Ejercicio 1:

El siguiente algoritmo simula la transmisión de calor en dos dimensiones.

```
function calor_dif(nx)
dz=0.05;
dt=0.005;
q=2;
alpz=0.01;
tfin=30;
conc=zeros(nx+2,nx+2);
conc(2,2)=q;
conaux=conc;
iter=0;
tim=0;
tiempo_calculo=0

while tim<tfin
  tim=tim+dt;
  conaux=conc;
  tic
  %Este es el doble bucle que hay que paralelizar
  for row=1:nx+2
   for col=1:nx+2
     rowU=max(1,row-1);rowD=min(nx+2,row+1);
     colL=max(1,col-1);colR=min(nx+2,col+1);

conc(row,col)=conaux(row,col)+dt*(conaux(rowU,col)+conaux(row,co
lL)-
4*conaux(row,col)+conaux(rowD,col)+conaux(row,colR))*alpz/(dz*dz
);

   end
  end
  % fin del doble bucle
tiempo_calculo=tiempo_calculo+toc;

  if tim<10
     conc(2,2)=q;
    end
   iter=iter+1;
   if rem(iter,500)==1
      contourf(conc(1:nx+1,:))
      colorbar
      tim
      pause
   end
end
tiempo_calculo
end
```

1) Paraleliza el bucle for row=1:nx+2 usando spmd. Se puede hacer de muchas formas, pero se sugiere hacerlo de forma parecida al ejemplo de las raíces (trasparencias 10 o 11 de sesión 2). paralelizando el bucle más externo. Como en esas transparencias, es necesaria una etapa fuera de la región paralela, para meter los resultados en un único array.

2) Paralelizar con arrayfun el doble bucle: de forma que el cálculo se ejecute en la GPU, de forma similar al uno de los ejemplos de la sesión 3

```
for row=1:nx+2
 for col=1:nx+2
   rowU=max(1,row-1);rowD=min(nx+2,row+1);
   colL=max(1,col-1);colR=min(nx+2,col+1);
  conc(row,col)=conaux(row,col)+dt*(conaux(rowU,col)+…
    conaux(row,colL)-4*conaux(row,col)+conaux(rowD,col)+…
    conaux(row,colR))*alpz/(dz*dz);

 end
end
```

No os preocupéis por que las versiones paralelizadas vayan más rápido o más lento que la versión secuencial: Lo importante es usar las instrucciones que se piden y que el resultado final sea el correcto.