Lecture 4: Automatic Truncation of Series for $\sin(x)$

1 Learning objectives

At the end of this class you should be able to...

- give numerical values equivalent to the logical expressions "true" and "false"
- implement a basic "`if` construct.
- write a MATLAB expression to test convergence of a series

2 Motivation

As x increases, more terms in the series representation of $\sin(x)$ are needed for convergence.

```
>> sn = nTermSine2(pi/3,5);

>> sn = nTermSine2(4*pi/3,5);

>> sn = nTermSine2(4*pi/3,10);
```

We want to add flexibility to the code: terminate the series only when "`term`" is small compared to "s".

3 "`if...end`" Syntax

```
if condition
  block of statements
end
```

The block of statements is executed only if the *`condition`* is true

Examples

```
if x<0
  disp('x is negative');
end
```

4 True and False

True:	`true`	1	Anything but zero or "
False:	`false`	0	

Examples

```
>> x = 5;
>> x < 0
ans =
     0
>> (x-5) > 0
ans =
     0
>> (x-3) > 0
ans =
     1
```

5 Automatic Truncation of Series for $\sin(x)$

5.1 Explore Truncation Criteria

Version 1:

```
if term < s/1000
    break;
end
```

what if `term` or `s` is negative?

Version 2:

```
if abs(term) < abs(s)/1000
    break;
end
```

Version 3:

```
if abs(term/s) < 1/1000
    break;
end
```

Version 4: Complete loop

```
tol = 5e-6;
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)
  if abs(term/s) < tol
      break;
  end
end
```

```
function s = nTermSine3(x,n)
% nTermSine3  Evaluate the n-term series approximation to sin(x)
%             Recursive evaluation of terms and check convergence
%
% Synopsis:  s = nTermSine3(x,n)
%
% Input: x = argument of sine(x)
%        n = number of terms in the series
%
% Output: s = approximation to sin(x) with a maximum of n terms
%             of the series.  Stop when abs(term/sum) < tol
%             where tol = 5e-6

term = x;
s = term;   %  initialize the sum and the sign of the term
tol = 5e-6;
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)
  if abs(term/s) < tol
    break;
  end
end
```

Testing:

```
>> sn = nTermSine3(4*pi/3,10);

>> sn = nTermSine3(4*pi/3,30);
```

6 `while` loop

```
while condition
  block of statements
end
```

The *`block of statements`* is repeated as long as *`condition`* is true.

6.1 Implement $\sin(x)$ series with a `while` loop

```
function s = nTermSine4(x)
% nTermSine4  Evaluate series approximation to sin(x). Use recursive
%             evaluation of terms, while loop and convergence che
%
% Synopsis:  s = nTermSine4(x)
%
% Input: x = argument of sine(x)
%
% Output: s = approximation to sin(x) with a max of n terms.  Stop when
%             abs(term/sum) < tol, where tol = 5e-6

term = x;   s = term;   %  initialize the sum and the sign of the term
tol = 5e-6;
fprintf('\n    i       term              s\n');
fprintf(' %4d  %18.13f  %8.5f\n',1,term,s);
i = 1;
while abs(term/s)>tol
  i = i + 2;
  term = -term*(x^2)/(i*(i-1));
  s = s + term;
  fprintf(' %4d  %18.13f  %8.5f\n',i,term,s)
end
```

```
>> sn = nTermSine4(4*pi/3);
```

Trouble: If convergence criterion is never met, the `while` loop will repeat indefinitely, i.e. it will be an *infinite loop*

Solution: Limit the number of times the while loop can be executed.

```
function s = nTermSine5(x,maxterms)
% nTermSine4  Evaluate series approximation to sin(x). Use recursive
%             evaluation of terms, while loop, convergence check and limit
%             maximum number of terms
%
% Synopsis:  s = nTermSine5(x,n)
%
% Input: x = argument of sine(x)
%        maxterms = maximum number of terms in the series
%
% Output: s = approximation to sin(x) with a max of n terms.  Stop when
%             abs(term/sum) < tol, where tol = 5e-6

term = x;  s = term;   %  initialize the sum and the sign of the term
tol = 5e-6;
fprintf('\n    i       term              s\n');
fprintf(' %4d  %18.13f  %8.5f\n',1,term,s);
i = 1;  n=1;
while abs(term/s)>tol && n<maxterms
  i = i + 2;  n = n + 1;
  term = -term*(x^2)/(i*(i-1));
  s = s + term;
  fprintf(' %4d  %18.13f  %8.5f\n',i,term,s)
end
```

```
>> sn = nTermSine5(4*pi/3,5);

>> sn = nTermSine5(4*pi/3,15);
```