Exercises: Overdetermined Linear Systems (Solutions)

2. (a) Writing down the interpolating conditions, we have

$$\left\{\begin{array}{l} y_1 = p(x_1) \\ y_2 = p(x_2) \\ y_3 = p(x_3) \end{array}\right\} \implies \left\{\begin{array}{l} 3 = c_1 - 1 \cdot c_2 \\ 6 = c_1 + 2 \cdot c_2 \\ 9 = c_1 + 5 \cdot c_2 \end{array}\right\}$$

Using matrix and vector notation, we can succinctly express the system as $\mathbf{y}$ "=" $X\mathbf{c}$:

$$\underbrace{\begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}}_{=:\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & -1 \\ 1 & 2 \\ 1 & 5 \end{bmatrix}}_{=:X} \underbrace{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}_{=:\mathbf{c}} .$$

(b) With X and $\mathbf{y}$ found in the previous part, we write the residual vector as

$$\mathbf{r} = X\mathbf{c} - \mathbf{y} = \begin{bmatrix} c_1 - c_2 - 3 \\ c_1 + 2c_2 - 6 \\ c_1 + 5c_2 - 9 \end{bmatrix},$$

whose squared 2-norm is given by

$$\|\mathbf{r}\|_2^2 = (c_1 - c_2 - 3)^2 + (c_1 + 2c_2 - 6)^2 + (c_1 + 5c_2 - 9)^2 =: g(c_1, c_2) .$$

(c) Setting $\nabla g = \mathbf{0}$, we obtain the following two equations:

$$\frac{\partial g}{\partial c_1} = 2(c_1 - c_2 - 3) + 2(c_1 + 2c_2 - 6) + 2(c_1 + 5c_2 - 9) = 0$$
$$\frac{\partial g}{\partial c_2} = -2(c_1 - c_2 - 3) + 4(c_1 + 2c_2 - 6) + 10(c_1 + 5c_2 - 9) = 0$$

Dividing both equations by 2, collecting like terms, and moving constants to the other side of equations, we obtain

$$\left\{\begin{array}{l} 3c_1 + \ \ 6c_2 = 18 \\ 6c_1 + 30c_2 = 54 \end{array}\right.$$

which, in turn, is written as a matrix equation

$$\begin{bmatrix} 3 & 6 \\ 6 & 30 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 18 \\ 54 \end{bmatrix} . \qquad (☺)$$

The problem never asked to solve for $\mathbf{c}$, so you may just stop here.

(d) By simple matrix calculation,

$$X^{\mathrm{T}}X = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 2 & 5 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ 1 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 30 \end{bmatrix} \quad \text{and} \quad X^{\mathrm{T}}\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 2 & 5 \end{bmatrix} \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix} = \begin{bmatrix} 18 \\ 54 \end{bmatrix}.$$

Thus, we verify that the normal equation $X^{\mathrm{T}}X\mathbf{c} = X^{\mathrm{T}}\mathbf{y}$ is indeed the same as the matrix equation ☺ obtained in the previous part. How nice!

3. See the function `lsqrfact` in Lecture 21.

4. Begin by constructing the matrix and the vector as described.

```
A = reshape(1:40, 10, 4);
x = ( [1:10].^2 )';
```

Now we want to decompose $\mathbf{x}$ into $\mathbf{u} + \mathbf{z}$, where $\mathbf{u}$ lies in $\mathcal{R}(A)$ and $\mathbf{z}$ is perpendicular to $\mathbf{u}$ and thus lies in $\mathcal{R}^{\perp}(A)$. This is achieved when $\mathbf{u}$ is an orthogonal projection of $\mathbf{x}$ onto $\mathcal{R}(A)$.

To this end, it is the most convenient to work with an orthonormal basis of the range of A rather than working with the columns of A. So let's grab one by doing Gram-Schmidt on columns of A.

Thankfully, we do not need to do this by hand and thus we will use the thin QR factorization of $A = \widehat{Q}\widehat{R}$; the columns of $\widehat{Q}$ form an orthogonal basis of $\mathcal{R}(A)$. For the sake of simplicity, we will simply call them `Q` and `R` in the code below.

```
[Q,~] = qr(A,0); % R is not needed, but we still need a placeholder.
```

The projection of $\mathbf{x}$ onto the range of A is now neatly written as

$$\mathbf{u} = (\mathbf{q}_1^{\mathrm{T}}\mathbf{x})\mathbf{q}_1 + (\mathbf{q}_2^{\mathrm{T}}\mathbf{x})\mathbf{q}_2 + (\mathbf{q}_3^{\mathrm{T}}\mathbf{x})\mathbf{q}_3 + (\mathbf{q}_4^{\mathrm{T}}\mathbf{x})\mathbf{q}_4 = \sum_{j=1}^{4} (\mathbf{q}_j^{\mathrm{T}}\mathbf{x})\mathbf{q}_j.$$

Once it is computed, $\mathbf{z}$ is found simply by

$$\mathbf{z} = \mathbf{x} - \mathbf{u}.$$

In MATLAB:

```
u = zeros(size(x));
for j = 1:4
    u = u + (Q(:,j)'*x)*Q(:,j);
end
z = x - u;
```

To confirm, try

```
u'*z     % are u and z indeed orthogonal?
```

If the outcome is small, `u` and `z` are nearly orthogonal numerically.

Alternately, observe that

$$\mathbf{z} = \mathbf{x} - \mathbf{u} = \left(I - \widehat{Q}\widehat{Q}^{\mathrm{T}}\right)\mathbf{x}.$$

So we can write a more compact code as

```
z1 = (eye(10) - Q*Q')*x;
u1 = x - z1;
u1'*z1
```

How close are the results to the previous ones?

```
norm(u-u1)
norm(z-z1)
```