

CAAM 453 · NUMERICAL ANALYSIS I

Problem Set 3 · Solutions

Posted Friday 25 September 2009. Due Monday 5 October 2009.

CAAM 453 students should complete 100 points worth of problems.

CAAM 553 students should complete 125 points worth of problems (4 and 5 strongly recommended).

Students are welcome to attempt more problems if they wish.

1. [25 points: (a)=10 points; (b)=7 points; (c)=8 points.]

Recall that for $\mathbf{A} \in \mathbb{C}^{n \times n}$, the linear system $\mathbf{A}\mathbf{c} = \mathbf{f}$ has a unique solution for any \mathbf{f} provided $\text{Ker}(\mathbf{A}) = \{\mathbf{0}\}$, where $\text{Ker}(\mathbf{A})$ denotes the kernel (null space) of \mathbf{A} .

If the kernel of \mathbf{A} is larger, i.e., if there is a nonzero vector $\mathbf{z} \in \text{Ker}(\mathbf{A})$, then there are two possibilities:

- If $\mathbf{f} \notin \text{Ran}(\mathbf{A})$, then there is *no solution* \mathbf{c} to the linear system $\mathbf{A}\mathbf{c} = \mathbf{f}$.
- If $\mathbf{f} \in \text{Ran}(\mathbf{A})$, then there are *infinitely many solutions* to the linear system $\mathbf{A}\mathbf{c} = \mathbf{f}$. In particular, if $\widehat{\mathbf{c}}$ satisfies $\mathbf{A}\widehat{\mathbf{c}} = \mathbf{f}$, then any \mathbf{c} of the form $\mathbf{c} = \widehat{\mathbf{c}} + \gamma\mathbf{z}$ is also a solution, where γ is an arbitrary constant.

With these facts in mind, please answer the following questions.

- (a) Suppose we wish to construct a polynomial $p_5 \in \mathcal{P}_5$ that interpolates a function $f \in \mathbb{C}^2[-1, 1]$ in the following (somewhat unusual) manner: $p_5(-1) = f(-1)$; $p_5'(-1) = f'(-1)$; $p_5(0) = f(0)$; $p_5''(0) = f''(0)$; $p_5(1) = f(1)$; $p_5'(1) = f'(1)$. Write down the linear system to determine the coefficients c_0, \dots, c_5 for p in the monomial basis: $p_5(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 + c_5x^5$.
- (b) What is the kernel of the matrix \mathbf{A} constructed in part (a)?
(You may use the MATLAB command `null(A, 'r')`.)
What does your answer imply about the existence and uniqueness of the interpolant p_5 ?
- (c) Consider the data: $f(-1) = -1$, $f'(-1) = 0$, $f(0) = 1$, $f''(0) = -2$, $f(1) = 3$, $f'(1) = 4$. Show that there are infinitely many choices for the polynomial p_5 that interpolate this data. Plot six of them. (Superimpose all on the same plot.)

Solution.

- (a) We seek the coefficients c_0, \dots, c_5 to the polynomial

$$p_5(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 + c_5x^5,$$

which will be determined by the five constraints

$$c_0 + c_1x_0 + c_2x_0^2 + c_3x_0^3 + c_4x_0^4 + c_5x_0^5 = f(x_0)$$

$$c_1 + 2c_2x_0 + 3c_3x_0^2 + 4c_4x_0^3 + 5c_5x_0^4 = f'(x_0)$$

$$c_0 + c_1x_1 + c_2x_1^2 + c_3x_1^3 + c_4x_1^4 + c_5x_1^5 = f(x_1)$$

$$2c_2 + 6c_3x_0 + 12c_4x_0^2 + 20c_5x_0^3 = f''(x_1)$$

$$c_0 + c_1x_2 + c_2x_2^2 + c_3x_2^3 + c_4x_2^4 + c_5x_2^5 = f(x_2)$$

$$c_1 + 2c_2x_2 + 3c_3x_2^2 + 4c_4x_2^3 + 5c_5x_2^4 = f'(x_2).$$

These equations can be written in the matrix form

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & x_0^4 & x_0^5 \\ 0 & 1 & 2x_0 & 3x_0^2 & 4x_0^3 & 5x_0^4 \\ 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 \\ 0 & 0 & 2 & 6x_1 & 12x_1^2 & 20x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 \\ 0 & 1 & 2x_2 & 3x_2^2 & 4x_2^3 & 5x_2^4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f'(x_0) \\ f(x_1) \\ f''(x_1) \\ f(x_2) \\ f'(x_2) \end{bmatrix}$$

With our values for the nodes $x_0 = -1$, $x_1 = 0$, $x_2 = 1$, we have

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 & -4 & 5 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f'(x_0) \\ f(x_1) \\ f''(x_1) \\ f(x_2) \\ f'(x_2) \end{bmatrix}.$$

(b) Using the `null` command or otherwise, one finds that the kernel is the span of the vector

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ -2 \\ 0 \\ 1 \end{bmatrix}.$$

Since the kernel is non-trivial, there are two possible situations: there will either be *no polynomial*, or there will be *infinitely many polynomials* that satisfy the six interpolation conditions. Which of the two depends on the actual interpolation conditions.

(c) Note that the vector

$$\mathbf{f} = \begin{bmatrix} f(x_0) \\ f'(x_0) \\ f(x_1) \\ f''(x_1) \\ f(x_2) \\ f'(x_2) \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \\ -2 \\ 3 \\ 4 \end{bmatrix}$$

is in the range of the coefficient matrix, since we can write

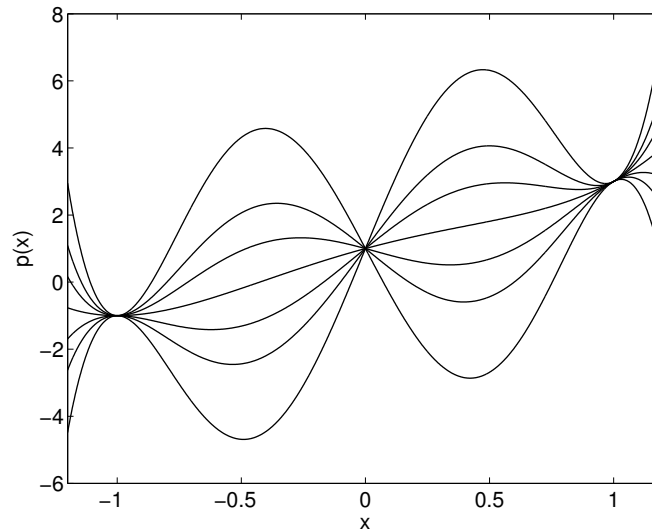
$$\begin{bmatrix} -1 \\ 0 \\ 1 \\ -2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 & -4 & 5 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Since \mathbf{f} is in $\text{Ran}(A)$, there are infinitely many choices for the coefficients c_0, \dots, c_5 that will satisfy the six constraints. All solutions have the form

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \gamma \begin{bmatrix} 0 \\ 1 \\ 0 \\ -2 \\ 0 \\ 1 \end{bmatrix}$$

for arbitrary γ .

The plot below shows polynomials for the above coefficients chosen with $\gamma = -16, -8, -4, 0, 4, 8, 16$.



2. [25 points: (a)=10 points; (b)=15 points]

The *Hermite interpolant* $h_n \in \mathcal{P}_{2n+1}$ of $f \in C^1[a, b]$ at the points $\{x_j\}_{j=0}^n$ can be written in the form

$$h_n(x) = \sum_{j=0}^n \left(A_j(x) f(x_j) + B_j(x) f'(x_j) \right),$$

where the functions A_j and B_j generalize the Lagrange basis functions:

$$\begin{aligned} A_j(x) &= (1 - 2\ell_j'(x_j)(x - x_j)) \ell_j^2(x) \\ B_j(x) &= (x - x_j) \ell_j^2(x), \end{aligned}$$

with $\ell_j(x) = \prod_{k=0, k \neq j}^n (x - x_k) / (x_j - x_k)$.

(a) Verify that

$$A_j(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k, \end{cases} \quad A_j'(x_k) = 0, \quad B_j(x_k) = 0, \quad B_j'(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k. \end{cases}$$

(b) The above expression for the Hermite interpolating polynomial mimics the *Lagrange form* of the standard interpolating polynomial. Devise a scheme for constructing Hermite interpolants that generalizes the *Newton form*. What are your new Newton-like basis functions for \mathcal{P}_{2n+1} ?

Solution.

(a) First consider $A_j(x_k)$. If $k \neq j$, then

$$A_j(x_k) = (1 - 2\ell_j'(x_j)(x_k - x_j)) \ell_j^2(x_k) = 0$$

since $\ell_j(x_k) = 0$ by construction. If $k = j$, then

$$A_j(x_k) = (1 - 2\ell_j'(x_j)(x_j - x_j)) \ell_j^2(x_j) = \ell_j^2(x_j) = 1,$$

since $\ell_j(x_j) = 1$ by construction.

Next consider $A'_j(x_k)$. To begin with,

$$A'_j(x) = -2\ell'_j(x_j)\ell_j^2(x) + 2(1 - 2\ell'_j(x_j)(x - x_j))\ell_j(x)\ell'_j(x).$$

For $k \neq j$, we have

$$A'_j(x_k) = -2\ell'_j(x_j)\ell_j^2(x_k) + 2(1 - 2\ell'_j(x_j)(x_k - x_j))\ell_j(x_k)\ell'_j(x_k) = 0,$$

since both terms in this sum have $\ell_j(x_k)$ terms. When $j = k$, we have

$$\begin{aligned} A'_j(x_k) &= -2\ell'_j(x_j)\ell_j^2(x_j) + 2(1 - 2\ell'_j(x_j)(x_j - x_j))\ell_j(x_j)\ell'_j(x_j) \\ &= -2\ell'_j(x_j)\ell_j^2(x_j) + 2\ell'_j(x_j) \\ &= 0, \end{aligned}$$

since $\ell_j(x_j) = 1$. Thus A_j and A'_j both perform as required.

It is simple to see that $B_j(x_k) = 0$ since $\ell_j(x_k) = 0$ if $k \neq j$, and $(x_k - x_j) = 0$ if $k = j$. Note that

$$B'_j(x) = \ell_j^2(x) + 2(x - x_j)\ell_j(x)\ell'_j(x).$$

If $k \neq j$, then $\ell_j(x_k) = 0$ and so $B'_j(x_k) = 0$. If $k = j$, then

$$B'_j(x_k) = \ell_j^2(x_j) + 2(x - j - x_j)\ell_j(x_j)\ell'_j(x_j) = 1,$$

since $\ell_j(x_j) = 1$.

- (b) The principle behind the Newton basis for standard polynomial interpolation is: find some constant function that interpolates at x_0 . Thus use this to find a linear function that interpolates at x_0 and x_1 , and so on. The Newton basis functions are $\{1, x - x_0, (x - x_0)(x - x_1), \dots, \prod_{j=0}^{n-1} (x - x_j)\}$. For Hermite interpolation, we will attempt to follow the same methodology. First, find $p_0 \in \mathcal{P}_0$ such that $p_0(x_0) = f(x_0)$:

$$p_0(x) = c_0 = f(x_0).$$

Thus, $p_0 = c_0 q_0(x)$, where the basis function $q_0(x) \equiv 1$.

Next, find a linear polynomial that interpolates both f and f' at x_0 : i.e., find $p_1 \in \mathcal{P}_1$ such that $p_1(x_0) = f(x_0)$ and $p'_1(x_0) = f'(x_0)$. In keeping with the Newtonian spirit, write p_1 in the form

$$p_1(x) = p_0(x) + c_1 q_1(x)$$

for some $q_1 \in \mathcal{P}_1$. Our challenge is to find c_1 and q_1 to satisfy the interpolation conditions. Since $p_0(x_0) = f(x_0)$, the interpolation condition $p_1(x_0) = f(x_0)$ implies $q_1(x_0) = 0$. Therefore, we conclude that q_1 has a root at x_0 ; this completely determines q_1 , since it is a linear polynomial and we are not concerned about scaling factors (which are absorbed by c_1):

$$q_1(x) = x - x_0.$$

Now c_1 is determined so that $p'_1(x_0) = c_0 q'_0(x_0) + c_1 q'_1(x_0) = f'(x_0)$:

$$c_1 = \frac{f'(x_0) - c_0 q'_0(x_0)}{q'_1(x_0)}.$$

So far, this basis is the same as the usual Newton basis. The next step introduces the critical difference. We want to construct $p_2 \in \mathcal{P}_2$ of the form

$$p_2(x) = p_1(x) + c_2 q_2(x)$$

- (a) Set up a linear system $\mathbf{A}\mathbf{c} = \mathbf{f}$ to determine the coefficients c_0, \dots, c_5 .
- (b) Write a MATLAB code to determine \mathbf{c} when $f(x, y) = e^x \sin y$ and the (x_j, y_j) pairs take the values listed in the following table.

j	0	1	2	3	4	5
x_j	0	0	1	1	2	2
y_j	0	2	0	2	1	3

Report your value for \mathbf{c} .

- (c) Plot your model function $p(x, y)$ over $x \in [-1, 3]$, $y \in [-1, 3]$ using MATLAB's `surf` command. Compare this plot to the similar plot for $f(x, y)$, which can be obtained in the following manner.

```
f = inline('exp(x).*sin(y)', 'x', 'y');
[xx,yy] = meshgrid(linspace(-1,3,25),linspace(-1,3,25));
zz = f(xx,yy);
figure(1), clf
surf(xx,yy,zz)
```

Please submit plots of both $p(x, y)$ and $f(x, y)$.

Solution.

- (a) The six interpolation conditions are

$$\begin{aligned} c_0 + c_1x_0 + c_2y_0 + c_3x_0y_0 + c_4x_0^2 + c_5y_0^2 &= e^{x_0} \sin y_0 \\ c_0 + c_1x_1 + c_2y_1 + c_3x_1y_1 + c_4x_1^2 + c_5y_1^2 &= e^{x_1} \sin y_1 \\ c_0 + c_1x_2 + c_2y_2 + c_3x_2y_2 + c_4x_2^2 + c_5y_2^2 &= e^{x_2} \sin y_2 \\ c_0 + c_1x_3 + c_2y_3 + c_3x_3y_3 + c_4x_3^2 + c_5y_3^2 &= e^{x_3} \sin y_3 \\ c_0 + c_1x_4 + c_2y_4 + c_3x_4y_4 + c_4x_4^2 + c_5y_4^2 &= e^{x_4} \sin y_4 \\ c_0 + c_1x_5 + c_2y_5 + c_3x_5y_5 + c_4x_5^2 + c_5y_5^2 &= e^{x_5} \sin y_5. \end{aligned}$$

These equations can be arranged into the matrix equation $\mathbf{A}\mathbf{c} = \mathbf{f}$:

$$\begin{bmatrix} 1 & x_0 & y_0 & x_0y_0 & x_0^2 & y_0^2 \\ 1 & x_1 & y_1 & x_1y_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & x_2y_2 & x_2^2 & y_2^2 \\ 1 & x_3 & y_3 & x_3y_3 & x_3^2 & y_3^2 \\ 1 & x_4 & y_4 & x_4y_4 & x_4^2 & y_4^2 \\ 1 & x_5 & y_5 & x_5y_5 & x_5^2 & y_5^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} e^{x_0} \sin y_0 \\ e^{x_1} \sin y_1 \\ e^{x_2} \sin y_2 \\ e^{x_3} \sin y_3 \\ e^{x_4} \sin y_4 \\ e^{x_5} \sin y_5 \end{bmatrix}.$$

- (b),(c) The following MATLAB code solves the system and produces the desired plots.

```
x = [0;0;1;1;2;2];           % x interpolation points
y = [0;2;0;2;1;3];           % y interpolation points

% set up coefficient matrix
A = [ones(6,1) x y x.*y x.*x y.*y];

% set up a grid in two dimensions and evaluate the function there
f = inline('exp(x).*sin(y)', 'x', 'y');
[xx,yy] = meshgrid(linspace(-1,3,25),linspace(-1,3,25));
zz = f(xx,yy);

% plot the function f on the grid
```

```

figure(1), clf
surf(xx,yy,zz)
set(gca,'fontsize',16)
xlabel('x'), ylabel('y'), zlabel('f(x,y)')

% find the coefficients c and plot the polynomial
figure(2), clf
c = A\f(x,y);
surf(xx,yy,c(1)+c(2)*xx+c(3)*yy+c(4)*xx.*yy+c(5)*xx.^2+c(6)*yy.^2)
set(gca,'fontsize',16)
xlabel('x'), ylabel('y'), zlabel('f(x,y)')

% output coefficients
format long
fprintf('coefficients, c:\n')
disp(c)

% check error at interpolation points
pxy = c(1)+c(2)*x+c(3)*y+c(4)*x.*y+c(5)*x.^2+c(6)*y.^2;
fxy = f(x,y);
fprintf('\nmaximum error at interpolation points = %10.7e\n', max(abs(fxy-pxy)))

```

The resulting output follows.

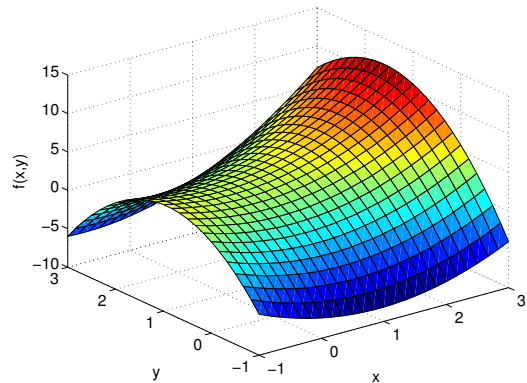
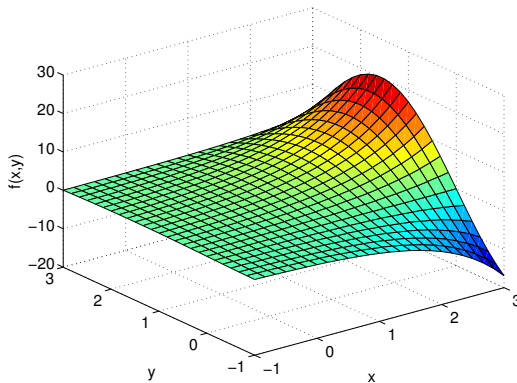
coefficients, c:

```

0
-0.94916310522349
5.05919300007085
0.78121462258957
0.94916310522349
-2.30227214332900

```

maximum error at interpolation points = 1.7763568e-15



4. [25 points: (a)=5 points; (b)=5 points; (c)=7 points; (d)=8 points]

When we have spoken about norms, projectors, and the like, we have usually been working with vectors and matrices, but these concepts generalize to a much broader setting. In this problem, you will apply these ideas to develop a bound on the accuracy of polynomial interpolation.

Let Π_n denote the linear operator that maps $f \in C[a, b]$ to the polynomial p_n that interpolates f at the distinct points $x_0, \dots, x_n, \{x_j\}_{j=0}^n \subset [a, b]$. In other words, $\Pi_n f = p_n$, where p_n is the unique polynomial of degree n (or less) for which $f(x_j) = p_n(x_j)$ for $j = 0, \dots, n$.

- (a) Explain why Π_n is a *projector*.
(What does $\Pi_n p_n$ equal if p_n is a polynomial of degree n ?)

For the rest of this problem, we use the following norm on $g \in C[a, b]$:

$$\|g\|_{L^\infty} = \max_{a \leq x \leq b} |g(x)|.$$

This norm obeys the three familiar norm axioms (e.g., the triangle inequality). It also induces the operator norm

$$\|\Pi_n\|_{L^\infty} = \max_{f \in C[a,b], f \neq 0} \frac{\|\Pi_n f\|_{L^\infty}}{\|f\|_{L^\infty}} = \max_{\|f\|_{L^\infty}=1} \|\Pi_n f\|_{L^\infty}.$$

(b) Show that if $x_0 = a$ and $x_1 = b$, then $\|\Pi_0\|_{L^\infty} = \|\Pi_1\|_{L^\infty} = 1$.

(c) Recall that we can write the polynomial $p_n = \Pi_n f$ in the Lagrange form

$$\Pi_n f = \sum_{j=0}^n f(x_j) \ell_j(x),$$

where ℓ_k denotes the k th Lagrange basis polynomial.

$$\text{Prove that } \|\Pi_n\|_{L^\infty} = \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|.$$

(d) Let p_* denote any polynomial of degree n (e.g., p_* minimizes $\|f - p\|_{L^\infty}$ over all $p \in \mathcal{P}_n$).

Prove that $\|f - p_n\|_{L^\infty} \leq (1 + \|\Pi_n\|_{L^\infty})\|f - p_*\|_{L^\infty}$.

(e) (Optional) Computationally estimate $\|\Pi_n\|_{L^\infty}$ for $n = 1, \dots, 20$ with (i) uniformly spaced points $x_j = -1 + 2j/n$ and (ii) Chebyshev points $x_j = \cos(j\pi/n)$ over $[-1, 1]$.

Solution.

(a) Given any $f \in C[a, b]$, let $p_n \in \mathcal{P}_n$ denote the polynomial that interpolates f at x_0, \dots, x_n . Hence, by definition of Π_n :

$$\Pi_n f = p_n.$$

Since $p_n \in \mathcal{P}_n$ and the interpolating polynomial is unique, we must have that $\Pi_n p_n = p_n$. It follows that

$$\Pi_n^2 f = \Pi_n \Pi_n f = \Pi_n p_n = p_n.$$

Since $\Pi_n^2 f = p_n = \Pi_n f$ for all $f \in C[a, b]$, it follows that $\Pi_n^2 = \Pi_n$. Hence, Π_n is a projector.

(b) The degree $n = 0$ interpolant to f at $x_0 = a$ is given by $p_0(x) = f(a)$. Hence

$$\|\Pi_0\|_{L^\infty} = \max_{f \in C[a,b], f \neq 0} \frac{\|\Pi_0 f\|_{L^\infty}}{\|f\|_{L^\infty}} = \max_{f \in C[a,b], f \neq 0} \frac{\|f(a)\|_{L^\infty}}{\|f\|_{L^\infty}} \leq \max_{f \in C[a,b], f \neq 0} \frac{\|f\|_{L^\infty}}{\|f\|_{L^\infty}} = 1.$$

Equality is attained when f is any nonzero constant, so $\|\Pi_0\|_{L^\infty} = 1$.

For $n = 1$, $p_1(x)$ is the line connecting $(a, f(a))$ with $(b, f(b))$. Hence $\|p_1(x)\|_{L^\infty} = \max\{|f(a)|, |f(b)|\}$.

We conclude that

$$\|\Pi_1\|_{L^\infty} = \max_{f \in C[a,b], f \neq 0} \frac{\|\Pi_1 f\|_{L^\infty}}{\|f\|_{L^\infty}} = \max_{f \in C[a,b], f \neq 0} \frac{\max\{|f(a)|, |f(b)|\}}{\|f\|_{L^\infty}} \leq \max_{f \in C[a,b], f \neq 0} \frac{\|f\|_{L^\infty}}{\|f\|_{L^\infty}} = 1.$$

Again, equality is attained when f is any nonzero constant, so $\|\Pi_1\|_{L^\infty} = 1$.

(c) We can write the interpolating polynomial $p_n = \Pi_n f$ using the Lagrange form,

$$p_n = \sum_{j=0}^n f(x_j) \ell_j(x).$$

Hence, we can develop an upper bound:

$$\begin{aligned}\|\Pi_n f\|_{L^\infty} &= \left\| \sum_{j=0}^n f(x_j) \ell_j(x) \right\|_{L^\infty} = \max_{x \in [a,b]} \left| \sum_{j=0}^n f(x_j) \ell_j(x) \right| \\ &\leq \max_{x \in [a,b]} \sum_{j=0}^n |f(x_j)| |\ell_j(x)| \leq \max_{x \in [a,b]} \|f\|_{L^\infty} \sum_{j=0}^n |\ell_j(x)|,\end{aligned}$$

from which it follows that

$$\|\Pi_n\|_{L^\infty} = \max_{f \in C[a,b], f \neq 0} \frac{\|\Pi_n f\|_{L^\infty}}{\|f\|_{L^\infty}} \leq \max_{f \in C[a,b], f \neq 0} \frac{\max_{x \in [a,b]} \|f\|_{L^\infty} \sum_{j=0}^n |\ell_j(x)|}{\|f\|_{L^\infty}} \leq \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|.$$

We thus have shown that proposed formula for $\|\Pi_n\|_{L^\infty}$ is an upper bound; it remains to show that this upper bound is attained. Let $\hat{x} \in [a, b]$ be a point for which

$$\sum_{j=0}^n |\ell_j(\hat{x})| = \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|.$$

Now define f to be any continuous function on $[a, b]$ such that

$$f(x_j) = \text{sign}(\ell_j(\hat{x})),$$

with $f(x) \in [-1, 1]$ for all $x \in [a, b]$. Thus $\|f\|_{L^\infty} = 1$ and

$$\begin{aligned}\|\Pi_n f\|_{L^\infty} &= \left\| \sum_{j=0}^n f(x_j) \ell_j(x) \right\|_{L^\infty} = \max_{x \in [a,b]} \left| \sum_{j=0}^n f(x_j) \ell_j(x) \right| \\ &\geq \left| \sum_{j=0}^n f(x_j) \ell_j(\hat{x}) \right| = \sum_{j=0}^n |\ell_j(\hat{x})| = \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|.\end{aligned}$$

Since $\|f\|_{L^\infty} = 1$, we have proved the lower bound $\|\Pi_n\|_{L^\infty} \geq \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|$. Hence

$$\max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)| \leq \|\Pi_n\|_{L^\infty} \leq \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|,$$

so we have proved the required formula for $\|\Pi_n\|_{L^\infty}$.

(d) We begin by noting that

$$\begin{aligned}\|f - p_n\|_{L^\infty} &= \|f - p_n - p_* + p_*\|_{L^\infty} \\ &= \|f - p_*\|_{L^\infty} + \|p_n - p_*\|_{L^\infty}.\end{aligned}$$

Notice that since p_* is a polynomial of degree- n (or less), we must have $p_* = \Pi_n p_*$ (see part (a)). Additionally, we have $p_n = \Pi_n f$. From these two observations it follows that

$$\begin{aligned}\|p_n - p_*\|_{L^\infty} &= \|\Pi_n f - \Pi_n p_*\|_{L^\infty} \\ &= \|\Pi_n\|_{L^\infty} \|f - p_*\|_{L^\infty}.\end{aligned}$$

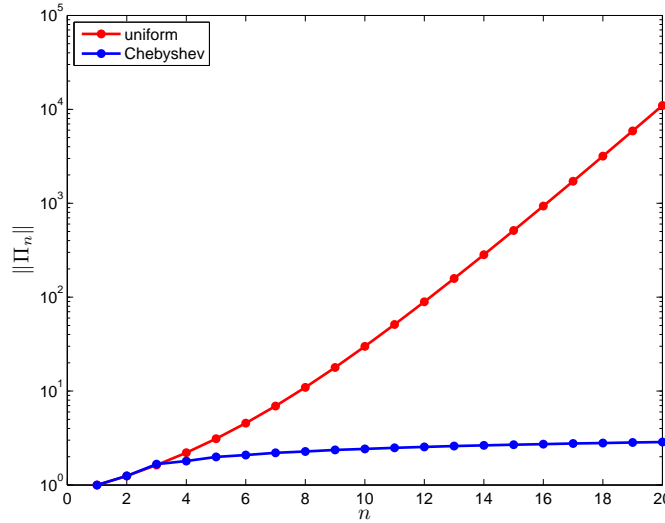
Inserting this result into our expression for $\|f\|_{L^\infty}$, we arrive at

$$\|f - p_n\|_{L^\infty} \leq (1 + \|\Pi_n\|_{L^\infty}) \|f - p_*\|_{L^\infty},$$

as required.

What is the significance of this result? We often construct interpolants in an effort to obtain polynomials that approximate $f \in C[a, b]$ throughout the interval $[a, b]$, not just at the interpolation points. This result allows us to compare the accuracy of this interpolant throughout $[a, b]$ (measured by $\|f - p_n\|_{L^\infty}$) to the accuracy obtained by the *best approximating polynomial* $p_* \in \mathcal{P}_n$ (measured by $\|f - p_*\|_{L^\infty}$). The scaling factor $1 + \|\Pi_n\|_{L^\infty}$ bounds how much worse the interpolant can be. The quantity $\|\Pi_n\|_{L^\infty}$ is independent of f , but highly dependent on the choice of interpolation points (see part (e)).

- (e) It is not difficult to write a MATLAB code to estimate the constants $\|\Pi_n\|_{L^\infty}$ for uniformly spaced points and Chebyshev points. The results are shown in the plot below: the numerics indicate that $\|\Pi_n\|_{L^\infty}$ grows *exponentially* for uniformly spaced points, but only *logarithmically* for Chebyshev points. Google ‘Lebesgue constants’ for more details!



5. [25 points: (a)=7 points; (b)=4 points; (c)=6 points; (d)=4 points; (e)=4 points]

Suppose the complex-valued function $f(z)$ of the variable $z \in \mathbb{C}$ is analytic in a region D of the complex plane whose boundary C is a simple closed contour. Furthermore, suppose the interpolation points x_0, \dots, x_n ($n \geq 1$) and the point x all lie in D .

- (a) Let $p_n \in \mathcal{P}_n$ denote the polynomial that interpolates f at x_0, \dots, x_n . For any $x \in D$, confirm the identity

$$f(x) - p_n(x) = \frac{1}{2\pi i} \int_C \frac{f(z)}{z-x} \prod_{j=0}^n \frac{x-x_j}{z-x_j} dz$$

by computing the integral on the right. (Hint: Consider the poles of the integrand, and use the Cauchy integral formula.)

For the rest of the problem, suppose that the real number x and the interpolation points x_0, \dots, x_n all lie in the real interval $[a, b]$, and define, for constant $K > 0$,

$$D = \{z \in \mathbb{C} : |z - t| < K \text{ for some } t \in [a, b]\}.$$

- (b) Plot (or draw) the boundary C of D for $[a, b] = [-1, 1]$ and $K = 1$.

- (c) Show that the length of the contour C is $2(b-a) + 2\pi K$, and that the integral formula in (a) leads to the bound

$$|f(x) - p_n(x)| < \frac{(b-a + \pi K)M}{\pi K} \left(\frac{b-a}{K}\right)^{n+1},$$

where M is such that $|f(z)| \leq M$ on C .

- (d) Deduce that if f is analytic on D for some $K > |b-a|$, then the sequence $\{p_n\}$ converges to f uniformly on $[a, b]$ as $n \rightarrow \infty$.
- (e) Show that the requirements for the conclusion in (d) *are not satisfied* by Runge's function, $f(x) = 1/(1+x^2)$ over $[a, b] = [-5, 5]$. For what values of α are the conditions satisfied by this f over $[a, b] = [-\alpha, \alpha]$?

[Süli and Mayers, Problem 6.11]

Solution.

- (a) The integrand

$$g(z) = \frac{f(z)}{z-x} \prod_{j=0}^n \frac{z-x_j}{z-x_j}$$

has poles at $z = x, x_0, x_1, \dots, x_n$. If $x = x_j$ for any $j \in \{0, \dots, n\}$, then $f(x) - p_n(x) = 0$ by construction of the interpolant p_n , so the ultimate result (part (c)) is trivial. If x is not equal to any of the interpolation points, then all the poles of the integrand g are simple. Hence, the integral reduces to a straightforward residue calculation:

$$\begin{aligned} \operatorname{res}(g, x_k) &= \lim_{z \rightarrow x_k} g(z)(z - x_k) \\ &= -f(x_k) \prod_{j=0, j \neq k}^n \frac{x - x_j}{x_k - x_j} \end{aligned}$$

and

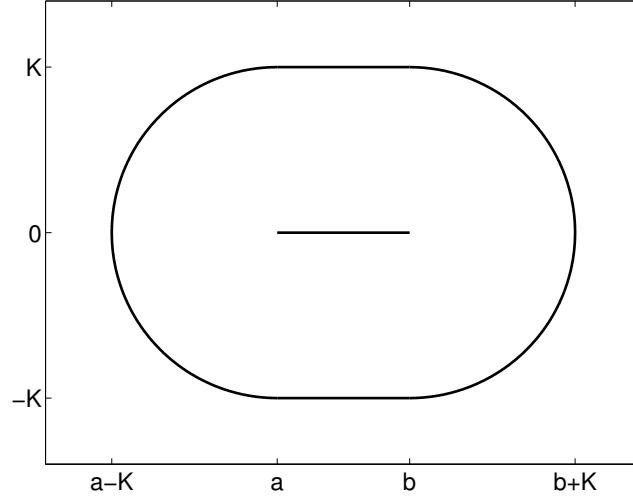
$$\begin{aligned} \operatorname{res}(g, x) &= \lim_{z \rightarrow x} g(z)(z - x) \\ &= f(x). \end{aligned}$$

Cauchy's integral formula then gives that

$$\begin{aligned} \frac{1}{2\pi i} \int_C g(z) dz &= \operatorname{res}(g, x) + \sum_{k=0}^n \operatorname{res}(g, x_k) \\ &= f(x) - \sum_{k=0}^n f(x_k) \prod_{j=0, j \neq k}^n \frac{x - x_j}{x_k - x_j} \\ &= f(x) - \sum_{k=0}^n f(x_k) \ell_k(x) \\ &= f(x) - p_n(x), \end{aligned}$$

where ℓ_k denotes the k th Lagrange basis polynomial.

- (b) The specified region is an oval whose boundary is a curve C that consists of two half-circles of radius K centered at a and b on the left and right, with line segments connecting them; see the figure below.



- (c) The arc-length of C is the sum of the perimeter of the circular arcs and line segments: $2\pi K + 2(b - a)$.

We can now bound the error by coarsely approximating the integral for $f - p_n$. (Note that for any $z \in C$, $|x - z| \geq K$ and $|x_k - z| \geq K$, and that $|x - x_k| \leq b - a$ for all $x \in [a, b]$.) Letting L_C denote the arc-length of C , we have

$$\begin{aligned} |f(x) - p_n(x)| &= \left| \frac{1}{2\pi i} \int_C \frac{f(z)}{z - x} \prod_{j=0}^n \frac{x - x_j}{z - x_j} dz \right| \\ &\leq \frac{L_C}{2\pi} \max_{z \in C} \frac{|f(z)|}{|z - x|} \prod_{j=0}^n \frac{|x - x_j|}{|z - x_j|} \\ &\leq \frac{L_C}{2\pi} \frac{M}{K} \frac{(b - a)^{n+1}}{K^{n+1}}. \end{aligned}$$

Substituting $L_C = 2\pi K + 2(b - a)$, this formula simplifies to

$$|f(x) - p_n(x)| = \frac{(b - a + \pi K)M}{\pi K} \left(\frac{b - a}{K} \right)^{n+1},$$

and, since $K > b - a$, this bound goes to zero as $n \rightarrow \infty$ independent of our choice of $x \in [a, b]$.

- (d) If $[a, b] = [-5, 5]$, the result requires that f be analytic on D with $k > |b - a| = 10$. Note that D contains the imaginary segment $(-10i, 10i)$, but $f(z) = (1 + z^2)^{-1}$ has poles at $z = \pm i$, so f is not analytic on D , and we cannot apply the bound in (c).

Now we seek an interval $[-\alpha, \alpha]$ over which we can apply the bound in (c). That is, we must select the interval $[-\alpha, \alpha]$ such that the contour C does not touch or enclose either pole $\pm i$, i.e., we must have $K < 1$. Recalling that $K > b - a = 2\alpha$ (to ensure that the $((b - a)/K)^{n+1}$ term goes to zero), we must have $\alpha < 1/2$.

In conclusion, polynomial interpolants to Runge's function for any interpolation points on the interval $[-1/2, 1/2]$ must always converge.

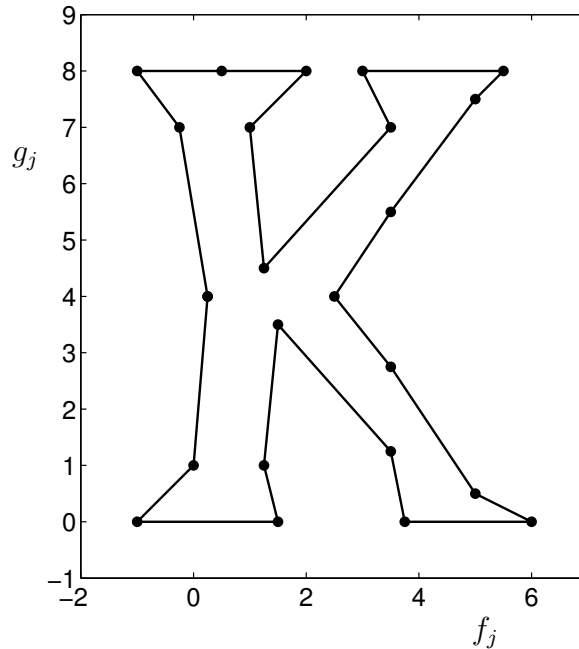
For further details, see: P. J. Davis, *Interpolation and Approximation*, Dover, 1975 (page 82) and the treatise: J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Domain*, 5th ed., American Mathematical Society, 1969.

6. [25 points: (a)=8 points; (b)=8 points; (c)=4 points; (d)=5 points]

Splines in font design. The font in which this text is set was designed by Donald Knuth using his remarkable METAFONT software. To make shapely letters, the font designer establishes fixed points that guide *Beziér curves*, defined via *Bernstein polynomials*. These curves do not interpolate the guide points, but a similar system based on spline functions, which do interpolate, has also been proposed. Here you will try your hand at spline font design: design a stylized ‘K’ character using cubic splines with natural boundary conditions. The craft would be the same if you were designing an airplane fuselage or a new sports car.

Consider the following table of data. The (f_j, g_j) values specify the skeleton for our ‘K’, as shown on the right. Our goal is to replace the straight lines by smooth curves generated from splines.

x_j	f_j	g_j
0	.25	4
1	0	1
2	-1	0
3	1.5	0
4	1.25	1
5	1.5	3.5
6	3.5	1.25
7	3.75	0
8	6	0
9	5	0.5
10	3.5	2.75
11	2.5	4
12	3.5	5.5
13	5	7.5
14	5.5	8
15	3	8
16	3.5	7
17	1.25	4.5
18	1	7
19	2	8
20	0.5	8
21	-1	8
22	-.25	7
23	.25	4



- (a) Write a MATLAB routine

```
function S = cBspline(x, x0, h)
```

that computes the value of a natural cubic B-spline at a point $x \in \mathbb{R}$, given the initial knot $x_0 \in \mathbb{R}$ and uniform grid spacing h , i.e., $f_j = x_0 + jh$.

- (b) Using your code from part (a), or otherwise, construct two *natural* cubic splines, one, called $S_1(x)$, interpolating the (x_j, f_j) values, the other, called $S_2(x)$, interpolating (x_j, g_j) . (Each spline should be the linear combination of $n + 3 = 26$ B-splines. Further details will be provided in the lecture and lecture notes; the variables f_j and g_j are defined in the MATLAB file `Kdata.m` on the class website.)
- (c) Produce a plot showing $S_1(x)$ and $S_2(x)$ for $x \in [0, 23]$, along with the points (x_j, f_j) and (x_j, g_j) , to verify that your splines interpolate the data as desired.
- (d) In a separate figure, plot $(S_1(x), S_2(x))$ for $x \in [0, 23]$. You should obtain a picture like the skeleton above, but with the straight lines replaced by more interesting curves. Superimpose the (f_j, g_j) points to verify that your splines interpolate the data points.

Solution.

```

(a) function Sx = cBspline(x, x0, h)
% return the value of the cubic B-spline B_{0,3}(x)
% assuming xj = x0+j*h

Sx = Bspline(3, x, x0, h);

function Sx = Bspline(k, x, x0, h)
% return the value of the B-spline B_{0,k}(x)
% assuming xj = x0+j*h

if k==0,
    Sx = (x >= x0) & (x < x0+h);
else
    Sx = (x-x0)/(k*h).*Bspline(k-1,x,x0,h) ...
        + (x0+(k+1)*h-x)/(k*h).*Bspline(k-1,x,x0+h,h);
end

(b) % spline curve fit of the letter "k"

xj = [ .25 0 -1 1.5 1 1.5 4 4 6 5 3.5 2.5 3.5 ...
        5 5.5 3.5 3.5 1.25 1 2 .5 -1 -.25 .25]';
yj = [ 4 1 0 0 1 3.5 .5 0 0 .5 2.5 4 5.5 ...
        7.5 8 8 7.25 4.5 7 8 8 8 7 4]';

n = length(xj)-1;
tj = [0:n]; % set up knots, tj = j

% Set up matrix that will determine spline coefficients.
% This matrix will be tridiagonal.
alpha = cBspline(1,0,1);
beta = cBspline(2,0,1);
A = zeros(n+3);
for j=0:n
    A(j+2,j+1:j+3) = [alpha beta alpha];
end
A(1,1:3) = [3 -6 3]; % natural spline boundary conditions
A(end,end-2:end) = [3 -6 3];
cx = A\[0;xj;0]; % solve for spline coefficients
cy = A\[0;yj;0]; % for (tj,xj) and (tj,yj)

% Construct the spline at a fine mesh of points
tt = linspace(0,n,1000)';
Sx = zeros(size(tt));
Sy = zeros(size(tt));
for j=1:n+3
    Sx = Sx + cx(j)*cBspline(tt,j-4,1);
    Sy = Sy + cy(j)*cBspline(tt,j-4,1);
end

% Plot the splines to show interpolations: (t,Sx) (t,Sy)
figure(1), clf
plot(tt, Sx, 'k-', 'linewidth', 2), hold on
plot(tt, Sy, 'k--', 'linewidth', 2)
plot(tj, xj, 'k.', 'markersize', 20)
plot(tj, yj, 'k.', 'markersize', 20)
set(gca, 'fontsize', 16)
legend('S_x(t)', 'S_y(t)', 2)

```

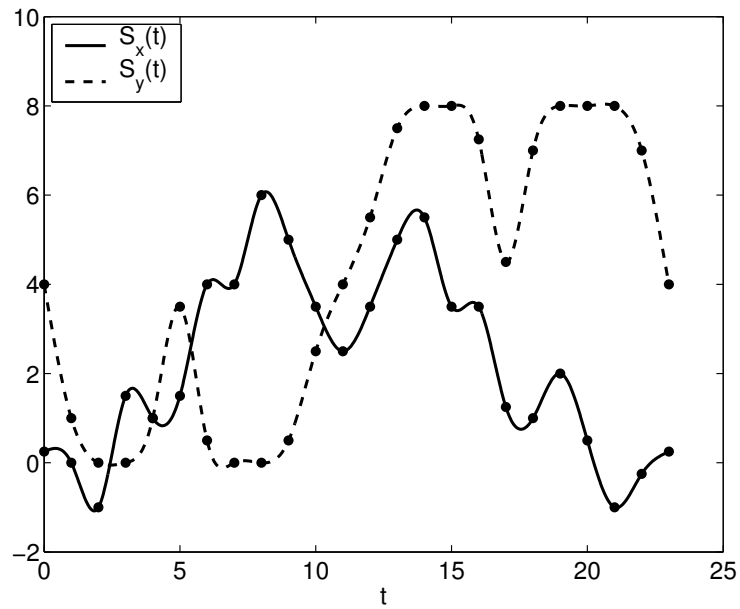
```

xlabel('t')

% Now plot the letter (Sx,Sy)
figure(2), clf
fill(Sx,Sy,[0.8 0.8 0.8]); hold on
plot(Sx,Sy,'b-','linewidth',2); hold on
plot(xj,yj, 'r.','markersize',20), hold on
plot(xj,yj, 'r-','linewidth',1.5)
axis equal
axis([-2 7 -1 9])
set(gca, 'fontsize', 16)

```

(c)



(d)

