Queens College, CUNY, Department of Computer Science

Numerical Methods

CSCI 361 / 761

Spring 2018

Instructor: Dr. Sateesh Mane

Final Spring 2018

solutions added

- **NOTE: It is the policy of the Computer Science Department to issue a failing grade to any student who either gives or receives help on any test.**
- This is an **open-book** test.
- **Any problem to which you give two or more (different) answers receives the grade of zero automatically.**
- This is a **take home exam.**
  Please submit your solution via email, as a file attachment, to `Sateesh.Mane@qc.cuny.edu`. The file name should have either of the formats:

  `StudentId_first_last_CS361_final_May2018`

  `StudentId_first_last_CS761_final_May2018`

  Acceptable file types are txt, doc/docx, pdf (also cpp, with text in comment blocks).
- **In all questions where you are asked to submit programming code, programs which display any of the following behaviors will receive an automatic F:**
  1. Programs which do not compile successfully (compiler warnings which are not fatal are excluded, e.g. use of deprecated features).
  2. Array out of bounds.
  3. Dereferencing of uninitialized variables (including null pointers).
  4. Operations which yield NAN or infinity, e.g. divide by zero, square root of negative number, etc. *Infinite loops.*
  5. Programs which do NOT implement the public interface stated in the question.
- **In addition, note the following:**
  1. Programs which compile and run successfully but have memory leaks will receive a poor grade (but not F).
  2. All debugging and/or output statements (e.g. `cout` or `printf`) will be commented out.
  3. Program performance will be tested solely on function return values and the values of output variable(s) in the function arguments.
  4. In other words, program performance will be tested solely via the public interface presented to the calling application. (I will write the calling application.)

General information

- **You are permitted to copy and use the code in the online lecture notes.**
- **Value of π to machine precision on any computer.**
  1. Some compilers support the constant `M_PI` for π, in which case you can write

     ```
     const double pi = M_PI;
     ```

  2. If your compiler does not support `M_PI`, the value of π can be computed via

     ```
     const double pi = 4.0*atan2(1.0,1.0);
     ```

- **64–bit computers**
  1. The questions in this exam do not involve problems of overflow.
  2. Solutions involving the writing of algorithms will not be judged if they work on a 64–bit instead of a 32–bit computer.

- **If you submit code, put all your code in ONE cpp file.**
  Else include all the code in your main docx or pdf or txt file.
  DO NOT SUBMIT MULTIPLE CPP FILES.

1 Question 1 (submit code)

- You are given input arrays a and b, both of length n and both of type `double`.
- Hence the array elements are `a[0],...,a[n-1]` and `b[0],...,b[n-1]`.
- **Write functions to efficiently calculate the following sums of products:**

$$S_1 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a[i]\, b[j]\,. \tag{1.1}$$

$$S_2 = \sum_{i=0}^{n-1} \sum_{j=0}^{i} a[i]\, b[j]\,. \tag{1.2}$$

- **Return** 0.0 **if** $n \leq 0$**.**
- The function signatures are

```
double sum_prod1(int n, const double a[], const double b[]);
double sum_prod2(int n, const double a[], const double b[]);
```

- *Hint: If you do not immediately see the answer, it may help to work out a few simple cases "by hand" for example* $n = 1$*,* $n = 2$*, etc., to see the general pattern.*

- **Both S_1 and S_2 contain $O(n^2)$ terms but can be computed in $O(n)$ steps.**

$$\begin{aligned} S_1 &= \Big(a[0] + a[1] + \cdots + a[n-1]\Big)\Big(b[0] + b[1] + \cdots + b[n-1]\Big) \\ &= \left(\sum_{i=0}^{n-1} a[i]\right)\left(\sum_{i=0}^{n-1} b[i]\right). \end{aligned}$$

- **Hence S_1 equals a product of two sums each of length n.**
- **Hence S_1 can be computed in $O(n)$ steps using one loop as follows:**

```
double sum_prod1(int n, const double a[], const double b[])
{
  if (n <= 0) return 0.0;
  double a_sum = 0.0;
  double b_sum = 0.0;
  for (int i = 0; i < n; ++i) {
    a_sum += a[i];
    b_sum += b[i];
  }
  double result = a_sum*b_sum;
  return result;
}
```

- **Computational complexity: $2n$ additions and one multiplication.**
- **It is actually not necessary to test for $n \leq 0$, the loop will deal with it.**

```
double sum_prod1(int n, const double a[], const double b[])
{
  double a_sum = 0.0;
  double b_sum = 0.0;
  for (int i = 0; i < n; ++i) {
    a_sum += a[i];
    b_sum += b[i];
  }
  double result = a_sum*b_sum;
  return result;
}
```

- **Next consider S_2:**

$$S_2 = a[0]b[0] + a[1]\big(b[0] + b[1]\big) + a[2]\big(b[0] + b[1] + b[2]\big) + \ldots$$

- **Schematic algorithm (B_{sum} is a temporary variable):**

$$\begin{aligned}
B_{\text{sum}} &= b[0] \\
S_2 &= a[0]B_{\text{sum}} \\
B_{\text{sum}} &= B_{\text{sum}} + b[1] \qquad (= b[0] + b[1]) \\
S_2 &= S_2 + a[1]B_{\text{sum}} \\
B_{\text{sum}} &= B_{\text{sum}} + b[2] \qquad (= b[0] + b[1] + b[2]) \\
S_2 &= S_2 + a[2]B_{\text{sum}} \\
&\vdots
\end{aligned}$$

- **Keep a running total `b_sum`, increment `b_sum += b[i]` at each step.**
- **Hence S_2 can be also computed in $O(n)$ steps using one loop as follows:**

```
double sum_prod2(int n, const double a[], const double b[])
{
  if (n <= 0) return 0.0;
  double result = 0.0;
  double b_sum = 0.0;
  for (int i = 0; i < n; ++i) {
    b_sum += b[i];
    result += a[i]*b_sum;
  }
  return result;
}
```

- **Computational complexity: $2n$ additions and n multiplications.**
- **Alternative, also works (keep running total `a_sum` and count down):**

```
double sum_prod2(int n, const double a[], const double b[])
{
  if (n <= 0) return 0.0;
  double result = 0.0;
  double a_sum = 0.0;
  for (int i = n-1; i >= 0; --i) {
    a_sum += a[i];
    result += b[i]*a_sum;
  }
  return result;
}
```

2 Question 2 (submit code)

- It was explained in class what is a Grey code and how to create a Grey code.
- You are given an input `nbits`, and define an integer $n = 2^{\text{nbits}}$.
- **Write a function to calculate a Grey code of length n.**
  1. The function signature is

     ```
     void Grey_code(int nbits, int n, int gc[], std::string gcstr[]);
     ```

  2. Here `gc[]` and `gcstr[]` are output arrays both of length n.
  3. The array `gc[]` contains the Grey code integers.
  4. **The value of `gc[i]` should lie in the interval** $0 \leq \texttt{gc[i]} \leq n-1$**.**
  5. The value of `gcstr[i]` contains the binary representation of `gc[i]`.
  6. (Optional, but better if you do it.) The string `gcstr[i]` should be padded so that its length is `nbits`.
- Your function will be tested using the following calling program

```
void Grey_code_test()
{
  for (int nbits = 2; nbits < something; ++nbits) {  // *** I will set the limits ***
    int n = 1 << nbits; // 2^n
    int gc[n];
    std::string gcstr[n];
    Grey_code(nbits, n, gc, gcstr);
    for (int i = 0; i < n; ++i) {
      std::cout << std::setw(10) << gc[i] << std::setw(10) << gcstr[i] << std::endl;
    }
    std::cout << endl;
  }
}
```

- Note the following:
  1. A Grey code is not unique.
  2. **All correct answers will be accepted *(but use nonnegative integers only).***
  3. You are permitted to call `Grey_code(...)` recursively.
  4. You are permitted to write "helper functions" to perform additional tasks and `Grey_code(...)` can call them.
  5. **Solutions which implement a lookup table which contains a Grey code to a large value of n will receive a failing grade.**

- **It is actually "Gray code" not Grey code, but that was my mistake.**
  1. **Any solution which returns a valid Gray code is acceptable.**
  2. **Computational efficiency was not judged.**
  3. **Solutions were graded only if the array `gcstr[]` contained a correct result.**
  4. **Some students misunderstood and returned `gc[i] = i`, this was excused.**
- **Here is one possible implementation, not necessarily the best.**
- **Students submitted numerous solutions with clever bit manipulations.**

```
void int_to_bits(int nbits, int b, std::string &s)
{
  s.clear();
  for (int i = 0; i < nbits; ++i) {
    int r = b % 2;
    std::string t = (r == 1) ? "1" : "0";
    s.insert(0, t);
    b /= 2;
  }
}

void Gray_code(int nbits, int n, int gc[], std::string gcstr[])
{
  if (n == 1) {
    gc[0] = 0;
    int_to_bits(nbits, gc[0], gcstr[0]);
    return;
  }
  else {
    int nhalf = n/2;
    Gray_code(nbits, nhalf, gc, gcstr);
    for (int j = 0; j < nhalf; ++j) {
      int k = nhalf + j;
      gc[k] = nhalf + gc[nhalf-1-j];
      int_to_bits(nbits, gc[k], gcstr[k]);
    }
  }
}
```

- **Here is a sample solution, for $n_{\text{bits}} = 4$ hence $n = 2^4 = 16$.**
- **If a student solution did not pad the strings, that was excused.**

i	gc[i]	gcstr[i]
0	**0**	**0001**
1	**1**	**0001**
2	**3**	**0011**
3	**2**	**0010**
4	**6**	**0110**
5	**7**	**0111**
6	**5**	**0101**
7	**4**	**0100**
8	**12**	**1100**
9	**13**	**1101**
10	**15**	**1111**
11	**14**	**1110**
12	**10**	**1010**
13	**11**	**1011**
14	**9**	**1001**
15	**8**	**1000**

3 Question 3 (submit code)

- It was derived in class that the Fourier series of a rectangular function of unit area, centered on $\theta = 0$ and half width θ_0 (so $f(\theta) = 0$ if $|\theta| > \theta_0$) is

$$a_j = \frac{1}{\pi} \frac{\sin(j\theta_0)}{j\theta_0} \qquad (j \geq 1)\,. \tag{3.1}$$

- In addition $a_0 = 1/\pi$ and $b_j = 0$ for all j.
- Let us construct a Fourier series with $a_0 = 1/\pi$ and $b_j = 0$ for all j and

$$a_j = \frac{1}{\pi} \frac{|\sin(j\theta_0)|}{j\theta_0} \qquad (j \geq 1)\,. \tag{3.2}$$

- **We shall use eq. (3.2) for a_j and set $\theta_0 = \pi/8$ in this question.**
- The function $f(\theta)$ is given by the sum of the Fourier series,

$$f_\infty(\theta) = \frac{1}{2\pi} + \frac{1}{\pi} \sum_{j=1}^{\infty} \frac{|\sin(j\theta_0)|}{j\theta_0} \cos(j\theta)\,. \tag{3.3}$$

- Exactly at $\theta = 0$ the function diverges logarithmically

$$f_\infty(0) = \frac{1}{2\pi} + \frac{1}{\pi} \sum_{j=1}^{\infty} \frac{|\sin(j\theta_0)|}{j\theta_0} \to \infty\,. \tag{3.4}$$

- For practical computation, we restrict the sum to a finite number of n terms:

$$f_n(\theta) = \frac{a_0}{2} + \sum_{j=1}^{n-1} a_j \cos(j\theta)\,. \tag{3.5}$$

- We employ a finite sum in eq. (3.5) hence we obtain a finite value for $f_n(0)$.
- **Compute the sum in eq. (3.5) for $f_n(\theta = 0)$ and fill the table below.**

n	$f_n(0)$
2^{16}	2 d.p.
2^{20}	2 d.p.
2^{24}	2 d.p.

- The function $f_\infty(\theta)$ also has weaker logarithmic singularities at $\theta = \pm\pi/4$, $\theta = \pm\pi/2$, $\theta = \pm 3\pi/4$ and $\theta = \pm\pi$.

- **Set $n = 2^{11} = 2048$ in the rest of this question.**
- **Plot a graph of $f_n(\theta)$ for $-\pi \le \theta \le \pi$.**
  1. Define $\theta_i = 2\pi i/n$, where $-(n/2) \le i \le (n/2)$, so $-\pi \le \theta_i \le \pi$.
     Therefore in this question $-1024 \le i \le 1024$.
  2. Compute the sum in eq. (3.5) at $\theta = \theta_i$ only.
  3. Plot the value of θ_i/π on the horizontal axis, so the numbers go from -1 to 1.
- **Calculate the area "under" the curve. State your answer to 3 decimal places.**
  1. The integral is given as follows:
  $$A_n = \int_{-\pi}^{\pi} f_n(\theta)\, d\theta\,. \tag{3.6}$$
  2. State your choice of integration method and any other relevant details.
  3. The integral in eq. (3.6) is an improper integral (but we know where the function $f_\infty(\theta)$ diverges).
  4. Explain how you will deal with the improper integral.
- The function $f_n(\theta)$ crosses zero once at $\theta = \theta_r$, where $0 < \theta_r < \pi/4$.
- **Employ a root finding algorithm to determine the value of θ_r to an accuracy of 4 decimal places.**
  1. State your choice of algorithm.
  2. Also state the value of your starting iterate(s).
  3. **Justify your choice** for your starting iterate(s).
  4. Display a table of your iterates to show convergence to 4 decimal places.
- **Calculate the area under the curve for $0 < \theta < \theta_r$, say A_r.**
  1. The integral is given by
  $$A_r = \int_{0}^{\theta_r} f_n(\theta)\, d\theta\,. \tag{3.7}$$
  2. *This calculation requires some precautions.*
  3. It is an improper integral because $f_\infty(0) = \infty$ hence the function diverges at one endpoint.
  4. Although the value of $f_n(0)$ is finite, that is cheating.
  5. We must employ the midpoint rule, which does not evaluate the function at the endpoints of the interval.
  6. Now we encounter some confusion about the meaning of n.
  7. We have used n to denote the number of Fourier harmonics in the sum in eq. (3.5).
  8. Let the number of subintervals for the midpoint rule be n_{mid}.
  9. **Calculate the value of the integral in eq. (3.7) using $n_{\text{mid}} = 1000$ subintervals.**
  10. **State your answer to 3 decimal places.**

- **Compute the sum in eq. (3.5) for $f_n(\theta = 0)$ and fill the table below.**

n	$f_n(0)$
2^{16}	**6.08**
2^{20}	**7.49**
2^{24}	**8.90**

- **A graph of $f_n(\theta)$ for $n = 2^{11}$ is plotted in Fig. 1. Note from the definition in eq. (3.5) that $f_n(\theta)$ is a sum of cosines hence it is an even function of θ.**

- **Area under the curve for $-\pi < \theta < \pi$.**

  1. **The function $f_n(\theta)$ is *defined by the sum in eq. (3.5).***
  2. **For a periodic function, the Fourier harmonics a_j are given by**

$$a_j = \frac{1}{\pi} \int_{-\pi}^{\pi} f_n(\theta) \, \cos(j\theta) \, d\theta \, .$$

  3. **Put $j = 0$ then**

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f_n(\theta) \, d\theta \, .$$

  4. **Hence the area A_n is simply $\pi a_0 = 1$:**

$$A_n = \int_{-\pi}^{\pi} f_n(\theta) \, d\theta = \pi a_0 = 1 \, .$$

  5. **A more long-winded derivation is to integrate the expression in eq. (3.5)**

$$\begin{aligned} A_n &= \int_{-\pi}^{\pi} f_n(\theta) \, d\theta \\ &= \int_{-\pi}^{\pi} \left[\frac{a_0}{2} + \sum_{j=1}^{n-1} a_j \, \cos(j\theta) \right] d\theta \\ &= \int_{-\pi}^{\pi} \frac{a_0}{2} \, d\theta + \sum_{j=1}^{n-1} a_j \underbrace{\int_{-\pi}^{\pi} \cos(j\theta) \, d\theta}_{=0} \\ &= \pi \, a_0 = 1 \, . \end{aligned}$$

  6. **This is an exact result.**
  7. **There is no need to perform a numerical integration, or to worry about improper integrals.**
  8. **This class is about analysis, not simply computation.**
  9. **The area would equal 1 even for $n \to \infty$.**
  10. **Technically, one must justify that $\int_{-\pi}^{\pi} |f_\infty(\theta)| \, d\theta < \infty$, but this can be done.**

- **Root θ_r.**
  1. **Any root finding algorithm for θ_r is acceptable.**
  2. **Solutions were not judged on the choice of initial bracket/iterate or the number of iterations.**
  3. **The value of the root to four decimal places is**
  $$\theta_r \simeq 0.7074\,.$$
  4. **This is suggestively close to $\sqrt{1/2} \simeq 0.7071$.**
  5. **This may be a numerical coincidence, or possibly the exact result (for $n \to \infty$) is $\theta_r = \sqrt{1/2}$. I do not know.**

- **The area under the curve, for $0 < \theta < \theta_r$, computed using the midpoint rule with 1000 subintervals, is**
$$A_r = \int_0^{\theta_r} f_n(\theta)\, d\theta \simeq 0.527\,.$$

- **Note that if we integrate over the symmetric interval $-\theta_r < \theta < \theta_r$, the area is $2A_r \simeq 1.054$ which is greater than $A_n = 1$.**
  1. **This is correct.**
  2. **The "area" is a signed area and can have negative contributions.**
  3. **Observe from the graph in Fig. 1 that $f_n(\theta)$ is negative for some values of θ.**
  4. **However, by definition $f_n(\theta) > 0$ in the interval $-\theta_r < \theta < \theta_r$.**
  5. **Hence there is no contradiction that the integral over the interval $-\theta_r < \theta < \theta_r$ exceeds that over the full interval $-\pi < \theta < \pi$.**
$$\int_{-\theta_r}^{\theta_r} f_n(\theta)\, d\theta > \int_{-\pi}^{\pi} f_n(\theta)\, d\theta\,.$$

- **There is another way to compute the area A_r, and we do not have to worry about improper integrals, but the formalism requires (a lot?) more theory.**
- **Note that because $f_n(\theta)$ is even in θ, the area is given by half the integral from $-\theta_r$ to θ_r:**

$$A_r = \frac{1}{2}\int_{-\theta_r}^{\theta_r} f_n(\theta)\,d\theta\,.$$

- **We can interpret this as an integral of $f_n(\theta)$ multiplied by a window function which extends from $-\theta_r \le \theta \le \theta_r$:**

$$g(\theta) = \begin{cases} 1 & (|\theta| \le \theta_r) \\ 0 & (|\theta| > \theta_r)\,. \end{cases}$$

- **Note that $g(\theta)$ is actually a window function with base $2\theta_r$ and height 1, hence its area is $2\theta_r$.**
- **The integral of $f_n(\theta)g(\theta)$ extends over the full circle $-\pi < \theta < \pi$:**

$$A_r = \frac{1}{2}\int_{-\pi}^{\pi} f_n(\theta)g(\theta)\,d\theta\,.$$

- ***We now invoke the convolution theorem.***
- **The Fourier series of a product is the convolution of the individual Fourier series.**
- **Define the complex Fourier coefficients F_k and G_k**

$$f_n(\theta) = \sum_{j=-\infty}^{\infty} F_j\,e^{ij\theta}\,, \qquad g_n(\theta) = \sum_{j=-\infty}^{\infty} G_j\,e^{ij\theta}\,.$$

- **In practice, F_j is nonzero only for $-n < j < n$ because of the definition of $f_n(\theta)$ as a finite sum of harmonics (see eq. (3.5)).**
- **Then**

$$\begin{aligned} A_r &= \frac{1}{2}\int_{-\pi}^{\pi} f_n(\theta)g(\theta)\,d\theta \\ &= \frac{1}{2}\int_{-\pi}^{\pi}\left(\sum_{j_1=-\infty}^{\infty} F_{j_1}\,e^{ij_1\theta}\right)\left(\sum_{j_2=-\infty}^{\infty} G_{j_2}\,e^{ij_2\theta}\right)d\theta \\ &= \frac{1}{2}\int_{-\pi}^{\pi}\left(\sum_{j_1,j_2} F_{j_1}G_{j_2}\,e^{i(j_1+j_2)\theta}\right)d\theta \\ &= \pi\sum_{j_1+j_2=0} F_{j_1}G_{j_2} \\ &= \pi\sum_{j=-\infty}^{\infty} F_jG_{-j} \\ &= \pi\sum_{j=-n+1}^{n-1} F_jG_{-j}\,. \end{aligned}$$

- **From the definition of the complex Fourier harmonics, we obtain**

$$F_j = \frac{a_{|j|}}{2} = \frac{1}{2\pi} \frac{|\sin(j\theta_0)|}{|j\theta_0|} .$$

- **For a window function of half width θ_r and area $2\theta_r$, we obtain**

$$G_j = \frac{2\theta_r}{2\pi} \frac{\sin(j\theta_r)}{j\theta_r} = \frac{\sin(j\theta_r)}{j\pi} .$$

- **Put it all together, noting that $F_{-j} = F_j$ and $G_{-j} = G_j$, and the area is**

$$A_r = \frac{\theta_r}{2\pi} \left[1 + 2 \sum_{j=1}^{n-1} \frac{|\sin(j\theta_0)|}{j\theta_0} \frac{\sin(j\theta_r)}{j\theta_r} \right] .$$

- **This yields the correct answer (try it), and does not require the introduction of an addition parameter "n_{mid}" for subintervals for numerical integration.**
- **This would be a very difficult exam question for you to solve.**

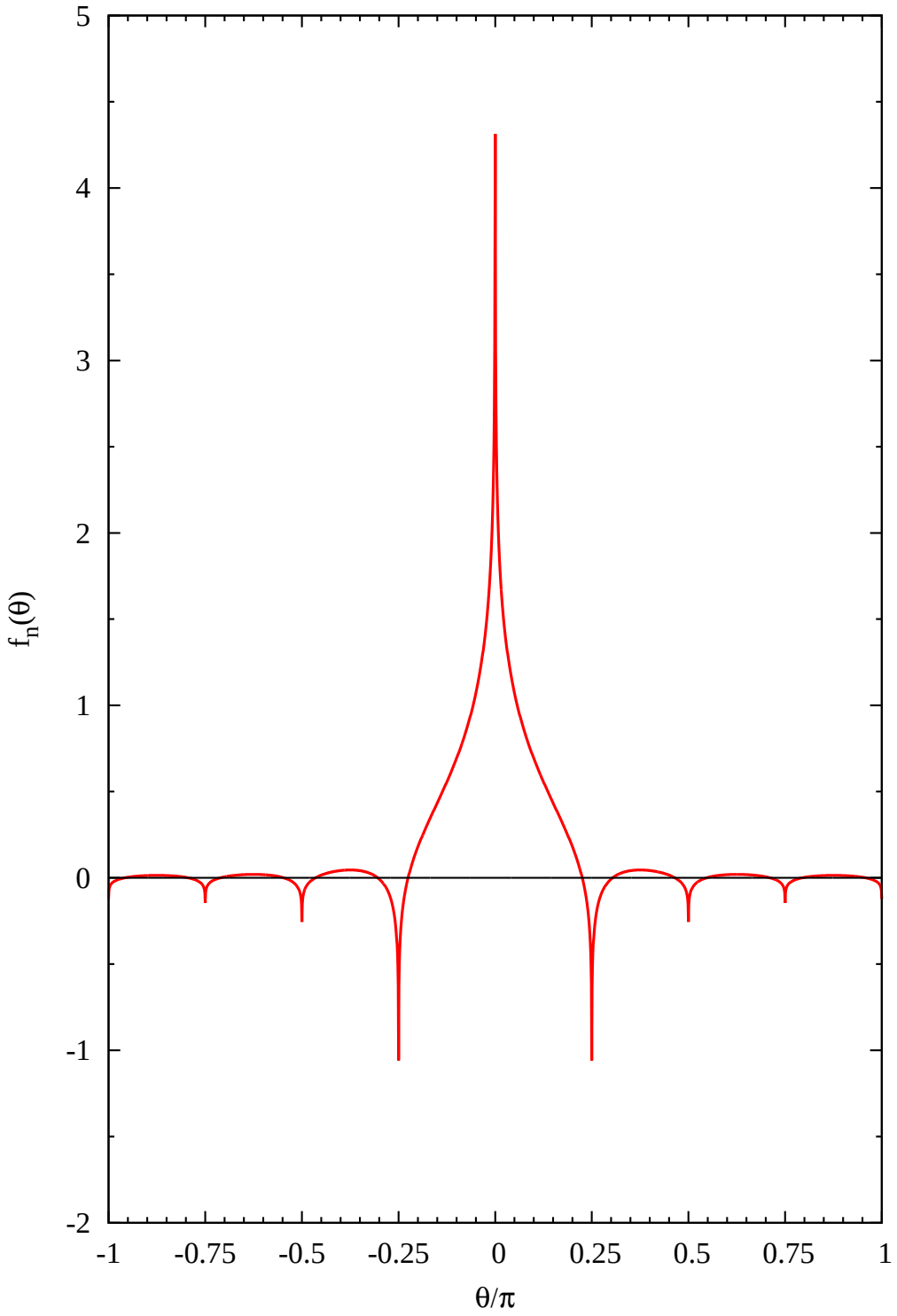

Figure 1: Graph of $f_n(\theta)$ in Question 3.

4 Question 4 (submit code)

- A **parametric curve** is a curve (x, y) where $x(t)$ and $y(t)$ are functions of a parameter t.
- We shall employ the following parametric curve:

$$\begin{aligned} x(t) &= (1 - s\rho)\cos(t) + s\rho\,\cos\Big(\frac{1-\rho}{\rho}\,t\Big)\,, \\ y(t) &= (1 - s\rho)\sin(t) - s\rho\,\sin\Big(\frac{1-\rho}{\rho}\,t\Big)\,. \end{aligned} \tag{4.1}$$

- We shall also require the following derivatives later in this question.

$$\begin{aligned} \frac{dx}{dt} &= -(1 - s\rho)\sin(t) - s(1-\rho)\,\sin\Big(\frac{1-\rho}{\rho}\,t\Big)\,, \\ \frac{dy}{dt} &= \quad (1 - s\rho)\cos(t) - s(1-\rho)\,\cos\Big(\frac{1-\rho}{\rho}\,t\Big)\,. \end{aligned} \tag{4.2}$$

- **Set the parameter values $\rho = 1/6$ and $s = 5/4 = 1.25$ in this question.**
- **Warning: the parameter t is not the same as the polar angle θ.**
  1. In polar coordinates (r, θ), $x = r\,\cos\theta$ and $y = r\,\sin\theta$ and $\theta = \arctan(y/x)$.
  2. It is obvious from eq. (4.1) that $t \neq \arctan(y/x)$, hence $\theta \neq t$ in general.
  3. However, it *is* true that $\theta = 0$ when $t = 0$ and $\theta = 2\pi$ when $t = 2\pi$.
  4. Hence the interval $0 \leq t \leq 2\pi$ yields a closed curve in the (x, y) plane.
- A graph of the curve is plotted in Fig. 2, for $0 \leq t \leq 2\pi$.
- **Calculate the area enclosed in the central portion of the curve.**
- That is to say, the area excluding the lobes.
- *Let's not panic.*
- First find the value of t such that the curve crosses the x-axis at the point marked α in Fig. 2.
- Call this value t_α. It is the smallest value $t > 0$ such that $y(t_\alpha) = 0$.
- **Employ a root finding algorithm to find t_α, which is the smallest value $t > 0$ such that $y(t_\alpha) = 0$.**
  1. State your choice of algorithm.
  2. Also state the value of your starting iterate(s).
  3. **Justify your choice** for your starting iterate(s).
  4. Iterate until the value of t_α converges to 4 decimal places.
  5. **State the value of t_α.**
  6. Display a table of your iterates to show convergence to 4 decimal places.

- **Find the second value of t where the curve self-intersects. Call the value t_β.**
  1. This is the point marked β in Fig. 2.
  2. **State the value of t_β and explain how you calculated it.**
  3. Just describe your algorithm, with enough detail so I can verify it is correct.
  4. You do not need to show iterates (or whatever) to calculate t_β.
- The area enclosed by the curve, from t_α to t_β is given by

$$A(t_\alpha, t_\beta) = \frac{1}{2} \int_{t_\alpha}^{t_\beta} \left(x \frac{dy}{dt} - y \frac{dx}{dt} \right) dt \,. \tag{4.3}$$

- By the six-fold symmetry, the total area of the central enclosed region is $A_{\text{tot}} = 6A(t_\alpha, t_\beta)$.
  1. **Calculate the value of A_{tot} using $n = 1000$ subintervals.**
  2. Use (i) trapezoid rule, (ii) midpoint rule, (iii) Simpson's rule.
  3. State your answers to 4 decimal places.
- Let us perform some algebra and trigonometry to process the integrand in eq. (4.3).

$$x \frac{dy}{dt} - y \frac{dx}{dt} = (1 - s\rho)^2 - s^2\rho(1-\rho) + s(2\rho - 1)(1 - s\rho) \cos\left(\frac{t}{\rho}\right). \tag{4.4}$$

- Then the area is

$$A_{\text{tot}} = 3\big[\,(1 - s\rho)^2 - s^2\rho(1-\rho)\,\big](t_\beta - t_\alpha) + 3s\rho(2\rho - 1)(1 - s\rho) \left[\sin\left(\frac{t_\beta}{\rho}\right) - \sin\left(\frac{t_\alpha}{\rho}\right) \right]. \tag{4.5}$$

- **Calculate the value of A_{tot} using eq. (4.5).**
- State your answer to 4 decimal places.
- This will give an estimate of the accuracy of the numerical integration.
- Of course, the value from eq. (4.5) is not exact because the values of t_α and t_β are not known exactly.

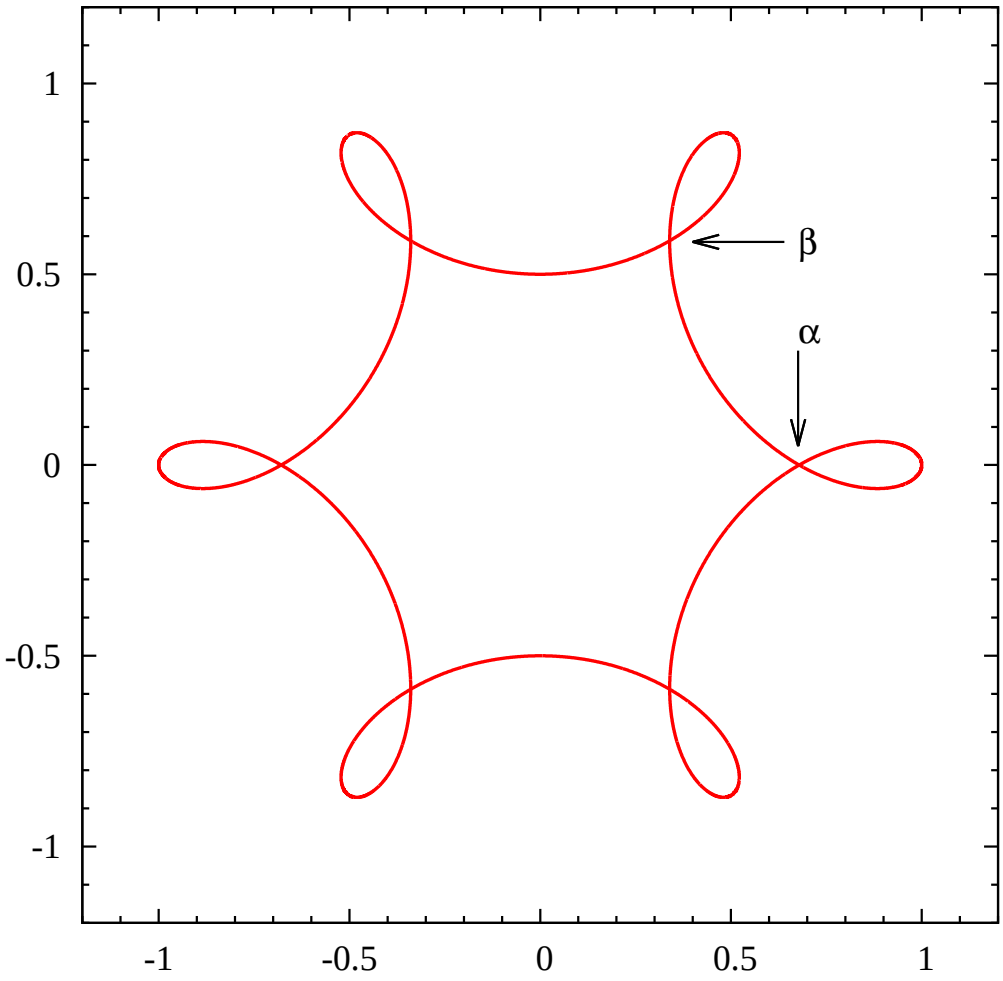

Figure 2: Graph of the curve in Question 4.

- **Value of t_α.**
  1. **Any root finding algorithm for t_α is acceptable.**
  2. **Solutions were not judged on the choice of initial bracket/iterate or the number of iterations.**
  3. **The value of the root to four decimal places is**

$$t_\alpha \simeq 0.2541\,.$$

- **Value of t_β.**
  1. **The exact relx2ation between t_β and t_α is**

$$t_\beta = \frac{\pi}{3} - t_\alpha\,.$$

  2. **Do you see why?**
  3. **By symmetry, the end-points of the "rightmost lobe" are $-t_\alpha$ and t_α.**
  4. **By the six-fold symmetry of the curve, the end-points of the other lobes are**

$$\begin{aligned}
1 \times \frac{2\pi}{6} \mp t_\alpha &= \left(\frac{\pi}{3} - t_\alpha, \quad \frac{\pi}{3} + t_\alpha\right), \\
2 \times \frac{2\pi}{6} \mp t_\alpha &= \left(\frac{2\pi}{3} - t_\alpha, \quad \frac{2\pi}{3} + t_\alpha\right), \\
3 \times \frac{2\pi}{6} \mp t_\alpha &= \left(\pi - t_\alpha, \quad \pi + t_\alpha\right), \\
4 \times \frac{2\pi}{6} \mp t_\alpha &= \left(\frac{4\pi}{3} - t_\alpha, \quad \frac{4\pi}{3} + t_\alpha\right), \\
5 \times \frac{2\pi}{6} \mp t_\alpha &= \left(\frac{5\pi}{3} - t_\alpha, \quad \frac{5\pi}{3} + t_\alpha\right).
\end{aligned}$$

  5. **Hence the arc we seek is given by $t_\alpha \le t \le (\pi/3) - t_\alpha$, hence $t_\beta = (\pi/3) - t_\alpha$.**
  6. **Solutions which employed this symmetry to calculate t_β received extra credit.**
  7. **This class is about analysis, not simply computation.**
  8. **Numerically, given the (calculated) value of t_α, the value of t_β is**

$$t_\beta \simeq 0.7931\,.$$

- **The value of the area of the central region, using all three numerical integration algorithms, is**

$$A_{\text{tot}} \simeq 1.3215\,.$$

- **The value of the area of the central region, using calculus, is the same (to four decimal places)**

$$A_{\text{tot}} \simeq 1.3215\,.$$

Lissajous figure (non-question)

- **There is no Lissajous figure.**
- I substituted the curve in Question 4 instead of a Lissajous figure (prettier).
- The curve in Question 4 is an example of a **hypotrochoid.**
- The special case $\rho = \frac{1}{2}$ is an ellipse.
- The special case $s = 1$ is a **hypocycloid.**
- For $s = 1$ and $\rho = 1/3$ the curve is called a **deltoid.**
- For $s = 1$ and $\rho = 1/4$ the curve is called an **astroid.**
- For $s > 1$ the curves develop lobes and self-intersect, as in Fig. 2.
- *No, there's nothing for you to calculate here.*
- Just look at the pretty pictures and learn their names.

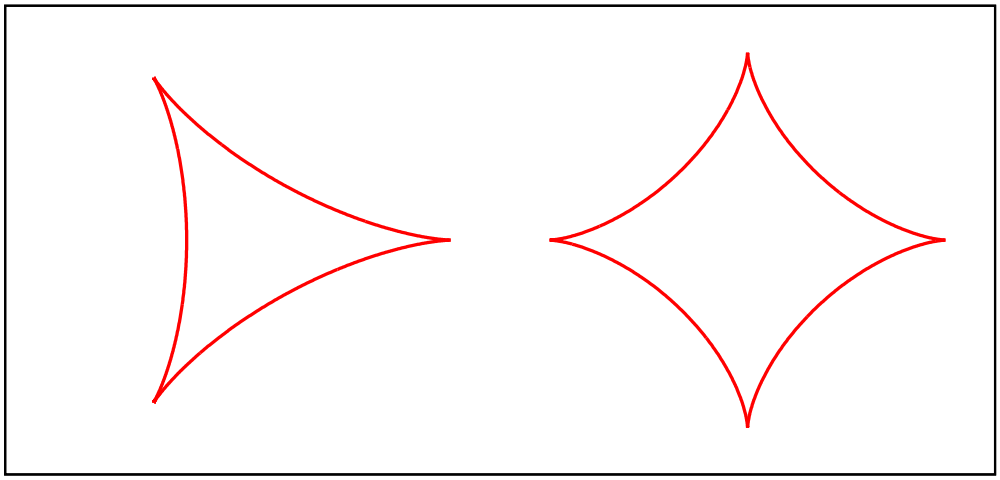

Figure 3: Plots of deltoid and astroid.

5 Question 5 (submit code)

- You are given the following first order ordinary differential equation:

$$\frac{dA}{dx} = 4\left(1 - |x|^{1.7}\right)^{1/3.2}. \tag{5.1}$$

- The initial condition is $A(1) = 0$ at $x = 1$.
- **Use Runge–Kutta fourth order `RK4` to integrate eq. (5.1) from $x = 1$ down to $x = 0$.**
- Submit your code as part of your solution.
- Use $n = 1000$ and a stepsize $h = -1/n = -1/1000$.
- Fill in the table below for the **absolute value** $|A(x)|$ as you proceed with the integration.

x	$\lvert A(x)\rvert$
1	0
0.75	4 d.p.
0.5	4 d.p.
0.25	4 d.p.
0	4 d.p.

- **This question was directly inspired by a student solution in Fall 2017.**
- Who says we need Simpson's rule, etc., to calculate the area of a superellipse?
- *Runge-Kutta will do the job very nicely and will moreover yield intermediate results for the areas of slices of the superellipse.*
- I have respect for students who think "outside the box" and have the imagination to use their knowledge in novel ways.

- **This question is directly related to Midterm 2 Question 2.**
- **This question is directly an application of a different style of numerical integration to calculate the area under a curve.**
- **Recall the superellipse in Midterm 2 Question 2.**

$$|x|^{1.7} + |y|^{3.2} = 1\,.$$

- **The area is given by**

$$A = 4\int_0^1 y(x)\,dx = 4\int_0^1 \left(1 - |x|^{1.7}\right)^{1/3.2} dx\,.$$

- **Instead of computing the integral using Simpson's rule, etc., we form an ordinary differential equation (see eq. (5.1)):**

$$\frac{dA}{dx} = 4\left(1 - |x|^{1.7}\right)^{1/3.2}.$$

- ***We compute the area by numerically integrating an ordinary differential equation.***
- **A slight twist is that the initial point is at $x = 1$ and the integeration proceeds backwards to $x = 0$.**
- **Hence the integration stepsize is $h = -1/n$, for n integration steps.**
- **As the integration procedure reaches $x = 0.75$, 0.5 and 0.25, we can read off the relevant areas.**
- **The table of integration results is as follows.**

x	$\lvert A(x)\rvert$
1	**0**
0.75	**0.5737**
0.5	**1.3993**
0.25	**2.3339**
0	**3.3227**

- **The value of $|A|$ for $x = 0$ should equal the value of the area calculated in Midterm 2 Question 2.**
- **It is a simple check on the correctness of the present calculation.**

6 Question 6 (submit code)

- You are given the following inhomogeneous ordinary differential equation:

$$\frac{d^2y}{dx^2} + y = \frac{\cos(x)}{\pi}\,. \tag{6.1}$$

- **Write down the homogeneous ordinary differential equation associated with eq. (6.1).**
- **Show that $y_1(x)$ and $y_2(x)$ are solutions of the associated homogeneous ordinary differential equation, where**

$$y_1(x) = \sin(x)\,, \qquad y_2(x) = \cos(x)\,. \tag{6.2}$$

- It is acceptable to substitute $y_1(x)$ and $y_2(x)$ into the associated homogeneous equation and show that they are solutions. It is not necessary to derive the solutions using calculus.
- **Show that $P(x)$ is a particular solution of eq. (6.1), where**

$$P(x) = \frac{x\sin(x)}{2\pi}\,. \tag{6.3}$$

- **Write down the general solution of eq. (6.1).**
- **Find the solution of eq. (6.1) which satisfies the following initial conditions at $x = 0$:**

$$y(0) = 0\,, \qquad \left[\frac{dy}{dx}\right]_{x=0} = 0\,. \tag{6.4}$$

- **Using the initial conditions in eq. (6.4), calculate the value of $y(x)$ at $x = 5\pi/2$.**
- Denote this value by y_* below, i.e. $y_* = y(5\pi/2)$.
- **Employ an auxiliary variable $v = dy/dx$ to express eq. (6.1) as a pair of coupled first order ordinary differential equations.**
- You are required to integrate the coupled equations numerically from $x_0 = 0$ forwards to $x_n = 5\pi/2$ in n steps and then backwards to $x_{2n} = 0$ also in n steps.
- Hence the stepsize is $h = (x_n - x_0)/n = 2.5\pi/n$.
- Let the value of y at $x = x_n = 5\pi/2$ be y_n and let the value of y at $x = x_{2n} = 0$ be y_{2n}.
- **Use Runge–Kutta fourth order `RK4` to integrate eq. (6.1) with the initial conditions in eq. (6.4). Find a value of n such that the following inequalities are satisfied:**

$$10^{-5} \le |y_n - y_*| \le 10^{-4}\,. \tag{6.5}$$

- ***Note the lower bound.*** The value of n should not be too small but it should also not be too large.

- **Next use `RK4` to integrate eq. (6.1) backwards to $x_{2n} = 0$.**
- **State the value of y_{2n} that you obtain.**
- **Submit your code for the integration as part of your solution (including your main program).**
- **You will be graded on your code as well as on your numerical results.**
- **You are permitted to copy code or code fragments from the online lecture notes.**
- **(Optional) Plot a graph of your numerical solution of eq. (6.1).**

- **The homogeneous ordinary differential equation is given by setting the right hand side in eq. (6.1) to zero:**

$$\frac{d^2y}{dx^2} + y = 0\,.$$

- **It is acceptable to substitute the given functions $y_1(x)$ and $y_2(x)$ into the homogeneous equation and verify that they are solutions.**

  1. **Given $y_1(x) = \sin(x)$, then**

$$\frac{dy_1}{dx} = \cos(x)\,, \qquad \frac{d^2y_1}{dx^2} = -\sin(x)\,.$$

  2. **Substituting into the homogeneous ordinary differential equation yields**

$$\frac{d^2y_1}{dx^2} + y_1 = -\sin(x) + \sin(x) = 0\,.$$

  3. **Given $y_2(x) = \cos(x)$, then**

$$\frac{dy_2}{dx} = -\sin(x)\,, \qquad \frac{d^2y_2}{dx^2} = -\cos(x)\,.$$

  4. **Substituting into the homogeneous ordinary differential equation yields**

$$\frac{d^2y_2}{dx^2} + y_2 = -\cos(x) + \cos(x) = 0\,.$$

  5. **Hence both $y_1(x) = \sin(x)$ and $y_2(x) = \cos(x)$ are solutions of the homogeneous equation.**

- **Given the particular solution $P(x)$ in eq. (6.3).**

  1. **The function and derivatives are**

$$\begin{aligned} P(x) &= \frac{x\sin(x)}{2\pi}\,, \\ \frac{dP}{dx} &= \frac{\sin(x)}{2\pi} + \frac{x\cos(x)}{2\pi} \\ \frac{d^2P}{dx^2} &= \frac{2\cos(x)}{2\pi} - \frac{x\sin(x)}{2\pi} \end{aligned}$$

  2. **Substitute in eq. (6.1) to obtain:**

$$\begin{aligned} \frac{d^2P}{dx^2} + P &= \frac{2\cos(x)}{2\pi} - \frac{x\sin(x)}{2\pi} + \frac{x\sin(x)}{2\pi} \\ &= \frac{\cos(x)}{\pi}\,. \end{aligned}$$

  3. **Hence $P(x)$ in eq. (6.3) is a particular solution of eq. (6.1).**

- **General solution of eq. (6.1) (c_1 and c_2 are arbitrary constants)**

$$\begin{aligned} y(x) &= P(x) + c_1y_1(x) + c_2y_2(x) \\ &= \frac{x\sin(x)}{2\pi} + c_1\sin(x) + c_2\cos(x)\,. \end{aligned}$$

- **Given initial conditions $y(0) = 0$ and $y'(0) = 0$ in eq. (6.4).**

  1. **The initial condition $y(0) = 0$ yields**

$$0 = 0 + c_1 \sin(0) + c_2 \cos(0) = c_2 \,.$$

  2. **Therefore $c_2 = 0$.**
  3. **The initial condition $y'(0) = 0$ (and $c_2 = 0$) yields**

$$\begin{aligned} 0 &= \left[\frac{\sin(x)}{2\pi} + \frac{x \cos(x)}{2\pi} \right]_{x=0} + c_1 y_1'(0) \\ &= 0 + 0 + c_1 \cos(0) \\ &= c_1 \,. \end{aligned}$$

  4. **Therefore $c_1 = 0$.**
  5. **Hence the solution which satisfies the initial conditions in eq. (6.4) is**

$$y(x) = P(x) = \frac{x \sin(x)}{2\pi} \,.$$

- **Using the initial conditions in eq. (6.4), the value of the solution at $x = 5\pi/2$ is**

$$y_* = y(5\pi/2) = P(5\pi/2) = \frac{5\pi \sin(5\pi/2)}{4\pi} = \frac{5}{4} = 1.25 \,.$$

- **The numerical integration can be carried out using the code supplied in the lectures. Alternatively, students may write their own code. All valid numerical routines are acceptable.**

  1. **The purpose of the inequalities in eq. (6.5) is to avoid solutions using $n = 1000$ or some other unnecessari;y large number.**
  2. **In practical applications, computations can be expensive and unnecessary calculations can be wasteful.**
  3. **The purpose of the inequalities is to test your understanding of the accuracy of the numerical results.**
  4. **This class is about analysis, not simply computation.**
  5. **The smallest value of n which satisfies the inequalities in eq. (6.5) is $n = 34$.**
  6. **The largest value of n which satisfies the inequalities in eq. (6.5) is $n = 57$.**
  7. **All solutions which state a value $34 \leq n \leq 57$ are acceptable.**
  8. **For $n = 34$ then $y_{2n} \simeq 3.11 \times 10^{-5}$.**
  9. **For $n = 57$ then $y_{2n} \simeq 2.36 \times 10^{-6}$.**

- **A graph of the solution $y(x) = P(x)$, using $n = 50$, is displayed in Fig. 4.**

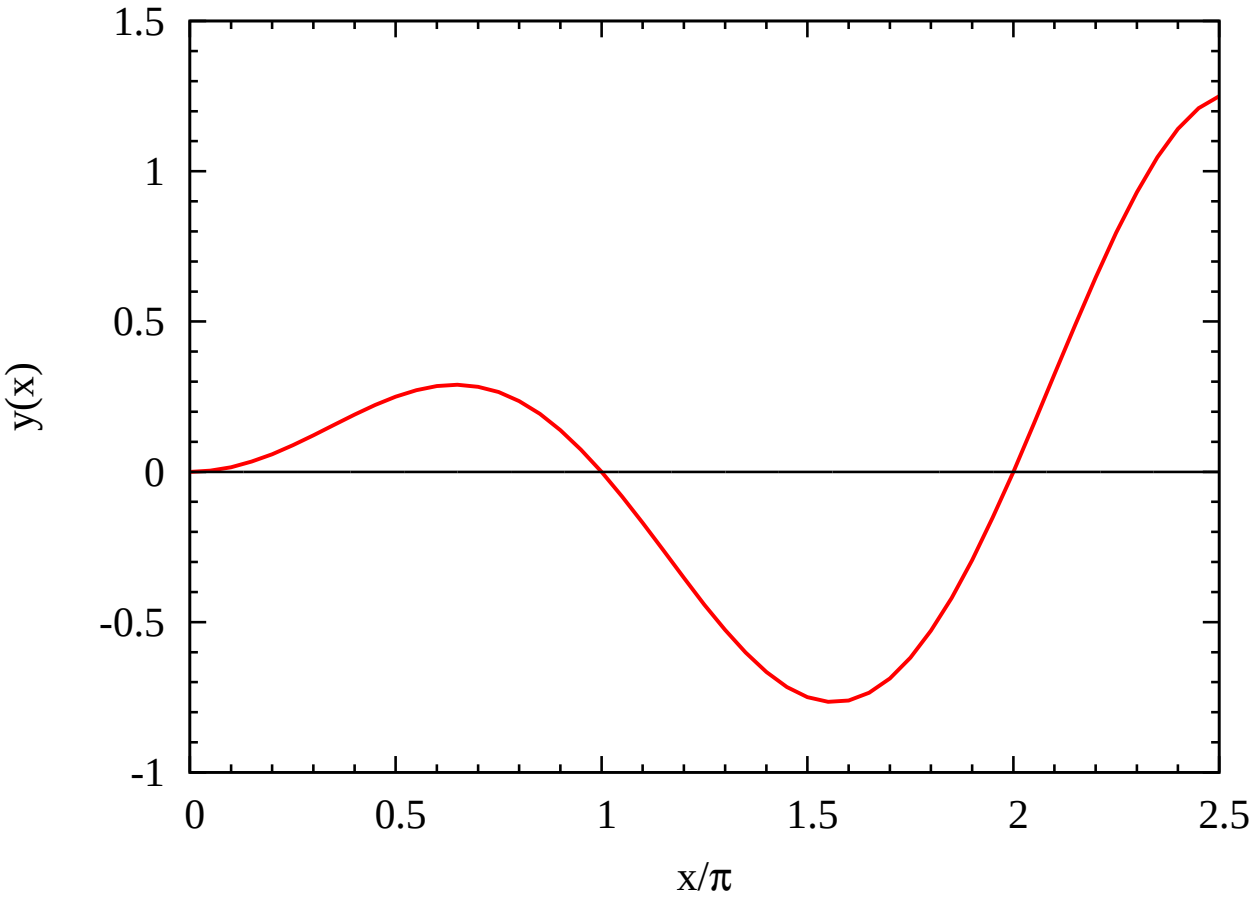

Figure 4: Graph of function $y(x)$ in Question 6.

7 Question 7 (submit code)

- The Bessel function $J_\nu(x)$ satisfies the following linear ordinary differential equation:

$$x^2 \frac{d^2y}{dx^2} + x \frac{dy}{dx} + (x^2 - \nu^2)y = 0\,. \tag{7.1}$$

- Here ν can be any number, even complex. **We shall use $\nu = 1.5$ in this question.**
- Define a set of $n+1$ equally spaced points with x_i via $x_0 = 0$ and $x_n = 20$.
- Hence the interval size we shall employ in this question is $h = 20/n$.
- **Express $y'(x_i)$ and $y''(x_i)$ using centered finite differences, for $i = 1, \ldots, n-1$.**
- We shall deal with the special cases $i = 0$ and $i = n$ later.
- **Substitute your expressions for the centered finite differences into eq. (7.1).**
- **Multiply through by h^2 to obtain an equation of the following form:**

$$a_i y_i + b_i y_{i-1} + c_i y_{i+1} = 0 \qquad (i = 1, \ldots, n-1)\,. \tag{7.2}$$

- **State your expressions for a_i, b_i and c_i, for $i = 1, \ldots, n-1$.**
- For the special cases, set $a_0 = 1$ **and** $c_0 = 0$ for $i = 0$ and $a_n = 1$ **and** $b_n = 0$ for $i = n$.
- You are given that $J_{1.5}(0) = 0$ and $J_{1.5}(20) \simeq -0.064663$. The above equations will satisfy the boundary conditions $y_0 = 0$ and $y_n = -0.064663$.
- If you have done everything correctly, you should obtain a tridiagonal set of equations of the following form. (Blanks denote zeroes in the tridiagonal matrix below.)

$$\begin{pmatrix} 1 & 0 & & & & & \\ b_1 & a_1 & c_1 & & & & \\ & b_2 & a_2 & c_2 & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & & & & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \mathbf{-0.064663} \end{pmatrix}. \tag{7.3}$$

- *The matrix in eq.* (7.3) *is NOT diagonally dominant. Do not worry.*
- **Solve eq. (7.3) using $n = 20000$.**
- Note that $J_{1.5}(0) = 0$. In addition, $J_{1.5}(x)$ oscillates and equals zero multiple times in the interval $0 < x < 20$.
- **Use your solution of the tridiagonal equations to determine where $J_{1.5}(x)$ equals zero in the interval $0 < x < 20$. State your answers to a tolerance of ± 0.001. Note that the tolerance is in the value of x, not y.**
- **(Optional) Plot a graph of $J_{1.5}(x)$ for $0 \le x \le 20$.**

- **Uniform stepsize h. Notation $x_i = ih$ and $y_i = y(x_i)$.**
- **Centered finite differences with stepsize h:**

$$\frac{dy}{dx} \simeq \frac{y_{i+1} - y_{i-1}}{2h}\,, \qquad \frac{d^2y}{dx^2} \simeq \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}\,.$$

- **Substitute your expressions for the centered finite differences into eq. (7.1).**

$$\begin{aligned} 0 &= x_i^2\,\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + x_i\,\frac{y_{i+1} - y_{i-1}}{2h} + (x_i^2 - \nu^2)y_i \\ &= x_i^2\,(y_{i+1} - 2y_i + y_{i-1}) + \frac{hx_i}{2}\,(y_{i+1} - y_{i-1}) + h^2(x_i^2 - \nu^2)y_i \\ &= \left[\, x_i^2 - \frac{hx_i}{2} \,\right] y_{i-1} + \left[\, -2x_i^2 + h^2(x_i^2 - \nu^2) \,\right] y_i + \left[\, x_i^2 + \frac{hx_i}{2} \,\right] y_{i+1}\,. \end{aligned}$$

- **Hence**

$$\begin{aligned} a_i &= -2x_i^2 + h^2(x_i^2 - \nu^2)\,, \\ b_i &= x_i^2 - \frac{hx_i}{2}\,, \\ c_i &= x_i^2 + \frac{hx_i}{2}\,. \end{aligned}$$

- **Expressions which did not multiply through by h^2 were also accepted.**
- **Given $x_0 = 0$, $x_n = 20$ and $n = 20000$. Therefore $h = (x_n - x_0)/n = 20/20000 = 0.001$.**
- **The numerical solution of eq. (7.3) can be carried out using the code supplied in the lectures. Alternatively, students may write their own code. All valid numerical routines are acceptable.**
- **A graph of $J_{1.5}(x)$ is plotted in Fig. 5.**

- **Use your solution of the tridiagonal equations to determine where $J_{1.5}(x)$ equals zero in the interval $0 < x < 20$. State your answers to a tolerance of ± 0.001. Note that the tolerance is in the value of x, not y.**
- **This is clearly a root finding problem.**
- **The task is to solve the equation $J_{1.5}(x) = 0$ and find all the roots in the interval $0 < x < 20$.**
  1. **The question specifies a tolerance of ± 0.001 in the value of x for the roots.**
  2. **How to solve this?**
  3. **Note that the stepsize in the values of x is $\Delta x = h = 0.001$.**
  4. **Hence the solution of the tridiagonal equations yields an array of values of y_i spaced at intervals of $\Delta x = h = 0.001$.**
  5. **Hence if we find a value of i such that y_{i-1} and y_i have opposite sign, then the root lies between x_{i-1} and x_i.**
  6. **The distance $|x_i - x_{i-1}|$ is $\Delta x = h = 0.001$ *and this is enough to satisfy the tolerance requirement for the value of the root.***
- **An acceptable solution for the value of the root is therefore $(x_{i-1} + x_i)/2$.**
- **The accuracy of the above solution is actually $\pm\frac{1}{2}h = \pm 0.0005$, think about it.**
- **There is no need to employ an iterative root finding algorithm.**
- **This class is about analysis, not simply computation.**
- **Hence we loop for $i = 1, \ldots, n$ and find the values of i such that y_{i-1} and y_i have opposite sign. (We exclude the root $x_0 = 0$.)**
- **From the graph in Fig. 5, the function $J_{1.5}(x)$ crosses zero five times in the interval $0 < x < 20$.**
  1. **Hence there are five roots.**
  2. **The above procedure finds them all.**
  3. **The graph serves as a useful cross-check that the roots have been located correctly.**
- **The roots are given as follows, up to the tolerance specified in the question.**

$(x_{i-1} + x_i)/2$	y_{i-1}	y_i
4.4945	1.5×10^{-4}	-2.2×10^{-4}
7.7265	-7.2×10^{-5}	2.1×10^{-4}
10.9055	2.9×10^{-5}	-2.1×10^{-4}
14.0675	-4.1×10^{-5}	1.7×10^{-4}
17.2215	1.4×10^{-4}	-4.7×10^{-5}

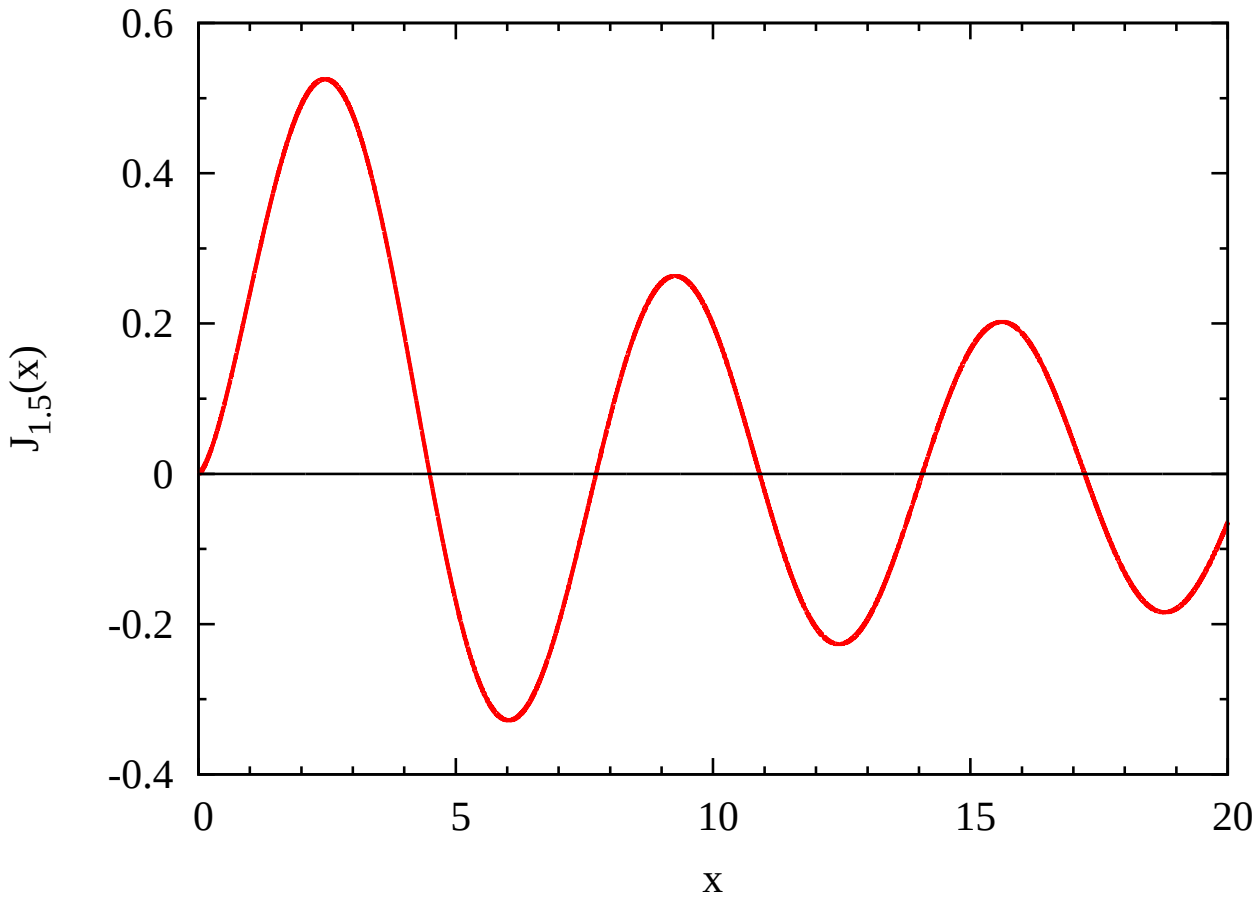

Figure 5: Graph of the Bessel function $J_{1.5}(x)$ in Question 7.

8 Question 8 (submit code (your function only))

- This is the question I promised near the beginning of the semester.
- This is adapted from a real research problem in my career.
- The device I had to design was called an *electrostatic quadrupole*.
- *Read the question carefully and don't panic.*
- A cross-section of the electrostatic quadrupole is shown in Fig. 6.
    1. There is an electric potential inside the quadrupole, call it V (for 'voltage').
    2. The electric potential is zero on the outer boundary (see Fig. 6).
    3. There are four electrodes, where the potential is set to fixed values.
    4. The electric potential is $V = 100$ on the left and right electrodes and $V = -100$ on the top and bottom electrodes (see Fig. 6).
    5. I solve the partial differential equation to calculate the value of the potential at the other points in the volume.
    6. *I perform the mathematical computation for you.*
- We measure the electric potential $V(\theta)$ on the dashed circle (see Fig. 6).
- We expand $V(\theta)$ in a Fourier series.
- Because of the four-fold symmetry, and also because the voltage on the electrodes form a $(+-+-)$ pattern, the only nonzero Fourier harmonics in $V(\theta)$ are

$$V(\theta) = a_2 \cos(2\theta) + a_6 \cos(6\theta) + a_{10} \cos(10\theta) + \cdots \tag{8.1}$$

- **The values of the Fourier coefficients a_2, a_6, a_{10}, etc., depend on the width w of the electrodes.**
- **Your task is to find the value of the width w such that $a_6 = 0$.**
- The procedure to do so is given below.
- Compile the supplied C++ code for the `class Node` and `class ElectrostaticQuadrupole`.
- Define an integer `digit = -4` if the first digit of your student id is odd and `digit = 4` if the first digit of your student id is even.
    1. For example if your student id is 12345678 or 34562345, then `digit = -4`.
    2. For example if your student id is 24681357 or 45623451, then `digit = 4`.
    3. Also set $n = 2^{12}$ to compute the values of $V(\theta)$ around the circle.

- Implement the code below. Use `digit` as an argument to the class constructor.

```
const int ngrid = 240;
const int n = 1 << 12;            // power of 2
std::vector<double> V(n, 0.0);

const int digit = -4;          // depends on student id
//const int digit = 4;        // depends on student id
ElectrostaticQuadrupole equad(ngrid, digit);
```

- The parameter n_{grid} sets the size of the grid inside the electrostatic quadrupole object, to solve a partial differential equation. In this question we hard code $n_{\text{grid}} = 240$. Do not change this value.

- **Now we iterate the value of w.**

  1. Choose a value for w.
  2. *The value of w must lie in the range $0 < w < 10$. The code in the class* `ElectrostaticQuadrupole` *contain validation checks to reject values of w outside this interval.*
  3. Run the code below.

```
double w = 4.0;         // choose some value for w
equad.SOR(w);
equad.circle(n, V);
```

  4. The class method "`SOR`" solves the partial differential equation.
  5. On my computer, one function call requires about 0.35 sec.
  6. The running time will vary depending on your computer.
  7. The class method "`circle`" computes the value of V at n points around the circle.
  8. The output value of V is $V_j = V(\theta_j)$, where $\theta = j = 2\pi j/n$, for $j = 0, \ldots, n-1$.

- So now you have an array (`std::vector`) V with entries $V(\theta_j)$.

- **Compute the coefficients a_j and b_j of the Fourier series for V.**

- *If you have done your work correctly, the only nonzero values will be $a_2, a_6, a_{10}, \ldots$. All the other values will be zero or very small.*

- Choose another value for w and calculate the values of the coefficients a_j and b_j again.

- **Iterate the value of w as follows:**

  1. Find two values w_1 and w_2 such that the value of a_6 has opposite signs at w_1 and w_2.
  2. Also $|w_1 - w_2| \leq 0.01$.

- **State the value of the average $w_{\text{avg}} = (w_1 + w_2)/2$ and fill the table below.**

- Fill the table below with your values. Note: the values are $a_6/\boldsymbol{a_2}$ and $a_{10}/\boldsymbol{a_2}$.

w		a_6/a_2	a_{10}/a_2
w_1	3 d.p.	4 d.p	4 d.p.
w_2	3 d.p.	4 d.p	4 d.p.
w_{avg}	3 d.p.	4 d.p	4 d.p.

- *If you do your work correctly, you will find the value of a_6 steps discontinuously across zero.*

- There is no value of w such that $a_6 = 0$. This is because the value of the electric potential is obtained via a numerical solution of a partial differential equation, and the number of points used is not enough for the values of a_2 and a_6, etc., to be continuous functions of w.

- Your overall code structure should look like this:

```
const int ngrid = 240;
const int n = 1 << 12;              // power of 2
std::vector<double> V(n, 0.0);

const int digit = -4;          // depends on student id
//const int digit = 4;         // depends on student id
ElectrostaticQuadrupole eq(ngrid, digit);

// *** begin iteration for value of w ***
w = something
equad.SOR(w);
equad.circle(n, V);

// *** compute Fourier coefficients of V
// *** find bracket such that value of a[6] has opposite sign at ends of bracket
// *** repeat iteration until tolerance is satisfied
// *** tabulate values of a[6]/a[2] and a[10]/a[2]
```

- **Some personal comments, for your information.**
- In the ideal design, we want only $a_2 \neq 0$ and all others zero.
- To achieve this, the shape of the requires electrodes must be hyperbolas.
- A hyperbolic shape is difficult to manufacture and moreover extends to $\pm\infty$,
- A real quadrupole electrode cannot extend to $\pm\infty$.
- Flat quadrupole electrodes are simple to manufacture.
- The design is robust and easy to install and maintain.
- By setting the width of the electrodes suitably, the leading error term (the unwanted a_6 Fourier harmonic) was eliminated.
- The Fourier harmonic a_{10} was also unwanted but not eliminated. However the magnitude of a_{10}/a_2 was small and negligible.
- The full expression for the potential in polar coordinates is

$$V(r,\theta) = a_2 r^2 \cos(2\theta) + a_6 r^6 \cos(6\theta) + a_{10} r^{10} \cos(10\theta) + \dots \tag{8.2}$$

- With elimination of the a_6 term, the value of the potential is

$$V(r,\theta) = a_2 r^2 \cos(2\theta) + a_{10} r^{10} \cos(10\theta) + \dots \tag{8.3}$$

- In the aboves question, the potential was calculated on the circle $r = 1$.
- For $r < 1$, the value of r^{10} is very small, hence the contribution from the a_{10} term is negligible.
- The above quadrupole design greatly simplified construction and maintenance and saved a lot of cost.
- A false color image of the electric potential is displayed in Fig. 7.

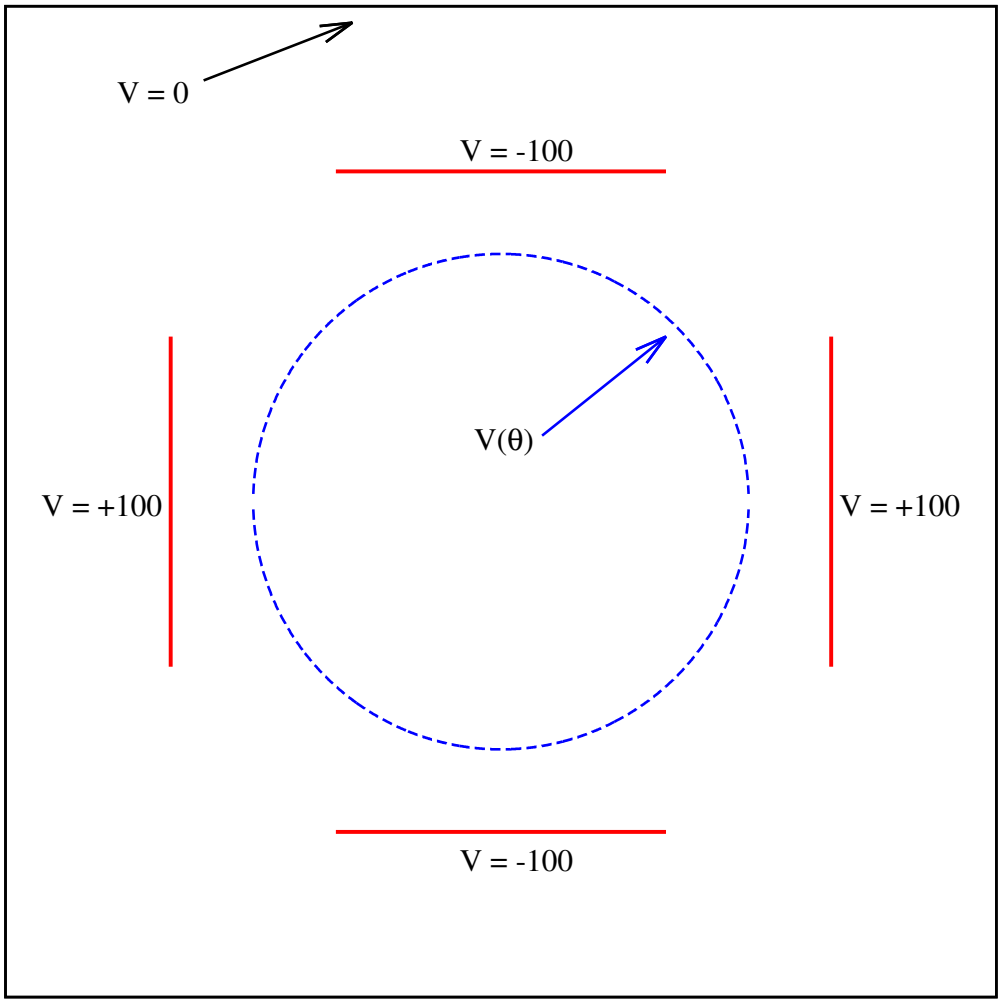

Figure 6: Cross-section of electrostatic quadrupole in Question 8.

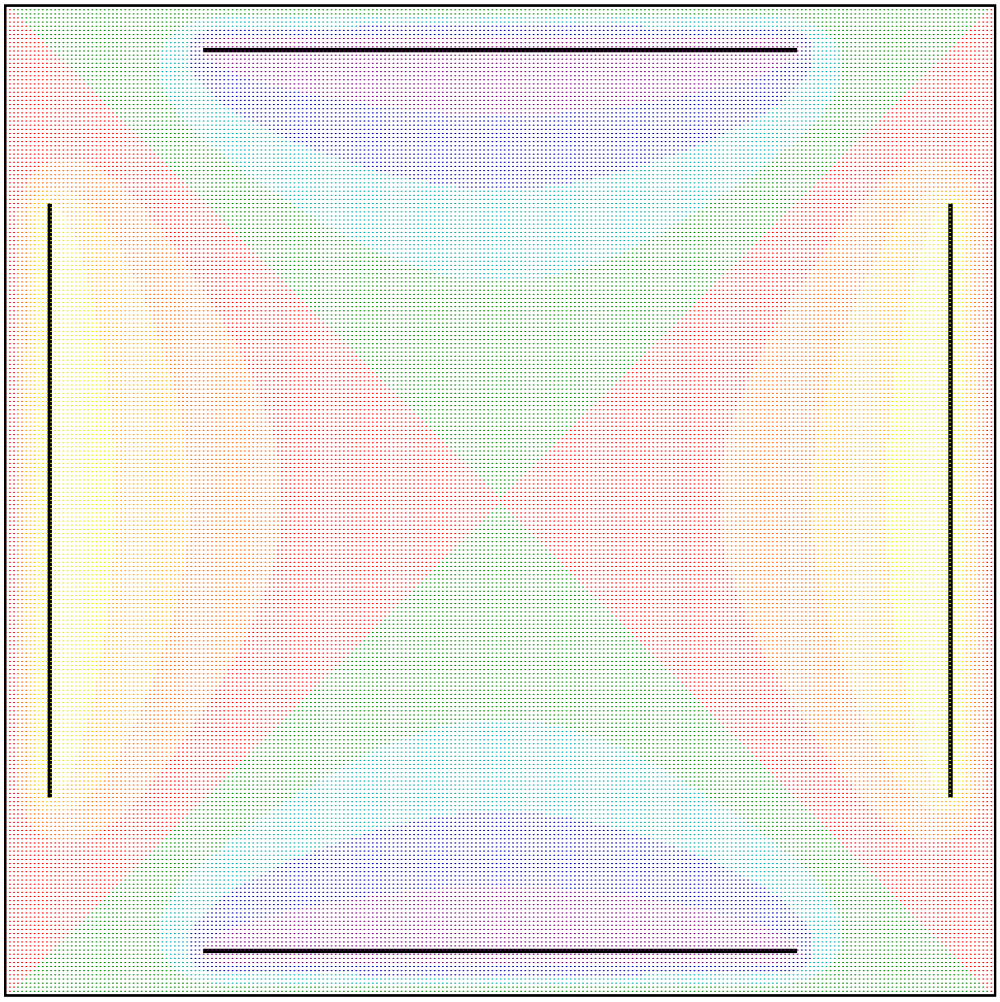

Figure 7: False color image of the electric potential for the electrostatic quadrupole in Question 8.

Background information (not for examination)

- The electric potential satisfies Laplace's equation, which in two dimensions is

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0\,. \tag{8.4}$$

- We express the partial derivatives as numerical differences on a rectangular grid with stepsizes (h, k) along x and y.

- Then we obtain, with indices (i, j) for x and y,

$$\frac{V_{i+1,j} + V_{i-1,j} - 2V_{ij}}{h^2} + \frac{V_{i,j+1} + V_{i,j-1} - 2V_{ij}}{k^2} = 0\,. \tag{8.5}$$

- It is simplest to employ a square grid. Hence we set $h = k$, which yields

$$\frac{V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} - 4V_{ij}}{h^2} = 0\,. \tag{8.6}$$

- This leads to the simple equation

$$V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} - 4V_{ij} = 0\,. \tag{8.7}$$

- We solve eq. (8.7) using the boundary conditions $V_{ij} = 0$ on the outer edges of the box and $V_{ij} = +100$ on the left and right electrodes and $V_{ij} = -100$ on the top and bottom electrodes. See Fig. 6.

- We solve eq. (8.7) as follows. Define a 'time' parameter t and define

$$\Delta V_{ij}^{(t)} = \left(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} - 4V_{ij}\right)^{(t)}\,. \tag{8.8}$$

- Then iterate for $t = 0, 1, \ldots$ via

$$V_{ij}^{(t+1)} = V_{ij}^{(t)} + \Delta V_{ij}^{(t)}\,. \tag{8.9}$$

- This technique is called **Gauss–Seidel iteration.**

- We iterate until $|\Delta V_{ij}^{(t)}| < \texttt{tol}$ at all points (i, j).

- This is a much more restrictive tolerance criterion than integration in one dimension.

- *Let's be greedy.* Let us try to 'jump ahead' to the final answer by introducing a parameter ω

$$V_{ij}^{(t+1)} = V_{ij}^{(t)} + \omega\,\Delta V_{ij}^{(t)}\,. \tag{8.10}$$

- We set $\omega > 1$ to try to 'take a bigger increment' to reach the solution more quickly.

- Actually, if $\Delta V_{ij}^{(t)}$ alternates in sign, it might be better to set $\omega < 1$.

- The technique in eq. (8.10) is called **successive overrelaxation (SOR).**
- If $0 < \omega < 1$ some authors called it *underrelaxation.*
- SOR is stable only if $0 < \omega < 2$.
- If there are totally n^2 points in the grid, each update of the grid requires n^2 computations.
- In addition, the Gauss-Seidel method requires $O(n^2)$ iterations in the time step t.
- SOR can reduce the number of iterations in t to $O(n)$.
- What is the optimal value of ω to employ?
    1. Nobody knows.
    2. There are results for special cases, but no general formula to determine the optimum.
    3. I set $\omega = 1.95$ in the code.
    4. I found that using $\omega = 1$ (Gauss-Seidel) required about 8 seconds for convergence.
    5. I found that using $\omega = 1.95$ (SOR) required about 0.35 seconds for convergence.
    6. I obtained the following approximate table of running times to obtain convergence.

ω	time
1.0	8.0
1.5	2.0
1.9	0.49
1.94	0.47
1.95	0.35
1.96	0.41
1.98	0.65

- You understand why the calculation function is called `SOR`.

```
#include <vector>
#include <cmath>
#include <complex>

class Node
{
public:
  Node() { x = 0; y = 0; V = 0; bdy = false; }
  double x;
  double y;
  double V;
  bool bdy;
};

class ElectrostaticQuadrupole
{
public:
  ElectrostaticQuadrupole(int n, int k) : _halfedge(5.0) {
    _n = n;
    _off = -k;
    allocate();
    init();
  }

  ~ElectrostaticQuadrupole() { clear(); }

private:
  void allocate() {
    _nodes = new Node*[_n+1];
    for (int i = 0; i <= _n; ++i) {
      _nodes[i] = new Node[_n+1];
    }
  }

  void clear() {
    for (int i = 0; i <= _n; ++i) {
      delete [] _nodes[i];
    }
    delete [] _nodes;
    _n = 0;
  }

  void init() {
    for (int i = 0; i <= _n; ++i) {
      bool bx = ((i==0) || (i==_n)) ? true : false;
```

```
    double x = -_halfedge + 2.0*_halfedge*double(i)/double(_n);
    Node * nx = _nodes[i];
    for (int j = 0; j <= _n; ++j) {
      bool by = ((j==0) || (j==_n)) ? true : false;
      double y = -_halfedge + 2.0*_halfedge*double(j)/double(_n);
      Node & nxy = nx[j];
      nxy.x = x;
      nxy.y = y;
      nxy.V = 0;
      nxy.bdy = (bx && by);
    }
  }
}

void set(double a) {
  for (int i = 0; i <= _n; ++i) {
    //double x = -_halfedge + 2.0*_halfedge*double(i)/double(_n);
    Node * nx = _nodes[i];
    Node & bottom = nx[0];
    Node & top = nx[_n];
    double &x = top.x;
    top.V = 0.0;
    bottom.V = 0.0;
    top.bdy = true;
    bottom.bdy = true;
  }

  for (int i = 0; i <= _n; ++i) {
    Node * nx = _nodes[i];
    int j = _n/20 + _off;
    Node & bottom = nx[j];
    Node & top = nx[_n-j];
    double &x = top.x;
    if (std::abs(x) <= 0.5*a) {
      top.V = -100.0;
      bottom.V = -100.0;
      top.bdy = true;
      bottom.bdy = true;
    }
  }

  for (int j = 0; j <= _n; ++j) {
    //double y = -_halfedge + 2.0*_halfedge*double(j)/double(_n);
    Node * left = _nodes[0];
    Node * right = _nodes[_n];
```

```
      Node & lj = left[j];
      Node & rj = right[j];
      double &y = lj.y;
      lj.V = 0.0;
      rj.V = 0.0;
      lj.bdy = true;
      rj.bdy = true;
    }

    for (int j = 0; j <= _n; ++j) {
      //double y = -_halfedge + 2.0*_halfedge*double(j)/double(_n);
      int i = _n/20 + _off;
      Node * left = _nodes[i];
      Node * right = _nodes[_n-i];
      Node & lj = left[j];
      Node & rj = right[j];
      double &y = lj.y;
      if (std::abs(y) <= 0.5*a) {
        lj.V = 100.0;
        rj.V = 100.0;
        lj.bdy = true;
        rj.bdy = true;
      }
    }

    for (int i = 1; i < _n; ++i) {
      Node * nx = _nodes[i];
      for (int j = 1; j < _n; ++j) {
        Node & nxy = nx[j];
        if (nxy.bdy == false) {
  nxy.V = 0;
        }
      }
    }
  }

public:
  int SOR(double a) {
    if ((a <= 0.0) || (a >= 2.0*_halfedge)) {
      return 1; // fail
    }
    set(a);
    const double tol = 1.0e-8;
    const double omega = 1.95;
    const int max_iter = 10000;
```

```
for (int iter = 0; iter < max_iter; ++iter) {
  double norm = 0;
  // chessboard
  // sweep i+j = even
  for (int i = 1; i < _n; i++) {
    Node * nmid = _nodes[i];
    Node * nleft = _nodes[i-1];
    Node * nright = _nodes[i+1];
    for (int j = 1; j < _n; j++) {
      if ((i+j)%2 == 1) continue;
      Node & n0 = nmid[j];
      if (n0.bdy == true) continue;

      Node & n1 = nright[j];
      Node & n2 = nmid[j+1];
      Node & n3 = nleft[j];
      Node & n4 = nmid[j-1];
      double delta_V = 0.25*(n1.V +n2.V +n3.V +n4.V) - n0.V;
      if (norm < delta_V*delta_V) {
        norm = delta_V*delta_V;
      }
      n0.V += omega*delta_V;
    }
  }
  // sweep i+j = odd
  for (int i = 1; i < _n; i++) {
    Node * nmid = _nodes[i];
    Node * nleft = _nodes[i-1];
    Node * nright = _nodes[i+1];
    for (int j = 1; j < _n; j++) {
      if ((i+j)%2 == 0) continue;
      Node & n0 = nmid[j];
      if (n0.bdy == true) continue;

      Node & n1 = nright[j];
      Node & n2 = nmid[j+1];
      Node & n3 = nleft[j];
      Node & n4 = nmid[j-1];
      double delta_V = 0.25*(n1.V +n2.V +n3.V +n4.V) - n0.V;
      if (norm < delta_V*delta_V) {
        norm = delta_V*delta_V;
      }
      n0.V += omega*delta_V;
    }
  }
```

```
    if (norm <= tol) break;
  }
  return 0; // success
}

void circle(int nc, std::vector<double> &V) const {
  const double pi = 4.0*atan2(1.0,1.0);
  const double r = _halfedge * 0.8;
  V.resize(nc, 0.0);

  for (int ic = 0; ic < nc; ++ic) {
    double theta = 2.0*pi*double(ic)/double(nc);
    double x0 = r*cos(theta);
    double y0 = r*sin(theta);
    int i0 = _n * (x0 / _halfedge + 1.0)/2.0;
    int j0 = _n * (y0 / _halfedge + 1.0)/2.0;
    bool interp = false;
    for (int i = i0-2; i <= i0+2; ++i) {
      double x1 = -_halfedge + 2.0*_halfedge*double(i)/double(_n);
      double x2 = x1 + 2.0*_halfedge/double(_n);
      if ((x1 > x0) || (x2 < x0)) continue;
      double lambda = (x0 - x1)/(x2 - x1);
      Node * nx1 = _nodes[i];
      Node * nx2 = _nodes[i+1];
      for (int j = j0-2; j <=j0+2; ++j) {
        double y1 = -_halfedge + 2.0*_halfedge*double(j)/double(_n);
        double y2 = y1 + 2.0*_halfedge/double(_n);
        if ((y1 > y0) || (y2 < y0)) continue;
        double mu = (y0 - y1)/(y2 - y1);
        Node & nx1y1 = nx1[j];
        Node & nx1y2 = nx1[j+1];
        Node & nx2y1 = nx2[j];
        Node & nx2y2 = nx2[j+1];
        double V_interp = (1.0-lambda)*(1.0-mu)*nx1y1.V
                          +lambda*(1.0-mu)*nx2y1.V
                          +lambda*mu*nx2y2.V
                          +(1.0-lambda)*mu*nx1y2.V;
        V[ic] = V_interp;
        interp = true;
        break;
      }
      if (interp == true) break;
    }
  }
}
```

```
private:
  // data
  int _n;
  int _off;
  Node **_nodes;
  const double _halfedge;
};
```

A student reported a bug in the C++ code for the electrostatic quadropole class.
I have confirmed the bug.
In practice, few students attempted this question and almost all solutions were correct.
The weight of this question was downgraded, effectively a bonus question.

- **This question requires a lot of reading but there is actually not much computation for you to do.**
- **Numerical routines are supplied to solve for the electric potential.**
- **Given $V(\theta)$, one can obtain its Fourier harmonics, in particular**

$$\begin{aligned} a_j &= \frac{1}{\pi} \int_0^{2\pi} V(\theta) \cos(j\theta)\, d\theta \\ &\simeq \frac{2}{n} \sum_{i=0}^{n-1} V(\theta_i)\, \cos(j\theta_i) \qquad\qquad (\theta_i = 2\pi i/n)\,. \end{aligned}$$

- **The computation of a_j is similar to that used to answer other questions.**
- **As stated in the question, the numerical accuracy of the calculation of the electric potential (based on the number of points in the grid) is such that the value of a_2 steps discontinously across zero. The best that one can do is to locate that discontinuity, i.e. find two closely spaced values of w such that a_2 has opposute signs at the two values.**

- **For students whose student id begins with an odd digit, here is a table of values of w, a_2, a_6/a_2 and a_{10}/a_2.**

w	a_2	a_6/a_2	a_{10}/a_2
4.30	44.0023	0.00429306	−0.0273792
4.31	44.0023	0.00429306	−0.0273792
4.32	44.0023	0.00429306	−0.0273792
4.33	44.0023	**0.00429306**	−0.0273792
4.34	44.2017	**−0.00120218**	−0.025722
4.35	44.2017	−0.00120218	−0.025722
4.36	44.2017	−0.00120218	−0.025722
4.37	44.2017	−0.00120218	−0.025722
4.38	44.2017	−0.00120218	−0.025722
4.39	44.2017	−0.00120218	−0.025722
4.40	44.2017	−0.00120218	−0.025722

- **For students whose student id begins with an odd digit, the answer is $w_{\text{avg}} = 4.335$.**

w		a_6/a_2	a_{10}/a_2
w_1	4.33	0.0043	−0.0274
w_2	4.34	−0.0012	−0.0257
w_{avg}	4.335	−0.0012	−0.0257

- **For students whose student id begins with an even digit, here is a table of values of w, a_2, a_6/a_2 and a_{10}/a_2.**

w	a_2	a_6/a_2	a_{10}/a_2
4.90	38.0864	0.000324352	−0.0152193
4.91	38.0864	**0.000324352**	−0.0152193
4.92	38.2442	**−0.00353241**	−0.0142112
4.93	38.2442	−0.00353241	−0.0142112
4.94	38.2442	−0.00353241	−0.0142112
4.95	38.2442	−0.00353241	−0.0142112
4.96	38.2442	−0.00353241	−0.0142112
4.97	38.2442	−0.00353241	−0.0142112
4.98	38.2442	−0.00353241	−0.0142112
4.99	38.2442	−0.00353241	−0.0142112
5.00	38.2442	−0.00353241	−0.0142112

- **For students whose student id begins with an even digit, the answer is $w_{\text{avg}} = 4.915$.**

w		a_6/a_2	a_{10}/a_2
w_1	4.91	0.0003	−0.0152
w_2	4.92	−0.0035	−0.0142
w_{avg}	4.915	0.0003	−0.0152

Statistics

- **There are 33 registered students, and the breakdown of grades is as follows.**

A+	**7**
A	**8**
A-	**5**
B+	**2**
B	**5**
B-	**5**
C	**1**

- **Two students obtained $A+$ for every midterm and the final.**
- **A histogram of the grades is plotted in Fig. 8.**
- **There is no "curve" in the grading.**
- **The grade distribution forms two peaks ($A-, A, A+$) and ($C, B-, B, B+$).**
- **Twenty students in the first set and thirteen students in the second set.**
- **It is a difficult class, but many students did very well.**

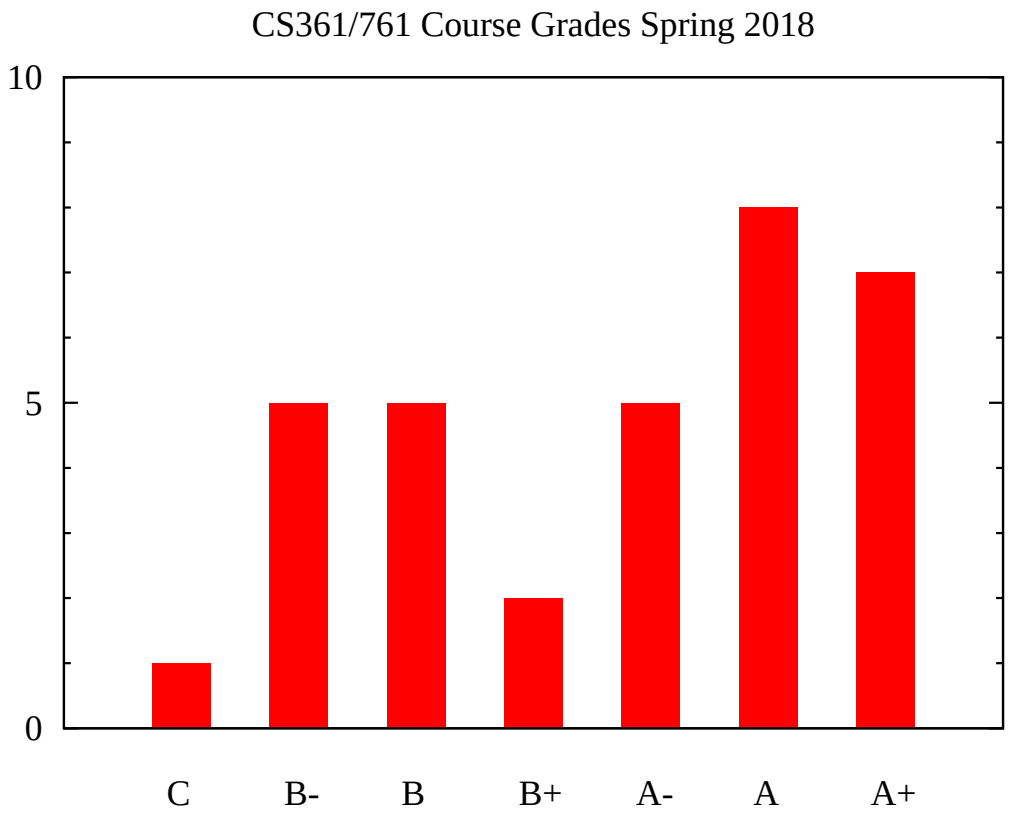

Figure 8: Histogram of course grades for CS361/761 Spring 2018.