2102 509 – Introduction to Optimization Techniques
Homework # 1: Due Date 1st October 2025.
Submit the homework in the class before lecture.

LATE HOMEWORK WILL NOT BE ACCEPTED. DUPLICATION OF HOMEWORK IS STRICTLY FORBIDDEN.

Name: นายภูเบศ สุวรรณโณ 6530316021 **Collaborator:**..

---

1. (20 points) Let f be a real-valued function of n variables.

1.1 Assume that f is continuously differentiable. Then prove that f is convex on the convex set S if and only if

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y \in S.$$

1.2 Assume that f is twice continuously differentiable. Then prove that f is convex on the convex set S if $\nabla^2 f(x)$ is positive semi-definite for all $x \in S$, by using the result obtained from Problem 1.1.

2. (15 points) Prove that if f is convex, then any stationary point is also a global minimizer.

3. (10 points) Let $f : \mathbb{R} \to \mathbb{R}$ be defined by

$$f(x) = x^5 - 5x^3 - 20x + 5$$

Determine the minimizer x^* of f ($x^* \geq 0$) with the accuracy of at least 10 significant digits by using Newton-Raphson method.

4. (20 points) Write your MATLAB functions for solving the one-dimensional optimization problem using quadratic interpolation and golden section methods. Print out your programs. The format of the functions are as follows.

function [x_{min}, f_{min}, IFLAG, IFunc] = quadractic(a, b, epsilon, itmax)
function [x_{min}, f_{min}, IFLAG, IFunc] = golden(a, b, epsilon, itmax)

where x_{min} is the estimate of the minimizer of f, f_{min} is value of $f(x_{min})$, IFLAG is set to be 0 if the search is successful and -999 otherwise, IFunc is the number of function evaluation calls, $[a, b]$ is a given interval that brackets x_{min}, epsilon is the parameter used in the stopping criterion, itmax denotes the maximum number of the iterations allowed. Write comments in the codes so that they can be examined.

(20 points) Then use the quadratic and golden functions to compute the value of x that minimizes the function f in Problem 3. Locate the value of x with at least 4 significant digit accuracy. Print the results for every iteration. Compare the obtained results with the exact solution in Problem 3 and make a discussion regarding the performance of both methods.

Suchin Arunsawatwong, 17th September 2025

นายภูเบศ สุวรรณโณ 6530316021

1. (20 points) Let f be a real-valued function of n variables.

1.1 Assume that f is continuously differentiable. Then prove that f is convex on the convex set S if and only if

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y \in S.$$

1.2 Assume that f is twice continuously differentiable. Then prove that f is convex on the convex set S if $\nabla^2 f(x)$ is positive semi-definite for all $x \in S$, by using the result obtained from Problem 1.1.

1.1 Let $f: \mathbb{R}^n \to \mathbb{R}$, f is continuously differentiable

เราจะกล่าวว่า f is a convex functions เมื่อ

$$f(\theta y + (1-\theta)x) \leq \theta f(y) + (1-\theta) f(x) \; ; \; \forall \theta \in [0,1]$$

หารตลอดด้วย θ ; $\dfrac{f(\theta y + (1-\theta)x) - f(x)}{\theta} \leq f(y) - f(x)$

พิจารณา $\displaystyle\lim_{\theta \to 0} \frac{f(\theta y + (1-\theta)x) - f(x)}{\theta} \leq f(y) - f(x)$

ทำการประมาณด้วย First order approximation

$$f(x + \theta(y-x)) - f(x) \simeq \cancel{f(x)} + \nabla f(x)^T \theta (y-x) - \cancel{f(x)}$$

แทนกลับจะได้

$$\lim_{\theta \to 0} \frac{\nabla f(x)^T \cancel{\theta} (y-x)}{\cancel{\theta}} \leq f(y) - f(x)$$

$$\therefore \quad f(y) \geq f(x) + \nabla f(x)^T (y-x) \; \#$$

ถ้า $f(y) \geq f(x) + \nabla f(x)^T (y-x)$ for all $x, y \in S$ จะแสดงว่า f is convex on set S

โดยที่ S เป็น convex set , เลือก x, y ใดๆ ที่ $x \neq y$ และ $\theta \in [0,1]$, ให้ $z = \theta x + (1-\theta) y$

จากผลลัพธ์ เราจะได้ $f(x) \geq f(z) + \nabla f(z)^T (x-z)$ — (1)

และ $f(y) \geq f(z) + \nabla f(z)^T (y-z)$ — (2)

(1) × θ ; $\theta f(x) \geq \theta f(z) + \theta \nabla f(z)^T (x-z)$ — (1)*

(2) × $(1-\theta)$; $(1-\theta) f(y) \geq (1-\theta) f(z) + (1-\theta) \nabla f(z)^T (y-z)$ — (2)*

(1)* + (2)* ; $\theta f(x) + (1-\theta) f(y) \geq f(z) + \nabla f(z)^T (y-z)$

และจาก definition $f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y)$

$\therefore$ f is a convex f^n on convex set S #

1.2 assume f is twice continuously differentiable

ทำการประมาณ f(y) รอบ x ด้วย second order approximation

$f(z) \simeq f(x) + \nabla f(x)^T (z-x) + \frac{1}{2}(z-x)^T \nabla^2 f(x)(z-x)$ และโจทย์กำหนดให้ $\nabla^2 f(x) \succeq 0$

เราจึงได้ว่า $(z-x)^T \nabla^2 f(x)(z-x) \geq 0 \quad \forall (z-x) \in \mathbb{R}^n$

ทำให้ทราบว่า $f(z) \geq f(x) + \nabla f(x)^T (z-x)$ จึงสามารถสรุปได้ว่า f is convex on the S

ถ้า $\nabla^2 f(x) \succeq 0$ สำหรับทุกๆ $x \in S$

นายภูเบศ สุวรรณโณ 6530316021

2. (15 points) Prove that if f is convex, then any stationary point is also a global minimizer.

จาก first-order condition; f is convex iff dom f is convex sets and

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) \quad , \forall x,y \in \mathrm{dom}\, f$$

RHS คือ first-order Taylor approximation ซึ่งเป็น global underestimator of f

หากพิจารณา ถ้า $\nabla f(x)=0$ จาก $f(y) \geq f(x) + \nabla f(x)^T(y-x)$ สำหรับ $\forall y \in \mathrm{dom}\, f$

เราจะได้ว่า $f(y) \geq f(x) \quad , \forall x,y \in \mathrm{dom}\, f$

เราจึงสรุปได้ว่า x เป็น global minimizer of f #

3. (10 points) Let $f : \mathbb{R} \to \mathbb{R}$ be defined by

นายภูเบศ ศุวรรณโณ 6530316021

$$f(x) = x^5 - 5x^3 - 20x + 5$$

Determine the minimizer x^* of f ($x^* \geq 0$) with the accuracy of at least 10 significant digits by using Newton-Raphson method.

Suppose that x^* is a root of $f(x)=0$ consider the Taylor series of f at the point x_k

Then we have $f(x_k+\Delta x) \simeq f(x_k) + f'(x_k)\Delta x_k$

If x_k+s is an good estimate of x^* so that

$$f(x^*) \simeq f(x_k) + \Delta x f'(x_k) = 0 \quad \text{and if } f'(x_k) \neq 0$$

we obtained $\Delta x = -\dfrac{f(x_k)}{f'(x_k)}$

Let $f(x) = x^5 - 5x^3 - 20x + 5$ เริ่มต้นที่ $x_0 = 0$

$k=0$; $x_0=0$ $f(0)=5$, $f'(0)=-20$, $\Delta x = -\dfrac{f(x_0)}{f'(x_0)} = 0.2500000000$

$x_1 = x_0 + \Delta x = 0+0.25 = 0.2500000000$

เงื่อนไข $|x_1 - x_0| = 0.25$ ซึ่ง มากกว่า 10^{-10} ตามโจทย์กำหนด จึงทำ iteration ต่อไป

$k=1$; $x_1 = 0.2500000000$;

$f(x_1) = -0.0771484375$

$f'(x_1) = -20.9179687500$

$\Delta x = -\dfrac{f(x_1)}{f'(x_1)} = -0.0036881419$

$x_2 = x_1 + \Delta x = 0.2463118581$

เงื่อนไข $|x_2 - x_1| = 0.0036881419$ ซึ่งมากกว่า 10^{-10} ตามโจทย์กำหนด จึงทำ iteration ต่อไป

$k=2$; $x_2 = 0.2463118581$;

$f(x_2) = -0.0000486644$

$f'(x_2) = -20.8916390114$

$\Delta x = -\dfrac{f(x_2)}{f'(x_2)} = -0.0000023294$

$x_3 = x_2 + \Delta x = 0.2463095287$

เงื่อนไข $|x_3 - x_2| = 0.0000023294$ ซึ่งมากกว่า 10^{-10} ตามโจทย์กำหนด จึงทำ iteration ต่อไป

$k=3$; $x_3 = 0.2463095287$;

$f(x_3) = 0.0000000006$

$f'(x_3) = -20.8916224944$

$\Delta x = -\dfrac{f(x_3)}{f'(x_3)} = 0.0000000000$

$x_4 = x_3 + \Delta x = 0.2463095287$

เงื่อนไข $|x_4 - x_3| = 0.0000000000$ ซึ่งน้อยกว่า 10^{-10} ดังนั้น หยุดที่ $k=3$

ดังนั้น จากวิธี Newton Raphson เราจะได้ $x^* = 0.2463095287$

ข้อที่ 4

นายภูเบศ สุวรรณโณ 6530316021

2102509 Introduction to Optimization Techniques
Problem 4 Homework 1

Phubet Suwanno ID:6530316021

October 4, 2025

Instruction

Write your Matlab functions for solving the one-dimensional optimization problem using quadratic interpolation and golden section methods. Print out your programs. The format of the functions are as follows.

```
[xmin, fmin, IFLAG, IFunc] = quadractic(a, b, epsilon, itmax)
[xmin, fmin, IFLAG, IFunc] = golden(f, a, b, epsilon, itmax)
```

where $x_{\min}$ is the estimate of the minimizer of f, $f_{\min}$ is the value of $f(x_{\min})$, `IFLAG` is set to 0 if the search is successful and -999 otherwise, `IFunc` is the number of function evaluations, $[a, b]$ is the given interval that brackets $x_{\min}$, `epsilon` is the parameter used in the stopping criterion, and `itmax` denotes the maximum number of iterations allowed. Then use the quadratic and golden functions to compute the value of x that minimizes the function f in Problem 3. Locate the value of x with at least 4 significant digit accuracy. Print the results for every iteration. Compare the obtained results with the exact solution in Problem 3 and make a discussion regarding the performance of both methods.

Inputs

- `f` : function handle
- `a, b` : interval endpoints
- `epsilon` : tolerance
- `itmax` : maximum iterations

Outputs

- `xmin` : estimated minimizer
- `fmin` : function value at minimizer
- `IFLAG` : termination flag
- `IFunc` : number of iterations

Write a `f.m` function to evaluate $f(x) = x^5 - 5x^3 - 20x + 5$

```
function y = f(lambda)
    y = lambda.^5 - 5*lambda.^3 - 20*lambda + 5;
end
```

1 Function: `quadractic`

Usage

```
[lmin, fmin, IFLAG, IFunc] = quadractic(a, b, epsilon, itmax)
```

Quadractic Interpolation code as `quadractic.m`

```matlab
function [xmin, fmin, IFLAG, IFunc] = quadractic(a, b, epsilon, itmax)
    % QUADRRACTIC  One-dimensional minimization by Quadratic Interpolation
    %
    % DEVELOPED BY PHUBET SUWANNO 6530316021 (BOOM)
    % PRESENTED TO ASSOC. PROF. DR. SUCHIN ARUNSAWATWONG
    % 2102509 INTRO OPTIMIZATION TECHNIQUE
    %
    %   [xmin, fmin, IFLAG, IFunc] = quadractic(a, b, epsilon, itmax)
    %
    %   This function finds the minimizer of f(lambda) in the interval [a,b]
    %   using the Quadratic Interpolation Method (Powell style).
    %
    %   Input arguments:
    %     a       - left endpoint of the initial interval
    %     b       - right endpoint of the initial interval
    %     epsilon - stopping tolerance (see eqn (5.2) in lecture notes)
    %     itmax   - maximum number of iterations allowed
    %
    %   Output arguments:
    %     lmin    - estimated minimizer of f
    %     fmin    - function value f(lmin)
    %     IFLAG   - flag (0 if successful, -999 otherwise)
    %     IFunc   - number of function evaluations
    %
    %   Notes:
    %   - The target function f(lambda) must be defined separately in f.m
    %     Example:
    %         function y = f(lambda)
    %             y = lambda.^5 - 5*lambda.^3 - 20*lambda + 5;
    %         end
    %   - Method uses 3 points (A,B,C), fits a parabola h(lambda),
    %     computes vertex lambda*, and updates according to 4 cases.
    %   - Stop criterion: |h(lambda*) - f(lambda*)| / |f(lambda*)| <= epsilon
    %
    %   Example of usage:
    %     [lmin,fmin,IFLAG,IFunc] = quadractic(-5,5,1e-6,100)

    % printing format (>= 4 digits; set to 10 for clarity)
    DIG  = 10; % number of digits to print
    fmt  = ['%.', num2str(max(4,DIG)), 'f'];   % e.g., '%.10f'
    A = a; C = b; B = 0.5*(a+b);
    fA = f(A); fB = f(B); fC = f(C);
    IFunc = 3; IFLAG = -999;
    k = 0;
    % header print
    fprintf('Iter  |  A  B  C  |  lambda*  f(lambda*) IFunc\n');
    fprintf('------+--------------------------------------\n');

    while k < itmax
        % use eq (5.1) in EE509 handout to find lambda*
        num = fA*(B^2 - C^2) + fB*(C^2 - A^2) + fC*(A^2 - B^2);
        den = 2*( fA*(B - C) + fB*(C - A) + fC*(A - B) );

        % To prevent the case where the denominator = 0, which would cause NaN and an error
        if abs(den) < 1e-14
            break;
        end

        lambda_star = num/den;
        fstar = f(lambda_star);
```

```
        IFunc = IFunc+1;

        % define a Parabola h(x) = a + bx + cx
        denom = (A-B)*(B-C)*(C-A);
        a_coef = (fA*B*C*(C-B) + fB*C*A*(A-C) + fC*A*B*(B-A)) / denom;
        b_coef = (fA*(B^2-C^2) + fB*(C^2-A^2) + fC*(A^2-B^2)) / denom;
        c_coef = -(fA*(B-C) + fB*(C-A) + fC*(A-B)) / denom;

        hstar = a_coef + b_coef*lambda_star + c_coef*lambda_star^2;

        % print this iteration
        fprintf(['%4d | ', fmt, '  ', fmt, '  ', fmt, '  |  ', fmt, '   ', fmt, '    %5d\n'], ...
                k, A, B, C, lambda_star, fstar, IFunc);

        % use eq (5.2) in EE509 Handout to find a Stop criterion
        if fstar ~= 0
            if abs(hstar - fstar)/abs(fstar) <= epsilon
                xmin = lambda_star;
                fmin = fstar;
                IFLAG = 0;
                return;
            end
        end

        % Update A,B,C follow by Cases
        if (lambda_star < B) && (fstar < fB)
            % Case 1: lambda* lies to the left of B and better than B
            C = B; fC = fB;
            B = lambda_star; fB = fstar;

        elseif (lambda_star < B) && (fstar >= fB)
            % Case 2: lambda* lies to the left of B and worse than B
            A = lambda_star; fA = fstar;

        elseif (lambda_star > B) && (fstar < fB)
            % Case 3: lambda* lies to the right of B and better than B
            A = B; fA = fB;
            B = lambda_star; fB = fstar;

        else
            % Case 4: lambda* lies to the right of B and worse than B
            C = lambda_star; fC = fstar;
        end
        k = k+1;
    end

    % if not converge
    [fmin, idx] = min([fA, fB, fC]);
    L = [A, B, C];      % save in array
    xmin = L(idx);

    % final line print for visibility
    fprintf('----> Stop by itmax or degenerate parabola. Return best of {A,B,C}.\n');
    fprintf(['Best lmin = ', fmt, ', fmin = ', fmt, ', IFLAG = %d, IFunc = %d\n'], xmin, fmin, IFLAG, IFunc);
end
```

Listing 1: Quadratic interpolation example

Test Output

```
>> [xmin, fmin, IFLAG, IFunc] = quadractic(0, 3, 1e-6, 1000)
Iter  |        A             B             C          |    lambda*        f(lambda*)        IFunc
------+--------------------------------------------------------------------------------
   0  | 0.0000000000  1.5000000000  3.0000000000  |  1.2155555556   -25.6376495989        4
   1  | 1.2155555556  1.5000000000  3.0000000000  |  1.6638742518   -38.5567667510        5
   2  | 1.5000000000  1.6638742518  3.0000000000  |  1.7887523019   -41.0791736178        6
   3  | 1.6638742518  1.7887523019  3.0000000000  |  1.8641914936   -42.1621164730        7
   4  | 1.7887523019  1.8641914936  3.0000000000  |  1.9150590789   -42.6601854484        8
   5  | 1.8641914936  1.9150590789  3.0000000000  |  1.9463881490   -42.8615995510        9
   6  | 1.9150590789  1.9463881490  3.0000000000  |  1.9665252235   -42.9452723205       10
   7  | 1.9463881490  1.9665252235  3.0000000000  |  1.9790250096   -42.9783235358       11
   8  | 1.9665252235  1.9790250096  3.0000000000  |  1.9869118746   -42.9915132250       12
   9  | 1.9790250096  1.9869118746  3.0000000000  |  1.9918222729   -42.9966753354       13
  10  | 1.9869118746  1.9918222729  3.0000000000  |  1.9948985380   -42.9987033943       14
  11  | 1.9918222729  1.9948985380  3.0000000000  |  1.9968159864   -42.9994942316       15
  12  | 1.9948985380  1.9968159864  3.0000000000  |  1.9980139499   -42.9998030543       16
  13  | 1.9968159864  1.9980139499  3.0000000000  |  1.9987609542   -42.9999233048       17
  14  | 1.9980139499  1.9987609542  3.0000000000  |  1.9992271711   -42.9999701529       18

xmin =
    1.9992

fmin =
  -43.0000

IFLAG =
     0

IFunc =
    18
```

2 Function: golden

Usage

```
[xmin, fmin, IFLAG, IFunc] = golden(f, a, b, epsilon, itmax)
```

Golden section method code as golden.m

```matlab
function [xmin, fmin, IFLAG, IFunc] = golden(a, b, epsilon, itmax)
% GOLDEN  One-dimensional minimization by golden-section search (derivative-
    free).
%
% DEVELOPED BY PHUBET SUWANNO 6530316021 (BOOM)
% PRESENTED TO ASSOC. PROF. DR. SUCHIN ARUNSAWATWONG
%
%   [xmin, fmin, IFLAG, IFunc] = GOLDEN(f, a, b, epsilon, itmax)
%
%   Finds an approximate minimizer of the real-valued scalar function f on
%   the closed interval [a, b] using the golden-section search. This method
%   is derivative-free and only requires function evaluations at scalar points.
%
%   INPUTS
%     f         - function handle to a real-valued scalar function f(x).
%                 The algorithm will evaluate f only at scalar x.
%     a, b      - scalars defining the initial interval with a < b.
%     epsilon   - positive scalar tolerance for the interval width; the loop
%                 stops when |b - a| <= epsilon.
%     itmax     - positive integer, maximum number of iterations allowed.
%
```

```
%   OUTPUTS
%     xmin     - estimated minimizer (midpoint of the final interval).
%     fmin     - f(xmin), function value at the estimated minimizer.
%     IFLAG    - termination flag:
%                  0 : converged (|b-a| <= epsilon),
%                  1 : reached maximum iterations without meeting tolerance,
%                -999: invalid input (e.g., a >= b, epsilon <= 0, or itmax <= 0).
%     IFunc    - number of iterations actually performed.
%
%   ASSUMPTIONS & NOTES
%   - f should be unimodal on [a, b] (i.e., has a single local minimum).
%     If this assumption is violated, the method may converge to a non-minimal point
%     or stall near a boundary.
%   - This implementation prints a per-iteration debug line showing (a, b, x1, x2,
%     f(x1), f(x2)). To suppress console output, comment out the fprintf lines
%     in the loop (marked as "print this iteration").
%   - No derivatives are used. Only function values are required.
%
%   COMPLEXITY
%   - The interval length shrinks by a constant factor (~0.618) per iteration.
%     Roughly, the number of iterations needed is
%         N     ceil( log(epsilon / (b0 - a0)) / log(0.618...) )
%     where [a0, b0] is the initial interval.
%
%   EXAMPLES
%     % Using a function file f.m:
%     %   function y = f(x), y = x.^5 - 5*x.^3 - 20*x + 5; end
%     [xmin, fmin, flag, k] = golden(@f, 0, 3, 1e-6, 2000);
%
%     % Using an anonymous function:
%     g = @(x) (x-2).^2 + 1;
%     [xmin, fmin, flag, k] = golden(g, 0, 5, 1e-8, 1000);
%
%   EDGE CASES
%   - If any input is invalid (a >= b, epsilon <= 0, itmax <= 0),
%     the function returns NaN outputs and IFLAG = -999.
%   - If the tolerance is very small relative to machine precision, the loop may
%     terminate by itmax; check IFLAG.

% Golden Section Search for 1D minimization
    % IFLAG = 0        converged
    % IFLAG = 1        reached max iterations
    % IFLAG = -999     error (i.e. input are not valid)

    % check my input
    if a >= b || epsilon <= 0 || itmax <= 0
        xmin = NaN; fmin = NaN;
        IFLAG = -999;
        IFunc = 0;
        return;
    end

    golden_ratio = (sqrt(5)-1)/2;
    x1 = b - golden_ratio * (b - a);
    x2 = a + golden_ratio * (b - a);

    k = 0;
    % header print
    fprintf('Iter | a     b    |  x1     x2   |  f(x1)  f(x2)\n');
```

```
    fprintf('---------------------------------------------\n');

    while (abs(b-a) > epsilon && k < itmax)
        k = k + 1;
        if f(x2) > f(x1)
            b = x2;
            x2 = x1;
            x1 = b - golden_ratio * (b - a);
        else
            a = x1;
            x1 = x2;
            x2 = a + golden_ratio * (b - a);
        end

        % print this iteration
        fprintf('%4d | %13.6e  %13.6e | %13.6e  %13.6e | %13.6e  %13.6e\n', ...
        k, a, b, x1, x2, f(x1), f(x2));
    end

    xmin = (a+b)/2;
    fmin = f(xmin);
    IFunc = k;

    if abs(b-a) <= epsilon
        IFLAG = 0;  % converged
    else
        IFLAG = 1;  % reached max iterations
    end

end
```

Listing 2: Golden section search example

Test Output

```
>> [xmin, fmin, IFLAG, IFunc] = golden(0, 3, 1e-6, 1000)
Iter |       a             b       |      x1            x2       |     f(x1)          f(x2)
-------------------------------------------------------------------------------------------
   1 |  1.145898e+00  3.000000e+00 |  1.854102e+00  2.291796e+00 | -4.203992e+01  -3.779857e+01
   2 |  1.145898e+00  2.291796e+00 |  1.583592e+00  1.854102e+00 | -3.656920e+01  -4.203992e+01
   3 |  1.583592e+00  2.291796e+00 |  1.854102e+00  2.021286e+00 | -4.203992e+01  -4.297701e+01
   4 |  1.854102e+00  2.291796e+00 |  2.021286e+00  2.124612e+00 | -4.297701e+01  -4.215343e+01
   5 |  1.854102e+00  2.124612e+00 |  1.957428e+00  2.021286e+00 | -4.291205e+01  -4.297701e+01
   6 |  1.957428e+00  2.124612e+00 |  2.021286e+00  2.060753e+00 | -4.297701e+01  -4.280747e+01
   7 |  1.957428e+00  2.060753e+00 |  1.996894e+00  2.021286e+00 | -4.299952e+01  -4.297701e+01
   8 |  1.957428e+00  2.021286e+00 |  1.981819e+00  1.996894e+00 | -4.298368e+01  -4.299952e+01
   9 |  1.981819e+00  2.021286e+00 |  1.996894e+00  2.006211e+00 | -4.299952e+01  -4.299806e+01
  10 |  1.981819e+00  2.006211e+00 |  1.991136e+00  1.996894e+00 | -4.299610e+01  -4.299952e+01
  11 |  1.991136e+00  2.006211e+00 |  1.996894e+00  2.000453e+00 | -4.299952e+01  -4.299999e+01
  12 |  1.996894e+00  2.006211e+00 |  2.000453e+00  2.002653e+00 | -4.299999e+01  -4.299965e+01
  13 |  1.996894e+00  2.002653e+00 |  1.999094e+00  2.000453e+00 | -4.299996e+01  -4.299999e+01
  14 |  1.999094e+00  2.002653e+00 |  2.000453e+00  2.001293e+00 | -4.299999e+01  -4.299992e+01
  15 |  1.999094e+00  2.001293e+00 |  1.999934e+00  2.000453e+00 | -4.300000e+01  -4.299999e+01
  16 |  1.999094e+00  2.000453e+00 |  1.999613e+00  1.999934e+00 | -4.299999e+01  -4.300000e+01
  17 |  1.999613e+00  2.000453e+00 |  1.999934e+00  2.000132e+00 | -4.300000e+01  -4.300000e+01
  18 |  1.999613e+00  2.000132e+00 |  1.999811e+00  1.999934e+00 | -4.300000e+01  -4.300000e+01
  19 |  1.999811e+00  2.000132e+00 |  1.999934e+00  2.000010e+00 | -4.300000e+01  -4.300000e+01
  20 |  1.999934e+00  2.000132e+00 |  2.000010e+00  2.000056e+00 | -4.300000e+01  -4.300000e+01
  21 |  1.999934e+00  2.000056e+00 |  1.999981e+00  2.000010e+00 | -4.300000e+01  -4.300000e+01
  22 |  1.999981e+00  2.000056e+00 |  2.000010e+00  2.000028e+00 | -4.300000e+01  -4.300000e+01
  23 |  1.999981e+00  2.000028e+00 |  1.999999e+00  2.000010e+00 | -4.300000e+01  -4.300000e+01
  24 |  1.999981e+00  2.000010e+00 |  1.999992e+00  1.999999e+00 | -4.300000e+01  -4.300000e+01
  25 |  1.999992e+00  2.000010e+00 |  1.999999e+00  2.000003e+00 | -4.300000e+01  -4.300000e+01
```

```
  26 |   1.999992e+00   2.000003e+00 |  1.999996e+00   1.999999e+00 | -4.300000e+01  -4.300000e+01
  27 |   1.999996e+00   2.000003e+00 |  1.999999e+00   2.000000e+00 | -4.300000e+01  -4.300000e+01
  28 |   1.999999e+00   2.000003e+00 |  2.000000e+00   2.000001e+00 | -4.300000e+01  -4.300000e+01
  29 |   1.999999e+00   2.000001e+00 |  2.000000e+00   2.000000e+00 | -4.300000e+01  -4.300000e+01
  30 |   2.000000e+00   2.000001e+00 |  2.000000e+00   2.000001e+00 | -4.300000e+01  -4.300000e+01
  31 |   2.000000e+00   2.000001e+00 |  2.000000e+00   2.000000e+00 | -4.300000e+01  -4.300000e+01

xmin =
    2.0000

fmin =
  -43.0000

IFLAG =
     0

IFunc =
    31
```

Discussion: Choice of Algorithm

Both quadratic interpolation and golden-section search were implemented and tested on the given function. The results show that the quadratic interpolation method converged to $x_{\min} \approx 1.9992$ with $f_{\min} \approx -43.0000$ using only 18 function evaluations, while the golden-section search converged to $x_{\min} \approx 2.0000$ with $f_{\min} \approx -43.0000$ but required 31 function evaluations.

In general, the **golden-section search** is more robust, since it only relies on interval reduction and function values; it always guarantees progress as long as the function is unimodal in the given interval. However, it converges linearly and therefore requires more iterations.

On the other hand, the **quadratic interpolation** method exploits curvature information by fitting a parabola through three points. When the function is smooth and well-behaved, this leads to faster convergence and fewer evaluations. The drawback is that it may fail if the function is nearly linear in the chosen interval, or if the denominator in the interpolation formula becomes very small, leading to numerical instability.

For the present problem, since the function is smooth and unimodal around the minimizer, the quadratic interpolation method is more efficient, achieving nearly the same accuracy as the golden-section search but with fewer function evaluations.

access the code