Problem 1 Fix integer $n \geq 1$, n points x_i with $|x_i| \leq 1$, n points y_j with $|y_j| \leq 1$, n coefficients f_j, and n coefficients g_j.

(a) Fix integer $k \geq 0$. Design an algorithm for evaluating

$$f(x) = \sum_{j=1}^{n} f_j (x y_j)^k$$

at n points x_i, in $O(n)$ operations.

(b) Find a polynomial $P(x)$ with complex coefficients such that

$$|P(x) - \mathrm{e}^{\mathrm{i}x}| \leq \epsilon$$

on the interval $|x| \leq 1$.

(c) Design an algorithm for approximating

$$g(x) = \sum_{j=1}^{n} g_j \mathrm{e}^{\mathrm{i}x y_j}$$

at n points x_i in $O(n)$ operations, with absolute error bounded by

$$\epsilon \sum_{j=1}^{n} |g_j|.$$

(d) Define the $n \times n$ matrix F by

$$F_{jk} = \mathrm{e}^{\mathrm{i}x_j y_k}.$$

Find a rank r independent of n and an $n \times n$ matrix B with elements

$$B_{jk} = \sum_{i=1}^{r} c_{ji} d_{ik}$$

such that B has rank at most r and absolute error

$$|F_{jk} - B_{jk}| \leq \epsilon$$

for all n.

Solution 1 (5 pts x 4 parts = 20 pts)

(a) First store the powers x_i^k and y_j^k for $i, j = 1, \ldots, n$; this requires $2nk = \mathcal{O}(n)$ multiplications. Next, store the sum

$$\sum_{j=1}^{n} f_j \, y_j^k.$$

This requires n multiplications and n additions, for a total of $2n = \mathcal{O}(n)$ additional operations. Finally, calculate

$$\left(x_i^k\right) \cdot \left(\sum_{j=1}^{n} f_j \, y_j^k\right) = \sum_{j=1}^{n} f_j \, (x_i y_j)^k.$$

for $i = 1, \ldots, n$. This is another $n = \mathcal{O}(n)$ multiplications. Altogether we performed $\mathcal{O}(n)$ operations.

(b) Let $P(x)$ be the degree-m Taylor polynomial of e^{ix},

$$P(x) = \sum_{j=0}^{m} \frac{i^j}{j!} x^j$$

so that

$$|P(x) - e^{ix}| \leq \frac{1}{(m+1)!}$$

for $|x| \leq 1$. Since $1/18! = 1.6 \times 10^{-16} \leq \epsilon$, any choice $m \geq 17$ will suffice.

(c) Let $P(x) = a_0 + \ldots + a_m \, x^m$ denote the polynomial in part (b). Since $|x_i| \leq 1$ and $|y_j| \leq 1$,

$$\sum_{j=1}^{n} g_j e^{i x_i y_j} = \sum_{k=0}^{m} \sum_{j=1}^{n} (g_j a_k)(x_i y_j)^k$$

up to an error of size $\epsilon \sum_{j=0}^{n} |g_j|$. Applying the algorithm in part (a) for each k shows that $\sum_{k=0}^{n} \sum_{j=1}^{n} (g_j a_k)(x_i y_j)^k$ can be performed in $\mathcal{O}(n)$ operations.

(d) Define

$$c_{jr} = \frac{(it_j)^{r-1}}{(r-1)!} \text{ and } d_{rk} = t_k^{r-1},$$

and form the $n \times (m+1)$ matrix $C = (c_{jr})$ and the $(m+1) \times n$ matrix $D = (d_{rk})$. Let $B = CD$ and thus

$$\operatorname{rank}(B) \leq \min\{\operatorname{rank}(C), \operatorname{rank}(D)\} \leq m + 1 = 18.$$

For all j and k, using part (b) gives

$$\begin{aligned}
|F_{jk} - B_{jk}| &= \left| e^{it_j t_k} - \sum_{r=0}^{m} \frac{(it_j)^{r-1}}{(r-1)!} t_k^{r-1} \right| \\
&= \left| e^{it_j t_k} - \sum_{r=0}^{m} \frac{(it_j t_k)^{r}}{r!} \right| \\
&< \epsilon.
\end{aligned}$$

Problem 2 Show that floating point arithmetic sums

$$s_n = \sum_{k=1}^{n} \frac{1}{k^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$$

with absolute error $\leq (2n+1)\epsilon$ from left to right, while summing from right to left gives absolute error $\leq (3 + \ln n)\epsilon$. Estimate the maximum accuracy achievable and the number of terms required in each case.

Solution 2 (10 pts x 2 parts = 20 pts)

Summing from left to right Define $a_k = \frac{1}{k^2}$ and $s_n = \sum_{k=1}^{n} a_k$, and let $s_n^\star$ be the result for s_n in floating point arithmetic when summing from left to right. Define e_n by $s_n^\star - s_n = e_n\epsilon$, where ϵ is machine precision. We note that $e_1 = 0$.

Adding an additional term to the right gives

$$\begin{aligned}
s_{n+1}^\star &= \mathrm{fl}(s_n^\star + \mathrm{fl}(a_{n+1})) \\
&= (s_n^\star + a_{n+1}(1+\epsilon_1))(1+\epsilon_2), \qquad \text{where } |\epsilon_1| \leq \epsilon, |\epsilon_2| \leq \epsilon \\
&= s_n^\star + a_{n+1} + a_{n+1}\epsilon_1 + s_n^\star\epsilon_2 + a_{n+1}\epsilon_2 + a_{n+1}\epsilon_1\epsilon_2 \\
&= s_{n+1} + e_n\epsilon + a_{n+1}\epsilon_1 + s_n\epsilon_2 + e_n\epsilon\epsilon_2 + a_{n+1}\epsilon_2 + a_{n+1}\epsilon_1\epsilon_2.
\end{aligned}$$

Thus

$$s_{n+1}^\star = s_{n+1} + e_n\epsilon + a_{n+1}\epsilon_1 + s_n\epsilon_2 + a_{n+1}\epsilon_2 + O(\epsilon^2),$$

which indicates

$$\begin{aligned}
|s_{n+1}^\star - s_{n+1}| &\leq |e_n\epsilon + a_{n+1}\epsilon_1 + s_n\epsilon_2 + a_{n+1}\epsilon_2| \\
&\leq (|e_n| + a_{n+1} + s_{n+1})\epsilon,
\end{aligned}$$

that is,

$$|e_{n+1}| \leq |e_n| + a_{n+1} + s_{n+1}.$$

Applying this inequality repeatedly and the estimate that

$$s_n = \sum_{k=1}^{n} a_k \leq \sum_{k=1}^{\infty} a_k = \frac{\pi^2}{6} < 2$$

to get

$$\begin{aligned}
|e_n| &\le |e_1| + \sum_{k=2}^{n} a_k + \sum_{k=2}^{n} s_k \\
&\le s_n + \sum_{k=2}^{n} s_k \\
&\le 2 + 2(n-1) = 2n + 1.
\end{aligned}$$

Therefore the absolute error is bounded by $(2n+1)\epsilon$.

Summing from right to left Let

$$b_k = \frac{1}{(n+1-k)^2}$$

for $1 \le k \le n$. Define

$$S_k = \sum_{j=1}^{k} b_j = \frac{1}{n^2} + \frac{1}{(n-1)^2} + ... + \frac{1}{(n-k+1)^2},$$

let $S_k^\star$ be the result for S_k in floating point arithmetic summing from left to right in the above sum, and let E_k be defined by $S_k^\star - S_k = E_k\epsilon$, where ϵ is machine precision.

Therefore $|e_1| \le b_1$, and

$$S_{k+1}^\star = \mathrm{fl}(S_k^\star + \mathrm{fl}(b_{k+1})).$$

We use the bounds

$$S_n \le 2 \text{ and } S_k \le (n-k+1)b_k.$$

Working as in part a, we get

$$\begin{aligned}
|e_n| &\le |e_1| + \sum_{k=2}^{n} b_k + \sum_{k=2}^{n} S_k \\
&\le S_n + \sum_{k=2}^{n} S_k \\
&\le S_n + \sum_{k=2}^{n} \sum_{j=1}^{k} b_j.
\end{aligned}$$

We change the order of summation to get

$$\begin{aligned}
|e_n| &\leq S_n + \sum_{k=2}^{n} b_1 + \sum_{j=2}^{n} \sum_{k=j}^{n} b_j \\
&= S_n + (n-1)b_1 + \sum_{j=2}^{n} (n-j+1)b_j \\
&\leq S_n + b_n + \sum_{j=1}^{n-1} (n-j+1)b_j \\
&\leq 3 + \sum_{j=1}^{n-1} \frac{1}{n-j+1} \\
&= 3 + \sum_{m=2}^{n} \frac{1}{m} && m = n-j+1 \\
&\leq 3 + \sum_{m=2}^{n} \int_{m-1}^{m} \frac{1}{x}\, dx \\
&= 3 + \int_{1}^{n} \frac{1}{x}\, dx \\
&= 3 + \ln n.
\end{aligned}$$

Therefore the absolute error is bounded by $(3 + \ln n)\epsilon$.

Problem 3 Suppose a and b are floating point numbers with $0 < a < b < \infty$. Show that

$$a \leq \mathrm{fl}\left(\sqrt{ab}\right) \leq b,$$

in IEEE standard floating point arithmetic if no overflow occurs.

Solution 3 (10 pts)

Since $a^2 < ab < b^2$, $\sqrt{\ }$ delivers the exact result correctly rounded, and rounding is monotone, we need only show that $\mathrm{fl}(\sqrt{a^2}) = a$. But $\mathrm{fl}(a^2) = a^2(1+\delta)$ for some $|\delta| \leq \epsilon$, so $\mathrm{fl}(\sqrt{a^2}) = \mathrm{fl}(a(1+\delta/2+O(\epsilon^2))) = a$ since rounding delivers the nearest floating-point number.

Problem 4 Design an algorithm to evaluate

$$f(x) = \frac{\mathrm{e}^x - 1 - x}{x^2}$$

in IEEE double precision arithmetic, to 12-digit accuracy for all machine numbers $|x| \leq 1$.

Solution 4 (10 pts)

Our algorithm is to approximate $f(x)$ by its nth order Taylor polynomial, i.e.

$$\begin{aligned} f(x) &\sim \frac{\sum_{k=0}^{n+2} \frac{x^k}{k!} - 1 - x}{x^2} \\ &= \sum_{k=2}^{n+2} \frac{x^{k-2}}{k!} \\ &= \sum_{k=0}^{n} \frac{x^k}{(k+2)!}, \end{aligned}$$

evaluated with IEEE standard floating point arithmetic.

First we bound the error in the approximation assuming exact arithmetic. There exists ξ_1 and ξ_2 depending on x and satisfying $|\xi_1|, |\xi_2| \leq |x| \leq 1$ such that

$$\begin{aligned} \left| \frac{\sum_{k=0}^{n+2} \frac{x^k}{k!} - 1 - x}{x^2} - \frac{e^x - 1 - x}{x^2} \right| \Big/ \left(\frac{e^x - 1 - x}{x^2} \right) &= \frac{\left| \sum_{k=0}^{n+2} \frac{x^k}{k!} - e^x \right|}{e^x - 1 - x} \\ &= \frac{\frac{|x^{n+3}|}{(n+3)!} e^{\xi_1}}{\frac{x^2}{2} \cdot e^{\xi_2}} \\ &= 2 \cdot \frac{|x^{n+1}|}{(n+3)!} e^{\xi_1 - \xi_2} \\ &\leq \frac{2}{(n+3)!} e^2. \end{aligned}$$

In order to achieve 12-digit (decimal) accuracy in exact arithmetic, we need

$$\frac{2}{(n+3)!} e^2 \leq 10^{-12},$$

that is, $n \geq 13$.

We also need to bound the relative error in floating-point evaluation of the approximating polynomial. Let s_n be the nth order Taylor polynomial and let s_n^* be the floating point approximation when evaluating the sum from left ($k = 0$) to right ($k = n$). We note that $|s_n| \leq f(1) = e - 2 < 1$. Letting e_n be defined by $e_n \epsilon = s_n^* - s_n$, we find that

$$|e_n| \leq |e_1| + \sum_{k=2}^{n} |s_k| + \sum_{k=2}^{n} \left| \frac{x^k}{(k+2)!} \right| \leq \frac{1}{2} + n - 1 + 1 = n + \frac{1}{2}.$$

We find that for $n \leq 450$ the error $|s_n^* - s_n|$ in the floating-point evaluation is bounded by 10^{-13}, so the contribution to the absolute error from the floating-point evaluations is negligible for small n compared to the truncation error from the Taylor approximation.

Problem 5 Figure out exactly what sequence of intervals is produced by bisection with the *arithmetic* mean for solving $x = 0$ with initial interval $[a_0, b_0] = [-1, 2]$. How many steps will it take to get maximum accuracy in IEEE standard floating point arithmetic?

Solution 5 (10 pts)

Consider the first few terms in the sequence of intervals:

$$\begin{aligned}[a_0, b_0] &= [-1, 2],\\ [a_1, b_1] &= [-1, 2^{-1}],\\ [a_2, b_2] &= [-2^{-2}, 2^{-1}],\\ [a_3, b_3] &= [-2^{-2}, 2^{-3}],\\ &\vdots\end{aligned}$$

This gives the pattern

$$[a_{2n}, b_{2n}] = [-2^{-2n}, 2^{-2n+1}],$$

and

$$[a_{2n+1}, b_{2n+1}] = [-2^{-2n}, 2^{-2n-1}].$$

We can prove the above pattern by induction on n.

Proof.

1. *Base case.* $[a_0, b_0] = [-1, 2]$ and $[a_1, b_1] = [-1, 2^{-1}]$ satisfy the pattern.
2. *Inductive step.* Assuming the pattern works for $n = k$, that is,

   $$[a_{2k}, b_{2k}] = [-2^{-2k}, 2^{-2k+1}],$$

   and

   $$[a_{2k+1}, b_{2k+1}] = [-2^{-2k}, 2^{-2k-1}],$$

   we have

   $$[a_{2k+2}, b_{2k+2}] = [2^{-1}(-2^{-2k} + 2^{-2k-1}), 2^{-2k-1}] = [-2^{-2k-2}, 2^{-2k-1}],$$

   and

   $$[a_{2k+3}, b_{2k+3}] = [-2^{-2k-2}, 2^{-1}(-2^{-2k-2} + 2^{-2k-1})] = [-2^{-2k-2}, 2^{-2k-3}],$$

   which satisfy the pattern for $n = k + 1$.

$\square$

By the pattern above, we have

$$[a_{1074}, b_{1074}] = [-2^{-1074}, 2^{-1073}].$$

The midpoint of this interval is given by

$$p_{1074} = \frac{-2^{-1074} + 2^{-1073}}{2} = \frac{2^{-1074}}{2} \in (0, 2^{-1074}).$$

In floating point arithmetic, p_{1074} will give 0, as the smallest subnormal number is $(-1)^0 2^{1-1023}(0 + 2^{-52}) = 2^{-1074}$, and any positive number smaller than that will result in underflow to 0.

Hence 1075 steps are needed to get maximum accuracy.

Problem 6 Implement a MATLAB function `bisection.m` of the form

```
function [r, h] = bisection(a, b, f, p, t)
% a: Beginning of interval [a, b]
% b: End of interval [a, b]
% f: function handle y = f(x, p)
% p: parameters to pass through to f
% t: User-provided tolerance for interval width
```

At each step $j = 1$ to n, carefully choose m as in bisection with the *geometric* mean (watch out for zeroes!). Replace $[a, b]$ by the smallest interval with endpoints chosen from a, m, b which keeps the root bracketed. Repeat until a f value exactly vanishes, $b - a \le t \min(|a|, |b|)$, or b and a are adjacent floating point numbers, whichever comes first. Return the final approximation to the root r and a $3 \times n$ history matrix `h[1:3,1:n]` with column `h[1:3,j]` $=$ $(a, b, f(m))$ recorded at step j. Try to make your implementation as foolproof as possible.

(a) (See BBF 2.1.7) Sketch the graphs of $y = x$ and $y = 2 \sin x$.

(b) Use `bisection.m` to find an approximation to within ϵ to the first positive value of x with $x = 2 \sin x$. Report the number of steps, the final result, and the absolute and relative errors.

(c) Use `bisection.m` as many times as needed to find approximations within ϵ to all solutions $x > 0$ of the equation

$$f(x) = \frac{1}{x} + \ln x - 2 = 0.$$

Report the number of steps, the final results, and the absolute and relative errors.

(d) Use `bisection.m` to solve the equation

$$f(x) = (x - \epsilon^3)^3 = 0$$

on the interval $[-1, 2]$. Report the number of steps, the final result, and the absolute and relative errors.

(e) Use `bisection.m` to solve the equation

$$f(x) = \arctan(x - \epsilon^2) = 0$$

on the interval $[-1, 2]$. Report the number of steps, the final result, and the absolute and relative errors.

Solution 6 (5 pts x 1 code + 5 parts = 30 pts)

Sample code follows (and is embedded in this PDF file):

```
function [r, h] = bisection(a, b, f, p, t)
% a: Beginning of interval [a, b]
% b: End of interval [a, b]
% f: function handle y = f(x, p)
% p: parameters to pass through to f
% t: User-provided tolerance for interval width

    h = [];

    while 1
        m = middle(a, b);
        fa = f(a, p);
        fb = f(b, p);
        fm = f(m, p);

        % Record step, terminate if f vanishes
        if fa == 0
            r = a;
            h = [h, [a; b; fm]];
            break
        elseif fb == 0
            r = b;
            h = [h, [a; b; fm]];
            break
        else
            r = m;
            h = [h, [a; b; fm]];
        end

        % Terminate if b - a is small
        if (b - a <= t * min(abs(a), abs(b)) || a == m || b == m)
```

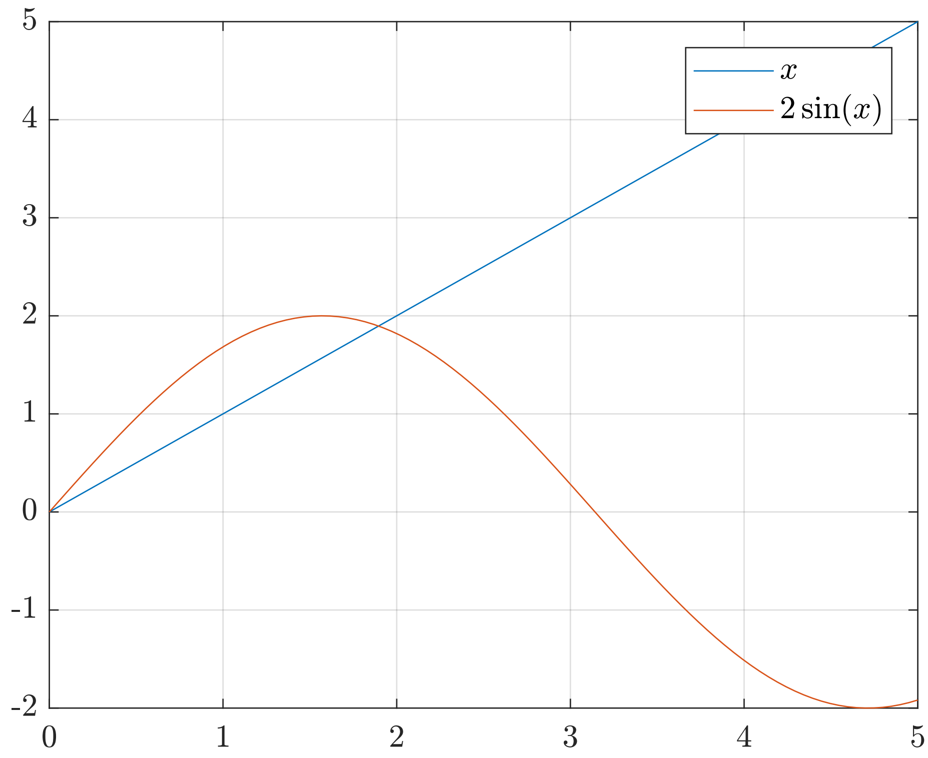
```
            break
        end

        % Bisect otherwise
        if sign(fa) ~= sign(fm)
            b = m;
        else
            a = m;
        end
    end

end

function m = middle(a, b)

    % Find the midpoint m
    if a == 0
        m = realmin;
    elseif b == 0
        m = -realmin;
    elseif sign(a) ~= sign(b)
        m = 0;
    else
        m = sign(a) * sqrt(abs(a)) * sqrt(abs(b));
    end

end
```

In this code the first recorded step is always $(a, b, f(m))$, where a and b are the input to `bisection.m`. There is always at least one step, and at most 65.

For the following problems we use the following sample code to display our results:

```
function bisection_results(a, b, f, p, t)

    [r, h] = bisection(a, b, f, p, t);
    matlab_result = fzero(@(x) f(x, p), r);
```

```

    abs_err = abs(matlab_result - r);
    rel_err = abs_err/abs(matlab_result);

  line = sprintf( ' steps = %d, r = %20.16g, abs err =%9.5g,
     rel err =%9.5g ', size( h, 2 ), r, abs_err , rel_err );
  disp( line )

end
```



(a) The plot follows:

(b) The output of `bisection_results.m` is as follows:

```
f =

@(x, p) x - 2 * sin (x)
```

```
octave:2> bisection_results(1, 3, f, 1, eps )
 steps = 53, r =    1.895494267033981, abs err =1.5543e-15, rel err =  8.2e-16
```

(c) There are two roots, so we run `bisection_results.m` twice with $[a, b] = [0.1, 1]$ and $[a, b] = [6, 7]$:

```
octave:2> bisection_results(0.1, 1, f, 1, eps )
 steps = 54, r =   0.3178444328993726, abs err =        0, rel err =        0
octave:3> bisection_results(6, 7, f, 1, eps )
 steps = 48, r =    6.305395279271691, abs err =        0, rel err =        0
```

(d) The output of `bisection_results.m` is as follows:

```
octave:1> f = @(x,p) ( x - eps^3)^3
f =

@(x, p) (x - eps ^ 3) ^ 3

octave:2> bisection_results( -1, 2, f, 1, eps )
 steps = 62, r = 1.094764425253763e-47, abs err =        0, rel err =        0
```

(e) The output of `bisection_results.m` is as follows:

```
octave:1> f = @(x,p) atan( x - eps^2)
f =

@(x, p) atan (x - eps ^ 2)

octave:2> bisection_results( -1, 2, f, 1, eps )
 steps = 64, r = 4.930380657631323e-32, abs err =        0, rel err =        0
```