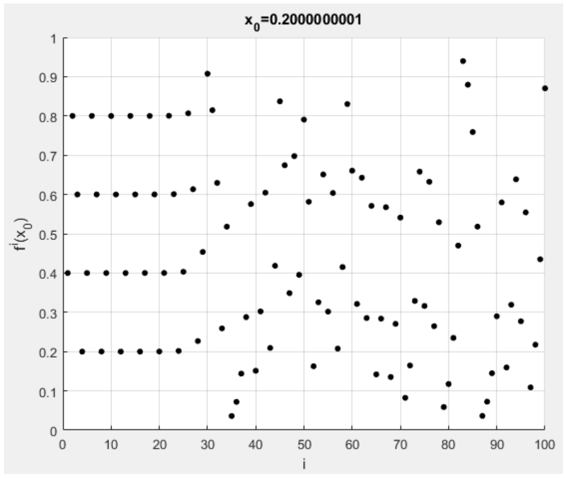


Fig 6. Plot of sequence x_0 = 0.2000000001, rounded

Observations: with x_0=1/pi: there is no repeating pattern to the graph once the error propagation is dealt with.

With x_0= 0.2000000001: prior to rounding a pattern is somewhat seen in the first 20 or so values, though these quickly devolve into noise, then go to 1 like the 1/pi graph. However, after rounding, the first 24 values are the same as the corresponding values in the graph for x_0=0.2 (Fig. 2), then vary from the pattern to the point of noise, around i=40. These both correspond to the assertions made in the description of the assignment: the unrounded data failed to reproduce the results (instead producing noise), and a very small change in the initial condition (of 0.0000000001) caused a significantly different outcome in a later state.

Matlab code

```
%implement f(x): fn is the copy of f that is used to compute f^i(x)
syms x f fn
a = 2*x;
b = 2*x-1;
f = piecewise((0<=x)&(x<=0.5), a, (0.5<x)&(x<=1), b);
%plot of f(x)
fplot(f,[0,1]);

%declare x_0 and init arrays to store the computed values
x_0 = 0.6;
fVals = zeros(1,100);
iVals = zeros(1,100);

%compute f^i(x_0)
fn = f;
for i = 1:100
       % find f^i(x_0)
       val = subs(fn,x,x_0);
       val = round(val,12);
             %store found value
       fVals(i) = val;
        iVals(i) = i;
       %display f(x_0), f^2(x_0), f^3(x_0) and f^100(x_0)
          if (i == 1 || i == 2 || i == 3 || i == 100)
              disp([i val]);
       end
          %compute f^i+1(x) = f(f^i(x))
           fn = subs(f,x,val);
end

%plot sequence
scatter(iVals,fVals, 20, 'black','filled');
hold on
grid on
xlabel('i');
ylabel('f^i(x_0)');
title('x_0 = 0.6'); %change to reflect x_0
```