Golden section search algorithm

This example tries to find the minimizer of $f(x) = 10(x-1)^4 - 4\sin(3x)$
The code for `golden.m` is written as follow:

```
function [xmin, fmin, IFLAG, IFunc, Ak, Bk, X1k, X2k] = golden( a, b, epsilon, itmax )

    tau = double((sqrt(5)-1)/2);    % golden ratio.
    k = 0;                  % number of iterations.

    x1=a+(1-tau)*(b-a);             % computing x1, x2 values
    x2=a+tau*(b-a);

    f_x1=f(x1);                     % computing f values at x1, x2
    f_x2=f(x2);

    Ak = [];  Ak(1) = a;            % list to store a_k.
    Bk = [];  Bk(1) = b;            % list to store b_k.
    X1k = []; X1k(1) = x1;          % list to store x1_k.
    X2k = []; X2k(1) = x2;          % list to store x2_k.

    if (f(a) > max(f_x2, f_x1) && f(b) > max(f_x2, f_x1))
        disp("This [a,b] interval is good.");

        while ((abs(b-a) > epsilon) && (k < itmax))

            k = k + 1; % new iteration.

            if (f_x1 < f_x2)
                % calculate new values according to the rules...
                b=x2;
                x2=x1;
                x1=a+(1-tau)*(b-a);
                f_x1=f(x1);
                f_x2=f(x2);
            else
                % calculate new values according to the rules...
                a=x1;
                x1=x2;
                x2=a+tau*(b-a);
                f_x1=f(x1);
                f_x2=f(x2);
            end

            Ak(k+1) = a;
            Bk(k+1) = b;
            X1k(k+1) = x1;
            X2k(k+1) = x2;
        end

        if (k == itmax)
            disp("too many iterations");
            IFLAG = -999;
        else
            disp("success!");
            IFLAG = 0;
        end

        IFunc = k;
        xmin = (x1+x2)/2;
        fmin = f(xmin);

    else
        disp("This [a,b] interval is not good. Please change the interval.");
        IFLAG = -999;
        xmin = 0; fmin = 0; IFunc = 0;
    end

```

```
end
```

The code for function f is written as follow:

```
function y = f(x)
    y = 10*(x-1)^4 - 4*sin(3*x);
end
```

This is the script used to test the function and printing the result. The easiest systematic way to find suitable b is to increment its value until it works.

```
a = 0;
b = 0.05;
epsilon = 1e-10;
itmax = 100;
IFLAG = -999;

while (IFLAG == -999) % iterates until find the suitable b.
    [xmin, fmin, IFLAG, IFunc, Ak, Bk, X1k, X2k] = golden( a, b, epsilon, itmax );
    b = b + 0.05; % increment b until the interval contains minimizer.
end

% print out the result.
fprintf('% 5s % 20s % 20s % 20s % 20s \n', 'Iter', 'a', 'x_1', 'x_2', 'b');
for i = 0:IFunc
    fprintf('% 5.2d % 20.10f % 20.10f % 20.10f % 20.10f \n', i, Ak(i+1), X1k(i+1), X2k(i+1),
     Bk(i+1));
end
```

And the result is

```
 Iter                    a                  x_1                  x_2                    b
   00          0.0000000000         0.3628677107         0.5871322893         0.9500000000
   01          0.3628677107         0.5871322893         0.7257354214         0.9500000000
   02          0.3628677107         0.5014708428         0.5871322893         0.7257354214
   03          0.5014708428         0.5871322893         0.6400739748         0.7257354214
   04          0.5014708428         0.5544125283         0.5871322893         0.6400739748
   05          0.5544125283         0.5871322893         0.6073542138         0.6400739748
   06          0.5544125283         0.5746344527         0.5871322893         0.6073542138
   07          0.5746344527         0.5871322893         0.5948563771         0.6073542138
   08          0.5871322893         0.5948563771         0.5996301259         0.6073542138
   09          0.5871322893         0.5919060381         0.5948563771         0.5996301259
   10          0.5919060381         0.5948563771         0.5966797869         0.5996301259
   11          0.5948563771         0.5966797869         0.5978067161         0.5996301259
   12          0.5948563771         0.5959833064         0.5966797869         0.5978067161
   13          0.5959833064         0.5966797869         0.5971102356         0.5978067161
   14          0.5966797869         0.5971102356         0.5973762675         0.5978067161
   15          0.5966797869         0.5969458188         0.5971102356         0.5973762675
   16          0.5966797869         0.5968442037         0.5969458188         0.5971102356
   17          0.5968442037         0.5969458188         0.5970086204         0.5971102356
   18          0.5968442037         0.5969070053         0.5969458188         0.5970086204
   19          0.5969070053         0.5969458188         0.5969698069         0.5970086204
   20          0.5969070053         0.5969309934         0.5969458188         0.5969698069
   21          0.5969309934         0.5969458188         0.5969549815         0.5969698069
   22          0.5969309934         0.5969401560         0.5969458188         0.5969549815
   23          0.5969401560         0.5969458188         0.5969493186         0.5969549815
   24          0.5969401560         0.5969436558         0.5969458188         0.5969493186
   25          0.5969401560         0.5969423190         0.5969436558         0.5969458188
   26          0.5969423190         0.5969436558         0.5969444820         0.5969458188
   27          0.5969423190         0.5969431452         0.5969436558         0.5969444820
   28          0.5969423190         0.5969428296         0.5969431452         0.5969436558
   29          0.5969428296         0.5969431452         0.5969433402         0.5969436558
   30          0.5969428296         0.5969430247         0.5969431452         0.5969433402
   31          0.5969430247         0.5969431452         0.5969432197         0.5969433402
   32          0.5969430247         0.5969430992         0.5969431452         0.5969432197
   33          0.5969430247         0.5969430707         0.5969430992         0.5969431452
   34          0.5969430707         0.5969430992         0.5969431167         0.5969431452
   35          0.5969430992         0.5969431167         0.5969431276         0.5969431452
   36          0.5969430992         0.5969431100         0.5969431167         0.5969431276
   37          0.5969430992         0.5969431059         0.5969431100         0.5969431167
```

```
38          0.5969431059          0.5969431100          0.5969431126          0.5969431167
39          0.5969431100          0.5969431126          0.5969431142          0.5969431167
40          0.5969431100          0.5969431116          0.5969431126          0.5969431142
41          0.5969431116          0.5969431126          0.5969431132          0.5969431142
42          0.5969431126          0.5969431132          0.5969431136          0.5969431142
43          0.5969431132          0.5969431136          0.5969431138          0.5969431142
44          0.5969431132          0.5969431134          0.5969431136          0.5969431138
45          0.5969431134          0.5969431136          0.5969431137          0.5969431138
46          0.5969431134          0.5969431135          0.5969431136          0.5969431137
47          0.5969431135          0.5969431136          0.5969431136          0.5969431137
48          0.5969431136          0.5969431136          0.5969431136          0.5969431137
```

The result up to 4 sig. fig. is 0.5969.