AM205: Assignment 1 solutions[*]

Problem 1 – Interpolating polynomials for the gamma function

Part (a)

We consider finding a polynomial $g(x) = \sum_{k=0}^{5} p_k x^k$ that fits the data points $(j, \Gamma(j))$ for $j = 1, 2, 3, 4, 5, 6$. Since there are a small number of data points, we can use the Vandermonde matrix to find the coefficients of the interpolating polynomial $g(x) = \sum_{k=0}^{5} g_k x^k$. The linear system is

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 \\ 1 & 3 & 9 & 27 & 81 & 243 \\ 1 & 4 & 16 & 64 & 256 & 1024 \\ 1 & 5 & 25 & 125 & 625 & 3125 \\ 1 & 6 & 36 & 216 & 1296 & 7776 \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 6 \\ 24 \\ 120 \end{pmatrix}. \tag{1}$$

The program `gamma_interp.py` solves this system, and shows that the coefficients are

$$(g_0, g_1, g_2, g_3, g_4, g_5) = (-35.00, 83.89, -70.88, 27.75, -5.13, 0.37). \tag{2}$$

Alternatively, in exact fractions, the solution is

$$(g_0, g_1, g_2, g_3, g_4, g_5) = \left(-35, \frac{5033}{60}, -\frac{567}{8}, \frac{111}{4}, -\frac{41}{8}, \frac{11}{30}\right). \tag{3}$$

Part (b)

We now consider finding a polynomial $p(x) = \sum_{k=0}^{4} p_k x^k$ that fits the transformed data points $(j, \log(\Gamma(j)))$ for $j = 1, 2, 3, 4, 5$. The coefficients are given by

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 \\ 1 & 3 & 9 & 27 & 81 & 243 \\ 1 & 4 & 16 & 64 & 256 & 1024 \\ 1 & 5 & 25 & 125 & 625 & 3125 \\ 1 & 6 & 36 & 216 & 1296 & 7776 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix} = \begin{pmatrix} \log 1 \\ \log 1 \\ \log 2 \\ \log 6 \\ \log 24 \\ \log 120 \end{pmatrix}. \tag{4}$$

The program `gamma_interp.py` shows that the coefficients are

$$(p_0, p_1, p_2, p_3, p_4, p_5) = (1.267, -2.187, 1.101, -0.2014, 0.02166, -0.0009721). \tag{5}$$

[*]Solutions were written Kevin Chen (TF, Fall 2014), Dustin Tran (TF, Fall 2014), and Chris H. Rycroft. Edited by Chris H. Rycroft.

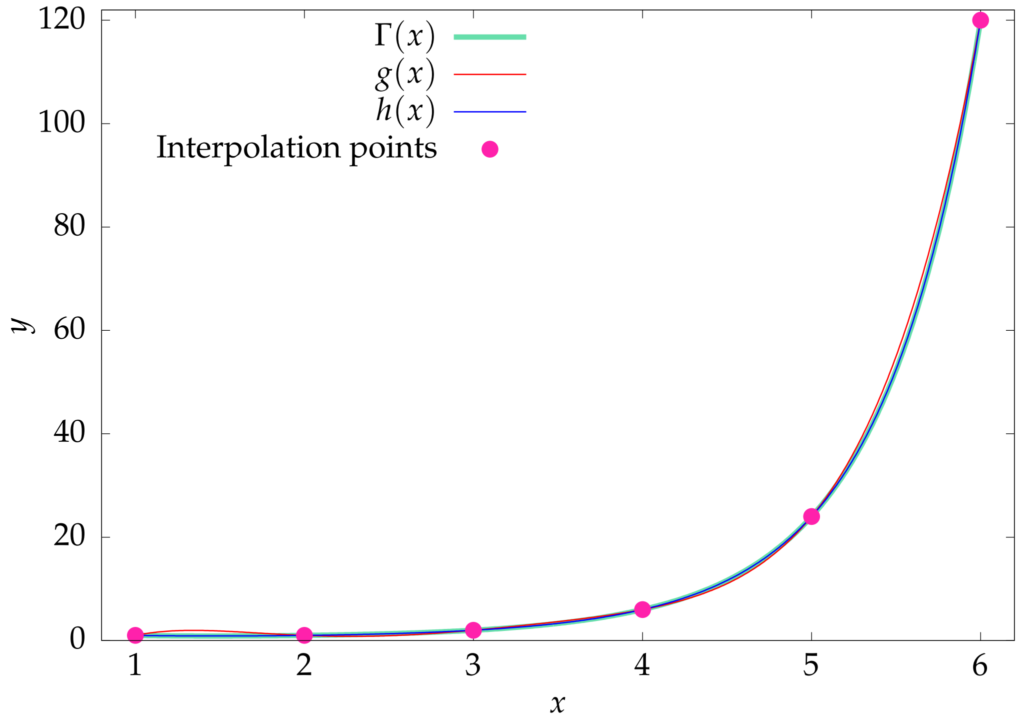

Figure 1: The gamma function $\Gamma(x)$ and the two interpolating polynomials $g(x)$ and $h(x)$ considered in problem 1.

Parts (c) and (d)

The program `gamma_interp.py` also outputs the gamma function and the two interpolating polynomials at 501 samples in the range $1 \leq x \leq 6$. The three functions $\Gamma(x)$, $g(x)$, and $h(x)$ are shown in Fig. 1. The function $h(x)$ is near-indistinguishable from the gamma function, whereas the function $g(x)$ is noticeably different, especially between $x = 1$ and $x = 2$. Since the gamma function grows rapidly, particularly near $x = 6$, its higher derivatives will grow very large, meaning that Cauchy's interpolation bound will be large. By taking the logarithm of the function values, the interpolation in part (b) does not feature such a large increase near $x = 6$. Hence, it should be expected that the interpolation will be more accurate.

The program `gamma_interp.py` also computes the maximum relative error by sampling the functions at 5001 equally spaced points over $1 \leq x \leq 6$. It reports that the maximum relative error of $g(x)$ is 1.21 and the maximum relative error of $h(x)$ is 0.00802, which is consistent with the curves in Fig. 1.

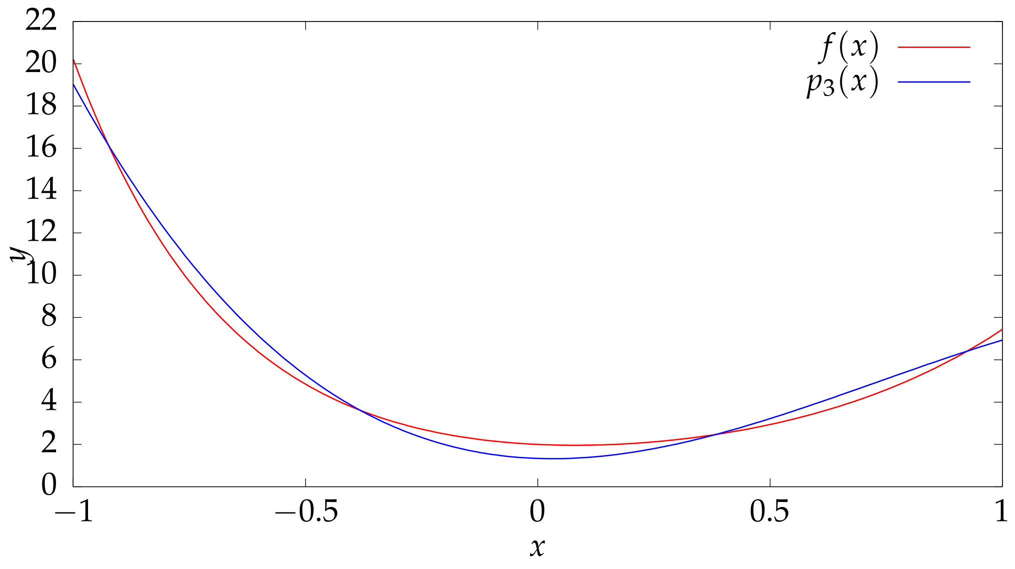

Figure 2: The function $f(x) = e^{-3x} + e^{2x}$ and the interpolating polynomial $p_3(x)$ considered in problem 2.

Problem 2 – Error bounds with Lagrange Polynomials

Parts (a) and (b)

Figure 2 shows the Lagrange polynomial $p_3(x)$ over the true function $f(x)$ using a slightly modified version of the in-class code example. Running the code, the infinity norm of the error is approximately 1.19249.

Part (c)

The difference between $f(x)$ and and the interpolating polynomial $p_{n-1}(x)$ can be expressed as

$$f(x) - p_{n-1}(x) = \frac{f^{(n)}(\theta)}{n!} \prod_{i=1}^{n} (x - x_i), \tag{6}$$

where θ is a specific value within the interval from -1 to 1. To obtain a bound on $\|f - p_{n-1}\|_\infty$, we consider the magnitude of the terms on the right hand side.

Since the x_i are at the roots of the nth Chebyshev polynomial $T_n(x)$, it follows that the product in Eq. 6 is a scalar multiple of this polynomial, *i.e.*,

$$\prod_{i=1}^{n} (x - x_i) = \lambda T_n(x) \tag{7}$$

where λ is some scaling constant. The coefficient in front of x^n on the left hand side is 1. Using properties of Chebyshev polynomials, the coefficient of x^n in $T_n(x)$ is 2^{n-1}. Hence $\lambda = 2^{-(n-1)}$. The Chebyshev polynomials satisfy $|T_n(x)| \leq 1$ for $x \in [-1, 1]$ and hence

$$\left| \prod_{i=1}^{n} (x - x_i) \right| \leq \frac{1}{2^{n-1}} \tag{8}$$

for $x \in [-1, 1]$.

Now consider the nth derivative of f, which is given by

$$f^{(n)}(\theta) = (-3)^n e^{-3\theta} + 2^n e^{2\theta}. \tag{9}$$

The maximum value of $|f^{(n)}(\theta)|$ can occur at two places: (i) at an internal maximum, or (ii) at one of the end points of the interval, $\theta = \pm 1$. Consider case (i) first. If n is odd, then

$$f^{(n+1)}(\theta) = 3^{n+1} e^{-3\theta} + 2^{n+1} e^{2\theta}, \tag{10}$$

and since both terms are positive, there is no value of θ where $f^{(n+1)}(\theta) = 0$. If n is even, then

$$f^{(n+1)}(\theta) = -3^{n+1} e^{-3\theta} + 2^{n+1} e^{2\theta} \tag{11}$$

Setting $f^{(n+1)} = 0$ gives

$$3^{n+1} e^{-3\theta} = 2^{n+1} e^{2\theta} \tag{12}$$

and hence $({}^3/_2)^{n+1} = e^{5\theta}$, so

$$\theta = \frac{(n+1) \log {}^3/_2}{5} \tag{13}$$

is a single solution. However, since

$$f^{(n+2)}(\theta) = |3^{n+2} e^{-3\theta} + 2^{n+2} e^{2\theta}| > 0 \tag{14}$$

it follows that this must be a minimum of $f^{(n)}$. Since $f^{(n)} > 0$, it must be a minimum of $|f^{(n)}|$ also. Hence, for all values of n there is no possibility that the maximum of $|f^{(n)}|$ occurs in the interior of the interval. Thus the only remaining possibilities are at the endpoints. Since the factor of $(-3)^n$ grows more rapidly in magnitude, the maximum will occur at $\theta = -1$, and hence

$$|f^{(n)}(\theta)| \leq |(-3)^n e^3 + 2^n e^{-2}|. \tag{15}$$

Combining the results from Eqs. 8 & 15 establishes that

$$\|f - p_{n-1}\|_\infty \leq \frac{|(-3)^n e^3 + 2^n e^{-2}|}{n!\, 2^{n-1}}. \tag{16}$$

Part (d)

There are many ways to find better control points, and this problem illustrates that while the Chebyshev points are a good set of points at which to interpolate a general unknown function, they are usually not optimal for a specific function.

One simple method is to examine where the maximum interpolation error is achieved. This is happens near $x = -1$. Hence if we move the first control point to the left, it will result in a better approximation of $f(x)$ within this region. In this case, we shift the first control point by -0.02, which leads to an infinity norm of 1.09283.

Problem 3 – The condition number

Part (a)

Throughout this equation, $\|\cdot\|$ is taken to mean the Euclidean norm. The first two parts of this problem can be solved using diagonal matrices only. Consider first

$$B = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \tag{17}$$

Then $\|B\| = 2$, $\|B^{-1}\| = 1$ and hence $\kappa(B) = 2$. Similarly, $\kappa(C) = 2$. Adding the two matrices together gives

$$B + C = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} = 3I \tag{18}$$

and hence $\kappa(B+C) = \|3I\|\,\|\frac{1}{3}I\| = 3 \times \frac{1}{3} = 1$. For these choices of matrices, $\kappa(B+C) < \kappa(B) + \kappa(C)$.

Part (b)

If

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{19}$$

then $\kappa(B) = 2$ and $\kappa(C) = 1$. Adding the two matrices together gives

$$B + C = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix} \tag{20}$$

and hence $\kappa(B+C) = 5$. Therefore $\kappa(B+C) > \kappa(B) + \kappa(C)$.

Part (c)

Let A be an invertible 2×2 symmetric matrix. First, note that

$$\|2A\| = \max_{v \neq 0} \frac{\|2Av\|}{\|v\|} = \max_{v \neq 0} \frac{2\|Av\|}{\|v\|} = 2 \max_{v \neq 0} \frac{\|Av\|}{\|v\|} = 2\|A\|. \tag{21}$$

Similarly, note that $\|(2A)^{-1}\| = \|\frac{1}{2}A^{-1}\| = \frac{1}{2}\|A^{-1}\|$. Hence

$$\kappa(2A) = \|2A\| \, \|(2A)^{-1}\| = 2\|A\| \times \frac{1}{2}\|A^{-1}\| = \|A\| \, \|A^{-1}\| = \kappa(A). \tag{22}$$

Now suppose that A is a symmetric invertible matrix. Then there exists an orthogonal matrix Q and a diagonal matrix D such that

$$A = Q^{\mathsf{T}} D Q. \tag{23}$$

Since $Q^{\mathsf{T}}Q = QQ^{\mathsf{T}} = I$, it follows that

$$A^2 = Q^{\mathsf{T}} D Q Q^{\mathsf{T}} D Q = Q^{\mathsf{T}} D^2 Q. \tag{24}$$

The matrix norm of $\|A\|$ is

$$\|A\| = \max_{v \neq 0} \frac{\|Q^{\mathsf{T}} D Q v\|}{\|v\|}. \tag{25}$$

Since Q corresponds to a rotation or reflection, it preserves distances under the Euclidean norm and hence $\|Qw\| = \|w\| = \|Q^{\mathsf{T}}w\|$ for an arbitrary vector w. Therefore

$$\|A\| = \max_{v \neq 0} \frac{\|DQv\|}{\|Qv\|} = \max_{u \neq 0} \frac{\|Du\|}{\|u\|} = \|D\| \tag{26}$$

where $u = Qv$. Similarly $\|A^{-1}\| = \|Q^{\mathsf{T}} D^{-1} Q\|$, and since D^{-1} is also diagonal it follows that $\|A^{-1}\| = \|D^{-1}\|$, so $\kappa(A) = \kappa(D)$. With reference to the condition number notes, $\kappa(A) = |\alpha\beta^{-1}|$ where α is the diagonal entry with largest magnitude and $|\beta|$ is the diagonal entry with the smallest entry with smallest magnitude.

Since D^2 is also diagonal, it follows that $\|A^2\| = \|D^2\|$. The diagonal entry of D^2 with the largest amplitude will be α^2, and the diagonal entry of D^2 with the smallest amplitude will be β^{-2}. Hence

$$\kappa(A^2) = |\alpha^2\beta^{-2}| = (\kappa(A))^2. \tag{27}$$

Part (d)

The result for that $\kappa(2A) = \kappa(A)$ is true for arbitrary 2×2 invertible matrices. The derivation that was considered in part (c) did not rely on the matrix being symmetric.

The result about $\kappa(A^2)$ does not generalize to arbitrary matrices. If

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{28}$$

then

$$A^2 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}. \tag{29}$$

One can numerically verify that $\kappa(A^2) = 5.828$ but $(\kappa(A))^2 = 6.854$, so the two do not agree.

Problem 4 – Periodic cubic splines

Parts (a), (b), and (c)

We aim to construct a cubic spline $s_x(t)$ of $\sin \frac{\pi t}{2}$ over the periodic interval $t \in [0,4)$, using four cubic representations over the ranges $[0,1)$, $[1,2)$, $[2,3)$, and $[3,4)$. This gives sixteen degrees of freedom. Each cubic must match the function value at either end of its range (giving eight constraints). The first and second derivatives must also be continuous at the interface between each cubic, giving another eight constraints. Hence, the number of free parameters and the number of constraints agree, so there should be a unique solution.

A possible pitfall is to try and find the solution using a library routine, such as SciPy's `interp1d`. However, the boundary conditions considered here are distinctly different from the defaults usually used by these routines. A standard (non-periodic) spline uses the conditions $s_x''(0) = 0$ and $s_x''(4) = 0$. On the other hand, to make the spline periodic, this problem uses the conditions $s_x''(0) = s_x''(4)$ and $s_x'(0) = s_x'(4)$. The different conditions fundamentally alter the solution.

To construct the spline, we make use of basis of cubics

$$\begin{aligned} c_0(t) &= t^2(3-2t), & c_1(t) &= -t^2(1-t), \\ c_2(t) &= (t-1)^2 t, & c_3(t) &= 2t^3 - 3t^2 + 1, \end{aligned} \tag{30}$$

which are chosen because of their favorable properties at $t = 0$ and $t = 1$ that are summarized in Table 1. Using these functions, the spline can be written as

$$s_x(t) = \begin{cases} c_0(t) + \alpha c_1(t) + \delta c_2(t) & \text{for } t \in [0,1), \\ c_3(t-1) + \beta c_1(t-1) + \alpha c_2(t-1) & \text{for } t \in [1,2), \\ -c_0(t-2) + \gamma c_1(t-2) + \beta c_2(t-2) & \text{for } t \in [2,3), \\ -c_3(t-3) + \delta c_1(t-3) + \gamma c_2(t-3) & \text{for } t \in [3,4), \end{cases} \tag{31}$$

where α, β, γ, and δ are unknown constants. The contributions of c_0 and c_3 are chosen to ensure each cubic matches $\sin \frac{\pi t}{2}$ at the end points. The pairwise occurences of each constant are chosen to ensure that the first derivative is continuous.

Function	$c_i(1)$	$c_i'(1)$	$c_i'(0)$	$c_i(0)$	$c_i''(1)$	$c_i''(0)$
$c_0(t) = t^2(3-2t)$	1	0	0	0	-6	6
$c_1(t) = -t^2(1-t)$	0	1	0	0	4	-2
$c_2(t) = (t-1)^2 t$	0	0	1	0	2	-4
$c_3(t) = 2t^3 - 3t^2 + 1$	0	0	0	1	6	-6

Table 1: Properties of the four cubics used as a basis to compute the cubic spline.

To set the free parameters, the second derivatives must be made continuous at each interface. At $t = 1, 2, 3, 4$, by reference to Tab. 1, this gives

$$-6 + 4\alpha + 2\delta = -6 - 2\beta - 4\alpha, \tag{32}$$

$$6 + 4\beta + 2\alpha = -6 - 2\gamma - 4\beta, \tag{33}$$

$$6 + 4\gamma + 2\beta = 6 - 2\delta - 4\gamma, \tag{34}$$

$$-6 + 4\delta + 2\gamma = 6 - 2\alpha - 4\delta, \tag{35}$$

respectively. In matrix form, this is

$$\begin{pmatrix} 8 & 2 & 0 & 2 \\ 2 & 8 & 2 & 0 \\ 0 & 2 & 8 & 2 \\ 2 & 0 & 2 & 8 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} 0 \\ -12 \\ 0 \\ 12 \end{pmatrix}, \tag{36}$$

which has the unique solution $(\alpha, \beta, \gamma, \delta) = (0, -\frac{3}{2}, 0, \frac{3}{2})$. Hence

$$s_x(t) = \begin{cases} c_0(t) + \frac{3}{2}c_2(t) & \text{for } t \in [0,1), \\ c_3(t-1) - \frac{3}{2}c_1(t-1) & \text{for } t \in [1,2), \\ -c_0(t-2) - \frac{3}{2}c_2(t-2) & \text{for } t \in [2,3), \\ -c_3(t-3) + \frac{3}{2}c_1(t-3) & \text{for } t \in [3,4). \end{cases} \tag{37}$$

Figure 3(a) shows a plot of $s_x(t)$ in comparison to $\sin \frac{\pi t}{2}$. There is a high level of agreement between the two functions.

Since $\cos \frac{\pi t}{2}$ is the same as $\sin \frac{\pi t}{2}$, but shifted by -1, it follows that it can be approximated by the spline

$$s_y(t) = \begin{cases} c_3(t) - \frac{3}{2}c_1(t) & \text{for } t \in [0,1), \\ -c_0(t-1) - \frac{3}{2}c_2(t-1) & \text{for } t \in [1,2), \\ -c_3(t-2) + \frac{3}{2}c_1(t-2) & \text{for } t \in [2,3), \\ c_0(t-3) + \frac{3}{2}c_2(t-3) & \text{for } t \in [3,4). \end{cases} \tag{38}$$

Figure 3(b) shows a plot of $s_y(t)$ in comparison to $\cos \frac{\pi t}{2}$.

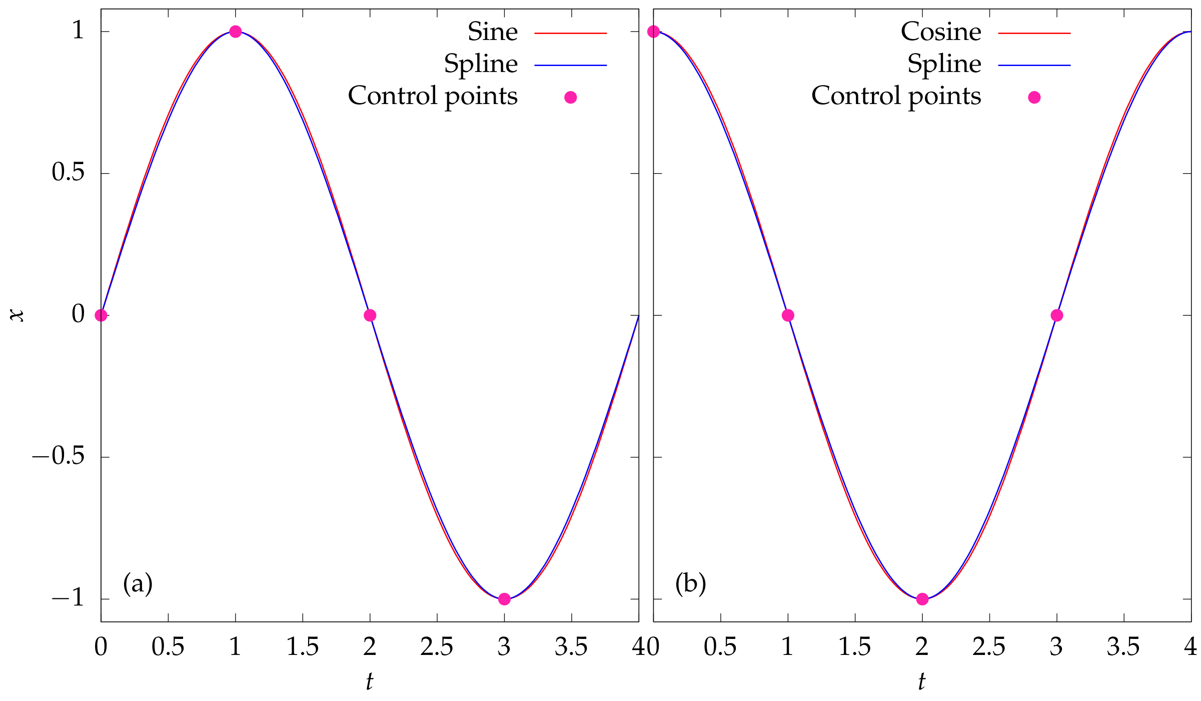

Figure 3: Comparison of (a) $\sin \frac{t\pi}{2}$ and the spline approximation $s_x(t)$, and (b) $\cos \frac{t\pi}{2}$ and the spline approximation $s_y(t)$.

Part (d)

Figure 4(a) shows a plot of the parametric curve $\mathbf{s}(t) = (s_x(t), s_y(t))$ compared to a true circle. The `am205_sol1_ex3.py` code integrates the area enclosed by the this curve by dividing it into a large number of triangles. Recall that if a $\mathbf{a}$ and $\mathbf{b}$ are vectors along two sides of a triangle, then the area is given by $\frac{1}{2}|\mathbf{a} \times \mathbf{b}|$. For the given problem, consider the triangle with vertices at the origin, $\mathbf{s}(t)$, and $\mathbf{s}(t + \Delta t)$. Then for Δt small, vectors along two sides are given by $\mathbf{s}(t)$ and $\mathbf{s}'(t)\Delta t$. Hence the total enclosed area is the sum of triangles such as this, and is hence given by the integral

$$A = \frac{1}{2} \int_0^4 |\mathbf{s}(t) \times \mathbf{s}'(t)| \, dt. \tag{39}$$

The program shows that $A = 3.05000$ to five decimal places using Python's integration routine. The integrand is a fifth order polynomial, and hence it can also be evaluated exactly using alternative means to determine that $A = \frac{61}{20}$. This differs from π by 3%.

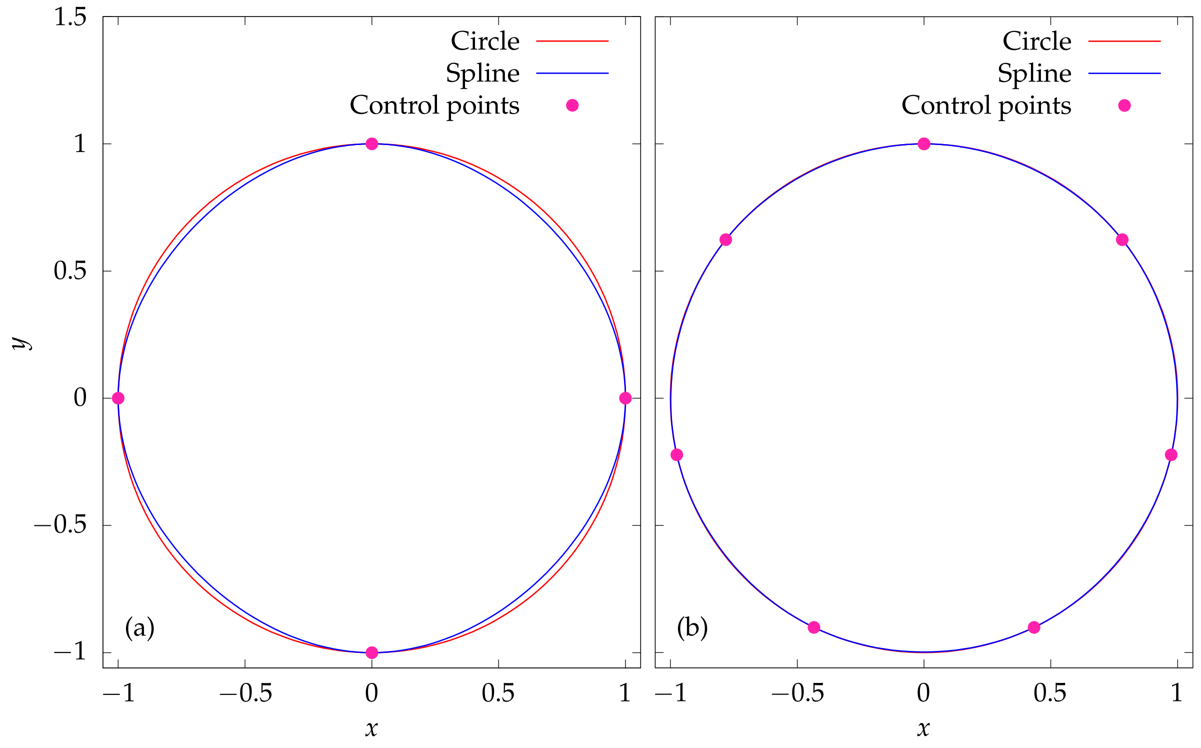

Figure 4: The parametric curve $\mathbf{s}(t) = (s_x(t), s_y(t))$ compared to a circle, for (a) four control points and (b) seven control points.

Part (e)

Consider calculating a spline $s^n_x(t)$ of $\sin \frac{\pi t}{2}$ on the periodic interval $[0, 4)$ using n control points at $t = t_0, t_1, \ldots, t_{n-1}$ where $t_k = \frac{4k}{n}$. By extending the argument from part (a), one can determine that in the interval $[t_k, t_{k+1})$, the spline is given by

$$s^n_x(t) = \left(\sin \frac{2k\pi}{n}\right) c_3(t_*) + \left(\sin \frac{2(k+1)\pi}{n}\right) c_0(t_*) + \alpha_{k+1} c_1(t_*) + \alpha_k c_2(t_*) \tag{40}$$

where $t_* = \frac{n}{4}(t - t_k)$ and the α_k are constants for $k = 0, \ldots, n-1$. Due to the periodicity, α_{n+k} is treated as equivalent to α_k. As in part (a), this form will satisfy that $s^n_x(t)$ match the function values at the end of each interval, and will have continuous first derivative. To set the α_k, the second derivatives must be considered. Ensuring continuity of the second derivative at t_k yields

$$-6 \sin \frac{2k\pi}{n} + 6 \sin \frac{2(k+1)\pi}{n} - 2\alpha_{k+1} - 4\alpha_k = 6 \sin \frac{2(k-1)\pi}{n} - 6 \sin \frac{2k\pi}{n} + 4\alpha_k + 2\alpha_{k-1}. \tag{41}$$

and hence

$$2\alpha_{k-1} + 8\alpha_k + 2\alpha_{k+1} = -6 \sin \frac{(k-1)\pi}{2n} + 6 \sin \frac{(k+1)\pi}{2n}. \tag{42}$$

By considering Eq. 42 for $k = 0, 1, \ldots, n-1$, a linear system is obtained for the α_k, which is the generalization of Eq. 36. The matrix is a circulant matrix, where the only entries are $(2, 8, 2)$ on diagonal lines that wrap around. Linear systems involving such matrices can be solved very rapidly in principle, by employing the fast Fourier transform. Once the α_k are determined, Eq. 40 gives the explicit form of the spline.

In a similar manner, a spline $s_y^n(t)$ of $\cos \frac{\pi t}{2}$ can be constructed. In part (c), this spline was constructed by noting that this function is the same as sine, but shifted by -1, which permitted the form of $s_y^n(t)$ to be written down immediately. However, here the same approach will not always work, since for a general n the positions of the control points may not precisely match when shifted by -1. It is therefore necessary to explicitly construct this spline, by using Eqs. 40 and 42 again, and replacing all instances of sine by cosine.

The program `am205_sol1_ex3e.py` constructs the two splines for an arbitrary number of control points n, when $n \geq 3$. Figure 4(b) shows a plot of the resultant circle approximation using seven points, which becomes near-indistinguishable from a true circle. The program also considers a range of different n and computes the approximation to π. The area integration is performed using three-point Gaussian quadrature[†] on each subinterval, which results in an exact answer up to truncation error. Figure 5 shows a plot of the relative error E_{rel} in the approximation of π as a function of n. The relative error is well-fit by the line $E_{\text{rel}} = 4.547 n^{-4.007}$ over the range $10 < x < 2000$. It is reasonable for the error to scale like fourth power of n. If sine and cosine are being well-fit by cubic polynomials, then the leading term in the approximation error will be quartic in size. For $x > 2000$, truncation error becomes visible, as expected.

Problem 5 – Image reconstruction using low light

Part (a)

In this question we are given a regular $M \times N$ photo of a still-life scene, plus three low-light photos of the same scene that are illuminated in red, green, and blue. Each pixel in the images can be represented as a three-component vector $\mathbf{p} = (R, G, B)$ for the red, green, and blue components. Let $\mathbf{p}_k^A$ be the kth pixel of the regular photo, and let $\mathbf{p}_k^B$, $\mathbf{p}_k^C$, and $\mathbf{p}_k^D$ be the kth pixel of the three low-light photos. Here, k is indexed from 0 up to $MN - 1$.

The question requires fitting a model for a regular photo pixel $\mathbf{p}_k$ in terms of the corresponding low-light pixels by minimizing

$$S = \frac{1}{MN} \sum_{k=0}^{MN-1} \| F^B \mathbf{p}_k^B + F^C \mathbf{p}_k^C + F^D \mathbf{p}_k^D + \mathbf{p}_{\text{const}} - \mathbf{p}_k^A \|_2^2. \tag{43}$$

[†]This will be explained in detail in Unit 3.

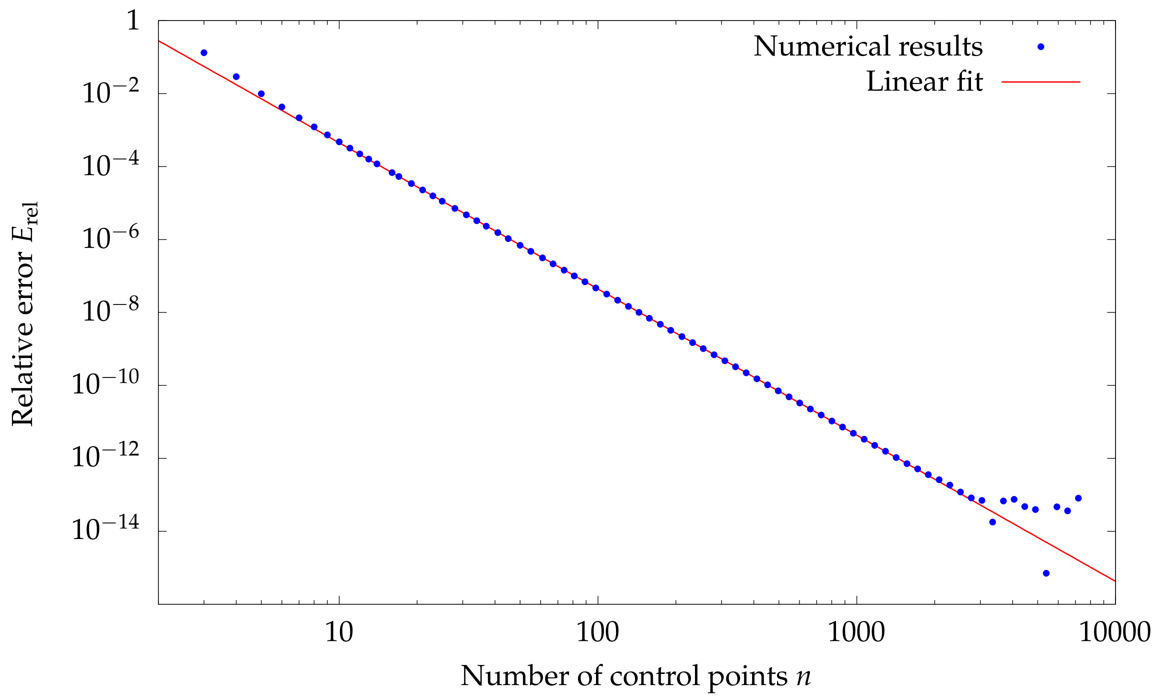

Figure 5: Relative error E_{rel} in the approximation of π using cubic splines as a function of the number of control points.

where F^B, F^C, and F^D are 3×3 matrices and $\mathbf{p}_{\mathrm{const}}$ is a vector. Note that this is equivalent to minimizing three separate quantities in each of the channels,

$$S = S_R + S_G + S_B. \tag{44}$$

Here

$$S_R = \frac{1}{MN} \sum_{k=0}^{MN-1} \left(F^{BR}\mathbf{p}_k^B + F^{CR}\mathbf{p}_k^C + F^{DR}\mathbf{p}_k^D + R_{\mathrm{const}} - R_k^A \right)^2. \tag{45}$$

where F^{BR}, F^{CR}, and F^{DR} are the rows of the F matrices corresponding to the red channel. Analogous expressions exist for the green and blue channels. Equation 45 is now a standard linear least squares problem for the ten parameters making up F^{BR}, F^{CR}, F^{DR}, and R_{const}. The program `pho_recons.py` solves this least squares problem, along with the analogus problems for the green and blue channels. It then uses the fitted parameters to make a reconstruction of the original photo using the three low light images.

Figure 6 shows a comparison between the original photo, and the one that is reconstructed using the low light images. Assume that the pixel values cover the range from 0 to 1. To properly visualize the reconstructed image, the pixel colors are truncated according to $(R,G,B) \to (T(R),T(G),T(B))$ where T is defined by

$$T(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \leq x < 1, \\ 1 & \text{if } x \geq 1. \end{cases} \tag{46}$$

Overall, the general match of colors in the scene is very good. There are some discrepancies due to differences in lighting. To visualize the differences in more detail, the pixel differences $\Delta \mathbf{p}_k = \mathbf{p}_k^{A*} - \mathbf{p}_k^A$ are computed. These values have both negative and positive components, and to visualize them, the negative parts and positive parts are plotted separately in Fig. 7. The square error per pixel is

$$S = \begin{cases} 0.00405 & \text{for } (M,N) = (400,300), \\ 0.00449 & \text{for } (M,N) = (800,600), \\ 0.00485 & \text{for } (M,N) = (1600,1200). \end{cases} \tag{47}$$

Overall, these square errors are very small.

Part (b)

The program `pho_recons.py` also uses the fitted model from part (a) to perform a reconstruction of a regular image of the still-life bear scene room using three low-light photos. Figure 8 shows a comparison between the original image and the reconstructed one. Again, the colors across the scene look realistic, which is remarkable given that it is based on a previously fitted model, and three images with very different lighting characteristics. The square error per pixel between the original image and the reconstruction is

$$T = \begin{cases} 0.00540 & \text{for } (M,N) = (400,300), \\ 0.00517 & \text{for } (M,N) = (800,600), \\ 0.00532 & \text{for } (M,N) = (1600,1200). \end{cases} \tag{48}$$

As expected, the square errors are larger than those from part (a), because the previous model is used, instead of finding the least squares fit for this new set of images. Given small variations in lighting and color, it is not surprising that the previous model is slightly more inaccurate when applied to this set of images. Nevertheless, the square error is still very small.

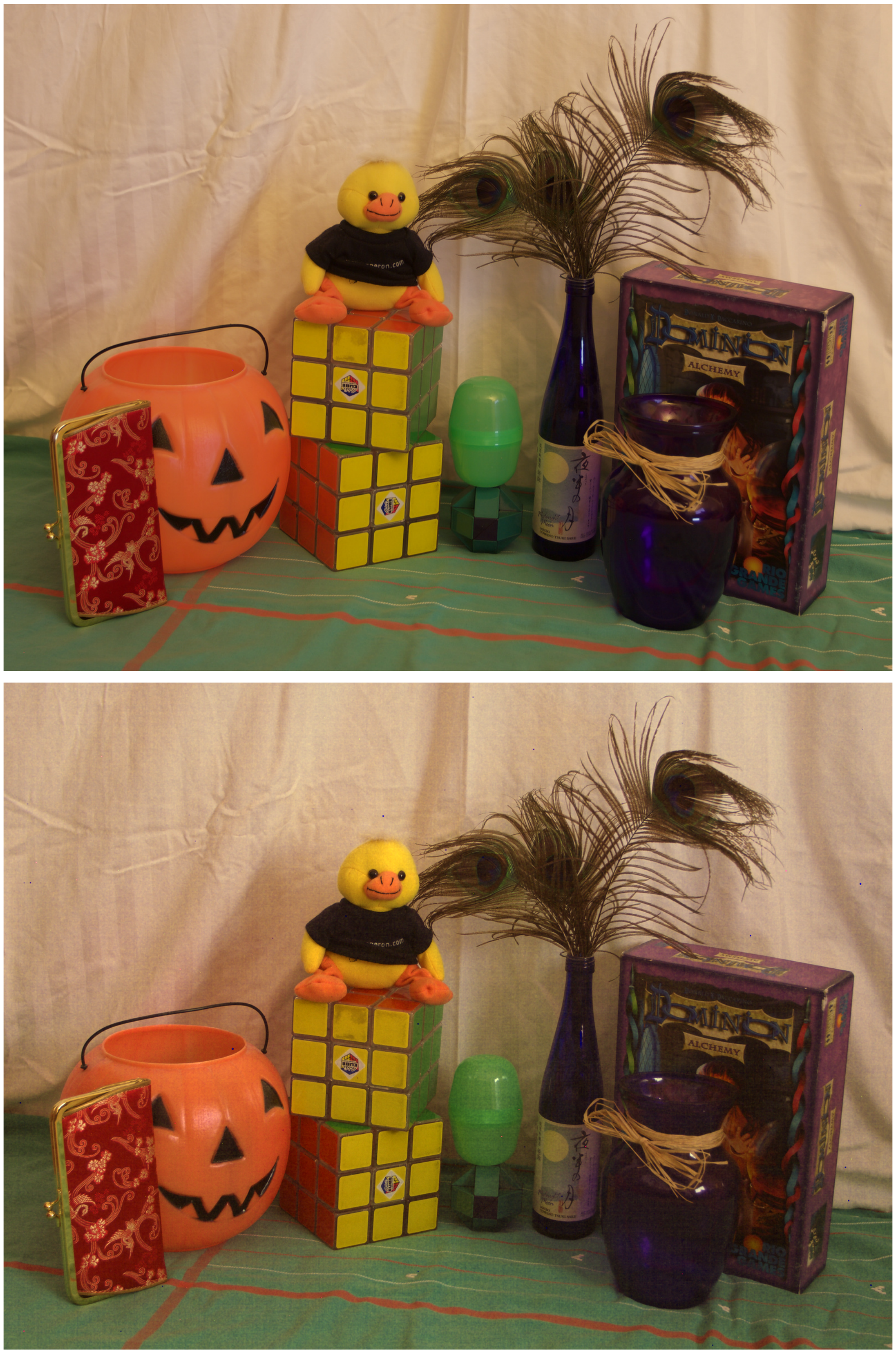

Figure 6: Comparison between the regular photo of the still-life scene (top), and the reconstructed image based on the best fit of the three low-light photos (bottom).

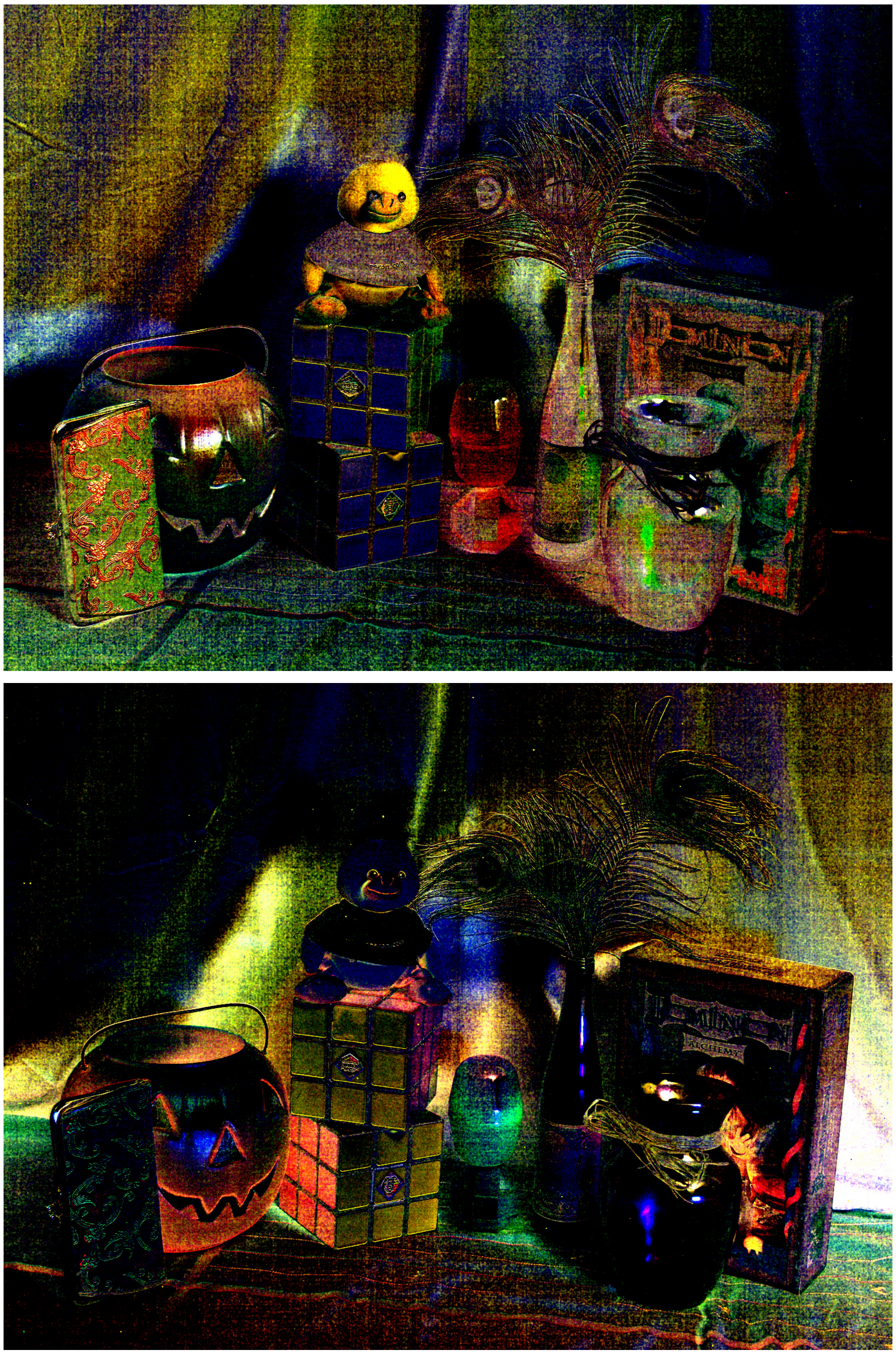

Figure 7: Differences between the reconstructed image and the regular image, given with pixel values $\Delta\mathbf{p}_k = \mathbf{p}_k^{A*} - \mathbf{p}_k^{A}$. Positive and negative color channel components are shown in the top and bottom images, respectively. Channels are scaled up by a factor of 10 to enhance the differences.

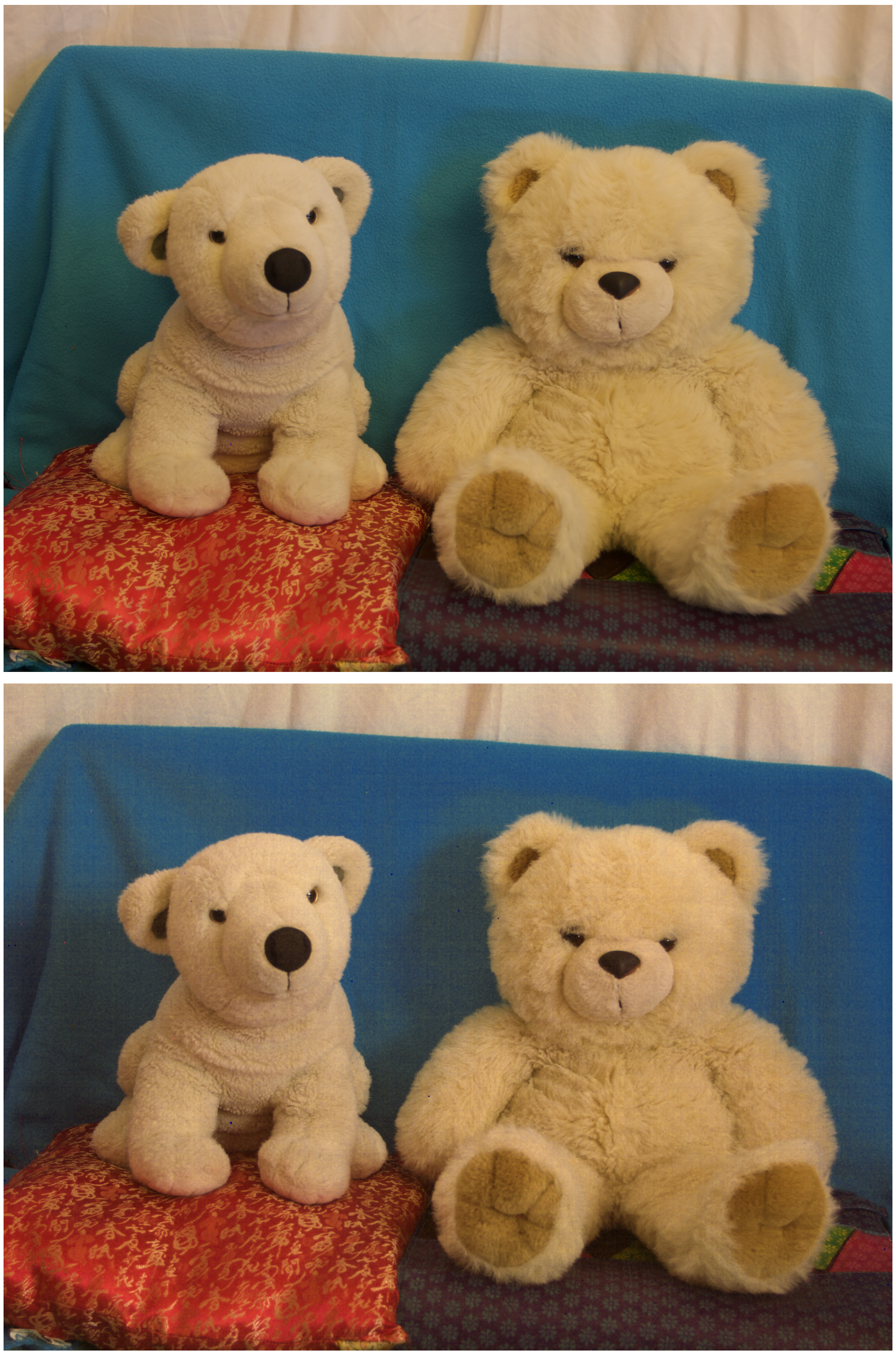
Figure 8: Comparison between the regular photo of two bears (top), and the reconstructed image based on three low-light photos and the previously fitted model (bottom).

Problem 6 – Determining hidden chemical sources

Part (a)

The time derivative of ρ_c is

$$\begin{aligned}\frac{\partial \rho_c}{\partial t} &= \frac{1}{4\pi b}\left(\frac{-1}{t^2} + \frac{-(x^2+y^2)}{4bt}\left(\frac{-1}{t^2}\right)\right)\exp\left(-\frac{x^2+y^2}{4bt}\right) \\ &= \frac{x^2+y^2-4bt}{16\pi b^2 t^3}\exp\left(-\frac{x^2+y^2}{4bt}\right). \end{aligned} \tag{49}$$

The x derivative of ρ_c is

$$\frac{\partial \rho_c}{\partial x} = \frac{-2x}{16\pi b^2 t^2}\exp\left(-\frac{x^2+y^2}{4bt}\right) \tag{50}$$

and the second x derivative is

$$\frac{\partial^2 \rho_c}{\partial x^2} = \frac{x^2-2bt}{16\pi b^3 t^3}\exp\left(-\frac{x^2+y^2}{4bt}\right). \tag{51}$$

By symmetry the second y derivative is

$$\frac{\partial^2 \rho_c}{\partial y^2} = \frac{y^2-2bt}{16\pi b^3 t^3}\exp\left(-\frac{x^2+y^2}{4bt}\right) \tag{52}$$

and hence

$$\nabla^2 \rho_c = \frac{x^2+y^2-4bt}{16\pi b^3 t^3}\exp\left(-\frac{x^2+y^2}{4bt}\right). \tag{53}$$

Comparing Eqs. 49 and 53 shows that $\partial_t \rho_c = b\nabla^2 \rho_c$ as required.

Part (b)

We now consider the case when $b = 1$ and 49 point sources of chemicals are introduced at $t = 0$ with different strengths, on a 7×7 regular lattice covering the coordinates $x = -3, -2, \ldots, 3$ and $y = -3, -2, \ldots, 3$. The concentration satisfies

$$\rho(\mathbf{x}, t) = \sum_{k=0}^{48} \lambda_k \rho_c(\mathbf{x} - \mathbf{v}_k, t), \tag{54}$$

Lattice site	$(0,0)$	$(1,1)$	$(2,2)$	$(3,3)$
St. Dev. of λ_k	21615	13609	2800	115

Table 2: Standard deviations in the λ_k when the measured concentrations are perturbed by a small normally distributed shift with mean 0 and variance 10^{-8}. The values are computed based on $N = 10^6$ random samples.

where $\mathbf{v}_k$ is the kth lattice site and λ_k is the strength of the chemical introduced at that site.

Two hundred measurements, $\rho_M(\mathbf{x}_i, t)$, at locations $\mathbf{x}_i$ and at $t = 4$ are provided. Estimating the concentrations can be viewed as a linear least squares problem, finding the λ_k such that

$$S = \sum_{i=0}^{199} \left| \rho_M(\mathbf{x}_i, t) - \sum_{k=0}^{48} \lambda_k \rho_c(\mathbf{x}_i - \mathbf{v}_k, t) \right|. \tag{55}$$

Even though Eq. 55 is quite complicated and involves the the expression for ρ_c, the parameters λ_k still enter linearly, and hence it can be solved using the linear least squares approach. The program `solve_cons.py` computes the λ_k and prints them. They are all positive, with a maximum value of approximately 252.

Part (c)

The program `stdev_cons.py` performs a sample of N computations of the λ_k when each of the ρ_M are perturbed by a small normally distributed shift with mean 0 and variance 10^{-8}. For each λ_k, the standard deviation is computed. Due to the sensitivity of the fitting procedure, these small differences in the measurements cause large alterations in the λ_k. Table 2 shows the standard deviations for the λ_k for four lattice sites, which show much larger variations than the actual λ_k values that were measured in part (b). The largest errors are at the central $(0, 0)$ lattice site, which is reasonable since it is furthest away from any of the measurements in the file, thus making it most difficult to estimate.