SOLUTIONS

Introduction to University Mathematics 2018

MATLAB Worksheet VI

die.mlx

Here's the solution to **Task 4**.

```
if r==6
        % case 1, if die shows 6
        fprintf('Congratulations! You rolled a 6.\n')
elseif r==5
        % case 2, if die shows 5
        fprintf('Oh so close!\n')
else
        % case 3, anything else
        fprintf('Bad luck. Try again!\n')
end
```

And here's the solution to **Task 5**.

```
r = randi(6);
if r==6
        fprintf('Congratulations! You rolled a 6.\n')
else
        fprintf('Bad luck. You rolled a %d. \n', r)
end
```

harmo.mlx

Clearly I’m not looking for a labour-intensive approach with a huge copy-and-pasted chunk of code.

Method I: Put another *for* loop around the previous code.

```
for m= 10:10:100
        % Harmonic series with m terms
        s = 0;
        for n=1:m
          s = s + 1/n;
        end
        fprintf(’The harmonic series with %d terms = %f \n’, m, s)
end
```

Another variation is to replace the inner *for* loop by a vectorised harmonic series (mix-and-match).

```
for m= 10:10:100
fprintf(’The harmonic series with %d terms = %f \n’, m, sum(1./[1:m]))
end
```

Whilst these methods give the right answers, they are highly inefficient. If we think about this carefully, a single calculation of the 100-term series should already give you the values of series with 10, 20, 30. . . terms.

Method II: After each line of the print-out, add 10 more terms to the previous answer.

```
s=0;
for m=10:10:100
        for n=m-9:m   % Add 10 terms at a time
            s = s+1/n;
        end
        fprintf(’The harmonic series with %d terms = %f \n’, m,s)
end
```

This method is about twice the speed of the previous methods.

Method III: Here is a nice variant of method II using a single *for* loop. We will display the result when n is a multiple of 10.

```
s=0;
for n=1:100;
    s=s+1/n;
    if rem(n,10)== 0  % display result if n is a multiple of 10
        fprintf(’The harmonic series with %d terms = %f \n’, n,s )
    end
end
```

The command `rem(n,10)` calculates the remainder when n is divided by 10. Similarly, if you know some modular arithmetic, you can use calculate $n \pmod{10}$ using the command `mod(n,10)`.

`rem(X,Y)` and `mod(X,Y)` are identical, except when `X` and `Y` have opposite signs: `rem` retains the sign of `X`, while `mod` retains the sign of `Y`.

“while” loop

Find the smallest odd number N such that

$$1 + \frac{1}{3^2} + \frac{1}{5^2} + \dots \frac{1}{N^2} > 1.2337$$

Ans: 908865 .

```
s = 0;
n = -1;
while s<=1.2337
        n = n+2;
        s = s+1/(n^2);
end
fprintf('N= %d \n', n)
```