1. **Price elasticity of demand.** The demand for $n$ different goods is a function of their prices:

$$q = f(p),$$

where $p$ is the price vector, $q$ is the demand vector, and $f : \mathbb{R}^n \to \mathbb{R}^n$ is the demand function. The current price and demand are denoted $p^*$ and $q^*$, respectively. Now suppose there is a small price change $\delta p$, so $p = p^* + \delta p$. This induces a change in demand, to $q \approx q^* + \delta q$, where

$$\delta q \approx Df(p^*)\delta p,$$

where $Df$ is the derivative or Jacobian of $f$, with entries

$$Df(p^*)_{ij} = \frac{\partial f_i}{\partial p_j}(p^*).$$

This is usually rewritten in terms of the **elasticity matrix** $E$, with entries

$$E_{ij} = \frac{\partial f_i}{\partial p_j}(p^*) \frac{1/q_i^*}{1/p_j^*},$$

so $E_{ij}$ gives the relative change in demand for good $i$ per relative change in price $j$. Defining the vector $y$ of relative demand changes, and the vector $x$ of relative price changes,

$$y_i = \frac{\delta q_i}{q_i^*}, \quad x_j = \frac{\delta p_j}{p_j^*},$$

we have the linear model $y = Ex$.

Here are the questions:

a) What is a reasonable assumption about the diagonal elements $E_{ii}$ of the elasticity matrix?

b) Goods $i$ and $j$ are called *substitutes* if they provide a similar service or other satisfaction (*e.g.*, train tickets and bus tickets, cake and pie, etc.). They are called *complements* if they tend to be used together (*e.g.*, automobiles and gasoline, left and right shoes, etc.). For each of these two generic situations, what can you say about $E_{ij}$ and $E_{ji}$?

c) Suppose the price elasticity of demand matrix for two goods is

$$E = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}.$$  

Describe the nullspace of $E$, and give an interpretation (in one or two sentences). What kind of goods could have such an elasticity matrix?
Solution.

a) The $i$th diagonal entry of $E$ relates $y_i$ to $x_i$, i.e., the demand for the $i$th good to its price. When the price of a product is increased, and all other prices are held constant, the demand for that product can be expected to decrease. Hence, the diagonal elements of $E$ should be negative. (Whether any good with a positive elasticity exists at all is a debated question, but most economists’ answer is no.)

b) The entry $E_{ij}$ relates the demand for good $i$ to the price of good $j$. A price increase in good $j$ leads to a decreased demand for that good. If good $i$ is a substitute, it also leads to an increased demand for good $i$, as some of the consumption switches to what now seems a more attractive price. Hence, $E_{ij}$ is positive. The same argument tells us that $E_{ji}$ is positive when goods $i$ and $j$ are substitutes.

If the goods are complements, the converse is true. Since the consumption of good $i$ is associated with consumption of good $j$, a decreased demand for good $i$ follows from the decreased demand for good $j$. Hence, $E_{ij}$ (and $E_{ji}$) is negative.

c) The nullspace consists of the vectors of the form

$$x = \alpha \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

with the scaling factor $\alpha$ any real number. If the two prices are changed by an equal amount, with one being increased and the other decreased, the demand is not affected at all. This can happen if the two goods are perfect complements, that is, if they are always consumed in fixed proportions (such as right and left shoes!) In this case, consumers only care about the total cost of the two goods, not the individual prices.

2. Halfspace. Suppose $a, b \in \mathbb{R}^n$ are two given points. Show that the set of points in $\mathbb{R}^n$ that are closer to $a$ than $b$ is a halfspace, i.e.:

$$\{ x \mid \|x - a\| \leq \|x - b\| \} = \{ x \mid c^T x \leq d \}$$

for appropriate $c \in \mathbb{R}^n$ and $d \in \mathbb{R}$. Give $c$ and $d$ explicitly, and draw a picture showing $a$, $b$, $c$, and the halfspace.

Solution. It is easy to see geometrically what is going on: the hyperplane that goes right between $a$ and $b$ splits $\mathbb{R}^n$ into two parts; the points closer to $a$ (than $b$) and the points closer to $b$ (than $a$). More precisely, the hyperplane is normal to the line through $a$ and $b$, and intersects that line at the midpoint between $a$ and $b$. Now that we have the idea, let’s try to derive it algebraically. Let $x$ belong to the set of points in $\mathbb{R}^n$ that are closer to $a$ than $b$. Therefore $\|x - a\| < \|x - b\|$ or $\|x - a\|^2 < \|x - b\|^2$ so

$$(x - a)^T (x - a) < (x - b)^T (x - b).$$

Expanding the inner products gives

$$x^T x - x^T a - a^T x + a^T a < x^T x - x^T b - b^T x + b^T b$$

2
or

\[-2a^T x + a^T a < -2b^T x + b^T b\]

and finally

\[(b - a)^T x < \frac{1}{2} (b^T b - a^T a). \tag{1}\]

Thus (1) is in the form \(c^T x < d\) with \(c = b - a\) and \(d = \frac{1}{2} (b^T b - a^T a)\) and therefore we have shown that the set of points in \(\mathbb{R}^n\) that are closer to \(a\) than \(b\) is a halfspace. Note that the hyperplane \(c^T x = d\) is perpendicular to \(c = b - a\).

3. **Linearizing range measurements.** Consider a single (scalar) measurement \(y\) of the distance or range of \(x \in \mathbb{R}^n\) to a fixed point or beacon at \(a\), i.e., \(y = \|x - a\|\).

a) Show that the linearized model near \(x_0\) can be expressed as \(\delta y = k^T \delta x\), where \(k\) is the unit vector (i.e., with length one) pointing from \(a\) to \(x_0\). Derive this analytically, and also draw a picture (for \(n = 2\)) to demonstrate it.

b) Consider the error \(e\) of the linearized approximation, i.e.,

\[e = \|x_0 + \delta x - a\| - \|x_0 - a\| - k^T \delta x.\]

The relative error of the approximation is given by \(\eta = e / \|x_0 - a\|\). We know, of course, that the absolute value of the relative error is very small provided \(\delta x\) is small. In many specific applications, it is possible and useful to make a stronger statement, for example,
to derive a bound on how large the error can be. You will do that here. In fact you will prove that

$$0 \leq \eta \leq \frac{\alpha^2}{2}$$

where \(\alpha = \|\delta x\|/\|x_0 - a\|\) is the relative size of \(\delta x\). For example, for a relative displacement of \(\alpha = 1\%\), we have \(\eta \leq 0.00005\), i.e., the linearized model is accurate to about 0.005\%. To prove this bound you can proceed as follows:

- Show that \(\eta = -1 + \sqrt{1 + \alpha^2 + 2\beta} - \beta\) where \(\beta = k^T\delta x/\|x_0 - a\|\).
- Verify that \(|\beta| \leq \alpha\).
- Consider the function \(g(\beta) = -1 + \sqrt{1 + \alpha^2 + 2\beta} - \beta\) with \(|\beta| \leq \alpha\). By maximizing and minimizing \(g\) over the interval \(-\alpha \leq \beta \leq \alpha\) show that \(0 \leq \eta \leq \frac{\alpha^2}{2}\).

**Solution.**

a) For the linearized model we have

$$\delta y = \left(\frac{\partial y}{\partial x}\right) \delta x$$

so all we have to do is to compute the matrix \(\partial y/\partial x\). Since \(y = \|x - a\|\) we have \(y^2 = (x - a)^T(x - a)\) and differentiating both sides with respect to \(x\) gives

$$2\frac{\partial y}{\partial x} y = 2(x - a)^T$$

and therefore

$$\frac{\partial y}{\partial x} = \frac{(x - a)^T}{y} = \frac{(x - a)^T}{\|x - a\|}$$

so \(\delta y = k^T\delta x\) with \(k = (x - a)/\|x - a\|\). Clearly, \(k\) points from \(a\) to \(x\) and is of length one since

$$k^Tk = \frac{(x - a)^T(x - a)}{\|x - a\|^2} = 1.$$
b) First we show that \( \eta = -1 + \sqrt{1 + \alpha^2 + 2\beta} - \beta \) where \( \beta = k^T \delta x / \| x_0 - a \| \). Note that

\[
e = \| x_0 + \delta x - a \| - \| x_0 - a \| - k^T \delta x
= \| x_0 - a \| \left( \frac{x_0 - a}{\| x_0 - a \|} + \frac{\delta x}{\| x_0 - a \|} - 1 + \frac{k^T \delta x}{\| x_0 - a \|} \right),
\]

and after dividing both sides by \( \| x_0 - a \| \) and using \( k = (x_0 - a)/\| x_0 - a \| \), \( \beta = k^T \delta x / \| x_0 - a \| \) and \( \eta = e / \| x_0 - a \| \) we get

\[
\eta = \| k + \frac{\delta x}{\| x_0 - a \|} \| - 1 - \beta.
\]

But

\[
\| k + \frac{\delta x}{\| x_0 - a \|} \| = \sqrt{\left( \frac{k}{\| x_0 - a \|} + \frac{\delta x}{\| x_0 - a \|} \right)^T \left( \frac{k}{\| x_0 - a \|} + \frac{\delta x}{\| x_0 - a \|} \right)}
= \sqrt{\| k \|^2 + 2 \frac{k^T \delta x}{\| x_0 - a \|} + \| \delta x \|^2 / \| x_0 - a \|^2}.
\]

Since \( \| k \| = 1 \) and by substituting the values for \( \alpha \) and \( \beta \) we have

\[
k + \frac{\delta x}{\| x_0 - a \|} = \sqrt{1 + 2\beta + \alpha^2}.
\]

Therefore (??) can be written as

\[
\eta = \sqrt{1 + 2\beta + \alpha^2} - 1 - \beta.
\]

It is easy to see that \( |\beta| \leq \alpha \). Simply we can use the Cauchy-Schwarz inequality for the vectors \( k \) and \( \delta x / \| x_0 - a \| \), i.e.,

\[
\left| \frac{k^T \delta x}{\| x_0 - a \|} \right| \leq \| k \| \frac{\| \delta x \|}{\| x_0 - a \|}
\]

and since \( \| k \| = 1 \) we immediately get \( |\beta| \leq \alpha \).

At this point, all we need to do to derive a bound on how large the error can be is to maximize and minimize the function \( g(\beta) = \sqrt{1 + 2\beta + \alpha^2} - 1 - \beta \) over the interval \( |\beta| \leq \alpha \) or \( -\alpha \leq \beta \leq \alpha \). The maximum or minimum of a smooth function \( g(\beta) \) over a given interval \( (-\alpha \leq \beta \leq \alpha) \) can only occur at the endpoints of the interval \( (\beta = \pm \alpha) \) or at the extremums (points \( \beta \) with \( g'(\beta) = 0 \)). For \( g(\beta) \) we have:

- **Value at endpoint \( \beta = \alpha \).**

  \[
g(\alpha) = \sqrt{1 + 2\alpha + \alpha^2} - 1 - \alpha
  = \sqrt{(1 + \alpha)^2} - 1 - \alpha
  = 1 + \alpha - 1 - \alpha
  = 0.
\]
4. Orthogonal complement of a subspace. If $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ we define $\mathcal{V}^\perp$ as the set of vectors orthogonal to every element in $\mathcal{V}$, i.e.,

$$\mathcal{V}^\perp = \{ x \mid \langle x, y \rangle = 0, \forall y \in \mathcal{V} \}.$$  

a) Verify that $\mathcal{V}^\perp$ is a subspace of $\mathbb{R}^n$. 

- Value at endpoint $\beta = -\alpha$.

$$g(-\alpha) = \sqrt{1 - 2\alpha + \alpha^2} - 1 - \alpha$$

$$= \sqrt{(1 - \alpha)^2} - 1 + \alpha$$

$$= |1 - \alpha| - (1 - \alpha)$$

$$= \begin{cases} 
0 & 0 \leq \alpha \leq 1 \\
2(\alpha - 1) & \alpha > 1.
\end{cases}$$

Therefore $g(-\alpha) \geq 0$ for all $\alpha$ because $2(\alpha - 1) > 0$ for $\alpha > 1$.

- Extremum value.

$$g'(\beta) = \frac{1}{\sqrt{1 + 2\beta + \alpha^2}} - 1.$$

Setting $g'(\beta) = 0$ we get $\sqrt{1 + 2\beta + \alpha^2} = 1$ or $1 + 2\beta + \alpha^2 = 1$ and therefore $\beta_{\text{ex.}} = -\alpha^2/2$. The function value at the extremum $\beta_{\text{ex.}} = -\alpha^2/2$ is

$$g(\beta_{\text{ex.}}) = \sqrt{1 - \alpha^2 + \alpha^2} - 1 + \frac{\alpha^2}{2}$$

$$= \frac{\alpha^2}{2}.$$ 

Clearly, $g(\beta) \geq 0$ for all $\beta$ satisfying $|\beta| \leq \alpha$ because the value of $g(\beta)$ at the endpoints $\beta = \pm \alpha$ and at the extremum $\beta = \alpha^2/2$ are all non-negative. Thus we have achieved the lower bound on the relative error $\eta$, i.e., we have shown that $\eta \geq 0$. For the upper bound we need to be a bit more careful. The upper bound we get is either $g(\alpha)$, $g(-\alpha)$ or $g(\beta_{\text{ex.}})$. First note that $g(\alpha) = 0$ is always less than or equal to $g(\beta_{\text{ex.}}) = \alpha^2/2 \geq 0$ so the choice of $g(\alpha)$ is immediately ruled out as the maximum of $g$. Now consider $g(-\alpha)$ and $g(\beta_{\text{ex.}})$. For $0 \leq \alpha \leq 1$ we obviously have $g(\beta_{\text{ex.}}) \geq g(-\alpha) = 0$. For $\alpha > 1$ we also have $g(\beta_{\text{ex.}}) \geq g(-\alpha) = 2(\alpha - 1)$ because $\alpha^2/2 \geq 2(\alpha - 1)$ is equivalent to $\alpha^2 - 4\alpha + 4 \geq 0$ which is true since $\alpha^2 - 4\alpha + 4 = (\alpha - 2)^2$ is a complete square. Thus, we achieve an upper bound on $g(\beta)$ for all $\beta$ satisfying $|\beta| \leq \alpha$ as $g(\beta) \leq \alpha^2/2$. Therefore we have shown that $\eta \leq \alpha^2/2$ and we are done.

(Note: when $\beta_{\text{ex.}}$ falls outside the interval $\beta \leq |\alpha|$, it is possible to achieve a tighter upper bound for $g$. In this case, the maximum of $g$ over $\beta \leq |\alpha|$ is obtained at the endpoint $\beta = -\alpha/2$. The extremum $\beta_{\text{ex.}} = -\alpha^2/2$ falls outside $\beta \leq |\alpha|$ when $\alpha^2/2 > \alpha$ or $\alpha > 2$. Therefore, a tighter upper bound on $\eta$ for $\alpha > 2$ becomes $\eta \leq g(-\alpha) = 2(\alpha - 1).$)
b) Suppose $\mathcal{V}$ is described as the span of some vectors $v_1, v_2, \ldots, v_r$. Express $\mathcal{V}$ and $\mathcal{V}^\perp$ in terms of the matrix $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_r \end{bmatrix} \in \mathbb{R}^{n \times r}$ using common terms (range, nullspace, transpose, etc.)

c) Show that every $x \in \mathbb{R}^n$ can be expressed uniquely as $x = v + v^\perp$ where $v \in \mathcal{V}$, $v^\perp \in \mathcal{V}^\perp$. 

*Hint:* let $v$ be the projection of $x$ on $\mathcal{V}$.

d) Show that $\dim \mathcal{V}^\perp + \dim \mathcal{V} = n$.

e) Show that $\mathcal{V} \subseteq \mathcal{U}$ implies $\mathcal{U}^\perp \subseteq \mathcal{V}^\perp$.

**Solution.**

a) We do not need to check all properties of a vector space to hold for $\mathcal{V}^\perp$, since many of them hold only because $\mathcal{V}^\perp \subseteq \mathbb{R}^n$ and the vector sum and scalar product definitions over $\mathcal{V}^\perp$ and $\mathbb{R}^n$ are the same. We only need to verify the following properties:

- $0 \in \mathcal{V}^\perp$.
- $\forall x_1, x_2 \in \mathcal{V}^\perp : x_1 + x_2 \in \mathcal{V}^\perp$.
- $\forall \alpha \in \mathbb{R}, \forall x \in \mathcal{V}^\perp : \alpha x \in \mathcal{V}^\perp$.

The first property comes from the fact that $\langle 0, y \rangle = 0$ for all $y \in \mathbb{R}^n$ and therefore $0 \in \mathcal{V}^\perp$. To verify the second property, we pick two arbitrary elements $x_1$ and $x_2$ in $\mathcal{V}^\perp$ and show that $x_1 + x_2 \in \mathcal{V}^\perp$. Let $y$ be any vector in $\mathbb{R}^n$. We have

$$
\langle x_1 + x_2, y \rangle = \langle x_1, y \rangle + \langle x_2, y \rangle = 0 + 0 \quad \text{(since } x_1 \in \mathcal{V}^\perp \text{ and } x_2 \in \mathcal{V}^\perp) = 0,
$$

and therefore $x_1 + x_2 \in \mathcal{V}^\perp$. Finally, we show that if $\alpha \in \mathbb{R}$ and $x \in \mathcal{V}^\perp$ then $\alpha x \in \mathcal{V}^\perp$. If $y \in \mathbb{R}^n$ is arbitrary

$$
\langle \alpha x, y \rangle = \alpha \langle x, y \rangle = \alpha \cdot 0 \quad \text{(since } x \in \mathcal{V}^\perp) = 0,
$$

which by definition of $\mathcal{V}^\perp$, proves that $\alpha x \in \mathcal{V}^\perp$ and we are done.

b) Expressing $\mathcal{V}$ in terms of the matrix $V$ is easy. The span of vectors $v_1, v_2, \ldots, v_r$ is simply all linear combinations of the columns of $V$ and therefore $\mathcal{V} = \text{range}(V)$. To express $\mathcal{V}^\perp$ in terms of $V$ we use the trivial fact that $x \in \mathcal{V}^\perp$ if and only if $x \perp v_i$ for $i = 1, \ldots, r$. (If $x \perp v_i$ then $x$ is orthogonal to any linear combination of the $v_i$’s and hence any element in $\mathcal{V}^\perp$. If $x \in \mathcal{V}^\perp$ then $x$ is specially orthogonal to the vectors $v_i \in \mathcal{V}^\perp$ for $i = 1, \ldots, r$.) Therefore $x \in \mathcal{V}^\perp$ if and only if $v_i^\top x = 0$ for $i = 1, \ldots, r$. In other words, using matrix notation, $x \in \mathcal{V}^\perp$ if and only if

$$
\begin{bmatrix} v_1^\top \\ v_2^\top \\ \vdots \\ v_r^\top \end{bmatrix} x = 0
$$
or \( V^T x = 0 \). Therefore \( \mathcal{V}^\perp = \text{null}(V^T) \).

c) Suppose that \( w_1, w_2, \ldots, w_k \) is an orthonormal basis for \( \mathcal{V} \). Consider the projection of \( x \) on \( \mathcal{V} \), i.e.,

\[
v := (w_1^T x)w_1 + (w_2^T x)w_2 + \cdots + (w_k^T x)w_k.
\]

Clearly, \( v \in \mathcal{V} \) because it is a linear combination of the basis vectors \( w_i \). Now we show that \( x - v \) (projection error) is an element in \( \mathcal{V}^\perp \). To do this we only have to verify that \( x - v \perp w_i \) or \( w_i^T (x - v) = 0 \) for \( i = 1, \ldots, k \). This is easy because

\[
w_i^T (x - v) = w_i^T x - w_i^T v = w_i^T x - (w_i^T x)w_i = 0
\]

since \( w_i^T w_j = 0 \) for \( i \neq j \) and \( w_i^T w_i = 1 \).

Now that \( x - v \in \mathcal{V}^\perp \), define \( v^\perp \in \mathcal{V}^\perp \) as \( v^\perp = x - v \) so \( x = v + v^\perp \) with \( v \in \mathcal{V} \) and \( v^\perp \in \mathcal{V}^\perp \). Now we show that the decomposition \( x = v + v^\perp \) is unique. Suppose that there are two ways to express \( x \) as the sum of elements in \( \mathcal{V} \) and \( \mathcal{V}^\perp \), i.e., \( x = v_1 + v_1^\perp \) and \( x = v_2 + v_2^\perp \) where \( v_1, v_2 \in \mathcal{V} \) and \( v_1^\perp, v_2^\perp \in \mathcal{V}^\perp \). Therefore \( v_1 + v_1^\perp = v_2 + v_2^\perp \) or \( v_1 - v_2 = v_1^\perp - v_2^\perp \). But \( v_1 - v_2 \in \mathcal{V} \) (because \( v_1, v_2 \in \mathcal{V} \)) and \( v_1^\perp - v_2^\perp \in \mathcal{V}^\perp \) (because \( v_1, v_2 \in \mathcal{V}^\perp \)), and by definition of \( \mathcal{V}^\perp \) we should have \( (v_1 - v_2) \perp (v_1^\perp - v_2^\perp) \) or \( (v_1 - v_2)^T (v_1^\perp - v_2^\perp) = 0 \). Now since \( v_1 - v_2 = v_1^\perp - v_2^\perp \) this implies that

\[
(v_1 - v_2)^T (v_1 - v_2) = \|v_1 - v_2\|^2 = 0
\]

and

\[
(v_1^\perp - v_2^\perp)^T (v_1^\perp - v_2^\perp) = \|v_1^\perp - v_2^\perp\|^2 = 0
\]

so \( v_1 = v_2 \) and \( v_1^\perp = v_2^\perp \) or the decomposition is unique.

d) This follows from the previous part. In part (??) we showed that any vector in \( \mathbb{R}^n \) can be expressed as the sum of two elements in \( \mathcal{V} \) and \( \mathcal{V}^\perp \). Therefore, if \( \{w_i\}_{i=1}^k \) is a basis for \( \mathcal{V} \) and \( \{u_i\}_{i=1}^l \) is a basis for \( \mathcal{V}^\perp \), for arbitrary \( x \in \mathbb{R}^n \) the scalars \( \alpha_i \) and \( \beta_i \) exist such that

\[
x = \sum_{i=1}^k \alpha_i w_i + \sum_{i=1}^l \beta_i u_i
\]

or the set of vectors \( \{w_1, \ldots, w_k, u_1, \ldots, u_l\} \) span \( \mathbb{R}^n \). In fact, the vectors \( w_i \) for \( i = 1, \ldots, k \) are orthogonal to the vectors \( u_i \) for \( i = 1, \ldots, l \) by the definition of \( \mathcal{V}^\perp \) and are therefore independent. Since the set of vectors \( \{w_1, \ldots, w_k, u_1, \ldots, u_l\} \) span \( \mathbb{R}^n \) and \( w_1, \ldots, w_k, u_1, \ldots, u_l \) are independent we get

\[
\dim \mathcal{V} + \dim \mathcal{V}^\perp = k + l = n.
\]

e) To show that \( \mathcal{U}^\perp \subseteq \mathcal{V}^\perp \) we take an arbitrary element \( x \in \mathcal{U}^\perp \) and prove that \( x \in \mathcal{V}^\perp \). Since \( x \in \mathcal{U}^\perp \) then \( x \perp y \) for all \( y \in \mathcal{U} \). But \( \mathcal{V} \subseteq \mathcal{U} \) so we also have \( x \perp y \) for all \( y \in \mathcal{V} \). By definition of \( \mathcal{V}^\perp \), this is nothing but to state that \( x \in \mathcal{V}^\perp \) and we are done.
5. Single sensor failure detection and identification. We have $y = Ax$, where $A \in \mathbb{R}^{m \times n}$ is known, and $x \in \mathbb{R}^n$ is to be found. Unfortunately, up to one sensor may have failed (but you don’t know which one has failed, or even whether any has failed). You are given $\tilde{y}$ and not $y$, where $\tilde{y}$ is the same as $y$ in all entries except, possibly, one (say, the $k$th entry). If all sensors are operating correctly, we have $y = \tilde{y}$. If the $k$th sensor fails, we have $\tilde{y}_i = y_i$ for all $i \neq k$.

The file `one_bad_sensor.m`, available on the course web site, defines $A$ and $\tilde{y}$ (as $\mathbf{A}$ and $\tilde{y}$). Determine which sensor has failed (or if no sensors have failed). You must explain your method, and submit your code.

For this exercise, you can use the `matlab` code `rank([F g]) == rank(F)` to check if $g \in \text{range}(F)$. (We will see later a much better way to check if $g \in \text{range}(F)$.)

**Solution.** Let $y^{(i)}$ be the measurement vector $y$ with the $i$th entry removed. Likewise, let $A^{(i)}$ be the measurement matrix with the $i$th row of $A$ removed. This corresponds to the system without the $i$th sensor.

If the $i$th sensor is faulty, we will almost surely have $y \notin \text{range}(A)$ (unless the sensor failure happens to give the same response $y_i$ as that predicted by $A$, which is highly unlikely). However, once we remove its faulty measurement, we will certainly have $y^{(i)} \in \text{range}(A^{(i)})$.

To test if a vector $z$ is in $\text{range}(C)$, we can use `matlab` and compare `rank([C z]) == rank(C)`. If they are equal, $z \in \text{range}(C)$. Otherwise `rank([C z]) == rank(C) + 1`. To find a faulty sensor, we remove one row of $A$ at a time, and use the above test.

The following `matlab` code solves the problem

```matlab
one_bad_sensor
for k=1:m
    withoutk=[1:k-1 k+1:m];
    Atent = A(withoutk,:);
    ytent = ytilde(withoutk);
    if rank([ Atent ytent ]) == rank(Atent)
        k
    end
end
```

The 11th sensor is faulty.

6. Reverse engineering a smoothing filter. A smoothing filter takes an input vector $u \in \mathbb{R}^n$ and produces an output vector $y \in \mathbb{R}^n$. (We will assume that $n \geq 3$.) The output $y$ is obtained as the minimizer of the objective

$$ J = J^{\text{track}} + \lambda J^{\text{norm}} + \mu J^{\text{cont}} + \kappa J^{\text{smooth}}, $$

where $\lambda$, $\mu$, and $\kappa$ are positive constants (weights), and

$$ J^{\text{track}} = \sum_{i=1}^{n} (u_i - y_i)^2, \quad J^{\text{norm}} = \sum_{i=1}^{n} y_i^2 $$
are the tracking error and norm-squared of \( y \), respectively, and
\[
J^{\text{cont}} = \sum_{i=2}^{n} (y_i - y_{i-1})^2, \quad J^{\text{smooth}} = \sum_{i=2}^{n-1} (y_{i+1} - 2y_i + y_{i-1})^2
\]
are measures of the continuity and smoothness of \( y \), respectively.

Here is the problem: You have access to one input-output pair, i.e., an input \( u \), and the associated output \( y \). Your goal is to find the weights \( \lambda \), \( \mu \), and \( \kappa \). In other words, you will reverse engineer the smoothing filter, working from an input-output pair.

a) Explain how to find \( \lambda \), \( \mu \), and \( \kappa \). (You do not need to worry about ensuring that these are positive; you can assume this will occur automatically.)

b) Carry out your method on the data found in \texttt{rev	extunderscore eng	extunderscore smooth	extunderscore data.m}. Give the values of the weights.

\textbf{Solution.} We first define matrices \( D_1 \in \mathbb{R}^{(n-1) \times n} \) and \( D_2 \in \mathbb{R}^{(n-2) \times n} \) as
\[
D_1 = \begin{bmatrix}
-1 & 1 & 0 & \cdots & 0 \\
0 & -1 & 1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & -1 & 1
\end{bmatrix}, \quad D_2 = \begin{bmatrix}
1 & -2 & 1 & 0 & \cdots & 0 \\
0 & 1 & -2 & 1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & 1 & -2 & 1
\end{bmatrix}.
\]

Multiplying a vector by these matrices gives the first and second differences:
\[
(D_1 y)_i = y_{i+1} - y_i, \quad (D_2 y)_i = y_{i+2} - 2y_{i+1} + y_i.
\]

Now we can write the objective components in the compact form
\[
J^{\text{track}} = \|y - u\|^2, \quad J^{\text{norm}} = \|y\|^2, \quad J^{\text{cont}} = \|D_1 y\|^2, \quad J^{\text{smooth}} = \|D_2 y\|^2.
\]

Since \( y \) minimizes \( J \), the gradient of \( J \) with respect to \( y \) must be zero, i.e.,
\[
\nabla_y J = 2(y - u) + 2\lambda y + 2\mu D_1^T D_1 y + 2\kappa D_2^T D_2 y = 0.
\]

We are interested in finding \( \lambda \), \( \mu \), and \( \kappa \), so we re-write these equations as a set of linear equations involving \( \lambda \), \( \mu \), and \( \kappa \):
\[
\begin{bmatrix}
y \\
D_1^T D_1 y \\
D_2^T D_2 y
\end{bmatrix}
\begin{bmatrix}
\lambda \\
\mu \\
\kappa
\end{bmatrix}
= u - y.
\]

This is a set of \( n \) equations in the three unknowns \( \lambda \), \( \mu \), and \( \kappa \). For \( n > 3 \) this is a set of overdetermined linear equations; but we know that the equations are solvable.

We didn’t require you to do so, but we can say when the matrix above is full rank (in which case, we can recover \( \lambda \), \( \mu \), and \( \kappa \) exactly). It is full rank precisely when \( y \) does not satisfy a two-term recursion, i.e., \( y_{i+1} = \alpha y_i + \beta y_{i-1} \) for some \( \alpha, \beta \in \mathbb{R} \). It was OK with us for you to simply assume the matrix is full rank, or to check that for the given \( y \), it is full rank.
We can calculate $\lambda$, $\mu$, $\kappa$ as
\[
\begin{bmatrix}
\lambda \\
\mu \\
\kappa
\end{bmatrix} = \begin{bmatrix} y & D_1^T D_1 y & D_2^T D_2 y \end{bmatrix}^\dagger (u - y).
\]
(It’s good practice to check that the computed $(\lambda, \mu, \kappa)$ do indeed satisfy the equations above, as they must.)

The following matlab script implements the solution.

```matlab
rev_eng_smooth_data;

% construct the first difference operator, D1
D1 = zeros(n-1,n);
for i = 2 : n
    D1(i-1,i-1) = -1;
    D1(i-1,i) = 1;
end

% construct the second difference operator, D2
D2 = zeros(n-2,n);
for i = 2:n-1
    D2(i-1,i-1) = 1;
    D2(i-1,i) = -2;
    D2(i-1,i+1) = 1;
end

% solve for the weights (i.e., lambda, mu, kappa)
weights = ([y D1'*D1*y D2'*D2*y]
\(u-y));

% let’s check that the equations are satisfied exactly (up to numerical errors)
norm([y D1'*D1*y D2'*D2*y]*weights-(u-y))

lambda = weights(1)
mu = weights(2)
kappa = weights(3)

We find that $\lambda = 0.1$, $\mu = 2$ and $\kappa = 10$. (The script verifies that the overdetermined equations are indeed satisfied.)

7. Estimating link delays from route latencies. We consider a communications network with $m$ links that connect $p$ nodes. There are $n$ routes in the network, and each route is a path from some source node, going along one or more links in the network, to a destination node. The routes are determined and known.

We associate a delay $d_i > 0$ with each link $i$, representing the time needed to travel the link. We use $d = (d_1, \ldots, d_m)$ to denote the vector of link delays in the network. We have a latency $l_j > 0$ associated with the route $j$, which corresponds to the total time needed to
travel from the source node to the destination node of the route. We use \( l = (l_1, \ldots, l_n) \) to denote the vector of route latencies in the network.

We say that the latency vector \( l \) is consistent with the underlying link delays if there exist a set of link delays which give that latency vector. In this problem we assume that all measured latency vectors are consistent.

Before we get to the questions, we define a matrix that might be useful. The route-link incidence matrix \( R \) specifies which routes are using which links and its \((i,j)\)th entry is defined as

\[
R_{ij} = \begin{cases} 
1 & \text{route } j \text{ utilizes link } i \\
0 & \text{otherwise}
\end{cases}
\]

a) When can we perfectly and without ambiguity recover all the link delays in the network given the route latencies? (Express your answer using defined terms such as \( l \), \( d \), and \( R \).)

b) True or False: If \( Ry = 0 \) for some \( y \in \mathbb{R}^n \) and \( l^Ty \neq 0 \), then \( l \) is not a consistent latency vector. State if this is a true or false statement and explain your reasoning.

c) A route latency vector \( l \) and a route-link matrix \( R \) are given in the Matlab file \texttt{route_latency_data.m}. If possible, find the link delays in the network from the latency data, otherwise state that this is not possible and give two different link delays both producing the same given latency.

d) Mr. Johnson (our favorite engineer) proposes the following method to compute link delays in the network from the latency data. Here is his proposal to the Boss.

We define the count matrix \( F \in \mathbb{R}^{m \times m} \) as follows: \( F_{ij} \) is the number of routes that utilize both the link \( i \) and \( j \). Therefore, \( F_{ii} \) is the number of routes utilizing the link \( i \).

For each link \( i \), we define \( g_i \) as the sum of all latencies \( l_j \), where \( j \) is over routes that contain link \( i \).

Then we compute the link delays as \( d = F^{-1}g \), where we require that \( F \) is invertible.

Choose one of the following:

- **Boss rewards Johnson since the method works whenever the delays can be perfectly recovered from the latencies.**
  
  By ‘works’ we mean that \( F \) is invertible, and that \( F^{-1}g \) is the unique \( d \) that gives the route latencies \( l \). If you believe this is the case, explain why.

- **Boss fires Johnson since the method can fail, even when the delays can be perfectly recovered from the latencies.**

  To justify the firing, give a specific example, where the delays can be perfectly recovered from the latency measurements, but the method above fails, i.e., either \( F \) is singular, or \( F^{-1}g \) does not have the required latency totals. (Please try to give as simplest example as you can think of.)
**Solution.** We first define the route-link incidence matrix \( R \in \mathbb{R}^{m \times n} \) in terms of its columns. We have \( R = [r_1 \cdots r_n] \), where \( r_i \in \mathbb{R}^m \) has ones at the entries corresponding to links used by the \( i \)th route.

a) The vector of route latencies can be represented in the form \( \mathbf{l} = R^T \mathbf{d} \). This is easy to see if we think of the matrix multiplication as the inner product between the rows of matrix \( R^T \) (which are columns of matrix \( R \)) and the vector \( \mathbf{d} \). Then the latency of the \( i \)th route is given by

\[
    l_i = r_i^T \mathbf{d} = \sum_{j=0}^{m} R_{ji} d_j,
\]

which is the sum of delays along the links used by the \( i \)th route.

Now the question is when can we solve (without ambiguity) for \( \mathbf{d} \) given an \( \mathbf{l} \). The answer is: delays \( d_1, \ldots, d_m \) can be determined from the latency data if \( R^T \) has zero nullspace (or equivalently the range of \( R \) spans \( \mathbb{R}^m \)).

*Comment:* Here it does not matter that all the vectors and the matrix \( R \) are nonnegative. By our assumption of consistency, we know that there exists a positive vector \( \mathbf{d} \) that gives a unique \( \mathbf{l} \) if \( \text{null}(R^T) = \{0\} \).

b) The statement is True. We note that \( y \in \mathbb{R}^n \) with \( Ry = 0 \) belongs to \( \text{null}(R) \). From the previous part we have that a consistent \( \mathbf{l} \) belongs to \( \text{range}(R^T) \). But these two spaces are complementary, i.e., they are orthogonal to each other and together span the whole space \( \mathbb{R}^n \). Therefore, \( l^T y = 0 \), since the two vectors are in the complementary subspaces. On the other hand, if we have \( l^T y \neq 0 \), then \( \mathbf{l} \) does not completely belong to \( \text{range}(R^T) \), and we conclude that \( \mathbf{l} \) is not consistent.

c) The given \( R^T \) matrix is a skinny, full rank matrix, and therefore has a zero nullspace (and is one-to-one). We should be able to find a unique link delay vector \( \mathbf{d} \) that generates the given latency vector \( \mathbf{l} \) if the latency is consistent, i.e., \( \mathbf{l} \in \text{range}(R^T) \). But in this problem we assume that \( \mathbf{l} \) is always consistent.

To find \( \mathbf{d} \) we need to construct a left inverse of \( R^T \), call it \( B \in \mathbb{R}^{m \times n} \), which will give \( BR^T = \mathbf{l} \) and \( \hat{\mathbf{d}} = Bl = BR^T \mathbf{d} = \mathbf{d} \). One such left inverse is the least-squares left inverse, i.e.,

\[
    B = ((R^T)^T (R^T)^{-1} (R^T)^T = (RR^T)^{-1} R \text{ which exists since } R \text{ and } R^T \text{ are full rank}.
\]

Matlab code that solves the problem is given below.

```matlab
% load the data
route_latency_data;

% route-link incidence matrix
R = [ ... ]
    1 1 1 1 1 0 0 0 0 0 1
    1 1 1 0 0 1 1 0 0 0 1
    0 0 0 1 0 1 0 1 1 0 1
    0 1 0 0 1 0 1 1 0 1 1
    0 0 0 1 1 0 0 1 1 1 1
    0 0 1 0 0 1 1 0 1 1 1
```
d) The proposed method is actually the least squares approximate solution for \( R^T d = l \), which in this case is given by
\[
d = (RR^T)^{-1} R l.
\]
To see this relationship we note that
\[
(RR^T)_{ij} = \sum_{k=1}^{n} (R)_{ik}(R^T)_{kj} = \sum_{k=1}^{n} (R)_{ik}(R)_{jk} = F_{ij}.
\]
(The entry \((R)_{ik}(R)_{jk} = 1\) whenever both \(i\) and \(j\) links are utilized by the route \(k\).) Thus, the matrix \( RR^T \) is exactly the count matrix \( F \). We can also write \( g = R l \). Therefore, our proposed method \( d = F^{-1} g \) is nothing more than
\[
d = F^{-1} g = (RR^T)^{-1} R l.
\]
Since \( l \in \text{range}(R^T) \), the least squares will give the exact solution to our problem, provided that we can invert the matrix \( F = RR^T \). We know that this matrix is invertible, i.e., \( \det(F) = \det(RR^T) \neq 0 \), whenever the nullspace of \( R^T \) is zero. Recall that this is the case when we can perfectly and without ambiguity determine the underlying link delays in the network. Therefore, we conclude that Johnson’s method works, whenever the delays can be perfectly recovered from the latencies. Mr. Johnson gets a fat bonus from the Boss.

8. Trace of a square matrix. The trace of a square matrix \( A \in \mathbb{R}^{n \times n} \), \( A = (a_{ij}) \), is defined to be the sum of its diagonal elements:
\[
\text{trace}(A) = \sum_{i=1}^{n} a_{ii}.
\]
It’s obvious that trace is linear,
\[
\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B) \quad \text{trace}(\alpha A) = \alpha \text{trace}(A),
\]
and that
\[ \text{trace}(A^T) = \text{trace}(A). \]

Less obvious is the following fact.

a) For \( A, B \in \mathbb{R}^{n \times n} \), show that
\[ \text{trace}(AB) = \text{trace}(BA). \]

b) The properties of the trace allow us to define an inner product of two square matrices \( A \) and \( B \) by
\[ \langle A, B \rangle = \text{trace}(AB^T). \]

This inner product then defines a norm of \( A \) as
\[ \|A\| = \{\text{trace}(AA^T)\}^{1/2}. \]

What is \( \|A\| \) in terms of the entries of \( A \)?

c) We can define a vectorize function, \( \text{vec}(A) \in \mathbb{R}^{n^2} \), which, given the matrix \( A \), stacks up the columns of \( A \) into a vector of length \( n^2 \) (first column followed by the second column, etc.) In Matlab \( \text{vec}(A) \) is given by \( A(:, :) \). Show that \( \langle A, B \rangle = \text{trace}(AB^T) = \text{vec}(A)^T \text{vec}(B) \).

**Solution.**

a) Write \( B = (b_{ij}) \). The \( ij \)-entry of the products \( AB \) and \( BA \) are
\[ (AB)_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}, \quad (BA)_{ij} = \sum_{k=1}^{n} b_{ik}a_{kj}. \]

The diagonal elements are given by
\[ (AB)_{ii} = \sum_{k=1}^{n} a_{ik}b_{ki}, \quad (BA)_{ii} = \sum_{k=1}^{n} b_{ik}a_{ki}. \]

Then
\[ \text{trace}(AB) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik}b_{ki}, \quad \text{trace}(BA) = \sum_{i=1}^{n} \sum_{k=1}^{n} b_{ik}a_{ki}. \]

Take the expression for \( \text{trace}(AB) \) and interchange the order of summation
\[ \text{trace}(AB) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik}b_{ki} = \sum_{k=1}^{n} \sum_{i=1}^{n} a_{ik}b_{ki} = \sum_{k=1}^{n} (BA)_{kk} = \text{trace}(BA). \]
b) We have

\[(AA^T)_{ij} = \sum_{k=1}^{n} a_{ik}a_{jk},\]

so that

\[(AA^T)_{ii} = \sum_{k=1}^{n} a_{ik}a_{ik} = \sum_{k=1}^{n} a_{ik}^2,\]

Then

\[\|A\|^2 = \text{trace}(AA^T) = \sum_{i,k} a_{ik}^2.\]

In words, \(\|A\|^2\) is the sum of the squares of all the entries in \(A\), and \(\|A\|\) is the square-root of this sum.

c) We have

\[(AB^T)_{ij} = \sum_{k=1}^{n} (A)_{ik}(B^T)_{kj} = \sum_{k=1}^{n} a_{ik}b_{jk},\]

so that

\[\langle A, B \rangle = \text{trace}(AB^T) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik}b_{ik}\]

By definition of \(\text{vec}(A)\), \(\text{vec}(B)\), and the inner product, \(\text{vec}(A)^T \text{vec}(B)\) gives the sum of inner products between the columns of \(A\) and columns of \(B\). That is

\[\text{vec}(A)^T \text{vec}(B) = \sum_{j=1}^{n} \left( \sum_{i=1}^{n} a_{ij}b_{ij} \right)\]

Interchanging the order of summation and labeling \(j\) as \(k\), we obtain

\[\text{vec}(A)^T \text{vec}(B) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik}b_{ik} = \text{trace}(AB^T) = \langle A, B \rangle.\]

9. Zeroing out the board. Bobbie and Reza are playing a game. This game is played on a 6 × 6 board as follows. First, Bobbie fills the board with 36 arbitrary real numbers and then Reza performs a sequence of actions:

At each step, Reza is allowed to choose one cell, and an arbitrary real number \(x\). Then he can add \(x\) to the selected cell and subtract \(x\) from all adjacent cells. (Some cells have four adjacent cells, some have three, and some have two.) Reza’s goal to perform a sequence of allowed actions to derive a table which consists of 36 zeros. If Reza can derive the table consisting of zeros, he wins the game, otherwise Bobbie is the winner.

a) If Bobbie writes 1 in a corner cell (and 0 elsewhere), can Reza win the game? If you believe the answer is positive, you should specify the sequence of actions Reza should take. If your answer is negative, you should prove that there is no possible sequence of actions that Reza can take to zero out the table.
b) Can Bobbie fill in the table so that Reza has no possible way of winning the game? If
your answer is positive, you should prove that there exists an initial table that Reza
cannot turn into zero. If your answer is negative, you should prove that Reza can turn
any initial table into zero with a sequence of allowed actions.

c) Solve part b for a $9 \times 9$ table.

Solution.

a) We represent every $6 \times 6$ table by a vector in $s \in \mathbb{R}^{36}$. Let $s^{(0)} \in \mathbb{R}^{36}$ represent the
initial configuration. The action associated with the real number $x$ and the $i$th cell is
equivalent to adding a vector $xa_i \in \mathbb{R}^{36}$ to $s$ where

$$(a_i)_j = \begin{cases} 
1 & j \text{ denotes the selected cell } i \\
-1 & j \text{ denotes an adjacent cell to the selected cell } i \\
0 & \text{ Otherwise.}
\end{cases}$$

Thus, the table derived after a sequence of actions will be $s^{(0)} + \sum_i x_i a_i = s^{(0)} + Ax$
where $A$ is a matrix whose columns are $a_i$’s. So we are looking for an input $x$ such that
$s^{(0)} + Ax = 0$. The following code solves the problem

```matlab
a=6;
A=[];
for i=1:a
    for j=1:a
        a=zeros(a^2,1);
        a(a*(i-1)+j)=1;
        if i>1
            a(a*(i-2)+j)=-1;
        end
        if i<a
            a(a*(i)+j)=-1;
        end
        if j>1
            a(a*(i-1)+j-1)=-1;
        end
        if j<a
            a(a*(i-1)+j+1)=-1;
        end
        A=[A,a];
    end
end
s = zeros(a*a,1);
s(1) = -1;
if rank([A,s]) == rank(A)
```

17
We see that the answer will be
\[
M = \frac{-1}{13} \begin{bmatrix}
17 & 2 & -9 & -6 & 1 & 3 \\
2 & -6 & -5 & 2 & 4 & 2 \\
-9 & -5 & 8 & 9 & -1 & -5 \\
-6 & 2 & 9 & 0 & -9 & -6 \\
1 & 4 & -1 & -9 & -2 & 8 \\
3 & 2 & -5 & -6 & 8 & 16
\end{bmatrix}.
\]

b) The problem is equivalent to verifying that the linear transformation defined by \(Ax\) is onto. Since \(A\) is square, it suffices to see if \(A\) has zero nullspace. The following code solves the problem.

```matlab
clc
for d=[6,9]
    A=[];
    for i=1:d
        for j=1:d
            a=zeros(d^2,1);
            a(d*(i-1)+j)=1;
            if i>1
                a(d*(i-2)+j)=-1;
            end
            if i<d
                a(d*(i)+j)=-1;
            end
            if j>1
                a(d*(i-1)+j-1)=-1;
            end
            if j<d
                a(d*(i-1)+j+1)=-1;
            end
            A=[A,a];
        end
    end
    fprintf('for d=%d, the dimension of nullspace is %d\n',d,d^2-rank(A))
end
```

We see that Reza can always win, since the transformation is onto.

c) The code from part b shows that the transformation is not onto, so Bobbie can select a good initialization to win.