Table of Contents

Project C: Hailstone numbers

Candidate number 2124124

```
clear, close all
```

Exercise C1

```
type sequence.m
```

```
function seq = sequence(a)
% Compute the sequence with entries, x_0 = a and for n>=1
% x_{n} = x_{n-1}/2     if x_{n-1} is even
%         3*x_{n-1}+1   if x_{n-1} is odd
% Stop when we have an entry that already appears in the sequence

x = a;
n = 1;
while 0 == 0
    if x(n)/2 == floor(x(n)/2)
        % then x(n) is even
        nextx = x(n)/2;
    else
        % x(n) is odd
        nextx = 3*x(n)+1;
    end
    if ismember(nextx,x)
        % this number already appears in the sequence
        x = [x, nextx];
        break
    else
        x = [x,nextx];
        n = n+1;
    end
end
seq = x;
```

Test the function sequence.m with a couple of starting values a.

```
seq1 = sequence(4)
```

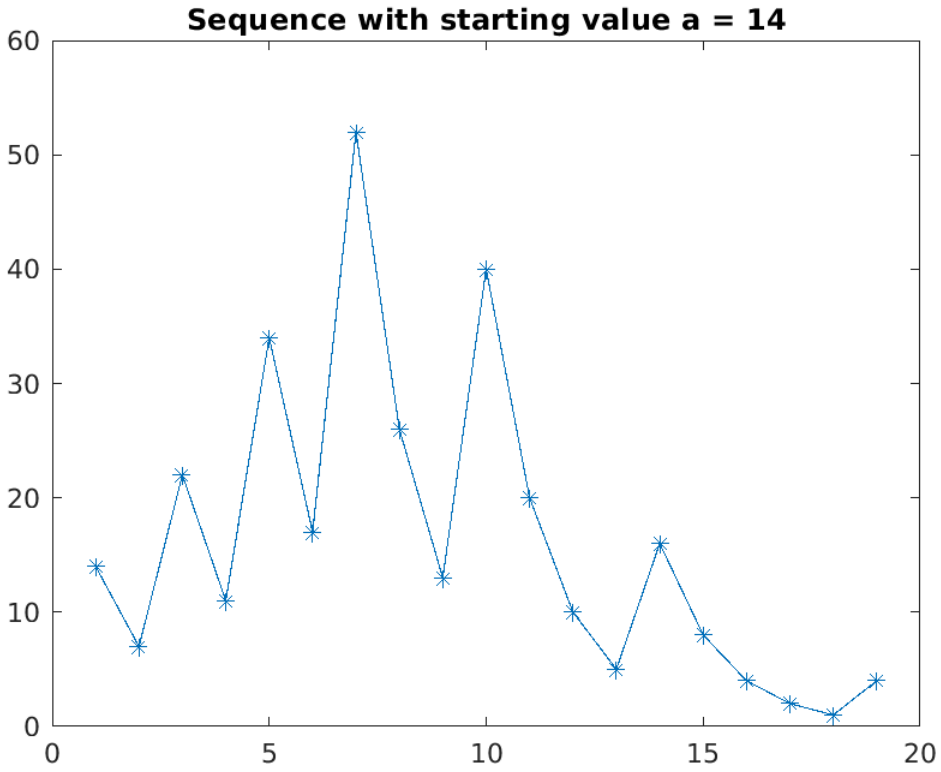
```
seq2 = sequence(11)
```

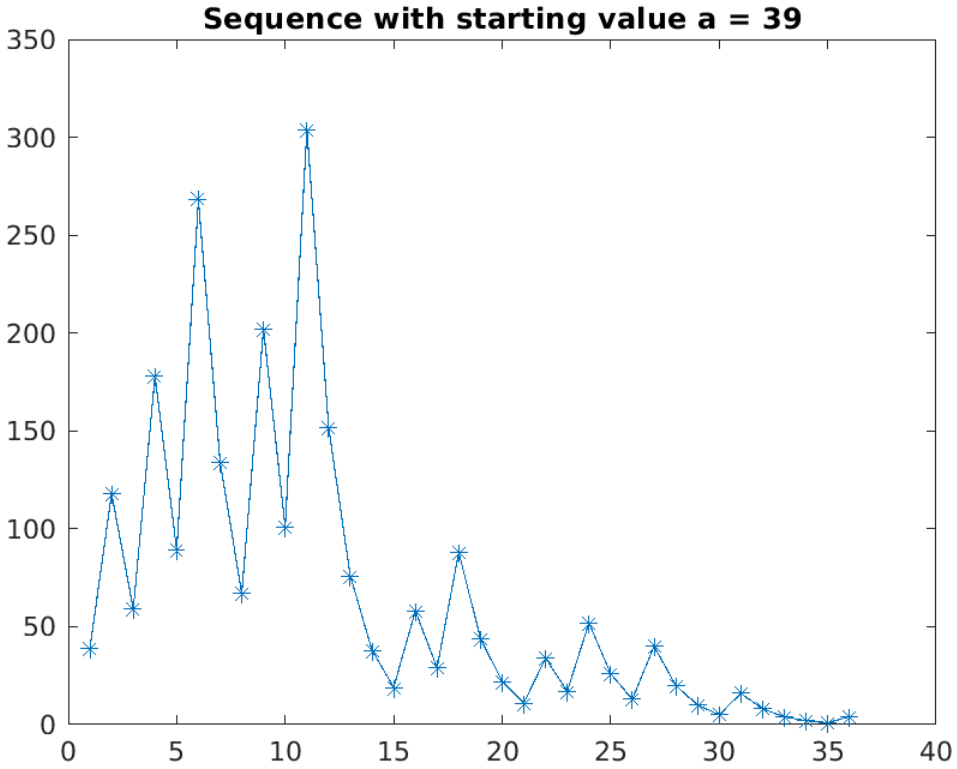
```
seq1 =
     4     2     1     4
seq2 =
  Columns 1 through 13
    11    34    17    52    26    13    40    20    10     5    16     8     4
  Columns 14 through 16
     2     1     4
```

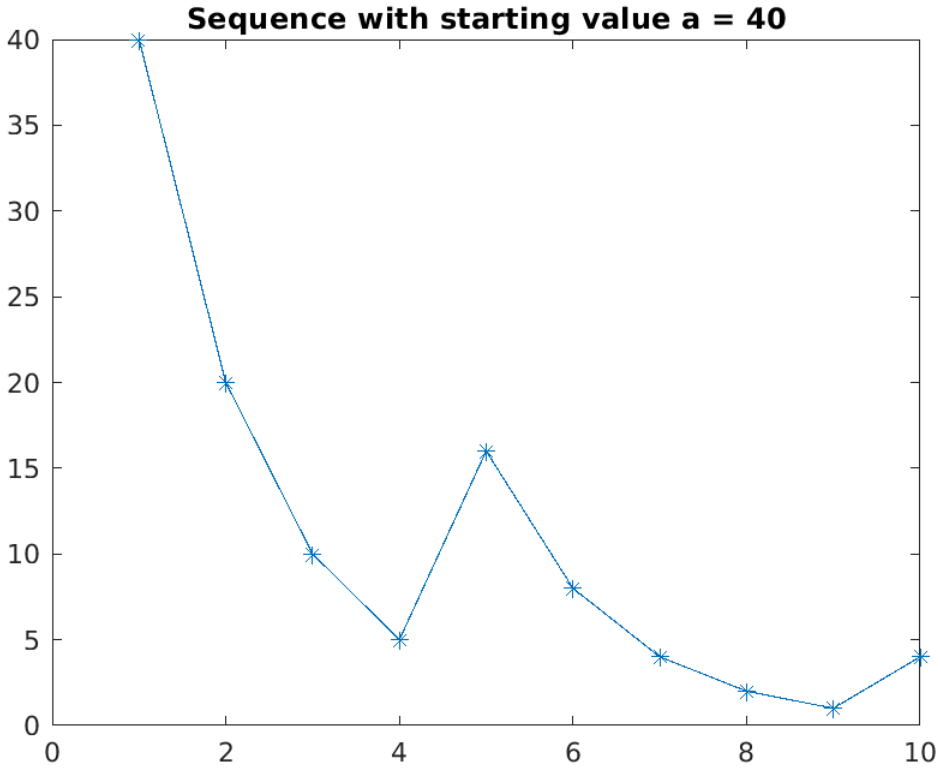
Exercise C2

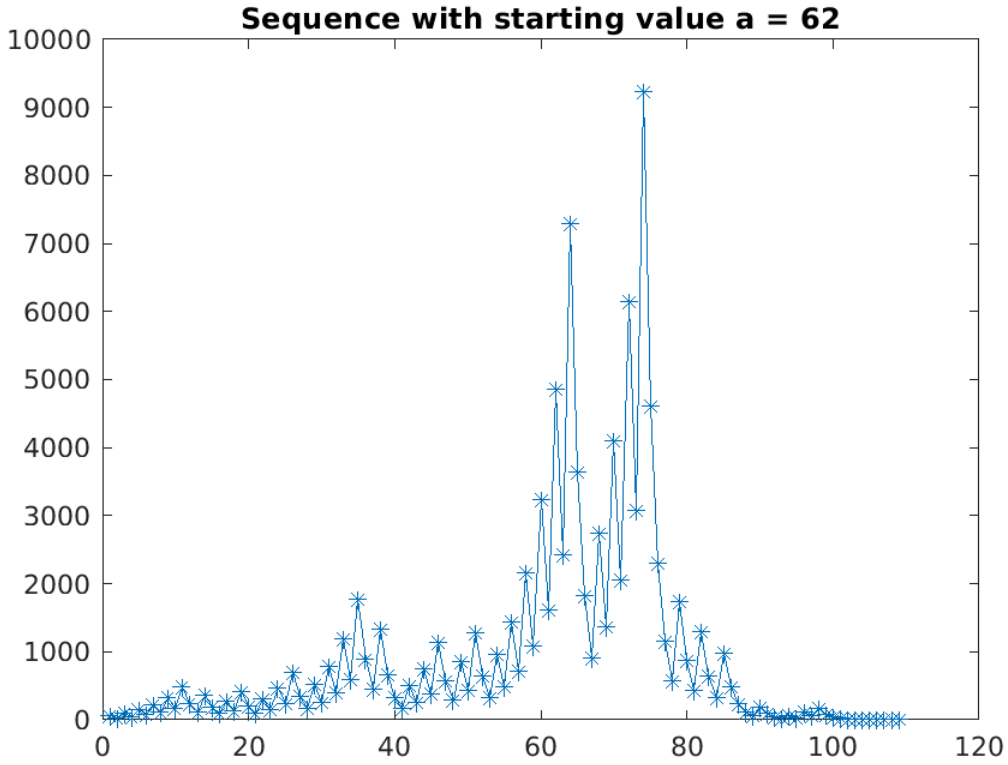
Experiment with different values of a.

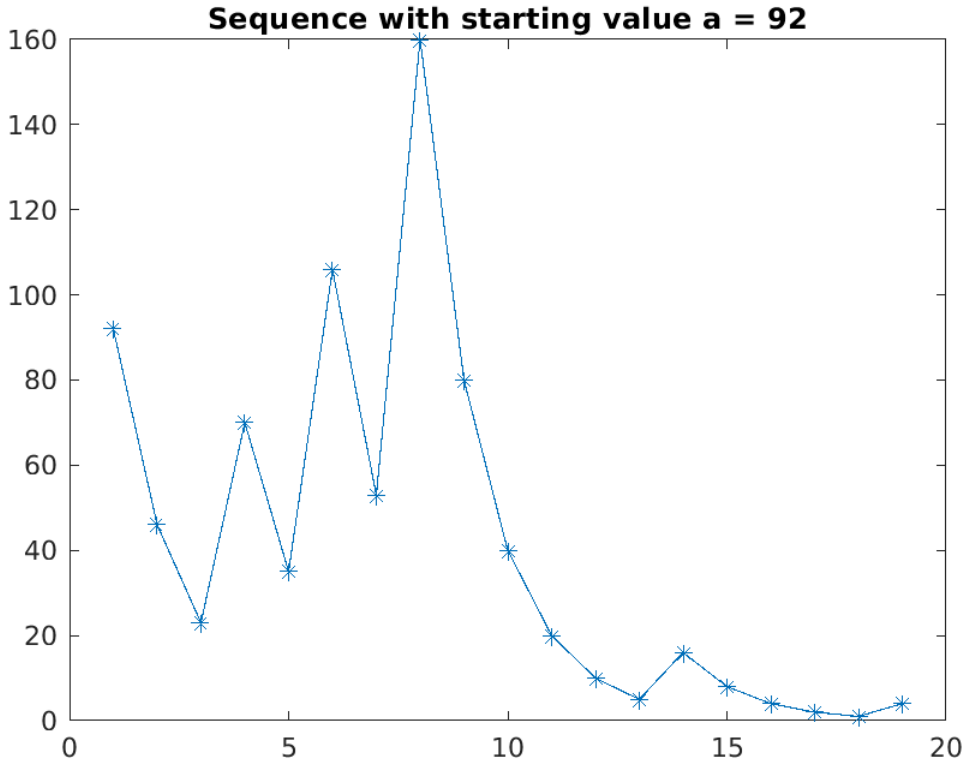
```
avals = [14, 39, 40, 62, 92];
for a = avals
    seq = sequence(a);
    figure
    plot(seq,'*-')
    title(['Sequence with starting value a = ',num2str(a)])
end
```




Sequence with starting value a = 39
350
300
250
200
150
100
50
0
0
5
10
15
20
25
30
35
40


Sequence with starting value a = 40
40
35
30
25
20
15
10
5
0
0
2
4
6
8
10


Sequence with starting value a = 62
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
120

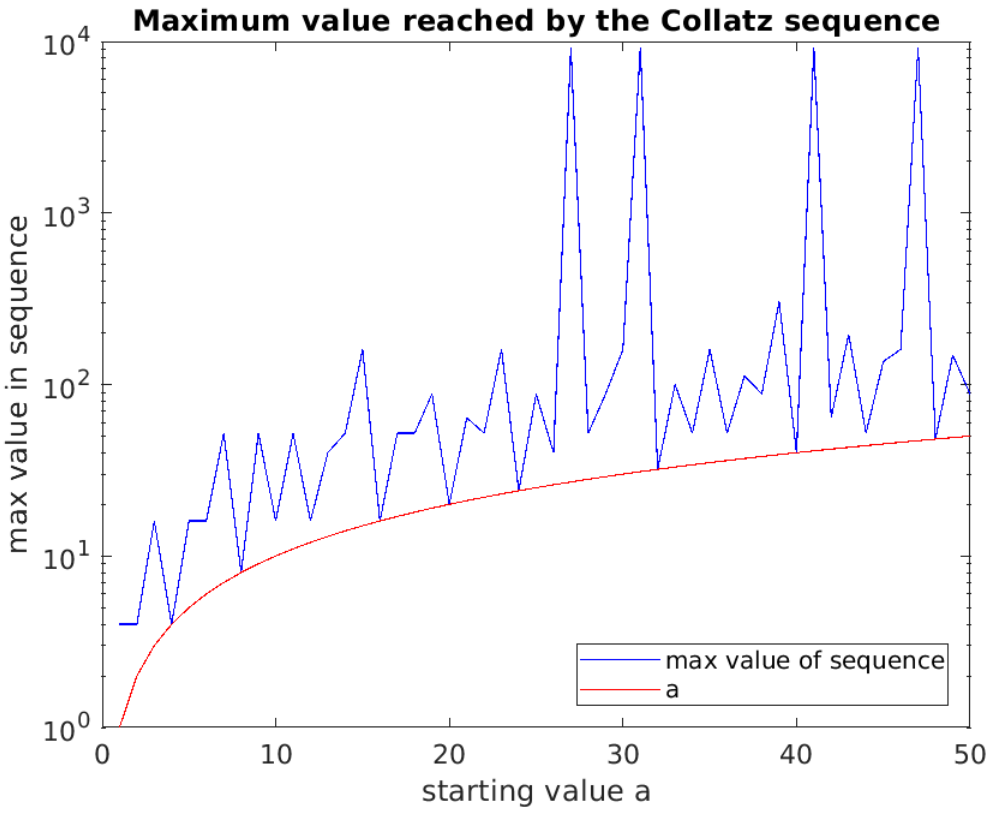

Sequence with starting value a = 92
160
140
120
100
80
60
40
20
0
0
5
10
15
20

These are called hailstone numbers because the sequence of numbers rises and falls a bit like bouncing hailstones, but eventually, for these values of a, falls to 1 (and then the sequence would just repeat 4, 2, 1).

Exercise C3

Find the maximum value in the sequence for each a from 1 to 50.

```
maxx = zeros(1,50);
for a = 1:50
    seq = sequence(a);
    maxx(a) = max(seq);
end
figure
semilogy(1:50,maxx,'b',1:50,1:50,'r')
legend('max value of sequence','a','location','southeast')
xlabel('starting value a')
ylabel('max value in sequence')
title('Maximum value reached by the Collatz sequence')
```



The maximum value in the sequence roughly increases with a. This is expected because the maximum must be at least a. Clearly if a is a power of 2, then a must be the maximum value in the sequence (except if a=2 when the maximum is 4). We can also see that if a=20, 24, 40 or 48 then a is also the maximum. In general though, the maximum is higher than a so the sequence must have increased at some point. In particular, if a is odd, then x1=3*a+1>a.

Exercise C4

Find the repeating part of the sequence.

```
avals = [14, 39, 40, 62, 92];
for a = avals
   seq = sequence(a);
   N = find(seq(1:end-1) == seq(end));
   disp(['a = ',num2str(a),': end of sequence is [',num2str(seq(N:end)),']'])
end
```

```
a = 14: end of sequence is [4  2  1  4]
a = 39: end of sequence is [4  2  1  4]
a = 40: end of sequence is [4  2  1  4]
a = 62: end of sequence is [4  2  1  4]
a = 92: end of sequence is [4  2  1  4]
```

The last part of the sequence, prior to repeats is always 4,2,1. The exception is when a=2 and we get 2,1,4.

Exercise C5

```
type sequence2.m
```

```
function seq = sequence2(a)
% Compute the sequence with entries, x_0 = a and for n>=1
% x_{n} = x_{n-1}/2     if x_{n-1} is even
%         3*x_{n-1}+1   if x_{n-1} is odd
% Stop when we have an entry that is 1

x = a;
n = 1;
while 0 == 0
    if x(n)/2 == floor(x(n)/2)
        % then x(n) is even
        nextx = x(n)/2;
    else
        % x(n) is odd
        nextx = 3*x(n)+1;
    end
    if nextx == 1
        % the sequence has got to 1
        x = [x, nextx];
        break
    else
        x = [x,nextx];
        n = n+1;
    end
end
seq = x;
```

Now check whether the sequences starting at a for a=2:10000 reach the value 1. If they do, the function sequence2.m will terminate for each a.

```
for a = 2:10000
    seq = sequence2(a);
end
disp('All cases checked')
```

```
All cases checked
```

Since the function sequence2.m completes for each value of a, the sequence gets to the value x(n)=1 for some n.

Published with MATLAB® R2022a