In the first example, `test_parrayfun_1.m`, the function:

\[ y(n) = \int_0^{\pi} \left(\cos^n(x) + \sin^{n-1}(x)\right) dx \]

is calculated for \( n \in [0, n_{\text{max}}] \), using four different ways. A large number of points is used, \( n_{\text{max}} = 10,000 \), in order to clearly show the calculation time saving when using multiple microprocessor cores by means of `parrayfun` of the `parallel` package.

**test_parrayfun_1.m**

```matlab
# In this script 4 ways of calculating the values of a one-dimensional function
# are compared, the time taken by each way is measured and it is verified that
# there are no discrepancies in the results.

pkg load parallel

nmax = 10000;  # Number of points where the function is calculated

function [a,b] = myfun(n);  # Function used in this test
    a = pi* (n-2)/n;
    f = @(x) (cos(x).^n + sin(x).^ (n-1));
    b = quadgk(f,0,a);
endfunction

# First method, using a for loop, defining the function to calculate
# within the loop.
 tic
    for n = 1:nmax;
        a1(n) = pi* (n-2)/n;
        b1(n) = quadgk(@(x) (cos(x).^n + sin(x).^ (n-1)),0,a1(n));
    endfor
 t1 = toc

# Second method, using a for loop and calling "myfun"
 tic
    for n = 1:nmax [a2(n),b2(n)] = myfun(n); endfor
 t2 = toc

# Third method, using arrayfun to call "myfun"
 tic
    ni = 1:nmax; [a3,b3] = arrayfun("myfun",ni);
 t3 = toc

# Forth method, using parrayfun to call "myfun"
 tic
    ni = 1:nmax; [a4,b4] = pararrayfun(4,@(n) myfun(n),ni);
 t4 = toc

# Are discrepancies in the results?
 discrepancies_1 = max(a2-a1) + max(b2-b1) + max(a3-a1)
 discrepancies_2 = max(b3-b1) + max(a4-a1) + max(b4-b1)

Results

\[
\begin{align*}
    t1 &= 19.212 \text{ sec} \\
    t2 &= 19.419 \text{ sec} \\
    t3 &= 19.324 \text{ sec} \\
    t4 &= 6.2121 \text{ sec} \\
    \text{discrepancies}_1 &= 0 \\
    \text{discrepancies}_2 &= 0
\end{align*}
\]

It can be seen that the `parrayfun` function, using all the 4 processor cores, divides the calculation time by 3.
In the second example, **test_parrayfun_2.m**, a 2D function:

\[
  z(x_0, y_0) = \int_{-L}^{L} \int_{-L}^{L} \left[ \cos\left(\frac{(x-x_0)^2 + (y-y_0)^2}{L}\right) \right]^2 \, dx \, dy \
\]

with \( x_0, y_0 \in [-0.8 \cdot L, 0.8 \cdot L] \)

is calculated for a two dimension array of points, \((x_0, y_0)\), 51 x 51 = 2601 points. In this case, the calculation time for each of these points is not negligible.

**test_parrarrayfun_2.m**

# In this script 2 ways of calculating the values of a two-dimensional function are compared, the time taken by each way is measured and it is verified that there are no discrepancies in the results. Each of the function values is calculated by means of a two-dimensional integral.

pkg load parallel

# Square root of the number of points where the function is calculated. 
npo = 51;

# Dimensions of the integration domain
L = 10; xa = -L; xb = L; ya = -L; yb = L;

# Function integrand definition
function intg = integrando(x,y,xo,yo,L)
    intg = \cos(((x-xo)^2 + (y-yo)^2)/L)^2;
endfunction

# Numerical integration definition
function res = Int_Num(xo,yo,L,xa,xb,ya,yb);
    res = dblquad(@(x,y) integrando(x,y,xo,yo,L), xa, xb, ya, yb);
endfunction

# Fist method, using two for loops, defining the function to calculate within the double loop.
tic
for m = 1:npo
    xo = L*0.8*((2*(m-1)/(npo-1))-1);
for l = 1:npo
    yo = L*0.8*((2*(l-1)/(npo-1))-1);
    INTENSITY_1(m,l) = dblquad(@(x,y) integrando(x,y,xo,yo,L), xa, xb, ya, yb);
endfor
endfor
t1 = toc

# Second method, using pararrayfun to call Int_Num
range = linspace(-L*0.8,L*0.8,npo);
[xo,yo] = meshgrid(range);
tic
INTENSITY_2 = pararrayfun(4,@(xo,yo) Int_Num(xo,yo,L,xa,xb,ya,yb),xo,yo);
t2 = toc
discrepancy = max(max(INTENSITY_2-INTENSITY_1))

Results

\[
\begin{align*}
  t1 &= 1789.05 \text{ sec} = 29 \text{ min } 49.05 \text{ sec} \\
  t2 &= 472.984 \text{ sec} = 7 \text{ min } 53 \text{ sec} \\
  t1/t2 &= 3.78 \\
  \text{discrepancy} &= 1.1369e-13 \quad \# \text{maximum discrepancy}
\end{align*}
\]
It can be seen that the `pararrayfun` function, using all the 4 processor cores, divides the calculation time by \textit{3.78}.

An additional advantage of using `pararrayfun` is that it informs you of the calculations that are already done, and therefore of what remains to be done. Ex:

`parcellfun`: 525/2061 jobs done

The calculation times for these two examples were obtained using a PC with a CPU Intel i52500K @ 3.30 GHz with 4 cores and 4 threads.