# Section 7: Least-squares Curve Fitting and Error Propagation

## Calibration: Matching Sensor Model Parameters to Data

A *sensor model* describes what output a sensor returns under different measurement conditions. For example, suppose that a temperature sensor returns a voltage $v$ given a temperature $t$. Let's assume that our hypothetical temperature sensor can be adequately modeled using the linear sensor model:

$$v = mt + b$$

where $m$ is the "slope" which converts temperature $t$ (in degrees C) to a voltage $v$, and $b$ is a bias in the sensor (also in units of volts). Usually, the parameters $m$ and $b$ are provided by the manufacturer. However, these parameters may vary between sensors (due to manufacturing differences, each sensor is not exactly the same) and/or they may depend on other factors (e.g., operating conditions). Thus, to *calibrate* the temperature sensor we can expose it to a variety of known temperatures $\{t_1, t_2, \cdots, t_N\}$, record the resulting voltages $\{v_1, v_2, \cdots, v_N\}$, and attempt to find the parameters $m$ and $b$. How many such measurements are required? If the sensor was an ideal, noise-less sensor then we would only need two measurements. The first measurement would be

$$v_1 = mt_1 + b$$

and choosing a unique temperature $t_2 \neq t_1$ the second measurement would be

$$v_2 = mt_2 + b \, .$$

Then we could combine this into a matrix equation of the form $\mathbf{y} = \mathbf{Ap}$:

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} t_1 & 1 \\ t_2 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} m \\ b \end{pmatrix}}_{\mathbf{p}}$$

and solve for the desired parameter vector by simply taking the inverse of $\mathbf{A}$ to obtain

$$\mathbf{p} = \mathbf{A}^{-1}\mathbf{y} \, .$$

However, in reality there is no ideal sensor! Instead, the measurements $y_i$ and even the "known" temperatures $t_i$ have some uncertainty. This implies that the values $t_1$ and $t_2$ do not exactly map to the values $v_1$ and $v_2$ using fixed parameters $m$ and $b$ of the sensor model. In other words, the equation $\mathbf{y} = \mathbf{Ap}$ above does not have an *exact* solution.

Rather than seeking the exact solution, perhaps we can find an approximate solution that fits the data. Intuitively, the more data we use the better this fit will be. If we take $N$ measurements $\{t_1, t_2, t_3, \cdots, t_{N-1}, t_N\}$ and $\{v_1, v_2, v_3, \cdots, v_{N-1}, v_N\}$ the equation we will want to solve will be of the

form:

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{N-1} \\ v_N \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} t_1 & 1 \\ t_2 & 1 \\ t_3 & 1 \\ \vdots & \vdots \\ t_{N-1} & 1 \\ t_N & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} m \\ b \end{pmatrix}}_{\mathbf{p}}$$

The problem is that the inverse operation $\mathbf{A}^{-1}$ only exists for square matrices (and even then, only under certain conditions). In any case, $\mathbf{A}^{-1}$ is used to find the exact solution which we have just concluded does not exist. What we need is an alternate way to "solve" the above equation and find the approximate $\mathbf{p}$. One approach is to use least-square linear regression.

## Least-Square Linear Regression

Suppose that we have a matrix equation:

$$\mathbf{y} = \mathbf{A}\mathbf{p}$$

where $\mathbf{y}$ is a vector of outputs, $\mathbf{p}$ is a vector of constant parameters, and $\mathbf{A}$ is a matrix that maps the parameters to the outputs. As discussed in the previous section, we assume the above equation has no exact solution, and instead we want an approximate $\mathbf{p}$. Not just any $\mathbf{p}$ will do, though; we want a $\mathbf{p}$ that is "good" in some sense. Since each measurement has some error we can define an error vector $\boldsymbol{\epsilon} = (\epsilon_1 \ \epsilon_2 \ \cdots \ \epsilon_N)^{\mathrm{T}}$ which accounts for this error so that

$$\mathbf{y} = \mathbf{A}\mathbf{p} + \boldsymbol{\epsilon}$$

Rearranging,

$$\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{A}\mathbf{p}$$

We may seek to find a parameter vector $\mathbf{p}$ vector that minimizes the error vector $\boldsymbol{\epsilon}$. Specifically, let us define the cost of a given vector $\mathbf{p}$ as the sum of the squares of the error vector:

$$J = ||\boldsymbol{\epsilon}||^2$$

The cost function $J$ is a scalar number. Using our new metric $J$, the "best" $\mathbf{p}$ will be the one that minimizes $J$.

---

**Side-note: Norm of a vector.** The square of the norm of a vector $\mathbf{x} = (x_1 \ x_2 \ \cdots x_N)$ is written $||\mathbf{x}||^2$ and is equal to the sum of the squares of the individual elements $||\mathbf{x}||^2 = x_1^2 + x_2^2 + \cdots + x_N^2$. This norm

can also be defined by multiplying the transpose of the vector by the vector itself:

$$\begin{aligned}
||\mathbf{x}||^2 &= \mathbf{x}^{\mathrm{T}}\mathbf{x} \\
&= (x_1\ x_2\ \cdots x_N)\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \\
&= x_1x_1 + x_2x_2 + \cdots + x_Nx_N \\
&= x_1^2 + x_2^2 + \cdots + x_N^2
\end{aligned}$$

**Transpose of a vector expression that is equal to a scalar.** Note that a matrix expression that is equal to a scalar can be transposed without changing the value of the scalar. Thus if

$$\mathbf{y}^{\mathrm{T}}\mathbf{A}\mathbf{p} = c$$

where $c$ is a scalar. Then the following is also true:

$$\mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y} = c$$

and thus we can replace $\mathbf{y}^{\mathrm{T}}\mathbf{A}\mathbf{p} = \mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y}$.

---

In light of the above side-notes, we can expand $J$ as follows:

$$\begin{aligned}
J &= \boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon} \\
&= (\mathbf{y} - \mathbf{A}\mathbf{p})^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\mathbf{p}) \\
&= (\mathbf{y}^{\mathrm{T}} - \mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}})(\mathbf{y} - \mathbf{A}\mathbf{p}) \\
&= \mathbf{y}^{\mathrm{T}}\mathbf{y} - \underbrace{\mathbf{y}^{\mathrm{T}}\mathbf{A}\mathbf{p}}_{\mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y}} - \mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{p} \\
&= \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{p}
\end{aligned}$$

Since we are seeking to find the $\mathbf{p}$ that minimizes $J$ we proceed by taking the derivative of $J$ with respect to $\mathbf{p}$.

$$\begin{aligned}
\frac{dJ}{d\mathbf{p}} &= \frac{d}{d\mathbf{p}}\left(\mathbf{y}^{\mathrm{T}}\mathbf{y}\right) - \frac{d}{d\mathbf{p}}(2\mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y}) + \frac{d}{d\mathbf{p}}(\mathbf{p}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{p}) \\
&= 0 - 2\mathbf{A}^{\mathrm{T}}\mathbf{y} + 2\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{p}
\end{aligned}$$

The last derivative in the above equation may not be obvious to you at this point. For now, think of this as related to the familiar derivative: $dx^2/dx = 2x$. Setting the derivative equal to zero $dJ/d\mathbf{p} = 0$, and rearranging, gives:

$$\begin{aligned}
\mathbf{A}^{\mathrm{T}}\mathbf{y} &= \mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{p} \\
(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y} &= \mathbf{p}
\end{aligned}$$

The quantity $\mathbf{A}^{+} \triangleq (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}$ is called the *pseudoinverse*. (The symbol $\triangleq$ means we are introducing a new term that is equivalent to the expression on the right-hand-side – this is slightly different then

using the equality symbol =.) The value

$$\mathbf{p} = \mathbf{A}^{+}\mathbf{y}$$

gives the parameter vector that matches the data best (in a least-squares sense).

---

**Example: Calibrating a temperature sensor**. Suppose that a temperature sensor was being calibrated by exposing it to a range of temperatures from 0 C to 30 C in increments of 3.333 C. At each temperature "station" the average voltage was recorded and the following data set was generated:

| Station $i$ | Temperature $t_i$ | Voltage $v_i$ |
|---|---|---|
| 1 | 0.00 | 1.85 |
| 2 | 3.33 | 2.92 |
| 3 | 6.67 | 3.48 |
| 4 | 10.00 | 3.10 |
| 5 | 13.33 | 3.96 |
| 6 | 16.67 | 4.13 |
| 7 | 20.00 | 5.48 |
| 8 | 23.33 | 5.00 |
| 9 | 26.67 | 5.50 |
| 10 | 30.00 | 6.40 |

The temperature sensor model is as described previously:

$$v = mt + b$$

To find the best parameters $m$ and $b$ that fit the data we define the parameter vector $\mathbf{p} = (m\ b)^{\mathrm{T}}$ and write the matrix equation in full expanded form as:

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{N-1} \\ v_N \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} t_1 & 1 \\ t_2 & 1 \\ t_3 & 1 \\ \vdots & \vdots \\ t_{N-1} & 1 \\ t_N & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} m \\ b \end{pmatrix}}_{\mathbf{p}}$$

$$\underbrace{\begin{pmatrix} 1.85 \\ 2.92 \\ 3.48 \\ 3.10 \\ 3.96 \\ 4.13 \\ 5.48 \\ 5.00 \\ 5.50 \\ 6.40 \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} 0.00 & 1 \\ 3.33 & 1 \\ 6.67 & 1 \\ 10.00 & 1 \\ 13.33 & 1 \\ 16.67 & 1 \\ 20.00 & 1 \\ 23.33 & 1 \\ 26.67 & 1 \\ 30.00 & 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} m \\ b \end{pmatrix}}_{\mathbf{p}}$$

Then computing $\mathbf{A}^+$ in MATLAB we find the best-fitting parameter vector is:

$$\mathbf{p} = \mathbf{A}^+\mathbf{y} = \begin{pmatrix} 0.134 \\ 2.171 \end{pmatrix}$$

Thus the estimated sensor model is

$$v = (0.134)t + 2.171$$

The data described above was generated from an "ideal" sensor model with $m = 0.15$ and $b = 2.0$ to which random noise was added. We find that the estimated parameters are fairly close to the ideal ones. The sensor model with ideal and estimated parameters, along with the raw data, is plotted in Fig. 1. The MATLAB code is also listed in the following.

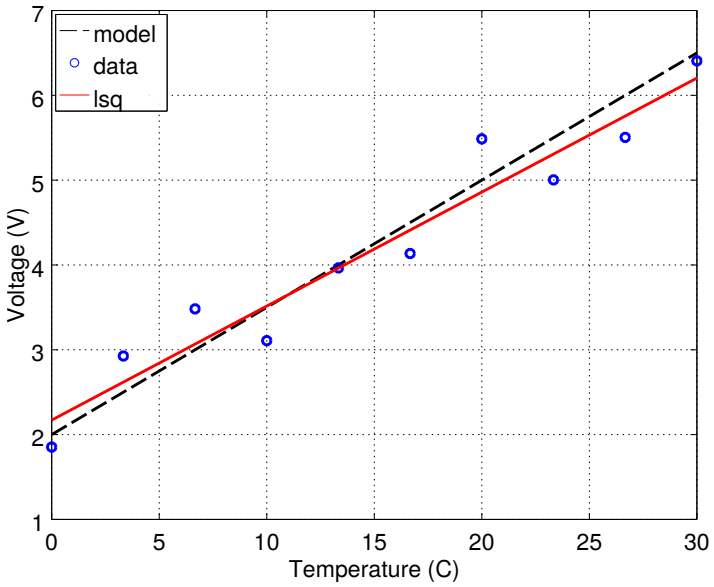



Figure 1: Temperature vs. Voltage Calibration Curve

Listing 1: Temperature vs. Voltage Calibration Curve

```
% ENGR 207: Least-Square Fitting of Temperature-Voltage Data
clear all; close all; clc; % prepare workspace
% sensor model: % v = m*t + b
m = 0.15; b = 2; % the true model parameters
numPts = 10; % generate ideal data from the model
t = linspace(0,30, numPts);
v = m*t + b;
plot(t,v,'k--','linewidth',2) % plot ideal data
hold on;
set(gca,'FontSize',16)
grid on;
xlabel('Temperature (C)'); ylabel('Voltage (V)');
% this is the data of average measured voltage at each temperature
dataAvg = [1.85304; 2.92669; 3.48231; 3.10769; 3.96549; 4.13511;
5.48647; 5.00322; 5.50429; 6.40635];
plot(t,dataAvg,'bo','linewidth',2);
% Want to form equation : y = Ap
```

```
% where p = [m b]; are the parameters we wish to estimate
% where y = [v1 v2 .. vN];  % are the N known outputs
% where A = [t1 t2 .. tN;   % are the N known inputs
%            1  1  .. 1   ];%
y = dataAvg;
A = [t; ones(size(t))]'; % this needs to be a tall matrix, so use transpose
p = pinv(A)*y % solve for params in a lsq sense
v_lsq = t*p(1) + p(2); % evaluate the model with the estimated params
plot(t,v_lsq,'r','linewidth',2) % plot results on same axes as before
legend('model','data','lsq','location','northwest') % move legend to upper left corner
print(1,'-dpdf','lsq_line.pdf'); % print figure to pdf
```

## Random Variables

You may recall from an earlier probability class that a *Gaussian* or *normal* probability density function (PDF) of a random variable $X$ is completely characterized by a mean $\mu$ and variance $\sigma^2$. A Gaussian PDF has the familiar "bell" curve shape as shown in Fig. 2. The Gaussian probability distribution is given by the following equation

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu).^2}{2\sigma^2}\right) \tag{1}$$

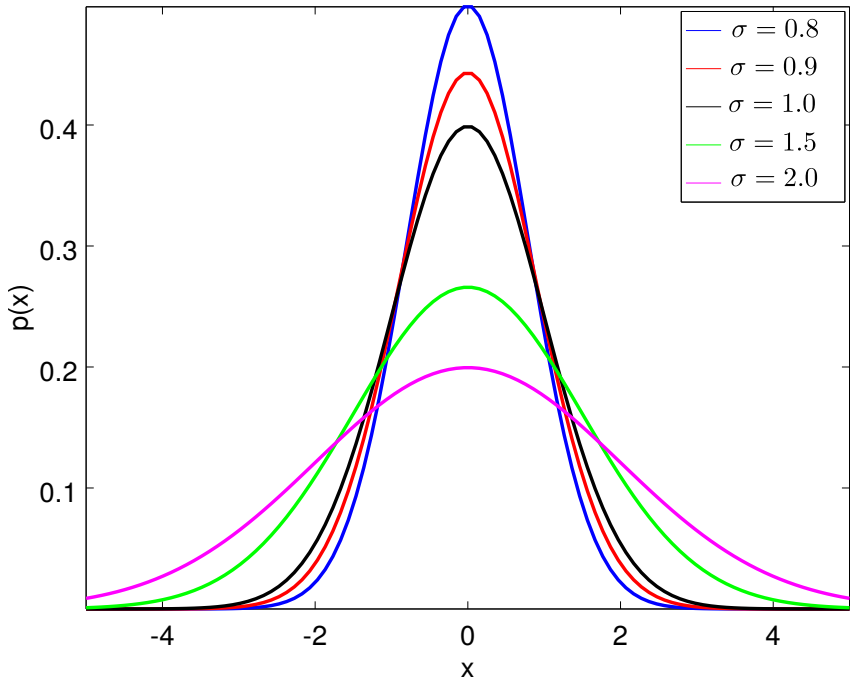



Figure 2: Gaussian probability distributions with a zero mean $\mu = 0$ and varying standard deviation $\sigma$

The mean $\mu$ tells us where the Gaussian is "centred" and the variance $\sigma^2$ (or standard deviation $\sigma$) tells us how "spread out" the distribution is. The larger the standard deviation $\sigma$ the wider the bell curve. We can compactly express the fact that a random variable $X$ is normally distributed with mean $\mu$ and variance $\sigma^2$ by writing

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

For example, the blue curve in Fig. 2 represents the random variable $X \sim \mathcal{N}(0, 0.8^2)$.

The probability of a random variable $X$ taking on a value between $a$ and $b$ is given by integrating the area under the PDF

$$\int_a^b p(x) = p[a \leq X \leq b] \tag{2}$$

The Gaussian PDF equation (1) is actually quite challenging to integrate. However, some values of the integral (2) are well known and can be found in a *cumulative probability table*. For example, the probability of $X$ taking on a value within one standard deviation of the mean is about 68%. This is written as:

$$p[\mu - \sigma < X \leq \mu + \sigma] = 0.68 \tag{3}$$

Similarly, the probability of $X$ taking on a value within two or three standard deviations of the mean is:

$$p[\mu - 2\sigma < X \leq \mu + 2\sigma] = 0.95 \tag{4}$$

$$p[\mu - 3\sigma < X \leq \mu + 3\sigma] = 0.997 \tag{5}$$

Refer to Fig. 3 for a graphical representation of this fact.

---

**Example: Determining $\mu$ and $\sigma$ from a given range and percent likelihood**. Suppose we are told that some percent $P$ of the time the Gaussian random variable $X$ is within some range given by a lower bound ($b_l$ and an upper bound $b_u$). Mathematically, this can be written

$$p[b_l \leq X \leq b_u] = P \, .$$

For example, we might be told that 68% of the time the random variable $X$ is within 6 and 9. This implies that

$$p[6 \leq X \leq 9] = 0.68$$

If the variable is Gaussian, then we know that for certain values of $P$ (e.g., 0.68, 0.95, or 0.997) the lower and upper bounds correspond to whole multiple standard deviations from the mean ($1\sigma, 2\sigma$ and $3\sigma$, respectively). For example, in the case of $P = 0.68$ we know from Eq. 3 that the lower bound is one standard deviation below the mean $b_l = \mu - \sigma$ and the upper bound is one standard deviation above the mean $b_u = \mu + \sigma$. It is obvious that the mean itself is just the average of the upper and lower bounds $\mu = (b_l + b_u)/2$. You can convince yourself of this by carrying out the computation:

$$\frac{b_l + b_u}{2} = \frac{(\mu - \sigma) + (\mu + \sigma)}{2} = \frac{2\mu}{2} = \mu$$

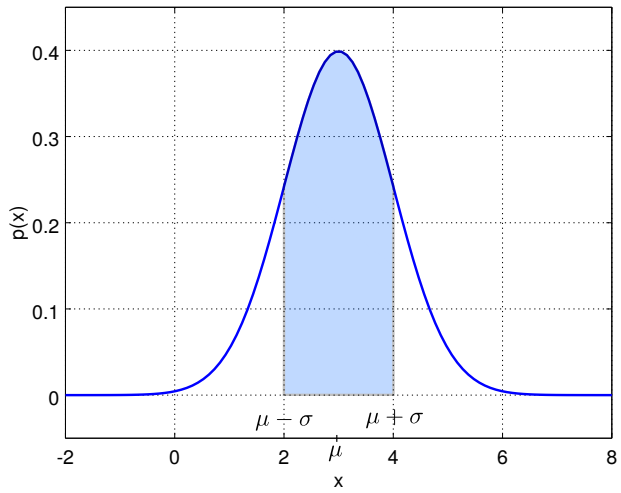
Using this knowledge we can solve for $\mu$ and $\sigma$. The mean is $\mu = (6 + 9)/2 = 7.5$ and the standard deviation is $\sigma = \mu - b_l = 7.5 - 6.0 = 1.5$ (obtained by rearranging the equations for $b_l$). We therefore conclude that the random variable is characterized as $X \sim \mathcal{N}(7.5, 1.5^2)$.

In this course we will only ask this type of question for the three $P$ values of (0.68, 0.95, or 0.997) listed above.

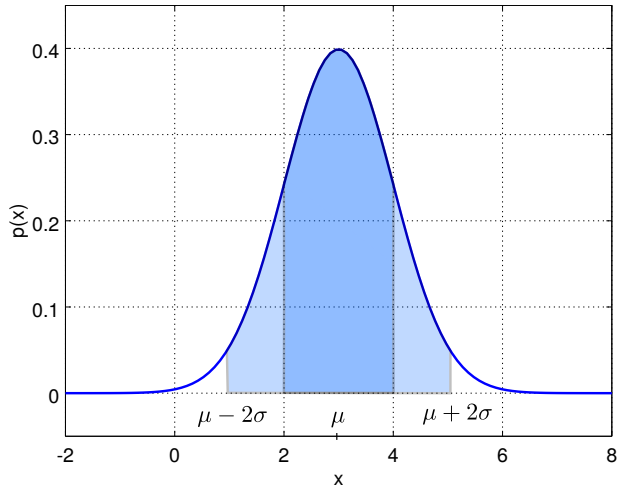
**Another Example:** If we are told that 99.7 % of the time the Gaussian random variable $X$ is within 0 and 12, then the mean is $\mu = (0 + 12)/2 = 6$ and (from Eq. 5) the lower bound is three standard

deviations below the mean. Thus, $\sigma = (\mu - b_l)/3 = (6 - 0)/3 = 2$. We therefore conclude that the random variable is characterized as $X \sim \mathcal{N}(6, 2^2)$.

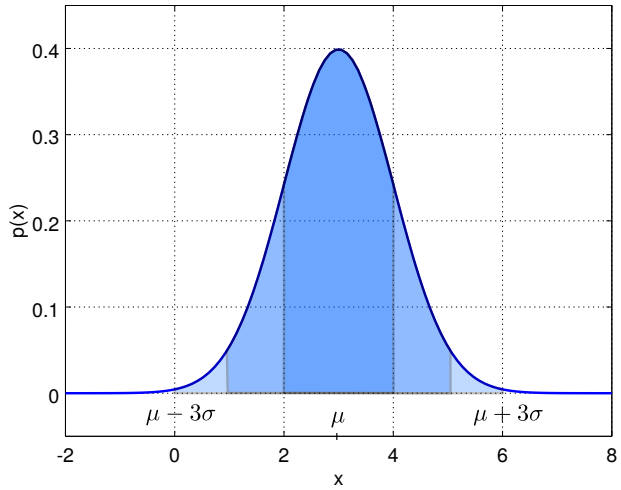
---




(a) 68% of the time the random number is within one standard deviation of the mean (i.e., between 2-4)




(b) 95% of the time the random number is within two standard deviations of the mean (i.e., between 1-5)




(c) 99.7% of the time the random number is within three standard deviations of the mean (i.e., between 0-6)

Figure 3: Probability distribution for the random number $X \sim \mathcal{N}(3, 1^2)$. The mean is $\mu = 3$, the variance is $\sigma^2 = 1^2$, and the standard deviation is $\sigma = 1$.

The PDF is defined for all possible values of $X$ (even those really far away from the mean have a

non-zero probability). We have 100 % certainty that the random variable $X$ must take on some value in the range $-\infty < X < \infty$ (it cannot remain undefined). This implies that the the area under every PDF is equal to 1

$$\int_{-\infty}^{\infty} p(x) = 1 \ .$$

## Random Vectors

The idea of probability can be extended to multiple dimensions and *random vectors*. Suppose we have a random vector $\mathbf{x}$ with two components $\mathbf{x} = (x_1 \ x_2)^{\mathrm{T}}$. The random vector has a corresponding vector mean

$$\boldsymbol{\mu}_{\mathbf{x}} = \begin{pmatrix} \mu_{x_1} \\ \mu_{x_2} \end{pmatrix}$$

As before, each element has its own variance $\sigma_{x_1}^2$ and $\sigma_{x_2}^2$, respectively. However, now that there is more than one random variable involved a pertinent question arises – is $x_1$ related to $x_2$ in any way? If the value $x_1$ has no influence on the value of $x_2$ we say the elements are *uncorrelated*. Mathematically, we write this by saying $\sigma_{x_1 x_2} = 0$. Similarly, if $x_2$ has no influence on $x_1$ we write $\sigma_{x_2 x_1} = 0$. Actually, these two variances are one and the same, and in general: $\sigma_{x_1 x_2} = \sigma_{x_2 x_1}$.

However, in some cases the two variables *are* correlated. Suppose that as $x_1$ increases $x_2$ tends to increase as well. In this case, $\sigma_{x_1 x_2} > 0$ will have a positive value. The opposite is also true: if $x_2$ tends to decrease as $x_1$ increases then $\sigma_{x_1 x_2} < 0$. The magnitude of $\sigma_{x_1 x_2}$ tells us how strongly these two random variables are correlated.

To full describe the probability distribution of a random vector $\mathbf{x}$ we must explicitly state the variances of each of these combinations of variables. This is written in the form of a *covariance matrix* $\mathbf{P}_{\mathbf{x}}$. In the 2D case we have:

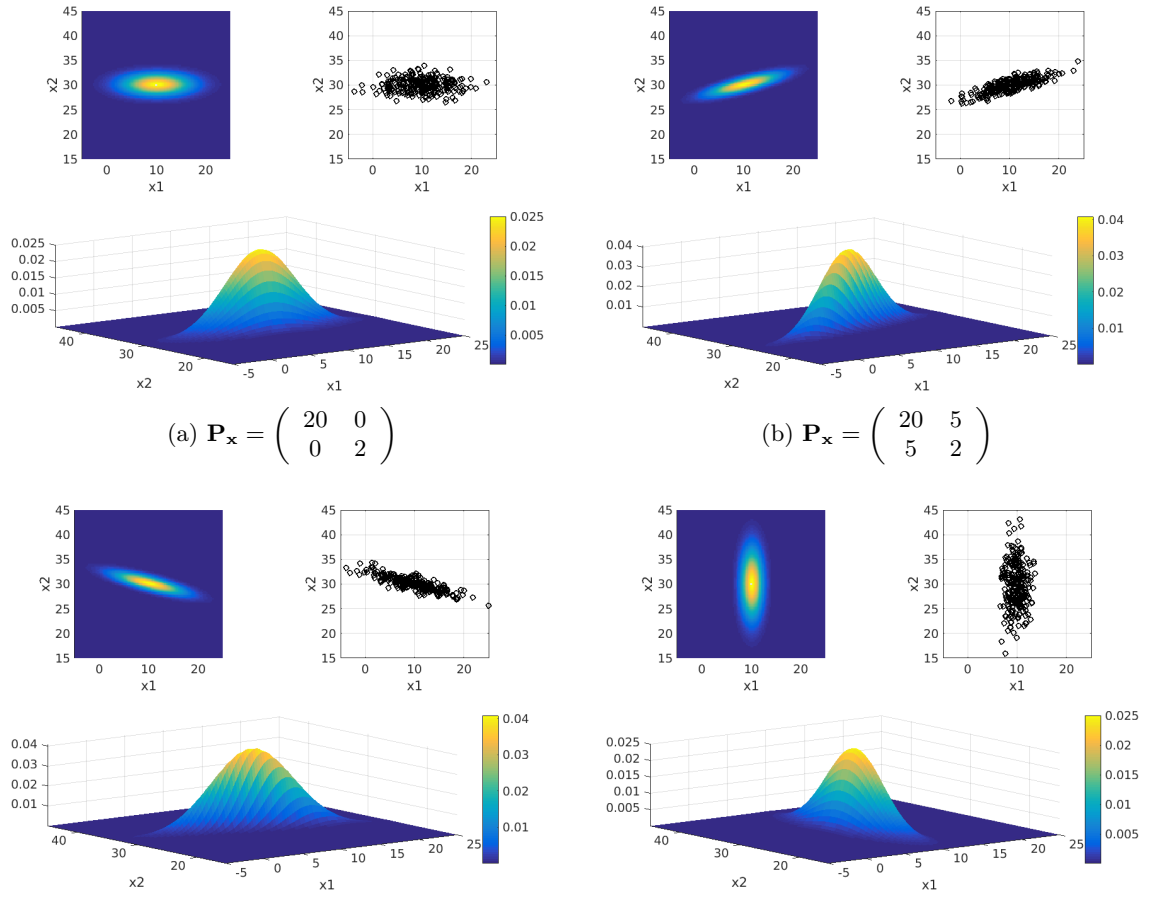
$$\mathbf{P}_{\mathbf{x}} = \begin{pmatrix} \sigma_{x_1}^2 & \sigma_{x_1 x_2} \\ \sigma_{x_2 x_1} & \sigma_{x_2}^2 \end{pmatrix}$$

where the entries of this matrix are as described in the preceding paragraphs. The probability density function of a multivariate Gaussian distribution is

$$p(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{|2\pi\mathbf{P}|}} \tag{6}$$

where the $|\cdot|$ operator denotes the determinant. As before, a random vector that has a normal/Gaussian distribution is denoted by $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \mathbf{P}_{\mathbf{x}})$.

Consider the numerical examples shown in Fig. 4 in which various probability distributions are plotted (using Eq. 6). Each distribution has the same mean $\boldsymbol{\mu}_{\mathbf{x}} = (10 \ \ 30)^{\mathrm{T}}$, but a varying covariance matrix $\mathbf{P}_{\mathbf{x}}$. In addition to plotting the (continuous) probability distribution, a set of 250 samples are drawn from each distribution (indicated by black markers) – notice that the shape of the samples is similar to the distribution. Fig. 4 is meant to illustrate how the four entires of the covariance matrix $\mathbf{P}_{\mathbf{x}}$ determine the shape of the PDF and the distribution of the samples. The term $\sigma_{x_1}^2$ determines how spread out the samples are in the $x_1$ direction. Similarly, $\sigma_{x_1}^2$ determines how spread out the samples are in the $x_2$ direction. The off-diagonal terms indicate in which direction the distribution is slanted. In the case, $\sigma_{x_1 x_2} = 5$ the two variables are positively correlated (as one increases, so does the other). In the case $\sigma_{x_1 x_2} = -5$ the opposite is true.




(a) $\mathbf{P_x} = \begin{pmatrix} 20 & 0 \\ 0 & 2 \end{pmatrix}$ (b) $\mathbf{P_x} = \begin{pmatrix} 20 & 5 \\ 5 & 2 \end{pmatrix}$

(c) $\mathbf{P_x} = \begin{pmatrix} 20 & -5 \\ -5 & 2 \end{pmatrix}$ (d) $\mathbf{P_x} = \begin{pmatrix} 2 & 0 \\ 0 & 20 \end{pmatrix}$

Figure 4: Probability distributions with mean $\boldsymbol{\mu} = (\mu_{x_1} \ \mu_{x_2})^{\mathrm{T}} = (10 \ \ 30)^{\mathrm{T}}$

## Error/Uncertainty Propagation

There are many instances in which a robot must infer how one uncertain quantity is related to another. For example, suppose a unicycle robot is trying to move forward by spinning its wheels. The robot's magnetometer may measure that it is currently facing north and send a command to spin the wheels 10 times. In an ideal world, the robot's belief that it is facing north would be true (without error) and it would move forward (perfectly) in a straight line by an amount equal to 10 times the radius of the wheel. However, in reality, there is uncertainty regarding how many times the wheel has rotated (e.g., perhaps it rotated 10.2 times or 9.6 times) and/or what orientation the vehicle was initial facing (e.g., perhaps it was actually a few degrees east or west of north because the sensor is noisy). Because the initial condition and the inputs are uncertain, the outcome of the action itself is also uncertain.

We can understand this situation mathematically by expression the uncertainty in the initial condition, the measurements, and/or the inputs as a random vector $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$. We are interested in what happens to this vector when some process occurs that results in a new vector $\mathbf{z}$. Further, we assume we have some model of this process $\mathbf{f}(\mathbf{x})$ which relates the input $\mathbf{x}$ to the output $\mathbf{z}$ (i.e., $\mathbf{z} = \mathbf{f}(\mathbf{x})$).

The question then becomes: what is the mean $\boldsymbol{\mu}_\mathbf{z}$ and covariance $\mathbf{P}_\mathbf{z}$ of the new random vector $\mathbf{z}$? We make the assumption that $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P}_\mathbf{z})$ – that $\mathbf{z}$ is also normally distributed. Although this is not always true it is a good first guess and this assumption is often used. The way in which we determine the new random vector's distribution is to use the first-order error propogation law.

## First-Order Error Propogation Law

Suppose we have the random vector $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$ and the (possibly) nonlinear equation

$$\mathbf{z} = \mathbf{f}(\mathbf{x}) \tag{7}$$

that generates a new random vector $\mathbf{z}$. We wish to determine the statistical properties of $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P}_\mathbf{z})$. The mean of $\boldsymbol{\mu}_\mathbf{z}$ is found by simply evaluating the the system (7) with the mean $\boldsymbol{\mu}_\mathbf{x}$:

$$\boldsymbol{\mu}_\mathbf{z} = \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}) \tag{8}$$

To determine the covariance $\mathbf{P}_\mathbf{z}$ we must first determine the *Jacobian* $\mathbf{J}_\mathbf{f}$ of the system (7). Suppose that $\mathbf{x} = (x_1,\ x_2,\ \cdots,\ x_n)^\mathrm{T}$ and $\mathbf{z} = (z_1,\ z_2,\ \cdots,\ z_m)^\mathrm{T}$. Clearly, the system (7) must also have $m$ equations (since there are $m$ elements in the output vector $\mathbf{z}$) and can be expanded as

$$\begin{gathered}\mathbf{z} = \mathbf{f}(\mathbf{x}) \\ \begin{pmatrix} z_1 \\ z_2 \\ \cdots \\ z_m \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \cdots, x_n) \\ f_2(x_1, x_2, \cdots, x_n) \\ \cdots \\ f_m(x_1, x_2, \cdots, x_n) \end{pmatrix}\end{gathered}$$

The Jacobian is defined by taking the partial derivative of each equation $f_i$ with respect to the $n$ variables $x_1, x_2, \cdots, x_n$:

$$\mathbf{J}_\mathbf{f} = \begin{pmatrix} \partial f_1/x_1 & \partial f_1/x_2 & \cdots & \partial f_1/x_n \\ \partial f_2/x_1 & \partial f_2/x_2 & \cdots & \partial f_2/x_n \\ \vdots & \vdots & \vdots & \vdots \\ \partial f_m/x_1 & \partial f_m/x_2 & \cdots & \partial f_m/x_n \end{pmatrix} \tag{9}$$

Computing this Jacobian symbolically gives a matrix that is a function of the variables $(x_1, x_2, \cdots, x_n)$. We can then *evaluate the Jacobian* at the mean $\boldsymbol{\mu}_\mathbf{x}$ by substituting the mean value $\mu_{x_1}$ for every $x_1$, the mean value $\mu_{x_2}$ for every $x_2$, and so on. To denote the evaluated Jacobian we write

$$\mathbf{J}_\mathbf{f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}$$

Finally, with this evaluated Jacobian in hand, we can compute the covariance $\mathbf{P}_\mathbf{z}$ using the following error-propogation law:

$$\mathbf{P}_\mathbf{z} = \left(\mathbf{J}_\mathbf{f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}\right) \mathbf{P}_\mathbf{x} \left(\mathbf{J}_\mathbf{f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}\right)^\mathrm{T} \tag{10}$$

Thus we have determined the mean $\boldsymbol{\mu}_\mathbf{z}$ and the covariance $\mathbf{P}_\mathbf{z}$ and therefore we have fully characterized the normally distributed random variable $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P}_\mathbf{z})$.

---

**Example: First-order Error Propagation (2D case)**.
Given the random vector $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$ where

$$\boldsymbol{\mu}_\mathbf{x} = \begin{pmatrix} 16 \\ 30 \end{pmatrix} \qquad \mathbf{P}_\mathbf{x} = \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}$$

find the mean $\boldsymbol{\mu}_\mathbf{z}$ and covariance $\mathbf{P}_\mathbf{z}$ which characterizes the random variable $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P}_\mathbf{z})$ that results from the transformation

$$\mathbf{z} = \mathbf{f}(\mathbf{x})$$
$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^{1/4} + x_2 \\ x_1 x_2 / 10 \end{pmatrix}$$

**Solution.** The mean $\boldsymbol{\mu}_z$ is obtained by propagating the mean $\boldsymbol{\mu}_\mathbf{x}$ through the system:

$$\boldsymbol{\mu}_\mathbf{z} = \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}) = \begin{pmatrix} 16^{1/4} + 30 \\ 30(16)/10 \end{pmatrix} = \begin{pmatrix} 32 \\ 48 \end{pmatrix}$$

Next, we must compute the Jacobian

$$\mathbf{J}_\mathbf{f} = \begin{pmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 \end{pmatrix} = \begin{pmatrix} \left(\frac{1}{4}\right) x_1^{-3/4} & 1 \\ x_2/10 & x_1/10 \end{pmatrix}$$

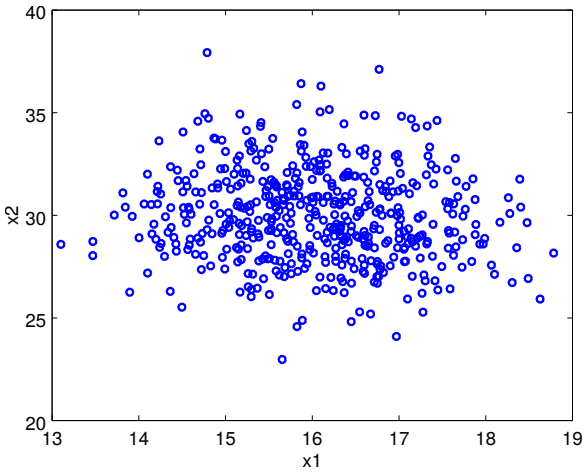
Then, evaluating the Jacobian at the mean $\boldsymbol{\mu}_x$

$$\mathbf{J}_\mathbf{f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}} = \begin{pmatrix} \left(\frac{1}{4}\right)(16)^{-3/4} & 1 \\ 30/10 & 16/10 \end{pmatrix} = \begin{pmatrix} 0.03125 & 1 \\ 3 & 1.6 \end{pmatrix}$$

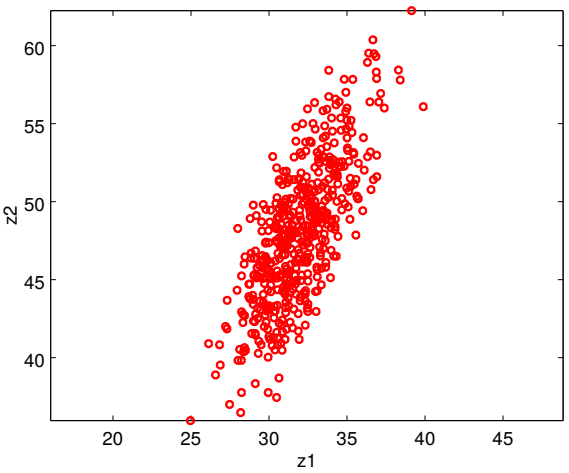
We have all the necessary elements to compute the covariance $\mathbf{P}_\mathbf{z}$ using the error-propagation law:

$$\begin{aligned} \mathbf{P}_\mathbf{z} &= \left(\mathbf{J}_\mathbf{f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}\right) \mathbf{P}_\mathbf{x} \left(\mathbf{J}_\mathbf{f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}\right)^\mathrm{T} \\ &= \begin{pmatrix} 0.03125 & 1 \\ 3 & 1.6 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix} \begin{pmatrix} 0.03125 & 1 \\ 3 & 1.6 \end{pmatrix}^\mathrm{T} \\ &= \begin{pmatrix} 5.001 & 8.094 \\ 8.094 & 21.800 \end{pmatrix} \end{aligned}$$

The solution can be visualized by generating samples from $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$ (Fig. 5a) and propagating them through the system (Fig. 5b).




(a) Samples representing $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$




(b) Samples representing $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P}_\mathbf{z})$

Figure 5: Transforming a 2D random vector: samples from Fig. 5a are propagated through the system $\mathbf{z} = \mathbf{f}(\mathbf{x})$ to generate the new random vector $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P}_\mathbf{z})$

**Example: First-order Error Propagation (1D case)**. Suppose we have a random variable $X \sim \mathcal{N}(\mu_x = 9, \sigma_x^2 = 0.2)$ that is transformed through the nonlinear function $z = f(x) = \sqrt{x}$ into a new random variable $Z \sim \mathcal{N}(\mu_z, \sigma_z^2)$. We wish to determine $\mu_z$ and $\sigma_z^2$.

**Solution.** The mean $\mu_z$ is found by propagating the mean $\mu_x$ through the transformation.

$$\mu_z = f(\mu_x) = \sqrt{9} = 3$$

The first-order error propagation law (10) reduces in the 1D case to:

$$\sigma_z^2 = \left( J_f \big|_{x=\mu_x} \right)^2 \sigma_x^2$$

where $J_f$ is the derivative of $f(x)$ with respect to $x$. Thus

$$J = \frac{d}{dx}\sqrt{x} = \frac{1}{2\sqrt{x}}$$

Evaluating with the mean $\mu_x$

$$J_f \big|_{x=\mu_x} = \frac{1}{2\sqrt{\mu_x}} = \frac{1}{2\sqrt{9}} = \frac{1}{6}$$

Finally, applying the 1D error-propagation law:

$$\sigma_z^2 = \left( J_f \big|_{x=\mu_x} \right)^2 \sigma_x^2 = \left( \frac{1}{6} \right)^2 (0.2) = 0.00556$$

Thus the random variable $Y$ is characterized as $Z \sim \mathcal{N}(3, 0.00556)$.

---

**Example: First-order Error Propagation (2D, with sample propagation)**.

Aircraft are often equipped with a device called a *pitot-static probe* which measures both the static pressure $p_s$ (that is only the ambient pressure at a given altitude and is independent of the aircraft's flight) and the total pressure $p_t$ (that includes the pressure resulting from the air flow and the ambient pressure). Based on these two pressure measurements, the speed $v$ and altitude $h$ of the aircraft can be determined.

Define the random input vector $\mathbf{x} = (p_t \;\; p_s)^{\mathrm{T}}$ and the random output vector $\mathbf{z} = (v \;\; h)^{\mathrm{T}}$. The two quantities are related by the system

$$\begin{gathered} \mathbf{z} = \mathbf{f}(\mathbf{x}) \\ \begin{pmatrix} v \\ h \end{pmatrix} = \begin{pmatrix} f_1(p_s, p_t) \\ f_2(p_s, p_t) \end{pmatrix} = \begin{pmatrix} \sqrt{k_1(p_t - p_s)} \\ k_2 \log(p_s/k_3) + k_4 \end{pmatrix} \end{gathered} \tag{11}$$

where

$$k_1 = 10.266 \qquad k_2 = -6341.726 \qquad k_3 = 22632.1 \qquad k_4 = 11000$$

Suppose that at a particular instant, the aircraft's pitot-static sensor returns a pressure reading (in units of Pascals)

$$\boldsymbol{\mu}_{\mathbf{x}} = \begin{pmatrix} 101228.869 \\ 101316.529 \end{pmatrix}$$

Further, let's represent the uncertainty associated with these measurements by the covariance matrix

$$\mathbf{P_x} = \begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix}$$

Thus, the measurement can be represented by the random vector $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P_x})$. We are interested in determining the mean $\boldsymbol{\mu}_\mathbf{z}$ and covariance $\mathbf{P_z}$ of the random vector $\mathbf{z}$ that is related to $\mathbf{x}$ through the system (11).

**Solution.** The mean $\boldsymbol{\mu}_\mathbf{z}$ is determined by evaluated the system (11) with the mean $\boldsymbol{\mu}_\mathbf{x}$

$$\boldsymbol{\mu}_\mathbf{z} = \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}) = \begin{pmatrix} \sqrt{k_1(101316.529 - 101228.869)} \\ k_2 \log(101228.869/k_3) + k_4 \end{pmatrix} = \begin{pmatrix} 30 & \text{m/s} \\ 1500 & \text{m} \end{pmatrix}$$

Then we proceed by computing the Jacobian of the system

$$\mathbf{J_f} = \begin{pmatrix} \partial f_1/p_s & \partial f_1/p_t \\ \partial f_2/p_s & \partial f_2/p_t \end{pmatrix} \tag{12}$$

where each partial derivative is evaluated separately:

$$\begin{aligned}
\frac{\partial f_1}{\partial p_s} &= \left(\frac{1}{2}\right)\left(\frac{1}{\sqrt{k_1(p_t - p_s)}}\right)(-k_1) = \frac{-k_1}{2\sqrt{k_1(p_t - p_s)}} \\
\frac{\partial f_1}{\partial p_t} &= \left(\frac{1}{2}\right)\left(\frac{1}{\sqrt{k_1(p_t - p_s)}}\right)(k_1) = \frac{k_1}{2\sqrt{k_1(p_t - p_s)}} \\
\frac{\partial f_2}{\partial p_s} &= k_2 \frac{1}{(p_s/k_3)}\left(\frac{1}{k_3}\right) = \frac{k_2}{p_s} \\
\frac{\partial f_2}{\partial p_t} &= 0
\end{aligned}$$

so that

$$\mathbf{J_f} = \begin{pmatrix} \frac{-k_1}{2\sqrt{k_1(p_t - p_s)}} & \frac{k_1}{2\sqrt{k_1(p_t - p_s)}} \\ \frac{k_2}{p_s} & 0 \end{pmatrix} \tag{13}$$

Evaluating this Jacobian with the mean $\boldsymbol{\mu}_\mathbf{x}$

$$\mathbf{J_f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}} = \begin{pmatrix} -0.17111 & 0.17111 \\ -0.06265 & 0.00000 \end{pmatrix} \tag{14}$$

Then using the error-propagation law

$$\begin{aligned}
\mathbf{P_z} &= \left(\mathbf{J_f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}\right) \mathbf{P_x} \left(\mathbf{J_f}\big|_{\mathbf{x}=\boldsymbol{\mu}_\mathbf{x}}\right)^{\mathrm{T}} \\
&= \begin{pmatrix} -0.17111 & 0.17111 \\ -0.06265 & 0.00000 \end{pmatrix} \begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix} \begin{pmatrix} -0.17111 & 0.17111 \\ -0.06265 & 0.00000 \end{pmatrix}^{\mathrm{T}} \\
&= \begin{pmatrix} 0.439168 & 0.107195 \\ 0.107195 & 0.039247 \end{pmatrix}
\end{aligned}$$

Since we have determined the mean $\boldsymbol{\mu}_\mathbf{z}$ and the covariance $\mathbf{P_z}$ we have fully characterized the normally distributed random variable $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \mathbf{P_z})$.

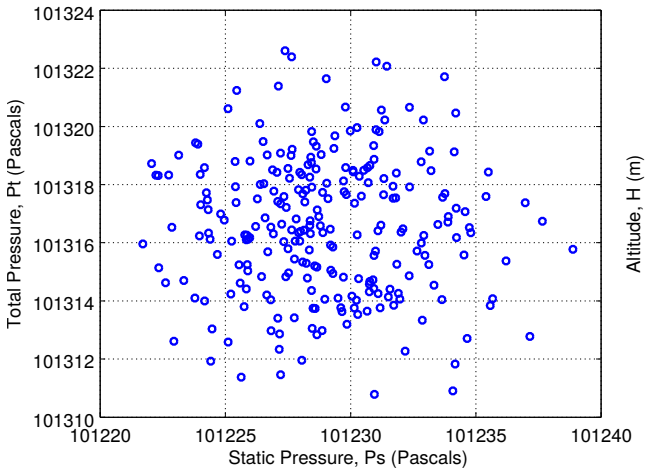
We can convince ourselves that our mean and covariance are correct by generating a collection

of samples from the distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$ (Fig. 6a), propagating each sample through the system (11) (Fig. 6b), and computing the statistics of this new population of samples. This procedure is implemented in the following MATLAB code and utilizes the `randSamplesWithMeanCov` and `meanCovarianceFromSamples` functions provided. The numerically computed statistics are:

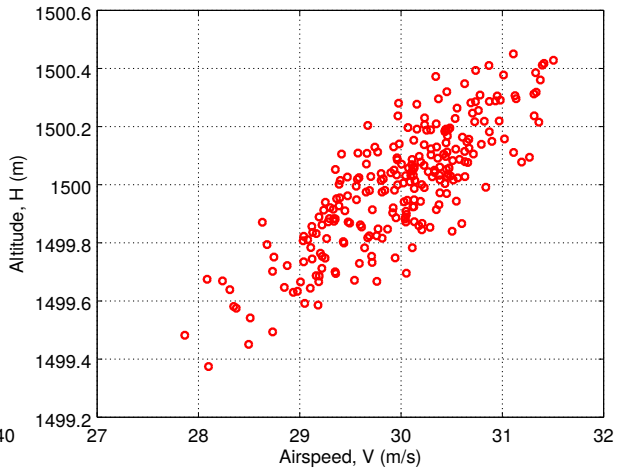
```
mu_VZ_samples =
    29.970
  1499.994

P_VZ_samples =
   0.446167   0.106127
   0.106127   0.037626
```

which is in close agreement with our analytical result. Since we are *approximating* the covariance to first-order (analytically) and using a finite set of samples (numerically), then some differences between the analytical and numerical results are expected.




(a) Generated Samples Of Random Vector $\mathbf{x}$




(b) Propagated Samples Of Random Vector $\mathbf{z}$

Listing 2: Error Propagation Analysis

```
% ENGR 207: Example of Error-Propagation for the Pitot-Static System
clear; close all; clc; % prepare workspace

% the constants are:
k1 = 10.266;
k2 = -6341.726;
k3 = 22632.1;
k4 = 11000;

% the random variable x (static / total pressure inputs) is defined by:
mu_PST = [101228.869   101316.529]; % mean
P_PST = [10 0;0 5] % covarance

% compute the mean and covariance of the random var z (speed / altitude output)
ps = mu_PST(1); % for convenience define ps, pt
pt = mu_PST(2);
mu_VZ = [sqrt(k1*(pt-ps)); k2*log(ps/k3)+k4]
% evaluate jacobian
J(1,1) = -k1/2/sqrt(k1*(pt-ps));
J(1,2) = k1/2/sqrt(k1*(pt-ps));
J(2,1) = k2/ps;
J(2,2) = 0;
P_VZ = J*P_PST*J' % from the error-propogation law

% simulate numerically
numSamples = 250; % number of samples
% generate samples from random var x mean and covariance
[samples] = randSamplesWithMeanCov(mu_PST, P_PST, numSamples);
for i=1:1:numSamples % go through each sample
  ps = samples(1,i); % for convenience define a temporary ps and pt
  pt = samples(2,i);
  samplesProp(1,i) = sqrt(k1*(pt-ps)); % propogate using the system equations
  samplesProp(2,i) = k2*log(ps/k3)+k4;
end
% compute statistics for the propogated samples
[mu_VZ_samples, P_VZ_samples] = meanCovarianceFromSamples(samplesProp)

% plots
figure;
plot(samples(1,:),samples(2,:),'bo','linewidth',2);
set(gca,'FontSize',16)
xlabel('Static Pressure, Ps (Pascals) '); ylabel('Total Pressure, Pt (Pascals)');
grid on;
figure;
plot(samplesProp(1,:),samplesProp(2,:),'ro','linewidth',2);
set(gca,'FontSize',16)
```

```
47 grid on;
48 xlabel('Airspeed, V (m/s)'); ylabel('Altitude, H (m)');
```

---