Lecture 3: Loops and Series for $\sin(x)$

1 Syntax of `for` loops

Unit increment:

```
for i = startValue:stopValue
  block of statements
end
```

Arbitrary increment:

```
for i = startValue:inc:stopValue
  block of statements
end
```

Examples

```
for i=1:10
  disp(i);
end
```

```
for k=0:2:20
  disp(k);
end
```

```
for k=0:2:21
  disp(k);
end
```

```
for j=10:-1:1
  disp(j);
end
```

```
x = [1 4 9 -7];
for i=1:length(x)
  disp(x(i));
end
```

```
for a=0:(pi/6):pi
  disp(a);
end
```

Example: Compute the average of elements in x

```
function ave = myAverage(x)
% myAverage  Compute average of elements in the input vector

n = length(x);
s = x(1);
for i=2:n
  s = s + x(i);
end
ave = s/n;
```

Example: Compute $n!$

```
function f = myfactorial(n)
% myfactorial  Compute factorial of n

f = 1;
for i=2:n
  f = f*i;
end
```

See also the built-in functions `cumprod` and `factorial`

2 Use `for` loops to evaluate n terms of a series

Example: Compute n terms of Series approximation to $\sin(x)$

```
function s = nTermSine1(x,n)
% nTermSine1  Evaluate the n-term series approximation to sin(x)
%             Simplest approach: evaluate each term from scratch
%
% Synopsis:  s = nTermSine1(x,n)
%
% Input: x = argument of sine(x)
%        n = number of terms in the series
%
% Output: s = approximation to sin(x) with n terms of the series

term = x;
s = term;   %  Initialize the sum and the sign of the term
sgn = 1;
fprintf('\n    i   sign    k        term             s\n');
fprintf(' %4d  %4d  %4d  %18.13f  %8.5f\n',1,sgn,1,term,s);
for i=2:n
  sgn = -sgn;          %  switch sign of term
  k = 2*i - 1;
  term = sgn*(x^k)/factorial(k);
  s = s + term;
  fprintf(' %4d  %4d  %4d  %18.13f  %8.5f\n',i,sgn,k,term,s)
end
```

Recursive evaluation of terms

Improve the efficiency of `nTermSine1` by eliminating redundant calculations. See Example 5.7, pp. 217–219. Reducing the number of calculations usually improves accuracy because the roundoff error present in each calculation is minimized: fewer calculations mean less roundoff.

Observe the Patterns:

$$s = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \cdots$$

1. The x part of each term is x^2 times the x part of the preceding term.
2. The factorial of the current term can be obtained by two additional multiplications with the factorial of the preceding term.

$$\frac{x^k}{k!} = \underbrace{\frac{x^{k-2}}{(k-2)!}}_{\text{previous term}} \; \frac{x^2}{k(k-1)}$$

3. If n terms are evaluated, the maximum power of x is $2n-1$.

Use preceding observations to write the following code chunk. (Not a complete program.)

```
xterm = x;    s = xterm;    sgn = 1;
for i=3:2:(2*n-1)
  sgn = -sgn;
  xterm = xterm*x^2;
  s = s + sgn*xterm/factorial(i);
end
```

This is called a *recursive* evaluation of the terms: calculate the current term by modifying the previous term.

The factorial calculation can also be done recursively

```
xterm = x;    fact = 1;    s = xterm;    sgn = 1;
for i=3:2:(2*n-1)
  sgn = -sgn;
  xterm = xterm*x^2;
  fact = fact*i*(i-1);
  s = s + sgn*xterm/fact;
end
```

Or, just combine the terms and eliminate `sgn`

```
term = x;    s = term;
for i=3:2:(2*n-1)
  term = -term*(x^2)/(i*(i-1));
  s = s + term;
end
```

The complete m-file is `nTermSine2.m` listed below.

```
function s = nTermSine2(x,n)
% nTermSine2  Evaluate the n-term series approximation to sin(x)
%             Recursive evaluation of each term
%
% Synopsis:  s = nTermSine2(x,n)
%
% Input: x = argument of sine(x)
%        n = number of terms in the series
%
% Output: s = approximation to sin(x) with n terms of the series

term = x;
s = term;   %  initialize the sum and the sign of the term
fprintf('\n    i       term             s\n');
fprintf(' %4d  %18.13f  %8.5f\n',1,term,s);
for i=3:2:(2*n-1)
  term = -term*(x^2)/(i*(i-1));
  s = s + term;
  fprintf(' %4d  %18.13f  %8.5f\n',i,term,s)
end
```