Iteration and recursion

Question 1 **10 marks**

Consider the nonlinear equation

$$f(x) = \sin(\pi x) - x^2$$

(**a**) Can you "solve the equation by hand", i.e. express the solution x^* explicitly in terms of elementary functions and the constant π?

No, this is impossible. There is no such explicit expression for the root x^*.

(**b**) Show that the equation $f(x) = 0$ has the same solution(s) as $g(x) = x$ if

$$g(x) = \frac{1}{2\pi}\sin(\pi x) - \frac{1}{2\pi}x^2 + x$$

$$\begin{aligned} g(x) = x \quad &\Leftrightarrow \\ \frac{1}{2\pi}\sin(\pi x) - \frac{1}{2\pi}x^2 + x = x \quad &\Leftrightarrow \\ \frac{1}{2\pi}\sin(\pi x) - \frac{1}{2\pi}x^2 = 0 \quad &\Leftrightarrow \\ \sin(\pi x) - x^2 = f(x) = 0 \end{aligned}$$

In fact, we can show that for any initial point in $(0, 2]$ the sequence $x_k = g(x_{k-1})$ converges to a unique solution x^*.

(**c**) Write a function that computes this sequence. Your function should:

- Take for input the an initial point, a maximal number of iterations and a tolerance for $|x_k - x_{k-1}|$ (the estimated error) and for $|f(x_k)|$ (the residual).
- Print to the command window the list of iterates, stopping when either the maximal number of iterations is reached or the estimated error **and** the residual are below their respective tolerance.
- Output the approximate solution, its estimated error and its residual.

Example with loop:

```
function [x,err,res] = iteration(f,g,x,epse,epsr,itmx)
%iteration: iterate the function g(x).
%   Input: functions f and g, initial point x, tolerances for |x_k-x_k-1| and |f(x_k)|
%   maximal number of iterations itmx. Output: x_k, its error and residual. Stops
%   after itmx iterations and if |x_k-x_k-1|<epse and |f(x_k)|<epsr.
fprintf('%d %e\n',0,x); % initial point
for i=1:itmx
    y=g(x);
    fprintf('%d %e\n',i,y);
    err=abs(y-x);
    res=abs(f(y));
    if err<epse && res<epsr
        fprintf('convergence\n');
        break
    end;
    x=y;
end
```

(**d**) If you used a `for` or `while` loop in (**c**), program a function with the same functionality using recursion (i.e. no explicit loops). If you used recursion in (**c**) then program a function with the same functionality using loops (i.e. no recursion).
Name your functions `iteration.m` for the version with a loop and `recursion.m` for the version without.

```
function [x,err,res] = recursion(f,g,x,epse,epsr,itmx)
%recursion: iterate the function g(x).
%   Input: functions f and g, initial point x, tolerances for |x_k-x_k-1| and |f(x_k)|;
%   maximal number of iterations itmx. Output: x_k, its error and residual. Stops
%   after itmx iterations and if |x_k-x_k-1|<epse and |f(x_k)|<epsr.
y=g(x);
err=abs(y-x);
res=abs(f(y));

fprintf('%e\n',x);
if itmx==0
   fprintf('max number of iterations reached\n');
   return;
end;
if err<epse && res<epsr
    fprintf('convergence\n');
    x=y;
    fprintf('%e\n',x);
else
    [x,err,res]=recursion(f,g,y,epse,epsr,itmx-1); % recursive call
end;

end
```

(**e**) What happens if you take $x_0 = 0$? What happens if you take $x_0 < 0$?

Since $x = 0$ is solution, the error and residual will be zero after one iteration. For negative x_0, the iterates diverge and approach $-\infty$.

Question 2 — 5 marks

(**a**) Write a function that implements the following pseudo-code:
Input: f, f', x, ϵ, N.
Output: x^*.

1. Repeat N times:
   (a) Set $y_1 = x$.
   (b) Take one Newton step, starting from y_1. Call the result y_2.
   (c) Take one Newton step, starting from y_2. Call the result y_3.
   (d) Set
   $$x = y_1 - \frac{(y_2 - y_1)^2}{y_3 - 2y_2 + y_1}$$
   (e) Display $|f(x)|$.
   (f) If $|f(x)| < \epsilon$ print "converged!", break.
2. Ouput $x^* = x$.

This algorithm is called Steffensen's iteration.

```
function [ xs ] = steffensen(f,fp,x,epsf,N)
%steffensen: Steffensen iteration for f(x)=0.
%   Input: f, f', intial point x, tolerance eps, maximal nr of iterations N
```

```
%   Output: in case of convergence: x such that |f(x)|<eps, otherwise:
%   Nth number of the Steffensen iteration.
for i=1:N
    xs=[x];
    dx=-f(x)/fp(x); % first Newton step
    x=x+dx;xs=[xs,x];
    dx=-f(x)/fp(x); % second Newton step
    x=x+dx;xs=[xs,x];
    x=xs(1)-(xs(2)-xs(1))^2/(xs(3)-2*xs(2)+xs(1)); % Steffensen's formula
    fprintf('it %d res=%e\n',i,abs(f(x)));
    if abs(f(x))<epsf
        fprintf('convergence!\n');
        break
    end;
end;

end
```

(**b**) Test your routine on the problem

$$\exp(-x^2 + x) - \frac{1}{2}x = 1.0836 \quad \text{(with initial guess } x = 1\text{)}$$

Show that Newton iteration does not converge quadratically, but your new iterative algorithm does.

Newton iteration gives this sequence of residuals (with the function from lecture 3):

```
iteration 1 residual 5.836000e-01 error 3.890667e-01
iteration 2 residual 1.207459e-01 error 1.545255e-01
iteration 3 residual 3.021620e-02 error 7.782351e-02
iteration 4 residual 7.656717e-03 error 3.972124e-02
iteration 5 residual 1.916870e-03 error 1.980267e-02
iteration 6 residual 4.619788e-04 error 9.183678e-03
iteration 7 residual 9.759948e-05 error 3.355400e-03
iteration 8 residual 1.291781e-05 error 6.038397e-04
```

which is obviously not quadratic. Steffensen's iteration, however, gives

```
it 1 res=3.735449e-03
it 2 res=7.347056e-05
it 3 res=3.268170e-05
it 4 res=1.750423e-07
it 5 res=4.418688e-14
```

and this sequence approaches the quadratic convergence $\epsilon_k \approx \epsilon_{k-1}^2$.
The reason for the slow convergence of Newton's method is, of course, that its condition number is large. The derivative of the function is about 0.02 at the solution…