

## 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:

```
1 function [xmin, fmin, IFLAG, IFunc, Ak, Bk, X1k, X2k] = golden( a, b, epsilon, itmax )
2
3     tau = double((sqrt(5)-1)/2);    % golden ratio.
4     k = 0;                          % number of iterations.
5
6     x1=a+(1-tau)*(b-a);              % computing x1, x2 values
7     x2=a+tau*(b-a);
8
9     f_x1=f(x1);                     % computing f values at x1, x2
10    f_x2=f(x2);
11
12    Ak = []; Ak(1) = a;              % list to store a_k.
13    Bk = []; Bk(1) = b;              % list to store b_k.
14    X1k = []; X1k(1) = x1;           % list to store x1_k.
15    X2k = []; X2k(1) = x2;           % list to store x2_k.
16
17    if (f(a) > max(f_x2, f_x1) && f(b) > max(f_x2, f_x1))
18        disp("This [a,b] interval is good.");
19
20        while ((abs(b-a) > epsilon) && (k < itmax))
21
22            k = k + 1; % new iteration.
23
24            if (f_x1 < f_x2)
25                % calculate new values according to the rules...
26                b=x2;
27                x2=x1;
28                x1=a+(1-tau)*(b-a);
29                f_x1=f(x1);
30                f_x2=f(x2);
31            else
32                % calculate new values according to the rules...
33                a=x1;
34                x1=x2;
35                x2=a+tau*(b-a);
36                f_x1=f(x1);
37                f_x2=f(x2);
38            end
39
40            Ak(k+1) = a;
41            Bk(k+1) = b;
42            X1k(k+1) = x1;
43            X2k(k+1) = x2;
44        end
45
46        if (k == itmax)
47            disp("too many iterations");
48            IFLAG = -999;
49        else
50            disp("success!");
51            IFLAG = 0;
52        end
53
54        IFunc = k;
55        xmin = (x1+x2)/2;
56        fmin = f(xmin);
57
58    else
59        disp("This [a,b] interval is not good. Please change the interval.");
60        IFLAG = -999;
61        xmin = 0; fmin = 0; IFunc = 0;
62    end
63
```

64 end

The code for the function  $f$  is written as follows:

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

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

```
1 a = 0;
2 b = 0.05;
3 epsilon = 1e-10;
4 itmax = 100;
5 IFLAG = -999;
6
7 while (IFLAG == -999) % iterates until find the suitable b.
8     [xmin, fmin, IFLAG, IFunc, Ak, Bk, X1k, X2k] = golden( a, b, epsilon, itmax );
9     b = b + 0.05; % increment b until the interval contains minimizer.
10 end
11
12 % print out the result.
13 fprintf('% 5s % 20s % 20s % 20s % 20s \n', 'Iter', 'a', 'x_1', 'x_2', 'b');
14 for i = 0:IFunc
15     fprintf('% 5.2d % 20.10f % 20.10f % 20.10f % 20.10f \n', i, Ak(i+1), X1k(i+1), X2k(i+1),
16         Bk(i+1));
17 end
```

And the result is

1	Iter	a	x_1	x_2	b
2	00	0.0000000000	0.3628677107	0.5871322893	0.9500000000
3	01	0.3628677107	0.5871322893	0.7257354214	0.9500000000
4	02	0.3628677107	0.5014708428	0.5871322893	0.7257354214
5	03	0.5014708428	0.5871322893	0.6400739748	0.7257354214
6	04	0.5014708428	0.5544125283	0.5871322893	0.6400739748
7	05	0.5544125283	0.5871322893	0.6073542138	0.6400739748
8	06	0.5544125283	0.5746344527	0.5871322893	0.6073542138
9	07	0.5746344527	0.5871322893	0.5948563771	0.6073542138
10	08	0.5871322893	0.5948563771	0.5996301259	0.6073542138
11	09	0.5871322893	0.5919060381	0.5948563771	0.5996301259
12	10	0.5919060381	0.5948563771	0.5966797869	0.5996301259
13	11	0.5948563771	0.5966797869	0.5978067161	0.5996301259
14	12	0.5948563771	0.5959833064	0.5966797869	0.5978067161
15	13	0.5959833064	0.5959833064	0.5971102356	0.5978067161
16	14	0.5966797869	0.5971102356	0.5973762675	0.5978067161
17	15	0.5966797869	0.5969458188	0.5971102356	0.5973762675
18	16	0.5966797869	0.5968442037	0.5969458188	0.5971102356
19	17	0.5968442037	0.5969458188	0.5970086204	0.5971102356
20	18	0.5968442037	0.5969458188	0.5969458188	0.5970086204
21	19	0.5969070053	0.5969458188	0.5969698069	0.5970086204
22	20	0.5969070053	0.5969309934	0.5969458188	0.5969698069
23	21	0.5969309934	0.5969458188	0.5969549815	0.5969698069
24	22	0.5969309934	0.5969401560	0.5969458188	0.5969549815
25	23	0.5969401560	0.5969458188	0.5969493186	0.5969549815
26	24	0.5969401560	0.5969436558	0.5969458188	0.5969493186
27	25	0.5969401560	0.5969423190	0.5969436558	0.5969458188
28	26	0.5969423190	0.5969436558	0.5969444820	0.5969458188
29	27	0.5969423190	0.5969431452	0.5969436558	0.5969444820
30	28	0.5969423190	0.5969428296	0.5969431452	0.5969436558
31	29	0.5969428296	0.5969431452	0.5969433402	0.5969436558
32	30	0.5969428296	0.5969430247	0.5969431452	0.5969433402
33	31	0.5969430247	0.5969431452	0.5969432197	0.5969433402
34	32	0.5969430247	0.5969430992	0.5969431452	0.5969432197
35	33	0.5969430247	0.5969430707	0.5969430992	0.5969431452
36	34	0.5969430707	0.5969430992	0.5969431167	0.5969431452
37	35	0.5969430992	0.5969431167	0.5969431276	0.5969431452
38	36	0.5969430992	0.5969431100	0.5969431167	0.5969431276
39	37	0.5969430992	0.5969431059	0.5969431100	0.5969431167

40	38	0.5969431059	0.5969431100	0.5969431126	0.5969431167
41	39	0.5969431100	0.5969431126	0.5969431142	0.5969431167
42	40	0.5969431100	0.5969431116	0.5969431126	0.5969431142
43	41	0.5969431116	0.5969431126	0.5969431132	0.5969431142
44	42	0.5969431126	0.5969431132	0.5969431136	0.5969431142
45	43	0.5969431132	0.5969431136	0.5969431138	0.5969431142
46	44	0.5969431132	0.5969431134	0.5969431136	0.5969431138
47	45	0.5969431134	0.5969431136	0.5969431137	0.5969431138
48	46	0.5969431134	0.5969431135	0.5969431136	0.5969431137
49	47	0.5969431135	0.5969431136	0.5969431136	0.5969431137
50	48	0.5969431136	0.5969431136	0.5969431136	0.5969431137

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