MA 580 Assignment 4

Kyle Hansen

31 October 2025

AI Use Statement: AI was used to write the `unpack_index` function in the MATLAB code, which gives exponents of the basis functions for least squares based on the index of flattened 2d data.

Question 1 Exercise 4.2.25

1(a) Image of Unit Circle

Let $A \in \mathbb{R}^{2\times 2}$ with singular values $\sigma_1 \geq \sigma_2 > 0$. Show that the set $\{Ax| \, \|x\|_2 = 1\}$ (the image of the unit circle) is an ellipse in $\mathbb{R}^2$ whose major and minor semiaxes have lengths σ_1 and σ_2 respectively.

Let $\mathcal{S}_2 = \{x| \, \|x\|_2 = 1\}$ be the 2−dimensional unit sphere. Then for every $y \in A(\mathcal{S}_2)$, there is some $x \in \mathcal{S}_2$ such that $y = Ax$. Since $\|x \in \mathcal{S}_2\|_2 = 1$,

$$1 = \|x\|_2^2 = \left\|A^{-1}Ax\right\|_2^2 = \left\|A^{-1}y\right\|_2^2 = \left\|V\Sigma^{-1}U^Ty\right\|_2^2. \tag{1}$$

Since V is an orthonormal matrix (and does not affect magnitude),

$$1 = \cdots = \left\|\Sigma^{-1}U^Ty\right\|_2^2. \tag{2}$$

Then, with $w = U^Ty$ (which is a rotation only),

$$1 = \cdots = \left\|\Sigma^{-1}w\right\|_2^2 = \left(\frac{w_1}{\sigma_1}\right)^2 + \left(\frac{w_2}{\sigma_2}\right)^2. \tag{3}$$

Then w is the set of vectors describing an ellipse with semiaxes σ_1, σ_2, and y describes a rotation of that same ellipse.

1(b) Hyperellipsoid

Let $A \in \mathbb{R}^{m\times n}, m \geq n, \operatorname{rank}(A) = n$. Show that the set $\{Ax| \, \|x\|_2 = 1\}$

is an n-dimensional hyperellipsoid with semiaxes $\sigma_1, \sigma_2, \ldots n$. Notice that the lengths of the longest and shortest semiaxes are maxmag(A) and minmag(A), respectively.

From the previous homework assignment, for $A \in \mathbb{R}^{m\times n}$, $A^\dagger A = I^{n\times n}$, and $AA^\dagger = I^{m\times m}$. Let $\mathcal{S}_n = \{x | \, \|x\|_2 = 1\}$ be the $n-$dimensional unit sphere. Then for every $y \in A(\mathcal{S}_n)$, there is some $x \in \mathcal{S}_n$ such that $y = Ax$. Since $\|xin\mathcal{S}_n\|_2 = 1$,

$$1 = \|x\|_2^2 = \left\|A^\dagger Ax\right\|_2^2 = \left\|A^\dagger y\right\|_2^2 = \left\|V\Sigma^\dagger U^T y\right\|_2^2 \tag{4}$$

since V is an orthonormal matrix (and does not affect magnitude),

$$1 = \cdots = \left\|\Sigma^\dagger U^T y\right\|_2^2. \tag{5}$$

Then, with $w = U^T y$ (which is a rotation only),

$$1 = \cdots = \left\|\Sigma^\dagger w\right\|_2^2 = \left(\frac{w_1}{\sigma_1}\right)^2 + \left(\frac{w_2}{\sigma_2}\right)^2 + \cdots + \left(\frac{w_n}{\sigma_n}\right)^2 + \cancel{(0 \cdot w_{n+1})^2} + \cdots \cancel{(0 \cdot w_m)^2} \tag{6}$$

then $w = U^T y = U^T Ax$ describes a hyperellipsoid with semiaxes $\sigma_1, \ldots, \sigma_n$. Since U^T is only a rotation with no scaling, Ax also describes such an ellipsoid.

Question 2 Stationary Iterative Methods – Invertible

Let $\mathbf{M} \in \mathbb{R}^{n\times n}$. Prove that $(\mathbf{I} - \mathbf{M})$ is invertible with $(\mathbf{I} - \mathbf{M})^{-1} = \sum_{n=0}^{\infty} \mathbf{M}^n$ if and only if $\rho(\mathbf{M}) < 1$. Recall that $\rho(\mathbf{M})$ is the spectral radius of $\mathbf{M}$.

This is a result of the Banach Lemma. Considering the sequence of partial sums

$$S_k = \sum_{l=0}^{k} M^l \tag{7}$$

for some $m > k$, using submultaplicativaty of matrices,

$$\|S_k - S_m\| = \left\|\sum_{l=k+1}^{m} M^l\right\| \leq \sum_{l=k+1}^{m} \left\|M^l\right\| \leq \sum_{l=k+1}^{m} \|M\|^l = \|M\|^{k+1} \sum_{l=0}^{m-k-1} \|M\|^l \tag{8}$$

then

$$\|S_k - S_m\| \leq \|M\|^{k+1} \sum_{l=0}^{m-k-1} \|M\|^l \tag{9}$$

which is a finite geometric series times $\|M\|^{k+1}$, and can be rewritten

$$\|S_k - S_m\| \leq \|M\|^{k+1} \frac{1 - \|M\|^{m-k}}{\|M\|}. \tag{10}$$

As k and m approach ∞, the difference between terms $\|S_k - S_m\|$ approaches zero, and the series is convergent when $\|M\| < 1$, since $\lim_{m\to\infty} \|M\|^{m+1} = 0$, and does not converge when $\|M\| \geq 1$. Since $\rho(M) \leq \|M\|$ for any induced norm, the series does not converge for $\rho \geq 1$.

Then let the sum of the infinite series be $S = \sum_{l=0}^{\infty} M^l = I + \sum_{l=1}^{\infty} M^l$. Then:

$$MS = M\left(\sum_{l=0}^{\infty} M^l\right) = \sum_{l=1}^{\infty} M^l = S - I \tag{11}$$

then

$$\begin{aligned} MS &= S - I && (12)\\ I &= S - MS && (13)\\ I &= (I - M)S && (14)\\ & && (15) \end{aligned}$$

then $S^{-1} = (I - M)$, or $S = (I - M)^{-1}$.

Question 3 Stationary Iterative Methods – Convergence

Prove that for every $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^n$ the iteration

$$\boldsymbol{x}_{k+1} = \mathbf{M}\boldsymbol{x}_k + \boldsymbol{b}, \quad k = 0, 1, 2, \ldots$$

converges to $(\mathbf{I} - \mathbf{M})^{-1}\boldsymbol{b}$ if and only if $\rho(\mathbf{M}) < 1$.

From the previous question, $(I - M)$ is nonsingular (with $(I - M)^{-1} = S = \sum_{l=0}^{\infty} M^l$) for $\rho(M) < 1$, so the solution to $(I - M)x = b$ exists. Let this exact solution be denoted

$$x^\star = (I - M)^{-1} b. \tag{16}$$

Then the error of each iterate is

$$\epsilon_{k+1} = x_{k+1} - x^\star = Mx_k + b - x^\star. \tag{17}$$

Also,

$$\begin{aligned} (I - M)x^\star &= b && (18)\\ x^\star - Mx^\star &= b && (19)\\ x^\star &= b + Mx^\star. && (20) \end{aligned}$$

Then from Equation 17,

$$\begin{aligned}
\epsilon_{k+1} = x_{k+1} - x^\star = Mx_k + \cancel{b - b} - Mx^\star \quad (21)\\
\epsilon_{k+1} = M(x_k) - M(x^\star) \quad (22)\\
\epsilon_{k+1} = M(x_k - x^\star) \quad (23)\\
\epsilon_{k+1} = M\epsilon_k \quad (24)
\end{aligned}$$

Then, from some initial guess x_0 with error ϵ_0,

$$\begin{aligned}
\epsilon_k &= M^k \epsilon_0 \quad (25)\\
\|\epsilon_k\| &= \left\|M^k \epsilon_0\right\| \leq \|M\|^k \cdot \|\epsilon_0\| \quad (26)
\end{aligned}$$

From the previous part, $\lim_{k\to\infty} \|M\|^k = 0$, so $\lim_{k\to\infty} \|\epsilon_k\| = 0$.

Question 4 2D Least-Squares

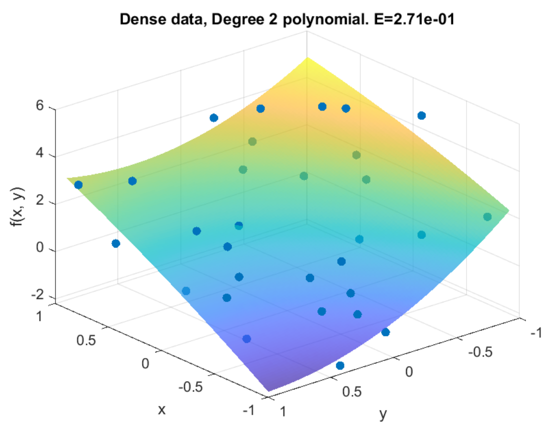
4(a) MATLAB Code Figure 2 and page 7 show the multivariate polynomial least-squares fit for the under- and over-determined cases, respectively. Both cases approximate the given data well with even a low-degree polynomial– the relative squared difference between the data and fit reached a maximum of 0.0179. The most appropriate choice of least-squares fit depends on the application, but the degree-1 may be enough for many cases.

```
load("data3d_validation.mat")
workspaces = ["data3d_dense.mat", "data3d_sparse.mat"]
names = ["Dense", "Sparse"]

for ws = [1,2,3]
    clear x1 y1
    load(workspaces(ws))


    data_length = length(x1);
    Density  = "Sparse";
    [X, Y] = meshgrid(linspace(min(x1), max(x1), 100), linspace(min(x2), max(x2), 100));


    for order = [2, 3, 4]
        dof = (order+1)*(order+2)/2;
```

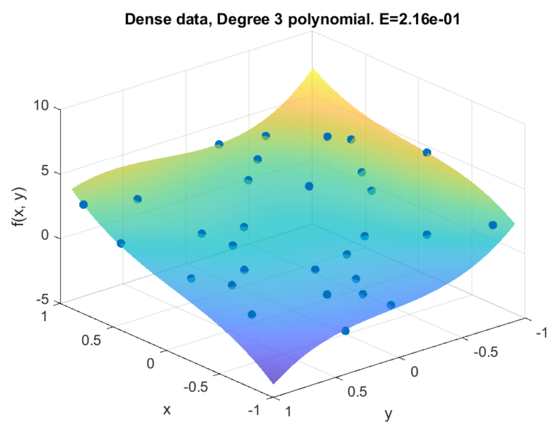
```
        if dof <= data_length
            fprintf("Using QR solver\n")
            c = qr_solve(x1, x2, f, order);
        else
            fprintf("Using SVD pseudoinverse")
            c = min_norm_lsqr(x1, x2, f, order);
        end

        Z = eval_poly(X, Y, c, order);

        figure()
        surf(X, Y, Z, 'FaceAlpha',0.7, LineStyle="none")
        hold on
        scatter3(x1, x2, f, 'filled')
        view(-130.2,32.4)
        xlabel("x")
        ylabel("y")
        zlabel("f(x, y)")
        title(sprintf("%s data, Degree %i polynomial. E=%.2e", names(ws), order, e))
        saveas(gcf, sprintf("%s%i.png", names(ws), order))

        e = norm(eval_poly(x_v, y_v, c, order)-f_v, 2)/norm(f_v, 2);

        figure()
        surf(X, Y, Z, 'FaceAlpha',0.7, LineStyle="none")
        hold on
        scatter3(x_v, y_v, f_v, 'filled')
        view(-130.2,32.4)
        xlabel("x")
        ylabel("y")
        zlabel("f(x, y)")
        title(sprintf("Validation data, %s data, Degree %i polynomial. E=%.2e", names(ws
        saveas(gcf, sprintf("validation_%s%i.png",names(ws), order))

    end

end

function x = qr_solve(x1, x2, f, n)
    m = length(x1);
    x1 = reshape(x1, [m 1]);
    x2 = reshape(x2, [m 1]);
    f = reshape(f, [m 1]);
    A = zeros([m (n+1)*(n+2)/2]);
    %
```

```
    col = 0;
    for i = 0:n
        for j = 0:(n-i)
            col = col + 1;
            for r = 1:m
                A(r, col) = (x1(r)^i)*(x2(r)^j);
            end
        end
    end
    [Q,R] = qr(A, 0);
    z = Q' * f;
    x = R\z;
end

function x = min_norm_lsqr(x1, x2, f, n)
    m = length(x1);
    x1 = reshape(x1, [m 1]);
    x2 = reshape(x2, [m 1]);
    f = reshape(f, [m 1]);
    A = zeros([m (n+1)*(n+2)/2]);
    col = 0;
    for i = 0:n
        for j = 0:(n-i)
            col = col + 1;
            for r = 1:m
                A(r, col) = (x1(r)^i)*(x2(r)^j);
            end
        end
    end
    [U, S, V] = svd(A);
    x = zeros(size(A, 2), 1);
    for r = 1:size(A, 1)
        x = x + (1/S(r,r))*U(:, r)'*f*V(:, r);
    end
end

function [i, j] = unpack_index(r, n)
    % find i by cumulative sum
    S = 0;
    for ii = 0:n
        prevS = S;
        S = S + (n - ii + 1);
        if r <= S
            i = ii;
            j = r - prevS - 1;
```

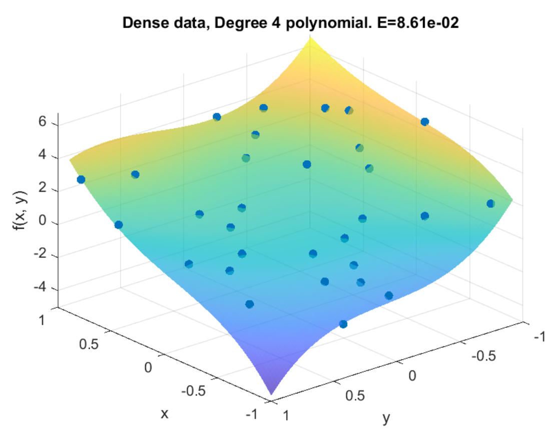
```
            return
        end
    end
end

function Z = eval_poly(X, Y, c, n)
    Z = zeros(size(X));
    for r = 1:((n+1)*(n+2)/2)
        [i, j] = unpack_index(r, n);
        Z = Z + (c(r).*(X.^i).*(Y.^j));
    end
end
```



(a)

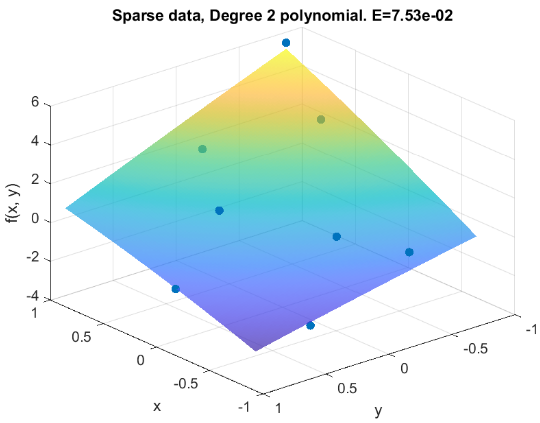


(b)

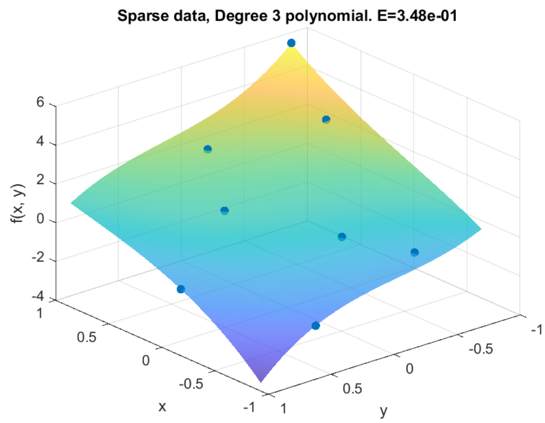


(c)

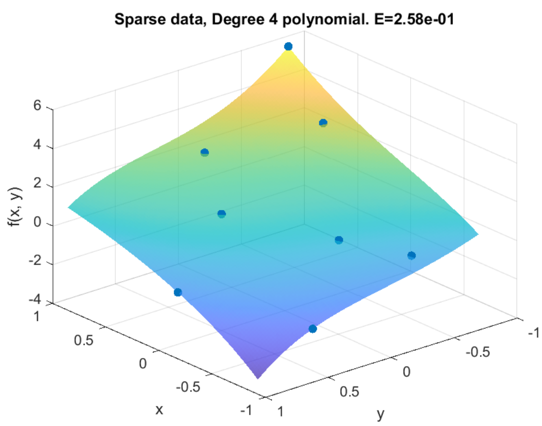
Figure 1: Overdetermined Case



(a)



(b)

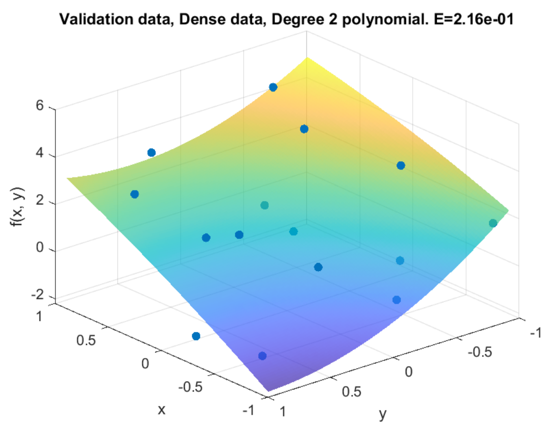


(c)

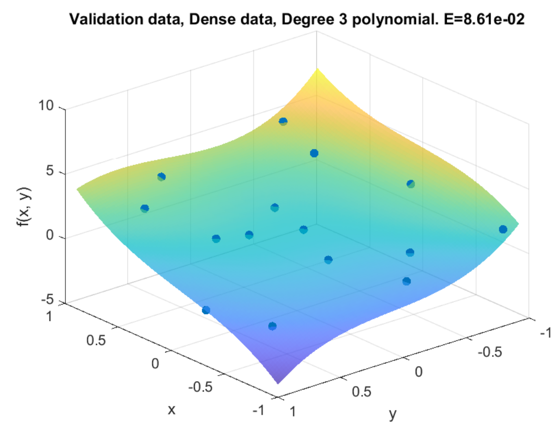
Figure 2: Underdetermined Case

4(b) Validation The overdetermined polynomial fits generally have lower error than the underdetermined, though the errors are close for the degree-2 polynomial. The underdetermined error actually increases with increasing polynomial order, while the overdetermined error does decrease as expected, though the error decreases little from $n = 3$ to $n = 4$ (less than 10 percent), even though there are additional 5 (+50 percent) degrees of freedom in the fit.

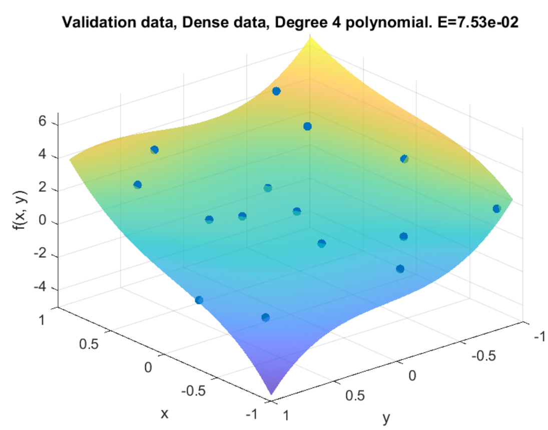
Sparse data may be desirable if only a low-degree fit is needed since it produces a similar approximation with less data needed, but for high-order polynomials the overdetermined case is certainly preferable if available.



(a)

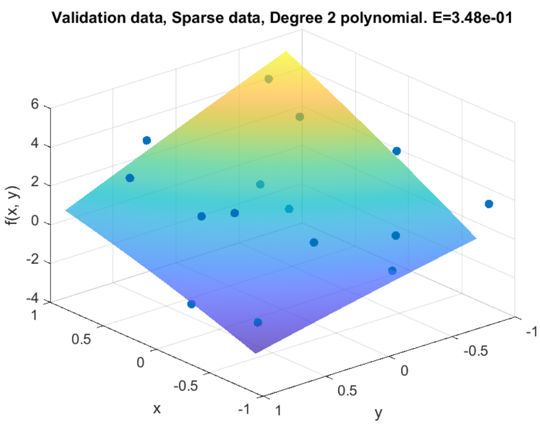


(b)

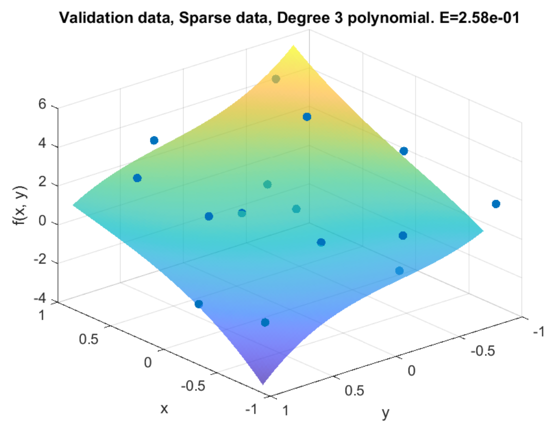


(c)

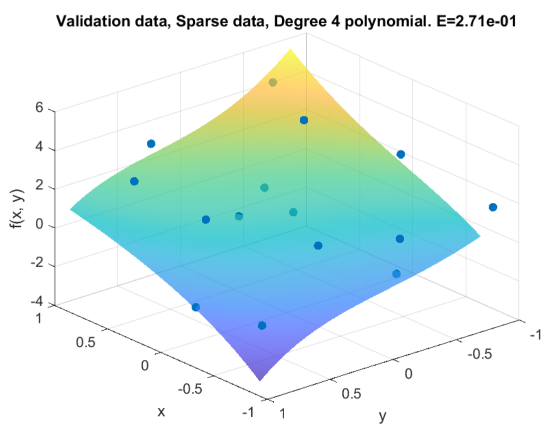
Figure 3: Underdetermined Case



(a)



(b)



(c)

Figure 4: Underdetermined Case