spmd basic examples

spmd - single program, multiple data html http://www.mathworks.com/examples/parallel-computing/400-using-gop-to-achieve-mpi_allreduce-functionality#4 /html

spmd - Single Program Multiple data

In contrast to parfor-loops, each worker executing an spmd-statement is assigned a unique value of `labindex` that let's you adrress the worker explicitly.

- An spmd block runs on the workers of the existing parallel pool. If no pool exists, spmd will start a new parallel pool, unless the automatic starting of pools is disabled in your parallel preferences. If there is no parallel pool and spmd cannot start one, the code runs serially in the client session.
- There are two forms of indexing a Composite, comparable to indexing a cell array:

`AA{n} returns the values of AA from worker n.`

`AA(n) returns a cell array of the content of AA from worker n`

First example

n=4; % Triton

```
n=2; % laptop
myPool = parpool(n);
```

```
spmd
    x = labindex
end
```

```
Lab 1:

  x =

       1

Lab 2:

  x =

       2
```

```
x
```

```
x =

   Lab 1: class = double, size = [1  1]
   Lab 2: class = double, size = [1  1]
```

Sum over workers:

```
spmd
    s = gplus(x);
   % s = gplus(x, 1);   % Return result (sum) in s{1} only

end
s
```

```
s =

   Lab 1: class = double, size = [1  1]
   Lab 2: class = double, size = [1  1]
```

```
[s{1},s{2}]
```

```
ans =

     3     3
```

Catenate: gcat

```
spmd
    y1 = gcat(x, 1); % Concatenate along rows.
    y2 = gcat(x, 2); % Concatenate along columns.
end
y1{1}
```

```
ans =

     1
     2
```

```
y2{1}
```

```
ans =

     1     2
```

Values from workers to client

```
v=[x{1} x{2}]
```

```
v =

     1     2
```

```
class(v)
```

```
ans = double
```

```
class(x)
```

```
ans = Composite
```

More generally

```
V=cell(1,2)
```

```
V =
    []    []
```

```
V(1:2)=x(1:2)
```

```
V =
    [1]    [2]
```

```
class(V)
```

```
ans = cell
```

```
class(x)
```

```
ans = Composite
```

Another example

spmd(3) % This is one way of choosing nr. of workers

```
spmd(2)    % 2 is default for laptop, thus is not needed
    if labindex==1
      R=rand(3,3);  % On worker #1
    else
      R=magic(4);   % On workers #2 (and #3)
    end
```

```
end
```

```
R
```

```
R =

   Lab 1: class = double, size = [3  3]
   Lab 2: class = double, size = [4  4]
```

Values computed inside the spmd-statement's body are returned in the form of **Composite object** on the client. It contains references to the values stored in the remote workers. These values can be retrieved using cell-array indexing, as noted above.

```
R{1:2}
```

```
ans =

    0.3246    0.6497    0.8691
    0.6618    0.1471    0.2843
    0.6349    0.1767    0.8364
ans =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

The values are retained until the pool is closed. As shown above, the values can be brought to the client as a cell array:

```
RR=cell(2,1);
RR(:)=R(:)
```

```
RR =
    [3×3 double]
    [4×4 double]
```

```
% After the pool is closed, composite objects disapper, the cell arrays
% (like RR) remain.
```

```
delete(gcp)
```

R{:} % Leads to error.

```
RR{:}
```

```
ans =

    0.3246    0.6497    0.8691
    0.6618    0.1471    0.2843
    0.6349    0.1767    0.8364
ans =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

spmd can do mpmd -- Multiple programs possible

```
flist={@cos,@sin}
```

```
flist =
    @cos    @sin
```

```
data1=(-pi:0.1:0)';
data2=(0:0.1:pi)';
datalist={data1,data2};
```

```
spmd
   y= flist{labindex}(datalist{labindex});
end
```

```
Starting parallel pool (parpool) using the 'local' profile ... connected to 2 workers.
```

```
y{1}
```

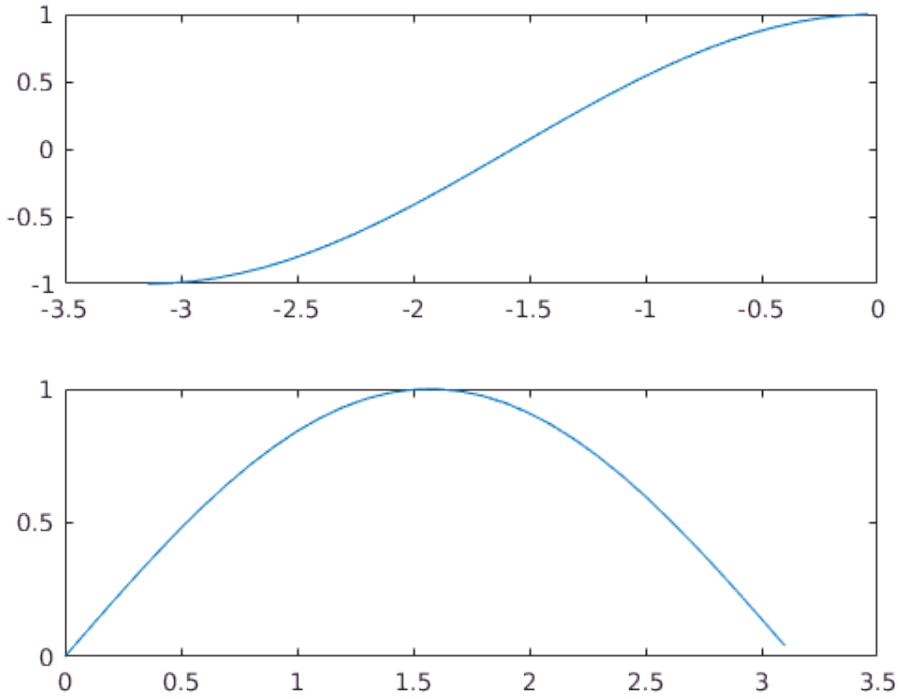
```
ans =

   -1.0000
   -0.9950
   -0.9801
   -0.9553
   -0.9211
   -0.8776
   -0.8253
   -0.7648
   -0.6967
   -0.6216
```

```
subplot(2,1,1)

 plot(data1,y{1})

subplot(2,1,2)
 plot(data2,y{2})
```




```
shg
% One way of piecewise plotting.
%
```

NOTE: Instead of cos and sin one can have extensive programs.

```
delete(gcp)
```