Supplemental material for Newton method

- A system of two non-linear equations.

$$0 = u'(c_0) - \beta R_1 u'(c_1)$$
$$0 = w_0 + \frac{w_1}{R_1} - c_0 - \frac{c_1}{R_1}$$

- If you do not specify the option of `fsolve`, the result may be too imprecise. Let's make the Newton method more precise.
- I specify the option for the newton method as
  `optimoptions('fsolve','Display','iter','OptimalityTolerance',1e-16);`
- I also add `tic;` and `tocl;` before/after the loop. It measures the computation time.

```
clear;clc;close all;
bt=0.95;sig=1.2;w0=0;w1=200; % parameters
numgrid = 100; % number of grids
Rvec = linspace(0.8,1.2,numgrid);
c0vec = zeros(1,numgrid);  % container of the resuts.
c1vec = zeros(1,numgrid);  % container of the resuts.

tic; % start stopwatch
for i=1:numgrid
    disp(i); % show iteration
    R = Rvec(i);
    parameters = [bt R sig w0 w1]; % packing
    fh = @(x) f(x,parameters);  % function handle
    % "..." means the same line continues.
    % In the default, the Newton method stops
    % when the difference in f(x) < 1e-6
    % But it is too imprecise, so let's change to 1e-16
    options = optimoptions('fsolve',...
        'OptimalityTolerance',1e-16);
    csolution = real(fsolve(fh,[100 100],options));
    c0vec(i) = csolution(1);
    c1vec(i) = csolution(2);
end
toc; % stop the stopwatch

plot(Rvec,c0vec);
hold on;
plot(Rvec,c1vec);

function y=f(ccc,p)
    % unpacking
    bt=p(1);R=p(2);sig=p(3);w0=p(4);w1=p(5);
    c0=ccc(1);c1=ccc(2);
    y(1) = c0^(-sig) - bt*R*c1^(-sig);
    y(2) = w0 + (w1/R) - c0 - (c1/R);
end
```

- If you change *numgrid* = 10000, the computation takes time. My compter spends 71 seconds.
- Take a look at CPU usage. You notice some CPU cores work hard, but others do not.
  - Windows: Windows Menu → Windows System → Task Manager → Performance → Open Resource Monitor
  - Mac: Applications → Utilities → Activity Monitor → Window Menu → CPU Usage
- All CPU cores should work hard. We can divide the 10000 iteration to each core. It is called *Parallel computing*. With 8 cores, the calculation takes 15 seconds.

```
clear;clc;close all;
bt=0.95;sig=1.2;w0=0;w1=200;
numgrid = 10000;
Rvec = linspace(0.8,1.2,numgrid);
c0vec = zeros(1,numgrid);
c1vec = zeros(1,numgrid);

parpool('local') % start parallel

tic;
parfor i=1:numgrid % parallel for loop
    disp(i);
    R = Rvec(i);
    parameters = [bt R sig w0 w1];
    fh = @(x) f(x,parameters);
    options = optimoptions('fsolve',...
        'OptimalityTolerance',1e-16);
    csolution = real(fsolve(fh,[100 100],options));
    c0vec(i) = csolution(1);
    c1vec(i) = csolution(2);
end
toc;

delete(gcp('nocreate')); % shutdown parallel

plot(Rvec,c0vec);
hold on;
plot(Rvec,c1vec);
```

```
function y=f(ccc,p)
    bt=p(1);R=p(2);sig=p(3);w0=p(4);w1=p(5);
    c0=ccc(1);c1=ccc(2);
    y(1) = c0^(-sig) - bt*R*c1^(-sig);
    y(2) = w0 + (w1/R) - c0 - (c1/R);
end
```