Efficient computation of robust average in wireless sensor networks using compressive sensing

Chun Tung Chou[1] Aleksandar Ignjatovic[1] Wen Hu[2]

[1] School of Computer Science and Engineering,
University of New South Wales, Australia
{ctchou,ignjat}@cse.unsw.edu.au
[2] Autonomous Systems Laboratory,
CSIRO ICT Centre, Brisbane, Australia
Wen.Hu@csiro.au

Technical Report
UNSW-CSE-TR-0915
29 October 2009

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Wireless sensor networks (WSNs) enable the collection of physical measurements over a large geographic area. It is often the case that we are interested in computing and tracking the spatial-average of the sensor measurements over a region of the WSN. Unfortunately, the standard average operation is not robust because it is highly susceptible to sensor faults (e.g. offset, stuck-at errors etc.) and variation of sensor measurement noises. In this paper, we propose a method to compute a robust average of sensor measurements, which appropriately takes sensor faults and sensor noise into consideration, in a bandwidth- and computational-efficient manner. At the same time, the proposed method can determine which sensors are likely to be faulty. Our method achieves bandwidth efficiency by exploiting compressive sensing. Instead of sending a block of sensor readings to the data fusion centre, each sensor performs random projections (as in compressive sensing) on the data block and sends the results of the projections (which we will refer to as the compressed data) to the data fusion centre. At the data fusion centre, we achieve computational efficiency by working directly with the compressed data, whose dimension is only a fraction of that of the original block of sensor data. In other words, our proposed method will work on the compressed data without decompressing them. By using the compressed data, our proposed method will determine which sensors are likely to be faulty as well as a robust average of the compressed data, which, after decompression (or compressive sensing reconstruction), will yield an approximation of the robust average of the original sensor readings. This means that the data fusion centre will only need to perform decompression once in order to obtain the robust average (rather than decompressing all the compressed data from all the sensors), therefore achieving computational efficiency. We apply our proposed method to the data collected from a number of WSN deployments to demonstrate its efficiency and accuracy.

1 Introduction

This paper considers the application of *compressive sensing* [CRT06, Don06, HN06, CW08] to wireless sensor networks (WSNs) [BJ05, KW05]. Compressive sensing is a collection of recently proposed sampling and signal reconstruction methods in Information Theory and Signal Processing. In particular, a promise of compressive sensing is that it can obtain a good approximation of an unknown data signal by performing a small number of generalised measurements, called *projections*, provided that the unknown signal is compressible. For the case of WSNs, it means that compressive sensing can be used to reduce the amount of data transmissions and therefore lower the energy consumption of the sensor nodes. In this paper, our goal is to use compressive sensing to realise a bandwidth-efficient as well as computationally-efficient method to compute a *robust average* in WSNs.

It is often the case that users of WSNs are interested to compute a spatial average of the sensor network data. Unfortunately, the standard average operation is *not robust* because it is highly susceptible to sensor faults (e.g. offset, stuck-at errors etc.) and variation of sensor measurement noises. Given that sensor faults are common in WSNs [GBS08, NRC$^+$09], it is important to appropriately modify the averaging process to take into account the presence of faults or variation in measurement noise. In this paper, we propose a method to compute a robust average of sensor measurements, which appropriately takes sensor faults and sensor noise into consideration, in a bandwidth- and computational-efficient manner. At the same time, the proposed method can determine which sensors are likely to be faulty. Our proposed method achieves bandwidth efficiency by having the sensors compressed the sensor readings by using projections and achieves computational efficiency by working on the compressed data, which has a smaller dimension compared with the original data.

Figure 1.1 depicts a "standard" method in which compressive sensing can be used to compute a robust average in a WSN. Instead of sending the original sensor measurements, each sensor performs projections on the original sensor measurements which result in a compressed data stream. These compressed data streams are then transmitted over the WSN to reach the data fusion centre where these data streams are decompressed to retrieve the original signals (or more precisely, an accurate approximation of the original signals because the compression is lossy). Based on the decompressed data streams, the data fusion centre can examine which of the signals are faulty or anomalous. Assuming that we are interested in an average of the data, we can now use the decompressed data streams to determine suitable weights for this averaging, e.g. by weighting noisy signals less than the clean signals. The key advantage of this method is that bandwidth will be saved by sending compressed data streams over the network. This is method is proposed in [DWBB06].

In this paper, we examine the alternative method depicted in Figure 1.2. For this method, the sensors again send compressed data streams to the data fusion centre. The main difference is the sequence of operations to be performed at the data fusion centre. Instead of first decompressing the compressed data to obtain the original data stream, our proposed method will work directly with the compressed data without decompressing them. Our proposed method determines a weight for each of the compressed data streams which reflects whether the sensor which produces the compressed data stream may be faulty or not. We then apply these weights to compute a robust average of the compressed data streams. A nice property of this robust average of the compressed data streams is that, upon decompressing (or applying the compressive sensing reconstruction method to) this stream, we will obtain an approximation of the robust average of the original sensor readings. The key advantage of our proposed method is a great reduction of computation requirement at the data fusion centre. Firstly, we work directly with the compressed data, whose dimension is only a fraction of that of the original sensor readings. Secondly, we only need to perform decompression (or compressive sensing reconstruction) once. Consider a large sensor networks with many sensors but only a small fraction of

them are faulty. The method described in Figure 1.1 will need to decompressed all the signals in order to locate the faulty sensors. However, our alternative method in Figure 1.2 achieves the same goal without decompressing any data. Also, our method will only need to perform decompression once. In the case where we are also interested to study the faulty data streams to find out what the sensor faults are, our method will only need to decompress those streams that we think are likely to contain faulty sensor readings. Since only a small fraction of sensors are likely to be faulty, this again represents a huge saving in computation requirement. In addition to using our proposed method for centralised computation in the data fusion centre, we show that, due to the use of compressed data, our averaging method can also be used by sensor nodes to determine faults locally.

Our proposed method exploits properties of compressive sensing, random matrices and a robust averaging algorithm of a specific form in order to achieve the goals mentioned in the last paragraph. Specifically, we use the property that compressive sensing uses a linear compression based on random matrices, which seems to have universal encoding property [CT06]. Due to the Johnson-Lindenstrauss Lemma [DG03] it has been shown in [Ach03] that the random matrices that are used in compressive sensing approximately preserve the ℓ_2 norm. Therefore, by using a robust averaging algorithm which applies linear weighting to the data streams and which uses only ℓ_2 norm of the data streams to compute the weights, we show that such a robust averaging algorithm can work directly with the compressed data.

The rest of this paper is organised as follows. In Section 2, we define the problem setting and give an overview of the proposed method. In Section 3, we present our robust averaging algorithm and show how it can be applied to compressed data directly. We give an interpretation of our averaging algorithm in terms of approximate maximum likelihood. We also study the effect of using compressed data, instead of the uncompressed data, on the averaging algorithm. In Section 4, we apply our proposed method to simulated data of a number of typical sensor faults to show that our proposed method can detect these faults. In Section 5, we apply our proposed method to data obtained from a number of outdoor WSN deployments. We show that our proposed method can work directly with compressed data and there is a reduction in both bandwidth and computation resource requirements. With these outdoor data, we also show that our method is locally stable and the perturbation on the final result due to using compressed data (instead of using the original sensor readings) is small. We also show the resource requirements of running our robust averaging algorithms on wireless sensor nodes. Section 6 discusses related work and Section 7 conclude the paper.

2 Problem statement and solution overview

2.1 Problem setting: Fault detection

We consider a wireless sensor network (WSN) with n sensors indexed by $s = 1, ..., n$. We assume that the sensors are time synchronised and at each time slot t, each sensor performs a measurement. We will use x_{st} to denote the sensor reading by sensor s at time slot t. We assume that the network works on one block of data with m consecutive data points in a block, therefore a block of data consists of $\{x_{st}\}$ with $s = 1, .., n$ and $t = 1, ..., m$.

As depicted in Figure 1.2, the sensors will not send their readings x_{st} directly to the data fusion centre in order to conserve bandwidth and energy. Instead, each sensor will perform projections [CW08] on its sensor readings and send only the results of the projections, which we call either the compressed data stream or the compressed data, to the data fusion centre. The action of, say sensor s,

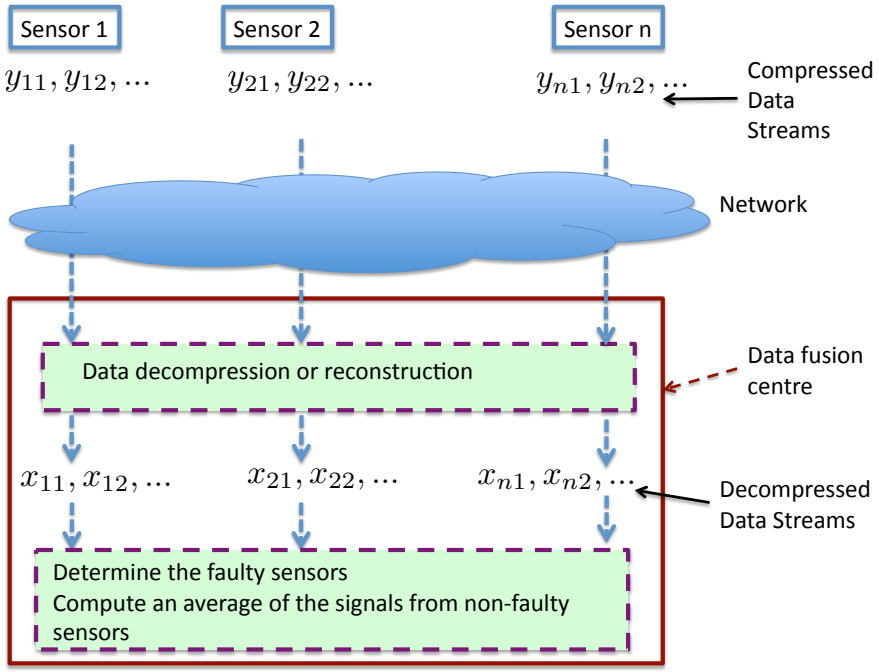

Figure 1.1: This method determines the anomalous signals or faulty sensors after decompressing the signals.

on projecting its data sequence x_{st} $(t = 1, ..., m)$ can be expressed in matrix form, as follows:

$$\underbrace{\begin{bmatrix} y_{s1} \\ y_{s2} \\ \vdots \\ y_{sp} \end{bmatrix}}_{\mathbf{y}_s} = \underbrace{\frac{1}{\sqrt{p}} \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ \phi_{p1} & \phi_{p2} & \cdots & \phi_{pm} \end{bmatrix}}_{\mathbf{\Phi}} \underbrace{\begin{bmatrix} x_{s1} \\ x_{s2} \\ \vdots \\ x_{sm} \end{bmatrix}}_{\mathbf{x}_s} \quad \text{or, more compactly } \mathbf{y}_s = \mathbf{\Phi}\mathbf{x}_s \quad (2.1)$$

The matrix $\mathbf{\Phi}$ is the projection matrix and the vector $\mathbf{y}_s$ is the result of the projection. (Note that all vector and matrix quantities will be written in boldface in this paper.) The theory of compressive sensing says that the elements ϕ_{ij} can be generated from Gaussian distribution with zero mean and unit variance, or symmetric Bernoulli distribution of random numbers $\{+1, -1\}$. We will assume that all the sensors use the same projection matrix for each block of data. Since the network is synchronised, this can be achieved by using a time dependent seed for the pseudo-random number generators. This also means that the sink knows the projection matrix that is being used and the sensors do not have send this information. The parameter p here is the number of projections to be used and we assume that all sensors use the same value of p.

The compressed data stream $\{y_{sk}\}$ (for $k = 1, ..., p$) is to be transmitted by sensor s to the data fusion centre of the network. For practical implementation, it will be easier for the sensors to send

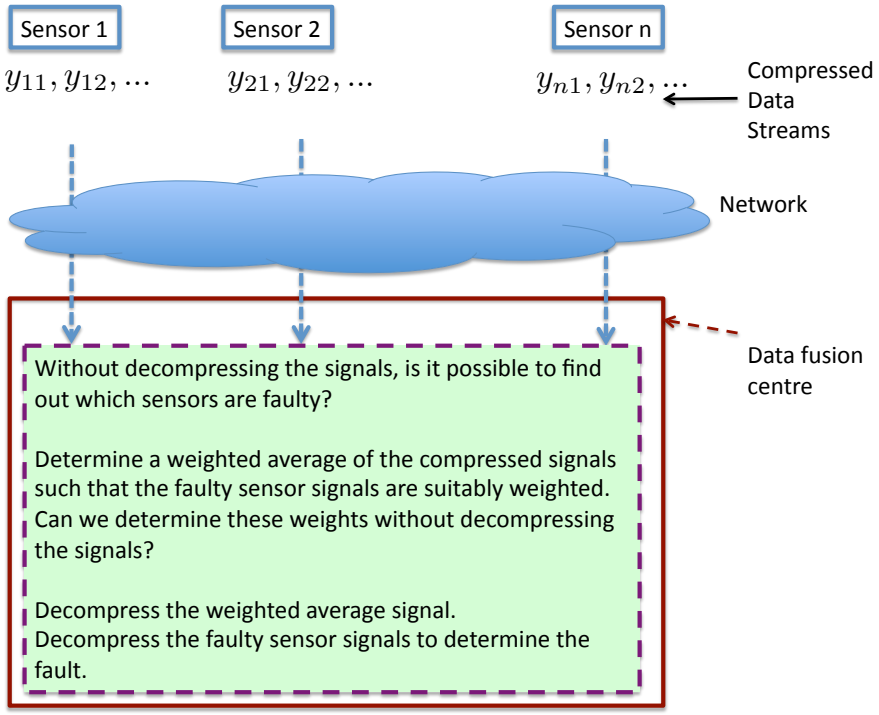

Figure 1.2: The new method that is studied in this paer. This method determines the anomalous signals or faulty sensors before decompressing the signals, i.e. using compressed data.

$\sqrt{p}y_{sk}$ to the data fusion centre and to choose symmetric Bernoulli distribution for the projection matrix, because only integer addition and subtraction are needed to perform at the end of the sensor. Furthermore, for suitable values of p, the transmission of the sequence $\{y_{sk}\}$ will require less number of bits than $\{x_{st}\}$. If the analogue-to-digital convertor on the sensor uses b bits, then sending the original sequence will need mb bits. However, sending the compressed sequence (assuming the aforementioned practical implementation) will require $p(b + 1 + \lceil \log_2 m \rceil)$ bits where $\lceil u \rceil$ denote the smallest integer larger than or equal to u because the range of y_{sk} is $[-m(2^b - 1), m(2^b - 1)]$. Thus, bandwidth is saved if $p(b + 1 + \lceil \log_2 m \rceil) < mb$ and we will show that using data from WSN deployments in Section 5. Note that we will continue to write projection using the convention in equation (2.1) which is commonly used in compressive sensing literature but noting that the practical implementation may be slightly different.

Based on the above setting, we are now ready to define the **fault detection problem**: Assuming that the compressed data streams $\{y_{sk}\}$ (for $s = 1, ..., n$ and $k = 1, ..., p$) are available at the data fusion centre, determine which sensors are faulty without decompressing the compressed data streams.

2.2 Problem setting: robust averaging

Our second goal is to obtain a robust average of the data streams from the sensor networks. In order to illustrate what we mean by that, we begin with the case where all sensors are working properly

(i.e. no sensor faults) and the noise affecting each sensor is independently and identically distributed (i.i.d.) with the zero mean and identical variance. Under these assumptions, and assuming that the data fusion centre has the data streams $\{x_{st}\}$, then the average reading over the sensor network at each time t is given by the arithmetic mean:

$$a_t \;=\; \frac{1}{n}\sum_{s=1}^{n} x_{st} \tag{2.2}$$

However, the arithmetic mean will no longer give a meaningful result when the noise variances are different or when some sensors are faulty (e.g. offset). It is well known in statistical estimation that the arithmetic mean can be seriously affected by outliers [Hub04]. Therefore, instead of using arithmetic mean, we seek to use a robust average of the data that correctly accounts for the heterogeneous noise variance and at the same time reduces the effect of faulty sensor readings. This robust average r_t is given by

$$r_t \;=\; \sum_{s=1}^{n} w_s x_{st} \tag{2.3}$$

where $w_s \geq 0$ (for $s = 1, ..., n$) and $\sum_{s=1}^{n} w_s = 1$. The weights w_s should have the property that they are small for those sensors that are faulty or have a larger noise variance. (Note that for each block of data, a weight is assigned to each sensor. The weight for each sensor can change from a block of data to another since a sensor may only display faults over a limited period of time. Note also that there is a debate in statistics on whether one should deal with outliers by first detecting them and then use only the cleaned data for estimation. We will not go in this debate but remark that we follow the robust statistics philosophy in [Hub04].) Hence, if we can determine the weight w_s for each sensor, we can also realise our goal of fault detection because a faulty sensor will have a smaller w_s compare with the other sensors. This is to be done by using the compressed data $\{y_{sk}\}$.

Let us, for the time being, assume that we have a method to determin these weights w_s. We will use $\mathbf{r}$ to denote the column vector $[r_1 \;\; r_2 \;\; ... \;\; r_m]^T$ (where ^{T} denotes matrix transpose). The robust average in equation (2.3) can be written in vector notation as:

$$\mathbf{r} \;=\; \sum_{s=1}^{n} w_s \mathbf{x}_s \tag{2.4}$$

Let us pre-multiply both sides of the above equation by the projection matrix $\mathbf{\Phi}$, we have

$$\mathbf{\Phi r} \;=\; \sum_{s=1}^{n} w_s \mathbf{\Phi x}_s \;=\; \sum_{s=1}^{n} w_s \mathbf{y}_s \tag{2.5}$$

where we have used the definition of the projected data streams $\mathbf{y}_s$ given in equation (2.1). Note in particular that $\mathbf{\Phi r}$ is in fact the compressed version of the robust average $\mathbf{r}$. Therefore, if the weights w_s are known, then one can obtain the compressed version of the robust average by applying the same weights to the compressed data streams $\mathbf{y}_s$. This means that one can readily obtain $\mathbf{r}$ by decompressing $\sum_{s=1}^{n} w_s \mathbf{y}_s$. This derivation also shows three of the ingredients which are needed for our scheme to work (note: there are a few more, for the estimation of w_s, which will be explained later): (1) The averaging operation must be linear. (2) The compression operation must be linear, which is the case for compressive sensing. (3) All sensors must use the same projection matrix, which can be realised by synchronising the sensor nodes.

Before moving on to the next section, where we will explain how the weights w_s can be computed from the compressed data $\{y_{sk}\}$, we will first explain how decompression (or reconstruction) can be done. For our case, decompression can be realised by using any compressive sensing reconstruction algorithm, e.g. basis pursuit [CT05], greedy pursuit [TG07]. For example, if we know that the robust average is sparse in the basis $\mathbf{\Psi} \in \mathbb{R}^{m\times m}$ (where the columns are the basis vectors), then we can obtain an approximation of $\mathbf{r}$ by:

$$\hat{\mathbf{r}} \;=\; \mathbf{\Psi}\hat{\mathbf{z}} \text{ where } \hat{\mathbf{z}} = \arg\min_{\mathbf{z}\in\mathbb{R}^m} \|\mathbf{z}\|_1 \text{ subject to } \mathbf{\Phi}\mathbf{\Psi}\mathbf{z} = \sum_{s=1}^{n} w_s \mathbf{y}_s \tag{2.6}$$

However, in cases where we do not know in which basis the robust average $\mathbf{r}$ is sparse, we use multiple bases in the decompression or reconstruction. Assuming that we know that the robust average may be sparse in the basis $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$, we compute an approximation of $\mathbf{r}$ by:

$$\hat{\mathbf{r}} \;=\; [\mathbf{\Psi}_1 \;\; \mathbf{\Psi}_2]\,\hat{\mathbf{z}} \text{ where } \hat{\mathbf{z}} = \arg\min_{\mathbf{z}\in\mathbb{R}^{2m}} \|\mathbf{z}\|_1 \text{ subject to } \mathbf{\Phi}\,[\mathbf{\Psi}_1 \;\; \mathbf{\Psi}_2]\,\mathbf{z} = \sum_{s=1}^{n} w_s \mathbf{y}_s \tag{2.7}$$

2.3 Estimating weights w_s from the compressed data

The aim of this section is to give an overview of how the weights w_s can be computed from the compressed data. The details of the algorithm will be given in Section 3. The key idea that we want to demonstrate in this section is that, if the algorithm for estimating the weights w_s satisfies certain properties, then one can in fact use exactly the same algorithm to estimate w_s from the original data $\mathbf{x}_s$ or from the compressed data $\mathbf{y}_s$.

Let us use $\mathcal{A}$ to describe the algorithm (which will be described in Section 3) which takes the original data streams $\mathbf{x}_s$ as the inputs and computes the weight w_s to be used for robust averaging. The algorithm $\mathcal{A}$ is in the form of a fixed point iteration in the (unknown) robust average $\mathbf{r}$. Specifically, the unknown robust average $\mathbf{r}$ is the solution of a fixed point equation of the form (compare with equation (2.4))

$$\mathbf{r} \;=\; \sum_{s=1}^{n} w_s(\|\mathbf{x}_1-\mathbf{r}\|_2, \|\mathbf{x}_2-\mathbf{r}\|_2, \ldots, \|\mathbf{x}_n-\mathbf{r}\|_2)\;\; \mathbf{x}_s \tag{2.8}$$

where we have used $w_s(\|\mathbf{x}_1-\mathbf{r}\|_2, \|\mathbf{x}_2-\mathbf{r}\|_2, \ldots, \|\mathbf{x}_n-\mathbf{r}\|_2)$ to show explicitly that w_s is a function of the ℓ_2-norm of $(\mathbf{x}_s-\mathbf{r})$ (for $s=1,...,n$).

By pre-multiplying equation (2.8) with $\mathbf{\Phi}$ and using equation (2.1), we have (c.f. equation (2.5)):

$$\tilde{\mathbf{r}} \;=\; \sum_{s=1}^{n} w_s(\|\mathbf{x}_1-\mathbf{r}\|_2, \|\mathbf{x}_2-\mathbf{r}\|_2, \ldots, \|\mathbf{x}_n-\mathbf{r}\|_2)\;\; \mathbf{y}_s \tag{2.9}$$

$$\text{where } \tilde{\mathbf{r}} \;\stackrel{\text{def}}{=}\; \mathbf{\Phi}\mathbf{r} \tag{2.10}$$

Note that $\tilde{\mathbf{r}}$ is in fact the compressed version of the robust average $\mathbf{r}$ that we want to find since once we can determine $\tilde{\mathbf{r}}$, we can determine $\mathbf{r}$ using the compressive sensing reconstruction algorithm mentioned earlier. Unfortunately, we cannot use equation (2.9) because it needs the data sequence $\{\mathbf{x}_s\}$. However, if it is true that

$$\|\mathbf{x}_s-\mathbf{r}\|_2 \;\approx\; \|\mathbf{\Phi}(\mathbf{x}_s-\mathbf{r})\|_2 = \|\mathbf{y}_s-\tilde{\mathbf{r}}\|_2 \;\; \forall s = 1,2,,,,n, \tag{2.11}$$

then equation (2.9) can be written as:

$$\tilde{\mathbf{r}} \approx \sum_{s=1}^{n} w_s(\|\mathbf{y}_1 - \tilde{\mathbf{r}}\|_2, \|\mathbf{y}_2 - \tilde{\mathbf{r}}\|_2, \ldots, \|\mathbf{y}_n - \tilde{\mathbf{r}}\|_2) \ \mathbf{y}_s \tag{2.12}$$

This shows that if the approximation in equation (2.11) holds, then one can solve for $\tilde{\mathbf{r}}$ from equation (2.12) using only the compressed data $\{\mathbf{y}_s\}$. Fortunately, the approximation (2.11) holds because of the choice of the projection matrix $\mathbf{\Phi}$ and the Johnson-Lindenstrauss Lemma. We quote the following version of the Johnson-Lindenstrauss Lemma from [Ach03].

Lemma 1 *(Johnson-Lindenstrauss Lemma [Ach03]) Let Q be an arbitrary set of q points in $\mathbb{R}^m$, represented by the vectors $\mathbf{d_1}, \mathbf{d_2}, ..., \mathbf{d_q}$. Given $\epsilon, \beta > 0$, let*

$$p_0 = \frac{4 + 2\beta}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \log q \tag{2.13}$$

For integer $p \geq p_0$, let $\mathbf{\Phi}$ be a random $p \times m$ matrix whose elements are generated from either: (1) a Gaussian distribution with zero mean and unit variance, or (2) a symmetric Bernoulli distribution of random numbers $\{+1, -1\}$, then with probability at least $1 - q^{-\beta}$, the following holds

$$(1 - \epsilon)\|\mathbf{d_i} - \mathbf{d_j}\|_2^2 \leq \|\frac{1}{\sqrt{p}}\mathbf{\Phi}(\mathbf{d_i} - \mathbf{d_j})\|_2^2 \leq (1 + \epsilon)\|\mathbf{d_i} - \mathbf{d_j}\|_2^2 \ \forall \mathbf{d_i}, \mathbf{d_j} \in Q \tag{2.14}$$

□

Since the projection matrices that are commonly used in compressive sensing obeys the Johnson-Lindenstrauss Lemma, the approximation in (2.11) holds. Therefore, if we can find an algorithm which is in the form of fixed-point equation (2.8), then we can input the compressed data $\mathbf{y_s}$ into this algorithm to obtain the weights w_s and the compressed weighted average $\tilde{\mathbf{r}}$. Note that in applying the Johnson-Lindenstrauss Lemma to (2.8), we need $q = n + 1$.

Based on the above discussion, we see that there are two additional ingredients that are needed in order to obtain the weights from the compressed data: (1) The projection matrix needs to satisfy the Johnson-Lindenstrauss Lemma. (2) The algorithm that determines the weights from the data can only use ℓ_2-norm of differences in $\mathbb{R}^m$ where m is the dimension of the original data vectors. We will provide the details of one such algorithm in the next section.

3 Fixed-point iteration for fault detection

In this section, we will derive an algorithm to detect whether sensors are faulty by determining a weight w_s for each sensor. The weight w_s will be used to weight the contribution of the signal from sensor s to the robust average in equation (2.3). Given there are n sensors, if sensor s is faulty, we will expect that w_s will be far smaller than $\frac{1}{n}$.

As discussed in Section 2, the algorithm in this section is of the form of equation (2.8). In this section, we will describe the algorithm assuming that the available data is $\{x_{st}\}$. Note that the algorithm can be applied to the compressed data simply by replacing $\{x_{st}\}$ with $\{y_{sk}\}$. We will study the perturbation on the weights w_s when the compressed data $\{y_{sk}\}$ is used instead of the original sensor measurements $\{x_{st}\}$ in Sections 3.3. We will also discuss a statistical interpretation of the robust averaging algorithm in this section.

3.1 Fixed point iteration

Our goal is to derive an algorithm from the data $\{x_{st}\}$ a set of weights $w_1, ..., w_n$ ($w_s \geq 0$ and $\sum_{i=1}^{n} w_s = 1$) where each weight reflects the trustworthiness of each sensor. Assuming for the time being that these weights are available, then we know the average behaviour of the sensor over this block of data is given by:

$$r_t = \sum_{s=1}^{n} w_s x_{st} \tag{3.1}$$

Recall that the weight w_s reflects how much we can trust sensor s. If sensor s is not faulty, then we expect that its measurements should not deviate much from the average behaviour of the other sensors. On the other hand, if sensor s is faulty, then its measurements should deviate somewhat from the average behaviour. Given that r_t is the average sensor behaviour at time t, we can use r_t as the expected average behaviour. (Note that r_t is what we want to estimate and is not known. The argument here assumes that r_t is known and we will show how r_t can be calculated later.) Therefore, if $x_{st} \approx r_t$, then sensor t is likely to be a working sensor and we expect w_s should be above $\frac{1}{n}$. On the other hand, if x_{st} differs from r_t by a great deal, then sensor s is likely to be faulty and w_s should be small. Based on this argument, we choose:

$$w_s \propto \frac{1}{\frac{\sum_{t=1}^{m}(x_{st} - r_t)^2}{\sum_{i=1}^{n}\sum_{t=1}^{m}(x_{it} - r_t)^2} + \lambda} \tag{3.2}$$

Note that the quantity $\sum_{t=1}^{m}(x_{st} - r_t)^2$ in the above expression is a measure of the deviation of the reading of sensor s (x_{st}) from the average behaviour r_t. If $\sum_{t=1}^{m}(x_{st} - r_t)^2$ is large, then w_s will be small and vice versa. The parameter λ is a small positive constant whose role will be clarified in Section 3.2.

Since the weights w_s must sum up to unity, we have

$$w_s = \frac{\frac{1}{\frac{\sum_{t=1}^{m}(x_{st} - r_t)^2}{\sum_{i=1}^{n}\sum_{t=1}^{m}(x_{it} - r_t)^2} + \lambda}}{\sum_{j=1}^{n} \frac{1}{\frac{\sum_{t=1}^{m}(x_{jt} - r_t)^2}{\sum_{i=1}^{n}\sum_{t=1}^{m}(x_{it} - r_t)^2} + \lambda}} \tag{3.3}$$

We have now derived both equations that we need, namely equations (3.1) and (3.3). Note that equation (3.1) expresses r_t as a function w_s while equation (3.3) expresses w_s in terms of r_t. Therefore, these two equations together form a fixed-point equation. The idea is to iterate between these two equations to estimate r_t using the following algorithm:

Let $w_s^{[\ell]}$ and $r_t^{[\ell]}$ be, respectively, the values of w_s and r_t at the ℓ-th iteration. Perform the following:

1. Initialise $\ell = 0$. $w_s^{[\ell]} = \frac{1}{n}$.
2. Compute $r_t^{[\ell+1]}$ from $w_s^{[\ell]}$ using equation (3.1).
3. Compute $w_s^{[\ell]}$ from $r_t^{[\ell]}$ using equation (3.3).
4. $\ell \leftarrow \ell + 1$.
5. If the iteration has converged, stop; otherwise, go back to Step 2.

If we use $\{x_{st}\}$ as the input of this algorithm, then the end result r_t is the robust average of the network data and w_s reflects how much we can trust sensor s. If w_s is much smaller than $\frac{1}{n}$, then it is likely that sensor s is faulty. If, instead, we use $\{y_{st}\}$ as the input of this algorithm, then we expect an approximation of $\tilde{\mathbf{r}}$ as well as that of w_s as the outputs.

Local sensor fault detection

We have assumed that all the calculations (robust averaging and compressive sensing reconstruction) will be carried out at the data fusion centre. The compressive sensing reconstruction algorithm is computational expensive for a typical wireless sensor. However, it is possible that the robust averaging algorithm can be carried out locally on a sensor. In this case, a sensor collects, via overhearing its neighbours' communication with the data fusion centre, the compressed data vectors $\mathbf{y_s}$ of its neighbours. The sensor then inputs these compressed data vector into the robust averaging fixed point iteration algorithm to compute the weights w_s for itself and all its neighbouring sensors. The sensor can decide, based on these weights, whether itself or any of its neighbouring sensors may be faulty. We will show in Section 5.3 results on the resource requirements of running the robust averaging algorithm on wireless sensor nodes.

3.2 Interpretation

Maximum likelihood interpretation when $\lambda = 0$

In this section, we provide an interpretation of the fixed point iteration algorithm as an approximate maximum likelihood estimator. Note that, when $\lambda = 0$, we can write the fixed point equation in terms of r_t alone by eliminating w_s from equations (3.3) and (3.1), as follows:

$$r_t = \sum_{s=1}^{n} \frac{\frac{1}{\sum_{t=1}^{m}(x_{st} - r_t)^2}}{\sum_{j=1}^{n} \frac{1}{\sum_{t=1}^{m}(x_{jt} - r_t)^2}} x_{st} \tag{3.4}$$

It can be shown that the above fixed point equation is identical to the first-order stationary condi-

tion of the following maximisation problem:

$$\max_{r_1,r_2,\ldots,r_t} \sum_{s=1}^{n} -\log(\sum_{t=1}^{m}(x_{st}-r_t)^2) \tag{3.5}$$

It can further be shown that the objective function in the above maximisation problem is in fact a log-likelihood function. Let us assume that the sensor reading of sensor s at time t, x_{st}, is generated from the Gaussian distribution with mean r_t and variance σ_i^2, i.e. $x_{st} \sim \mathcal{N}(r_t, \sigma_s^2)$. In other words, the model assumes that all sensors should have the same mean reading r_t at time t but the measurement noise of different sensors can be different. It can be shown that the log-likelihood function for the data sequences $\{x_{st}\}$ with r_t and σ_s as the unknown parameters is:

$$L \;=\; \log(\text{Prob}(\{x_{st}\}|\{r_t,\sigma_s\})) \;=\; \text{Constant} \; - m\sum_{s=1}^{n}\log(\sigma_s) - \sum_{s=1}^{n}\sum_{t=1}^{m}\frac{(x_{st}-r_t)^2}{2\sigma_s^2} \tag{3.6}$$

By differentiating the log-likelihood function L with respect to σ_s, we have at the maximum of L,

$$\sigma_s^2 \;=\; \frac{1}{m}\sum_{t=1}^{m}(x_{st}-r_t)^2 \tag{3.7}$$

After substituting σ_s^2 from equation (3.7) into equation (3.6), we have

$$L \;=\; \text{Constant} \; - \sum_{s=1}^{n}\log(\sum_{t=1}^{m}(x_{st}-r_t)^2) \tag{3.8}$$

Therefore, for $\lambda = 0$, the fixed point iteration can be interpreted as a maximum likelihood estimator of the unknown average r_j. Note in particular that the sensors can be affected by noise of different variances. This means that the fixed point iteration proposed earlier will weight the sensor readings according to the correct size of the noise variance. However, the function (3.8) can be unbounded, therefore a small positive constant λ is needed to improve the algorithm's numerical properties.

Note that the above maximum likelihood interpretation assumes that the fault is a Gaussian noise. When this assumption does not hold, maximum likelihood estimator can be viewed as a minimiser of the Kullback-Leibler divergence between the true and assumed distributions [Whi82].

The $\lambda \neq 0$ case

For a small positive constant λ, the fixed point iteration algorithm can therefore be interpreted as an approximate maximum likelihood estimator. Therefore, the fixed point iteration will interpret offset or stuck-at-fault as having a large deviation from the average behaviour (in other words, a large variance σ_s^2) and will therefore weight those sensors with these faults lightly. In the next section, we will study the behaviour of this algorithm under a number of commonly found fault models in WSNs.

Note that if λ is a large number, then the weights w_s in equation (3.3) is almost equal to $\frac{1}{n}$, therefore it is not recommended to choose a large λ. We will also prove in Section 4.1 that, for stuck-at faults, a small value of λ will only create a small bias in the estimation of the weights w_s.

We will study the convergence behaviour of the fixed point iteration using data collected from various outdoor WSN deployments in Section 5. We find that the fixed point iteration converges quickly and this makes it ideal for implementation in wireless sensor nodes which have only limited computation power.

Maximum likelihood interpretation when compressed data is used

We argue earlier that, when $\lambda = 0$, the fixed point iteration for robust averaging can be interpreted as a maximum likelihood estimator where the original sensor measurements x_{st} are generated from $\mathcal{N}(r_t, \sigma_s^2)$ with unknown parameters r_t and σ_s^2. The maximum likelihood interpretation continues to hold when the compressed data y_{sk} is used instead, except that the variance becomes larger. The following derivation will also show the trade-off between efficiency and accuracy in using compressed data.

Let us decompose x_{st} as the sum of r_t and a noise term e_{st}, as follows:

$$x_{st} \quad = \quad r_t + e_{st} \tag{3.9}$$

where $e_{st} \sim \mathcal{N}(0, \sigma_s^2)$. The compressed data y_{sk} can be written as:

$$y_{sk} \quad = \quad \frac{1}{p}\sum_{t=1}^{m} \phi_{kt} x_{st} \quad = \quad \underbrace{\sum_{t=1}^{m} \frac{1}{p}\phi_{kt} r_t}_{\tilde{r}_k} + \underbrace{\sum_{t=1}^{m} \frac{1}{p}\phi_{kt} e_{st}}_{\tilde{e}_{sk}} \tag{3.10}$$

Note that the noise term $\tilde{e}_{sk}$ is a sum of Gaussian distributed random variables, therefore $\tilde{e}_{sk}$ is also Gaussian distributed. By using the facts that (1) ϕ_{kt} is a random variable with zero mean and unit variance; (2) $\phi_{k_1 t_1}$ is independent of $\phi_{k_2 t_2}$ if either $k_1 \neq k_2$ or $t_1 \neq t_2$; (3) e_{st_1} is independent of e_{st_2}; (4) ϕ_{kt} is independent of e_{st}; it can be shown that

$$\mathbb{E}[\tilde{e}_{sk}] \quad = \quad 0 \; \forall s = 1, .., n, k = 1, ..., p \tag{3.11}$$

$$\mathbb{E}[\tilde{e}_{s_1 k_1} \tilde{e}_{s_2 k_2}] \quad = \quad \begin{cases} \frac{m}{p}\sigma_s^2 & \text{for } s_1 = s_2 \text{ and } k_1 = k_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where $\mathbb{E}$ denotes expectations. Therefore, if the compressed data $\{y_{sk}\}$ is used to determine the robust average instead, the assumption that the noise affecting each compressed datum y_{sk} is corrupted by an independent Gaussian noise continues to hold. This means that the maximum likelihood interpretation continues to hold even if the compressed data $\{y_{sk}\}$ is used instead. Note that the variance of the noise affecting the compressed datum y_{sk} is $\frac{m}{p}\sigma_s^2$. Since $m > p$, this noise variance is larger than that affecting the original sensor measurement. This derivation also shows the price that is being paid by using the compressed data for robust averaging is an increase in variance. This also shows that there is a trade-off between bandwidth-efficiency and computation-efficiency (which is achieved by using a small p) and accuracy (which is by using a large p).

3.3 Effect of using compressed data on the weights w_s

The derivation of the fixed point iteration in Section 3.1 assumes that the sensor readings $\{x_{st}\}$ are available to compute the weights w_s. Since our goal is to compute these weights from the compressed data $\{y_{sk}\}$ and use the Johnson-Lindenstrauss approximation (2.11), the aim of this section is to study the perturbation on the weights w_s due to the use of compressed data. We first set up the framework for performing the perturbation analysis.

In Section 3.1, the fixed-point iteration is described as an iteration between $\mathbf{w}$ and $\mathbf{r}$ using equations (3.1) and (3.3). By eliminating the variable $\mathbf{r}$, it can be shown that the fixed-point iteration can

be written as the iteration of the following two set of equations:

$$v_s = \|\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i}\|_2^2 \quad \text{for } s = 1, ..., n \tag{3.13}$$

$$w_s = \frac{\dfrac{1}{\dfrac{v_s}{\sum_{j=1}^{n} v_j} + \lambda}}{\sum_{i=1}^{n} \dfrac{1}{\dfrac{v_i}{\sum_{j=1}^{n} v_j} + \lambda}} \quad \text{for } s = 1, ..., n \tag{3.14}$$

We will use $\mathbf{v}$ to denote the vector whose s-th element is v_s. In order to facilitate the perturbation analysis, we define two operators. Let $\mathbf{T_{wv}}$ denote the operator that maps $\mathbf{w}$ to $\mathbf{v}$ defined by equation (3.13) and $\mathbf{T_{vw}}$ denote the operator that maps $\mathbf{v}$ to $\mathbf{w}$ by using equation (3.14). If the fixed point of equations (3.13) and (3.14) are given by $\mathbf{w^0}$ and $\mathbf{v^0}$, then

$$\mathbf{v^0} = \mathbf{T_{wv}}(\mathbf{w^0}) \tag{3.15}$$

$$\mathbf{w^0} = \mathbf{T_{vw}}(\mathbf{v^0}) \tag{3.16}$$

Let us consider the situation if compressed data $\mathbf{y_s}$ is used instead. In this case, equation (3.13) is replaced by

$$v_s = \|\mathbf{y_s} - \sum_{i=1}^{n} w_i \mathbf{y_i}\|_2^2 = \|\mathbf{\Phi}(\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i})\|_2^2 \tag{3.17}$$

Note that the right-hand-side (RHS) of (3.17) is an approximation of the RHS of (3.13) based on the Johnson-Lindenstrauss approximation. This approximation can be written as

$$\|\mathbf{\Phi}(\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i})\|_2^2 = (1 + \epsilon_s)\|\mathbf{x_s} - \sum_{i=1}^{n} w_i \mathbf{x_i}\|_2^2 \tag{3.18}$$

where ϵ_s is the relative error in using the Johnson-Lindenstrauss approximation (compare with equation (2.14)).

If compressed data $\mathbf{y_s}$ is used, the fixed point algorithm iterates between equations (3.17) and (3.14). Let $\hat{\mathbf{T}}_{\mathbf{wv}}$ denote the operator that maps $\mathbf{w}$ to $\mathbf{v}$ defined by equation (3.17). Let $\mathbf{w^1}$ and $\mathbf{v^1}$ be the solution of the fixed-point iterations using compressed data, then we have:

$$\mathbf{v^1} = \hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) \tag{3.19}$$

$$\mathbf{w^1} = \mathbf{T_{vw}}(\mathbf{v^1}) \tag{3.20}$$

Our goal is to derive the perturbation on the weights $\Delta\mathbf{w} = \mathbf{w^1} - \mathbf{w^0}$. In addition, we let $\Delta\mathbf{v} = \mathbf{v^1} - \mathbf{v^0}$.

Local perturbation analysis

In this section, we derive the first order approximation for the perturbation $\Delta\mathbf{w}$ assuming that ϵ_s is small. First, we use equations (3.15) and (3.19) to obtain:

$$\Delta\mathbf{v} = \hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^0}) \tag{3.21}$$

$$= (\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1})) + (\mathbf{T_{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^0})) \tag{3.22}$$

$$\approx \mathbf{V^1 e} + \left.\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}\right|_{\mathbf{w^0}} \Delta\mathbf{w} \tag{3.23}$$

where $\mathbf{V^1}$ is a diagonal matrix obtained by "diagonalising" the vector $\mathbf{v^1}$, $\mathbf{e}$ is a vector whose s-th element is ϵ_s, and $\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}$ is a Jacobian matrix. (Expression of the Jacobian matrix is given in Appendix A.) Note that the first term in equation (3.23) represents the error in using the Johnson-Lindenstrauss approximation and the second term is obtained from Taylor series expansion.

By using equations (3.16) and (3.20), and Taylor series expansion, it can be shown that

$$\Delta \mathbf{w} \approx \left. \frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}} \right|_{\mathbf{v^0}} \Delta \mathbf{v} \tag{3.24}$$

Given equations (3.23) and (3.24), the perturbation on the weights $\Delta \mathbf{w}$ due to using the compressed data can be written as:

$$\Delta \mathbf{w} = \underbrace{\left(\mathbf{I} - \underbrace{\left. \frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}} \right|_{\mathbf{v^0}} \left. \frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}} \right|_{\mathbf{w^0}}}_{\mathbf{F}} \right)^{-1} \left. \frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}} \right|_{\mathbf{v^0}} \mathbf{V^1}}_{\mathbf{G}} \mathbf{e} \tag{3.25}$$

where $\mathbf{I}$ denotes the identity matrix. If the perturbation ϵ_s obeys $|\epsilon_s| \leq \epsilon$, i.e. $\|\mathbf{e}\|_\infty \leq \epsilon$, then by using standard result in linear algebra on the induced ∞-norm [HJ90], the largest possible perturbation in the weights w_s is given by

$$\|\Delta \mathbf{w}\|_\infty = \|\mathbf{G}\|_\infty \epsilon \tag{3.26}$$

Therefore the ∞-norm of the matrix $\mathbf{G}$ tells us how much perturbation we expect on the weight w_s. If $\|\mathbf{G}\|_\infty$ is small, then the approximation (2.11) will cause only small change in w_s. In addition, the matrix $\mathbf{F}$ gives information on the local stability of the fixed point. In addition, if all the eigenvalues of the matrix $\mathbf{F}$ is bounded by unity, then the fixed point is locally stable [ÅM08].

Note that the above perturbation analysis assumes that we evaluate the Jacobian matrices at the fixed point $\mathbf{w^0}$ and $\mathbf{v^0}$ determined by the original data $\{\mathbf{x_s}\}$. Since the original data is assumed to be not available, we will evaluate the size of the perturbation by using the fixed point given by the compressed data instead. This is straightforward for the Jacobian matrix $\frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}}$. However, the Jacobian matrix $\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}}$ depends on the data; in fact, the (i,j)-element of the matrix is:

$$\left[\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}} \right]_{ij} = 2\mathbf{x_j}^T (\sum_{q=1}^{n} w_q \mathbf{x_q} - \mathbf{x_i}) \tag{3.27}$$

Since the projection matrix $\mathbf{\Phi}$ approximately preserves the ℓ_2-norm with a high probability, we expect that the projection matrix $\mathbf{\Phi}$ will also preserve the inner product because the inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$ can be computed by:

$$\mathbf{u}^T \mathbf{v} = \frac{\|\mathbf{u}+\mathbf{v}\|_2^2 - \|\mathbf{u}-\mathbf{v}\|_2^2}{4} \tag{3.28}$$

Therefore, the Jacobian $\frac{\partial \mathbf{T_{vw}}}{\partial \mathbf{v}}$ can be evaluated by using the compressed data by:

$$\left[\frac{\partial \mathbf{T_{wv}}}{\partial \mathbf{w}} \right]_{ij} \approx 2\mathbf{y_j}^T (\sum_{q=1}^{n} w_q \mathbf{y_q} - \mathbf{y_i}) \tag{3.29}$$

In Section 5, we will evaluate the size of perturbation and local stability of the fixed point method by using data collected from WSN deployments. We will also compare the accuracy of evaluating the perturbation and stability by using the original data and the compressed data in Section 5.

Large perturbation analysis

The local perturbation analysis is Section 3.3 applies when the value of ϵ is small. However, for the practical situation considered in this paper, this is generally not true. For example, for $m = 200$ and $p = 80$, we use Monte-Carlo simulation to find that there is a 95% probability that ϵ is in the range of $[-0.3, 0.3]$. Therefore, in this section, we derive an expression for the effect on w_s when the perturbation ϵ is large using the functional method for stability analysis of nonlinear feedback control systems [Zam66].

Proposition 1 *The perturbation of* $\|\mathbf{w^1} - \mathbf{w^0}\|$ *obeys*

$$\|\mathbf{w^1} - \mathbf{w^0}\| \quad \leq \quad \frac{\|\mathbf{T}(\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T}_{\mathbf{wv}})\|}{1 - \|\mathbf{T}(\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T}_{\mathbf{wv}})\|_L}\|\mathbf{w^0}\| \tag{3.30}$$

where $\mathbf{T} = (\mathbf{T_{vw}}^{-1} - \mathbf{T_{wv}})^{-1}$ *and* $\| \bullet \|_L$ *denotes the Lipschitz operator norm. Note that the norm can be* $1-,2-$ *or* ∞*-norm, as long as the same type of norm is used.*

The proof of this proposition can be found in Appendix B. We will use this result to study the effect of the perturbation on the weights using data from an outdoor WSN in Section 5.

3.4 Choice of robust estimator

There are a number of other possible choices of robust estimators in the literature. One such possibility is the ℓ_1-norm estimator, which when specialise in our case, aims to find $\mathbf{r} \in \mathbb{R}^m$ such that

$$\min_{\mathbf{r}} \sum_{s=1}^{n} \|\mathbf{x}_s - \mathbf{r}\|_1 \tag{3.31}$$

In fact, the t-the element of $\mathbf{r}$ equals to the median of $x_{1t}, x_{2t}, ..., x_{nt}$. We will write this as

$$\mathbf{r} \quad = \quad \text{median}(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}) \tag{3.32}$$

It is possible to apply the framework of Figure 1.2 to the ℓ_1-norm estimator too since Cauchy distributed random projection matrices [LHC07, Ind06] or Pareto distributed sparse random matrices [Li07] are known to preserve the ℓ_1-norm. The procedure is as follows:

1. Each sensor uses the same Cauchy distributed or Pareto distributed sparse random matrix $\mathbf{\Phi}$ and compute $\mathbf{y_s} = \mathbf{\Phi x_s}$. Sensor s $(= 1, ..., n)$ sends $\mathbf{y_s}$ to the data fusion centre.

2. The fusion centre computes $\tilde{\mathbf{r}} = \text{median}(\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n})$. (Note that other estimators can be used in this step, see [LHC07].)

3. The robust ℓ_1 estimate $\mathbf{r}$ can be reconstructed from $\mathbf{\Phi}$ and $\tilde{\mathbf{r}}$.

Note that there are two issues with using the above estimation procedure. Firstly, the procedure may not be bandwidth efficient because floating point numbers will need to be transmitted to the data fusion centre. Secondly, the reconstruction step can be numerically sensitive because the projection matrix $\mathbf{\Phi}$ may not satisfy the *Restricted Isometry Properties* [CT05] for compressive sensing reconstruction due to the high variance of Cauchy and Pareto distributions. Therefore, in this paper, we focus on using the robust estimator in section 3.1.

4 Behaviour under commonly encountered fault models in WSNs

In this section, we will apply our fixed point iteration for robust averaging and fault detection to simulation data that emulates a number of commonly encountered faults in WSNs. The type of faults to be considered are stuck-at faults, offset and variance degradation fault [GBS08]. The aim of this section is to show that our fixed point iteration can deal with these faults. Firstly, we will show that the sensor which have these faults will have a low weighting factor w_s while the other sensors will have a larger value of weights w_s. Secondly, we will show that the computation of these weighting factors w_s, by using the original sensor measurements or by using the compressed data, give similar results. Thirdly, we will show that the robust average obtained from the original sensor measurements and that obtained from the compressed data are similar.

Note that in this section, we will not study behaviour such as convergence rate, bandwidth savings and stability of fixed point. We will study these in Section 5 using data from WSN deployments.

For the case of stuck-at fault, we will also study it analytically in Section 4.1 to show that our robust averaging algorithm can be designed to give an arbitrarily small bias in the robust average.

4.1 Stuck-at fault

The stuck-at fault has been observed in a number of WSN deployments [GBS08]. This fault occurs when the sensor readings are being stuck at a value which is not related to the true sensor measurement. It can be modelled as

$$x_{st} \quad = \quad c \ \forall t$$

where c is a constant.

We will first study it using simulation and then in Section 4.1, we will carry out an analytical study of applying our robust averaging algorithm when stuck-at faults occur.

Simulation study

Our simulation consists of 10 sensors and a block of data consists of 100 data points. The true signal is a sinusoid. The data is plotted in Figure 4.1. Sensors 1, 2 and 3 have the stuck-at faults. The readings from these three sensors stay at 40, 50 and -10 all the time. In addition, the other 7 sensors have a measurement noise with a standard deviation varying between 1 and 2.

We use Bernoulli distributed projection matrices and 40 projections. We apply our robust averaging algorithm to both the original sensor measurements, as well as to the compressed data. Figure 4.2 show the weights computed from using these two sets of data. It can be seen that the weights computed by these two sets of data are almost identical. Furthermore, Sensors 1-3 have a smaller weights, thus indicating that they are likely to be faulty.

Finally, we show the robust average computed in Figure 4.3. The dashed blue line show the readings from sensor 10, which is a working sensor, as the reference. The thick red line shows the robust average computed using the original sensor reading. The thick green line shows the robust averaging computed using the compressed data. We see that there is almost no loss in fidelity in using the compressed data to compute the robust average.

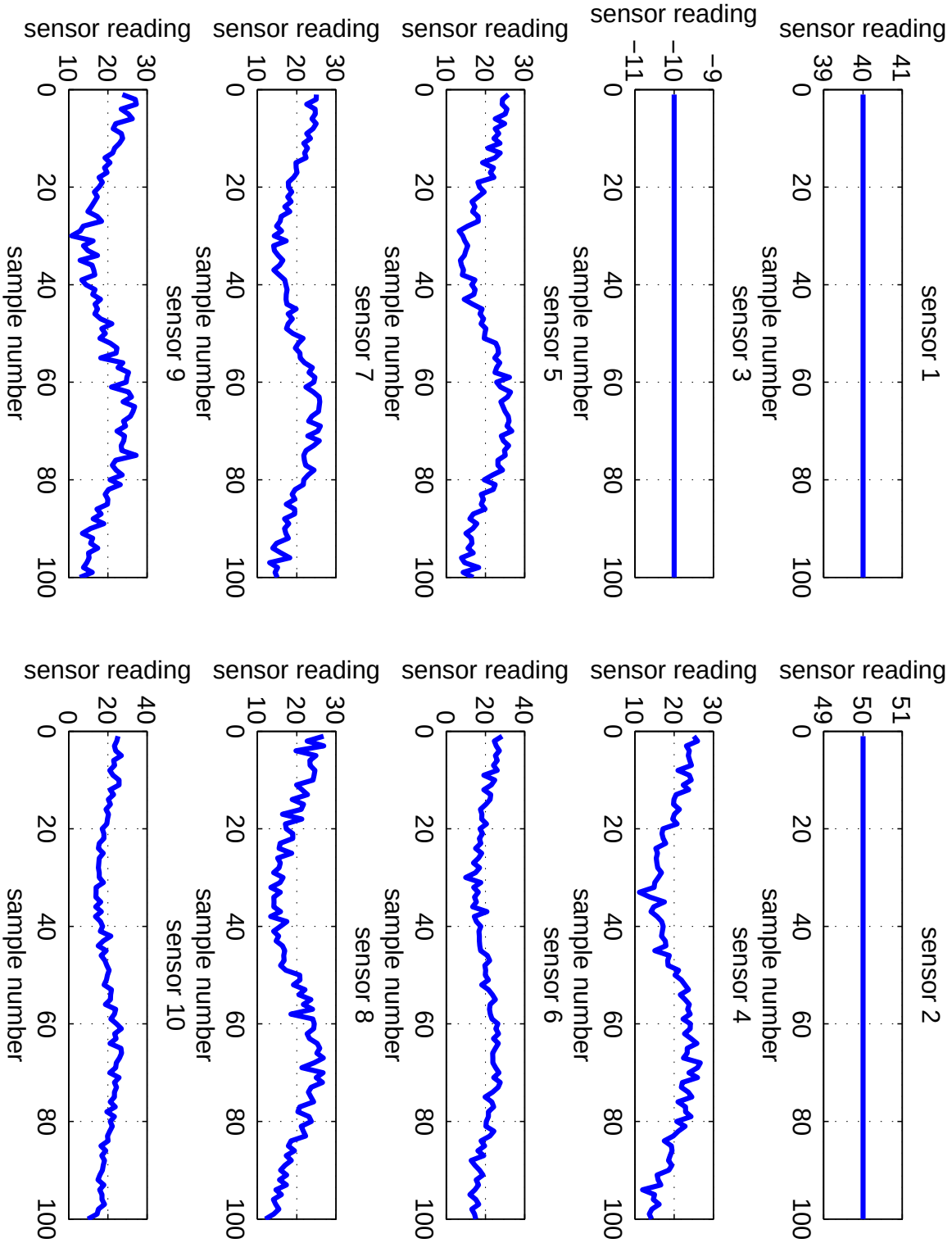

Figure 4.1: Simulated data for the stuck-at fault experiment. Sensors 1–3 have stuck-at faults. Noise of different variances are added to the other 7 sensor readings.

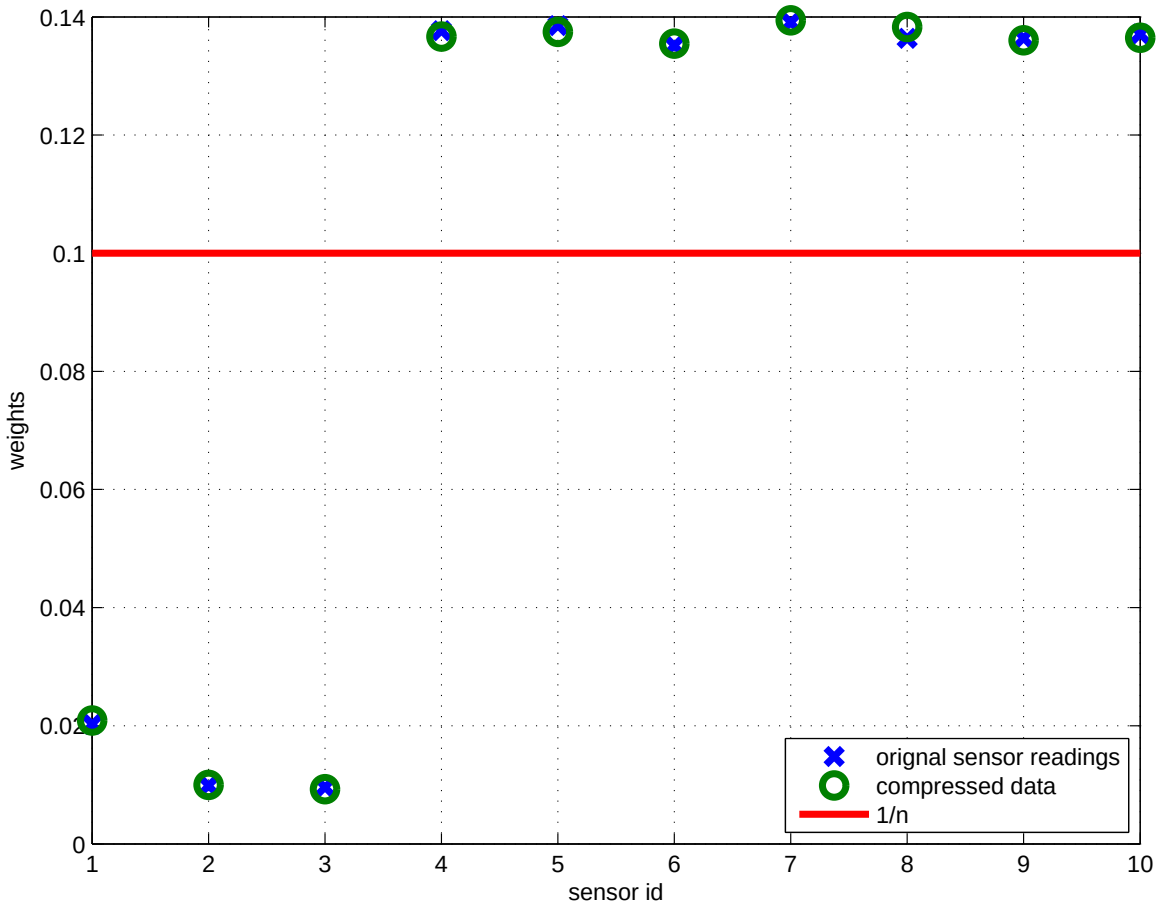

Figure 4.2: The weighs w_s from our robust averaging algorithm. The weights computed from the original sensor readings are shown in circles and those computed from the compressed data are shown in crosses. The line is at the level of $\frac{1}{10}$ which is the expected weights when no sensors are faulty.

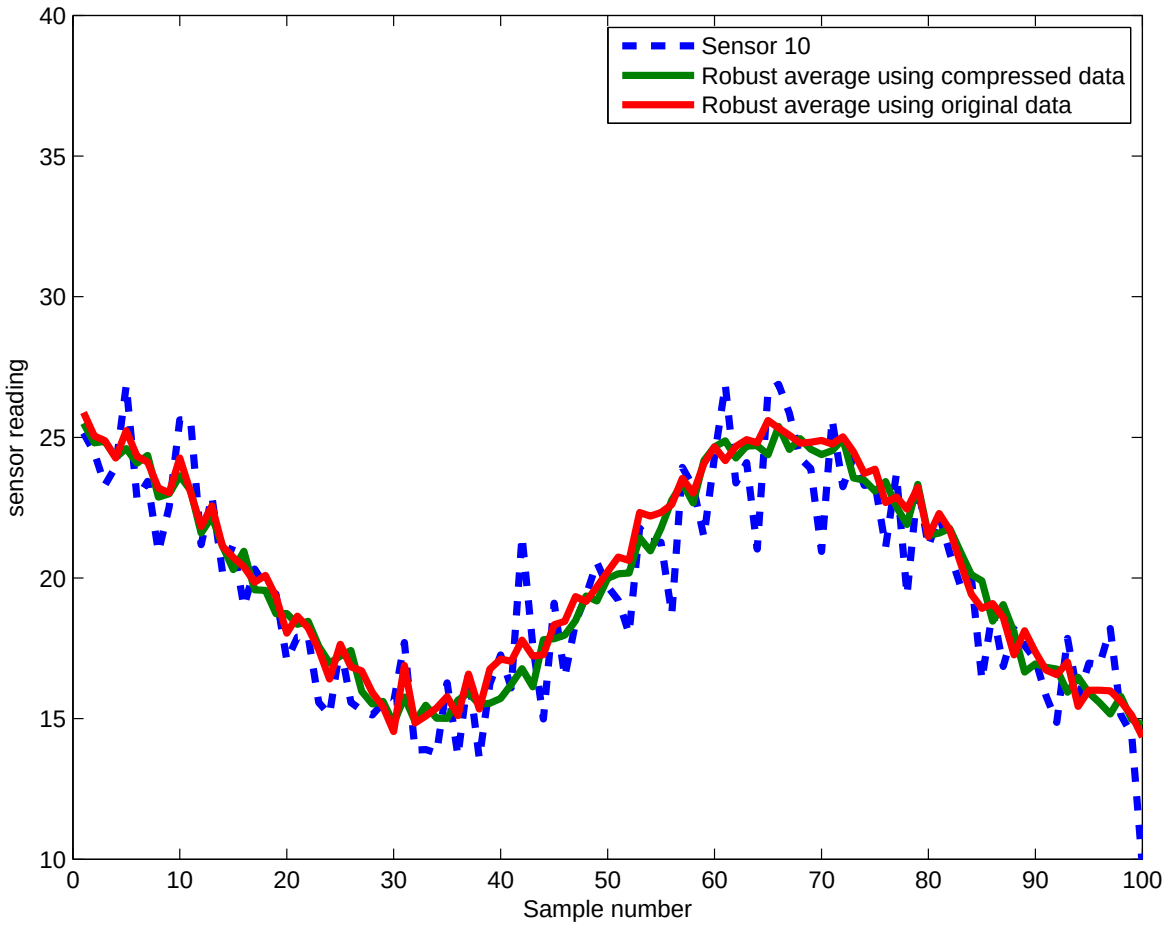

Figure 4.3: The dashed blue line shows the readings from sensor 10, which is a working sensor. The red line shows the robust average computed using the original sensor reading. The green line shows the robust averaging computed using the compressed data.

Analytical study

In this section, we use analytical tools to investigate the property of our robust averaging algorithm in the presence of stuck-at faults. Our goal is to show that the positive constant λ in our robust averaging algorithm will only produce a small change in the weights and a small bias in the robust average.

We consider a network of n sensors. The sensor readings for these sensors are given by:

$$x_{st} = \begin{cases} X & \text{for Sensors } s = 1, ..., n_0, \forall t = 1, .., m \\ X + D & \text{for Sensors } s = n_0 + 1, ...n, \forall t = 1, .., m \end{cases}$$

For this model, we assume that n_0 sensors are functioning correctly and they all register the correct reading X. The other $n_1 = n - n_0$ sensors are stuck-at the same wrong value which is $X + D$. We further assume that there are more working sensors than faulty sensors, i.e. $n_0 > n_1$. Under these assumptions, all the n_0 working sensors will have the same weight (denoted by α_0) and all the n_1 faulty sensors also have the same weights (denoted by α_1).

By using equation (**??**) (with $\epsilon_i = 0$ since the original data is considered), we have

$$v_i = \sum_{t=1}^{m} (X - \sum_{i=1}^{n} x_{st})^2 \tag{4.1}$$

$$= \begin{cases} mn_1^2\alpha_1^2 D^2 & \text{for Sensors } s = 1, ..., n_0, \forall t = 1, .., m \\ mn_0^2\alpha_0^2 D^2 & \text{for Sensors } s = n_0 + 1, ...n, \forall t = 1, .., m \end{cases} \tag{4.2}$$

By using equation (3.3), it can be shown that

$$\alpha_0(v_1 + \lambda \sum_{s=1}^{n} v_s) = \alpha_1(v_n + \lambda \sum_{s=1}^{n} v_s) \tag{4.3}$$

By substituting the expressions of v_i in equation (4.2) in the above equation, we have

$$\alpha_0(n_1^2\alpha_1^2 + \lambda n_0 n_1(n_1\alpha_1^2 + n_0\alpha_0^2)) = \alpha_1(n_0^2\alpha_0^2 + \lambda n_0 n_1(n_1\alpha_1^2 + n_0\alpha_0^2)) \tag{4.4}$$

Since all the weights must sum up to unity, we also have

$$n_0\alpha_0 + n_1\alpha_1 = 1 \tag{4.5}$$

By using equations (4.4) and (4.5), we can solve for α_1 and α_2. Also, the robust average is

$$r_t = X + n_1\alpha_1 D \ \ \forall t \tag{4.6}$$

This also means that the bias in the robust average is $n_1\alpha_1 D$. Therefore, the smaller value of α_1 (= weight of the faulty sensor), the smaller the bias in the robust average.

For $\lambda = 0$, it can be shown that the admissible solution is $\alpha_0 = \frac{1}{n_0}$ and $\alpha_1 = 0$. Therefore the weights for the faulty sensors are zero. This also means that the robust average is exactly X which is the correct value.

We will now show that, if the parameter λ if chosen sufficiently small, it will only produce a small bias. We first divide both sides of equation (4.4) by α_0. After substituting $\beta = \frac{\alpha_1}{\alpha_0}$ and re-arrange the result as a polynomial in λ, we have that β is the root of this following equation:

$$f(\beta) = \lambda \underbrace{[n_0 n_1(n_1\beta^2 + n_0)(\beta - 1)]}_{f_\lambda(\beta)} + \underbrace{\beta(n_0^2 - n_1^2\beta)}_{f_0(\beta)} = 0 \tag{4.7}$$

First note that $f(0) = -\lambda n_0^2 n_1 < 0$, i.e. $f(0)$ is strictly negative. With the assumption that $n_0 > n_1$, we have $f_0(\beta)$ is strictly positive for all $0 < \beta < 1$ and $f_\lambda(\beta)$ is strictly negative for all $0 < \beta < 1$. Therefore, for sufficiently small λ, $f(\beta_0)$ is strictly positive for some $\beta_0 \in (0, 1)$. Therefore, there is a root for equation (4.7) in the range $(0, \beta_0)$. In fact, β_0 can be made arbitrarily small (by having a sufficiently small λ), therefore $\beta = \frac{\alpha_1}{\alpha_0}$ can be made arbitrarily small, i.e. the weights of the faulty sensors α_1 can be made as close to 0 as possible. Therefore, with a small enough λ, the weights of the sensors will only change by a small amount. Since the bias in the robust average is proportional to α_1, therefore the small positive constant will result in only a small bias.

4.2 Offset fault

The offset fault has also been observed in a number of WSN deployments [GBS08]. This fault occurs when the reported sensor reading is corrupted by an additive offset. Let $\tilde{x}_{st}$ be the true sensor reading that sensor s should report at time t, then sensor s is said to suffer from the offset fault if the sensor reading reported by sensor s is:

$$x_{st} = \begin{cases} \tilde{x}_{st} + c & \text{with probability } p_o \\ \tilde{x}_{st} & \text{with probability } 1 - p_o \end{cases}$$

where c is a constant additive offset with corrupts the data with a probability of p_o.

Our simulation consists of 10 sensors and a block of data consists of 100 data points. The true signal is a sinusoid. The data is plotted in Figure 4.4. Sensors 1, 2 and 3 have the offset faults. The offset c for Sensors 1, 2 and 3 are respectively 40, 50 and -20. The probability for the offset to occur for Sensors 1, 2 and 3, are respectively, 0.2, 0.9 and 0.75. The remaining 7 sensors have a measurement noise with a standard deviation varying between 1 and 2.

We use Bernoulli distributed projection matrices and 40 projections. We apply our robust averaging algorithm to both the original sensor measurements, as well as to the compressed data. Figure 4.5 show the weights computed from using these two sets of data. It can be seen that the weights computed by these two sets of data are almost identical. Furthermore, Sensors 1-3 have a smaller weights, thus indicating that they are likely to be faulty.

Finally, we show the robust average computed in Figure 4.6. The dashed blue line show the readings from sensor 10, which is a working sensor, as the reference. The thick red line shows the robust average computed using the original sensor reading. The thick green line shows the robust averaging computed using the compressed data. We see that there is almost no loss in fidelity in using the compressed data to compute the robust average.

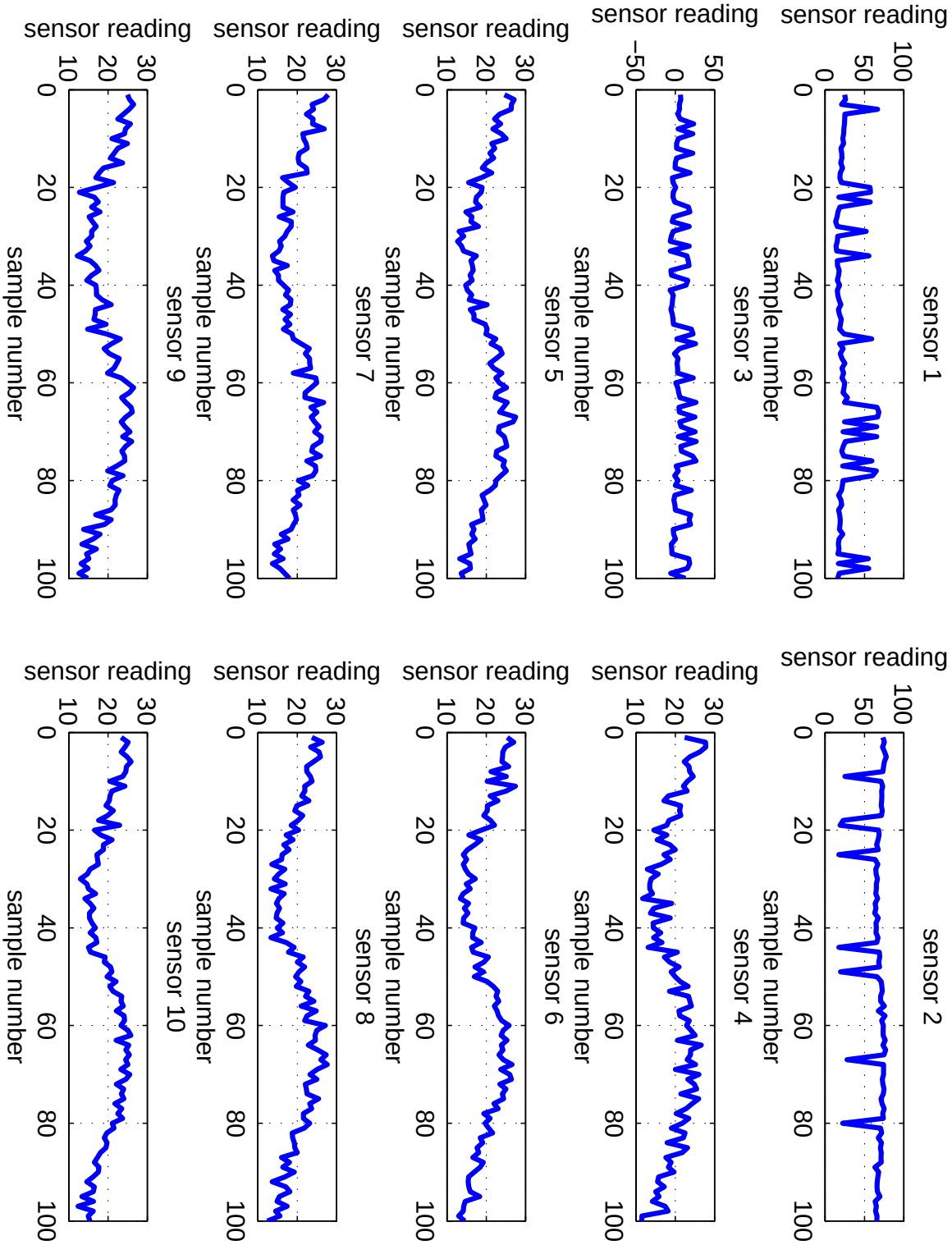

Figure 4.4: Simulated data for the offset fault experiment. Sensors 1–3 have offset faults. Noise of different variances are added to the other 7 sensor readings.

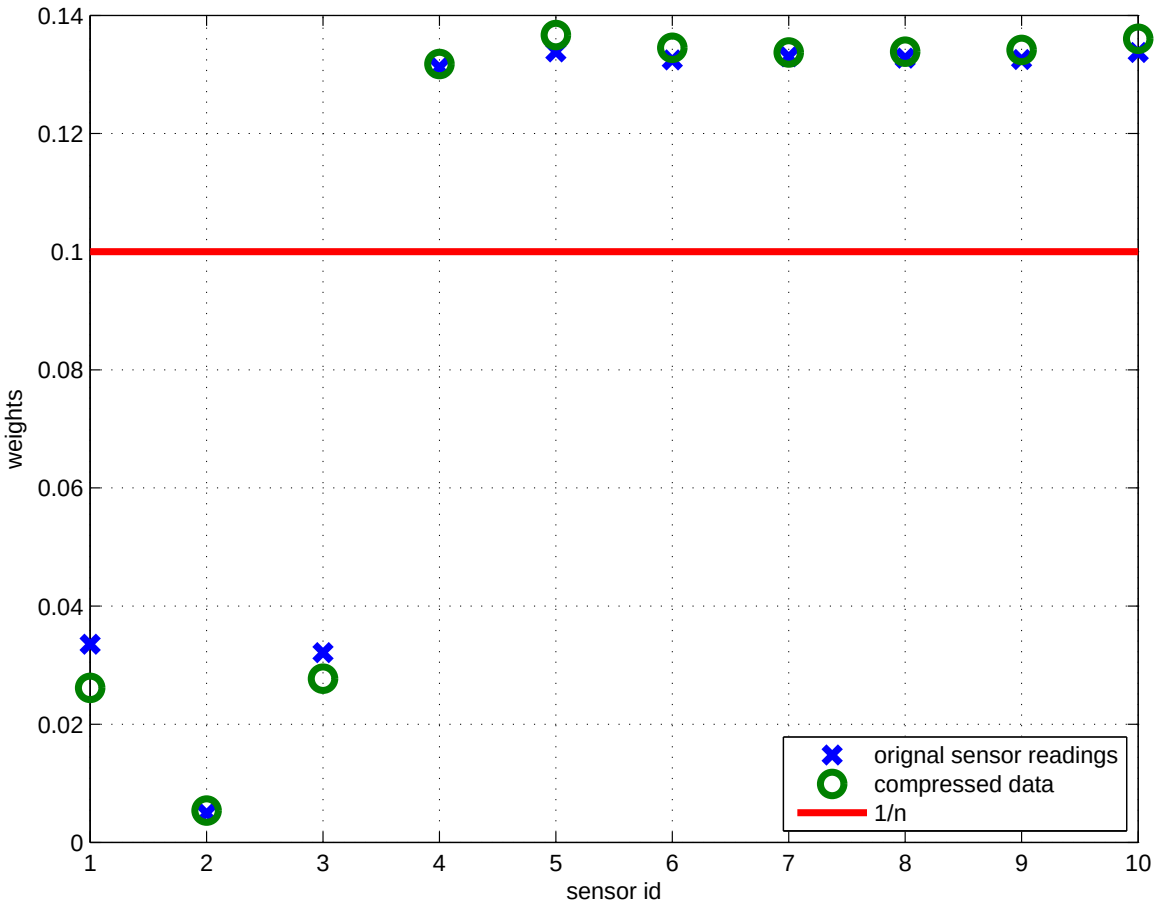

Figure 4.5: The weighs w_s from our robust averaging algorithm. The weights computed from the original sensor readings are shown in circles and those computed from the compressed data are shown in crosses. The line is at the level of $\frac{1}{10}$ which is the expected weight when no sensors are faulty.

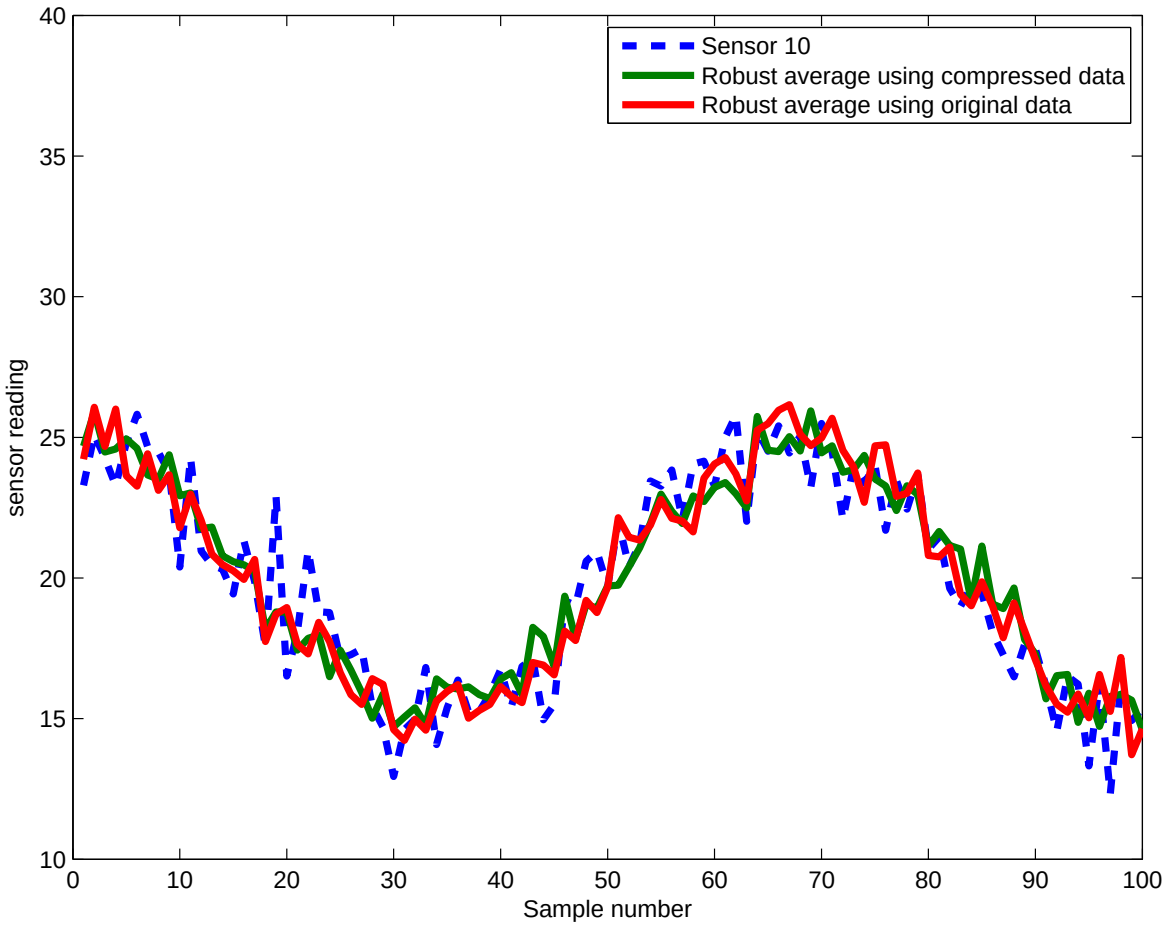

Figure 4.6: The dashed blue line show the readings from sensor 10, which is a working sensor. The red line shows the robust average computed using the original sensor reading. The green line shows the robust averaging computed using the compressed data.

4.3 Variance degradation fault

The variance degradation fault has been observed in a number of WSN deployments [GBS08]. This fault occurs when the noise variance becomes larger over time. The variance degradation fault can be modelled as an additive Gaussian distributed noise of larger variance compared with the other sensors. Let $\tilde{x}_{st}$ be the true sensor reading that sensor s should report at time t, then the actual reading that will be reported by sensor s at time t is:

$$x_{st} \quad = \quad \tilde{x}_{st} + e_{st}$$

where e_{st} is a Gaussian distributed random variable of zero mean and variance σ_s^2.

Our simulation consists of 10 sensors and a block of data consists of 100 data points. The true signal is a sinusoid. The data is plotted in Figure 4.7. Sensors 1, 2 and 3 have the variance degradation faults. The variances for Sensors 1, 2 and 3, are respectively, 62.7, 25.0 and 31.5. The remaining 7 sensors have a measurement noise with a variance varying between 1.6 and 3.6.

We use Bernoulli distributed projection matrices and 40 projections. We apply our robust averaging algorithm to both the original sensor measurements, as well as to the compressed data. Figure 4.8 show the weights computed from using these two sets of data. It can be seen that the weights computed by these two sets of data are almost identical. Furthermore, Sensors 1-3 have a smaller weights, thus indicating that they are likely to be faulty.

Finally, we show the robust average computed in Figure 4.9. The dashed blue line show the readings from sensor 10, which is a working sensor, as the reference. The thick red line shows the robust average computed using the original sensor reading. The thick green line shows the robust averaging computed using the compressed data. We see that there is almost no loss in fidelity in using the compressed data to compute the robust average.

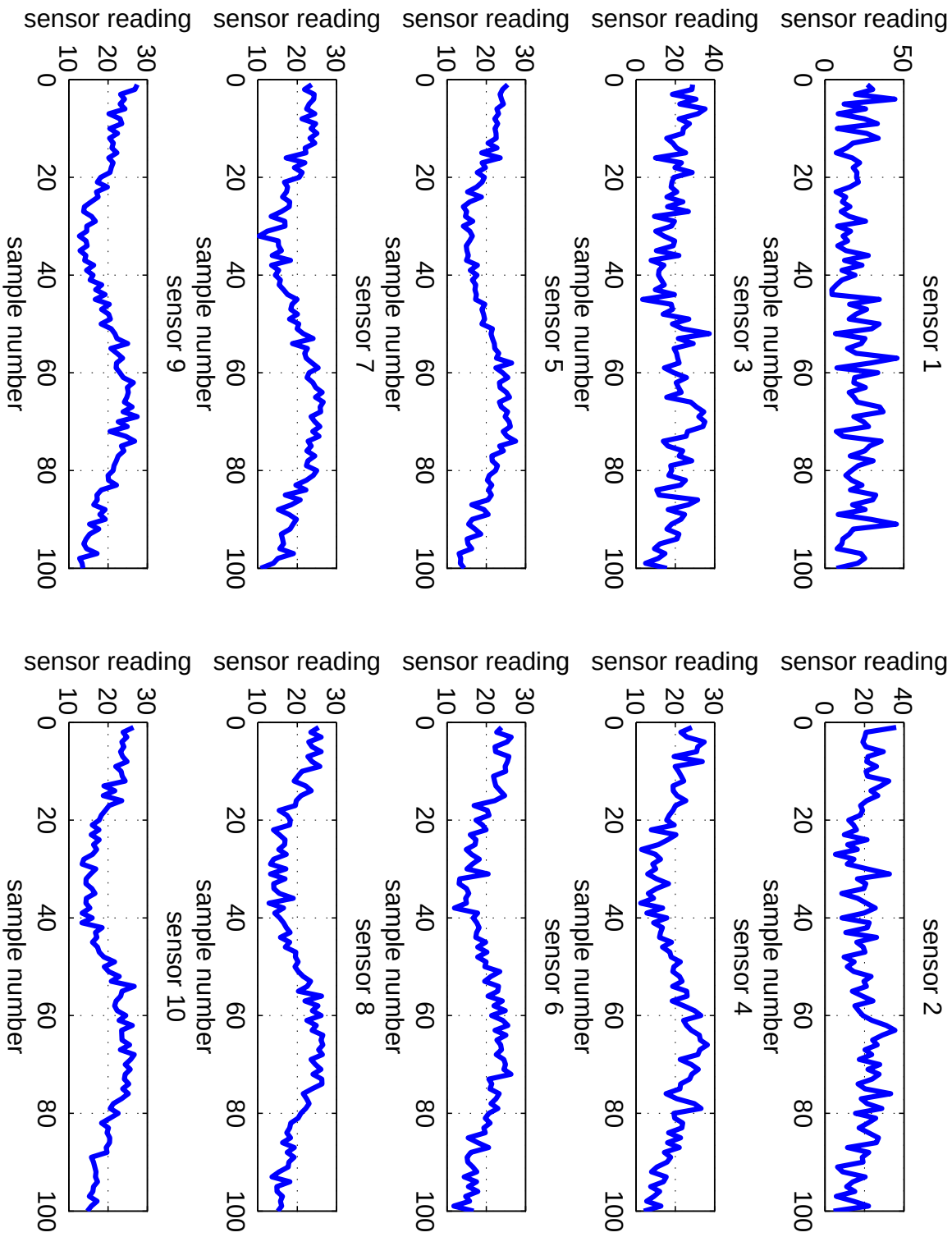

Figure 4.7: Simulated data for the stuck-at fault experiment. Sensors 1–3 have higher noise variances compared with the rest of the sensors. Noise of different variances are added to the other 7 sensor readings.

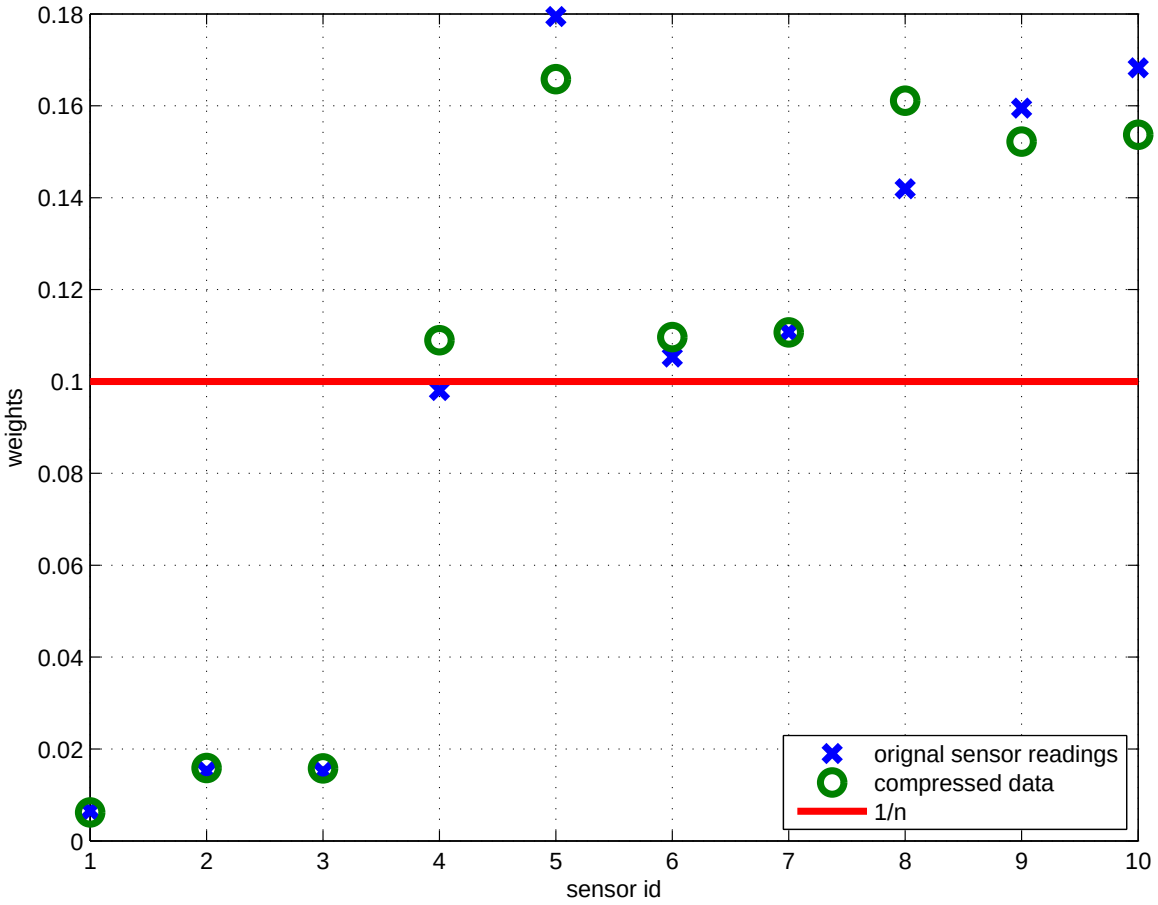

Figure 4.8: The weighs w_s from our robust averaging algorithm. The weights computed from the original sensor readings are shown in circles and those computed from the compressed data are shown in crosses. The line is at the level of $\frac{1}{10}$ which is the expected weight when no sensors are faulty.

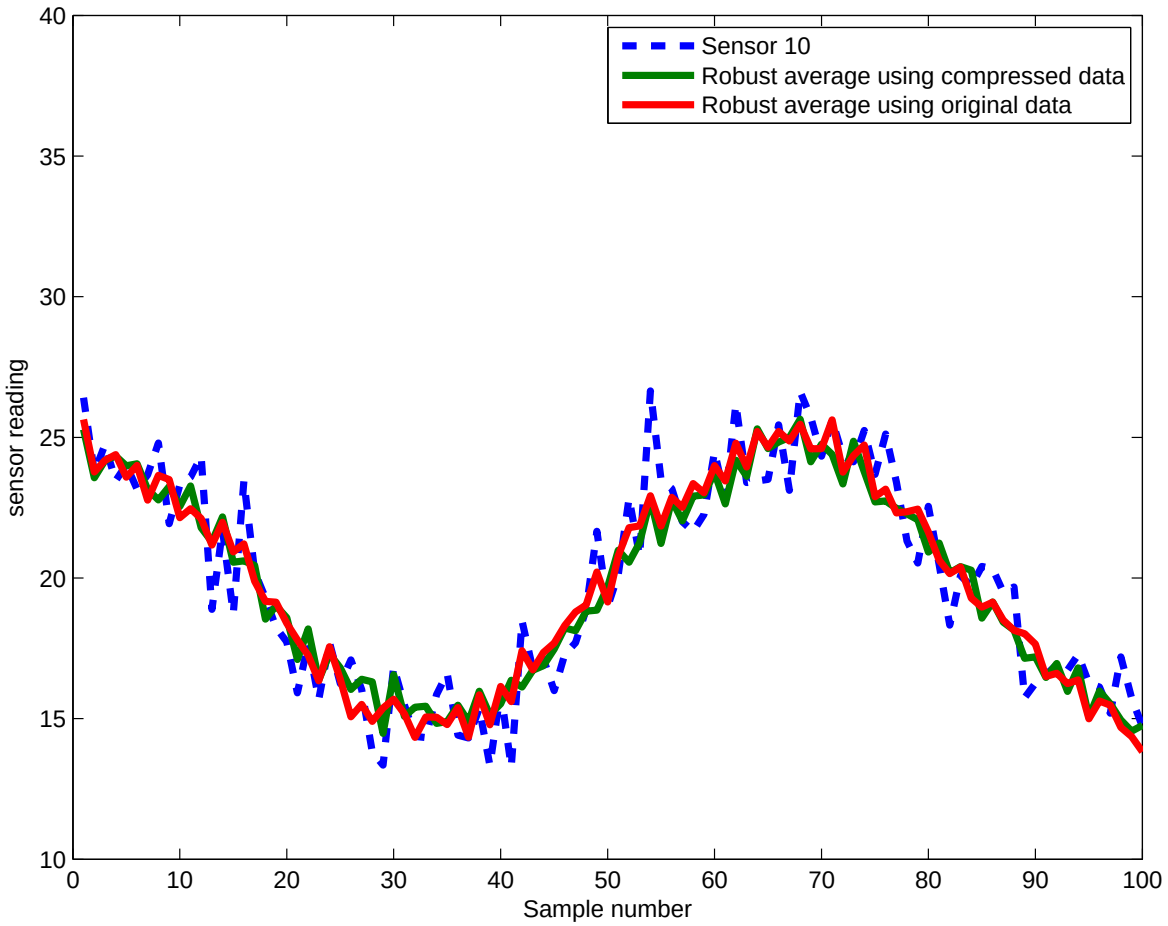

Figure 4.9: The dashed blue line show the readings from sensor 10, which is a working sensor. The red line shows the robust average computed using the original sensor reading. The green line shows the robust averaging computed using the compressed data.

4.4 Mixture of different types of faults in a network

In this section, we consider a simulation where all the three sensor faults described earlier appear in the same wireless sensor network. Our simulation consists of 10 sensors and a block of data consists of 100 data points. The true signal is a sinusoid. The data is plotted in Figure 4.10. Sensors 1, 2 and 3 are faulty. Sensor 1 has a higher noise variance of 25 while Sensors 4-10 have a smaller variance between 1 and 4. Sensor 2 suffers the stuck-at fault and the sensor reading stays at 50 all the time. Sensor 3 has an offset of 40 with a probability of 0.75.

We use Bernoulli distributed projection matrices and 40 projections. We apply our robust averaging algorithm to both the original sensor measurements, as well as to the compressed data. Figure 4.11 show the weights computed from using these two sets of data. It can be seen that the weights computed by these two sets of data are almost identical. Furthermore, Sensors 2-3 have a smaller weights, thus indicating that they are likely to be faulty. Note that Sensor 1 has a weight which is smaller than Sensors 4-10 but a larger weight compared with Sensors 2-3. This is reasonable since the stuck-at fault (Sensor 2) and offset fault (Sensor 3) produce data that does not resemble the average behaviour. Therefore, the weights for Sensors 2 and 3 are lower. Although Sensor 1 has larger variance, the true data hidden behind the noise is similar to what is found in Sensors 4-10, therefore Sensor 1 has a weight which is in-between.

Finally, we show the robust average computed in Figure 4.12. The dashed blue line show the readings from sensor 10, which is a working sensor, as the reference. The thick red line shows the robust average computed using the original sensor reading. The thick green line shows the robust averaging computed using the compressed data. We see that there is almost no loss in fidelity in using the compressed data to compute the robust average.

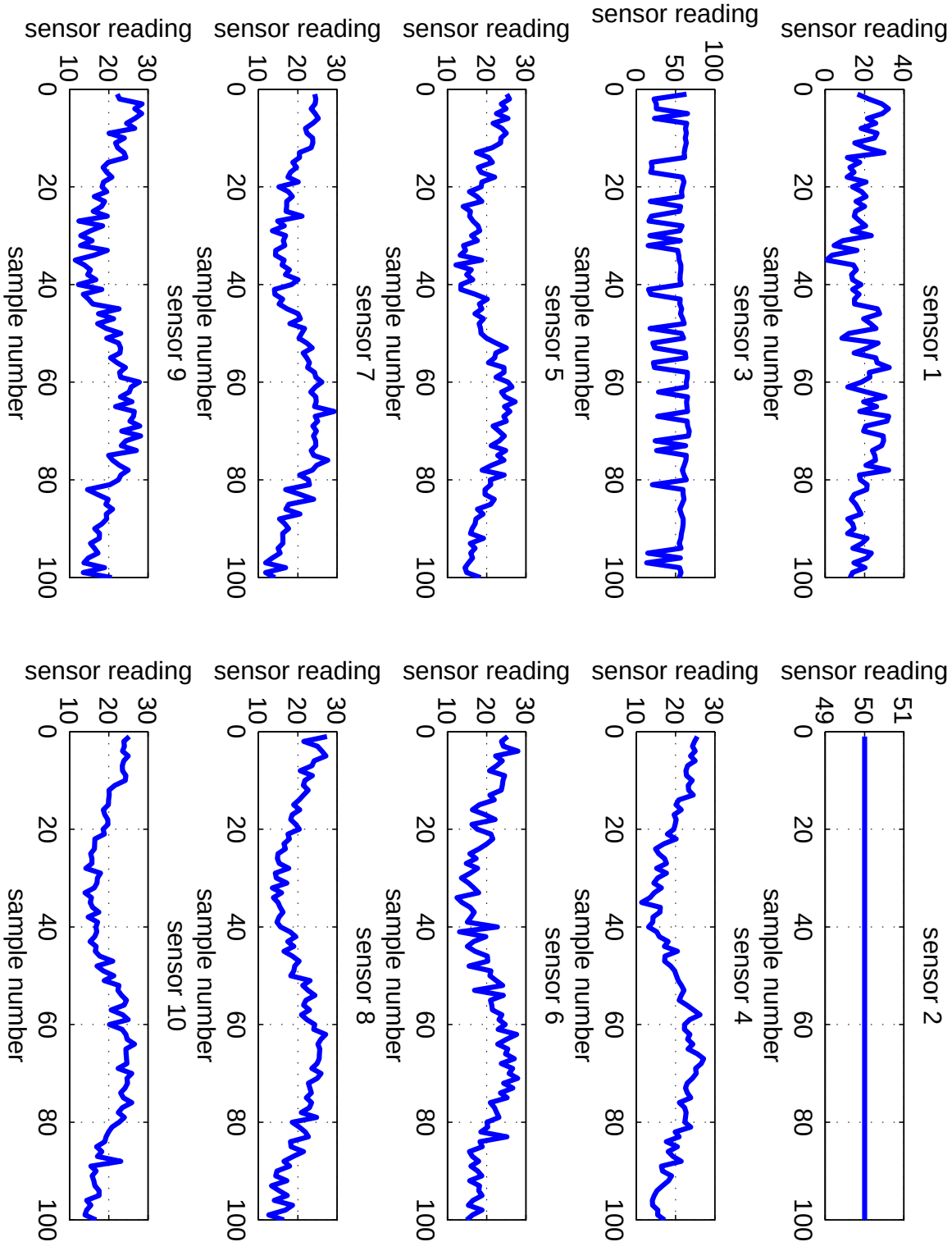

Figure 4.10: Simulated data with three different types of sensor faults. Sensor 1 has a higher noise variance. Sensor 2 has stuck-at fault. Sensor 3 has an offset fault. Noise of different variances are added to the other 7 sensor readings.

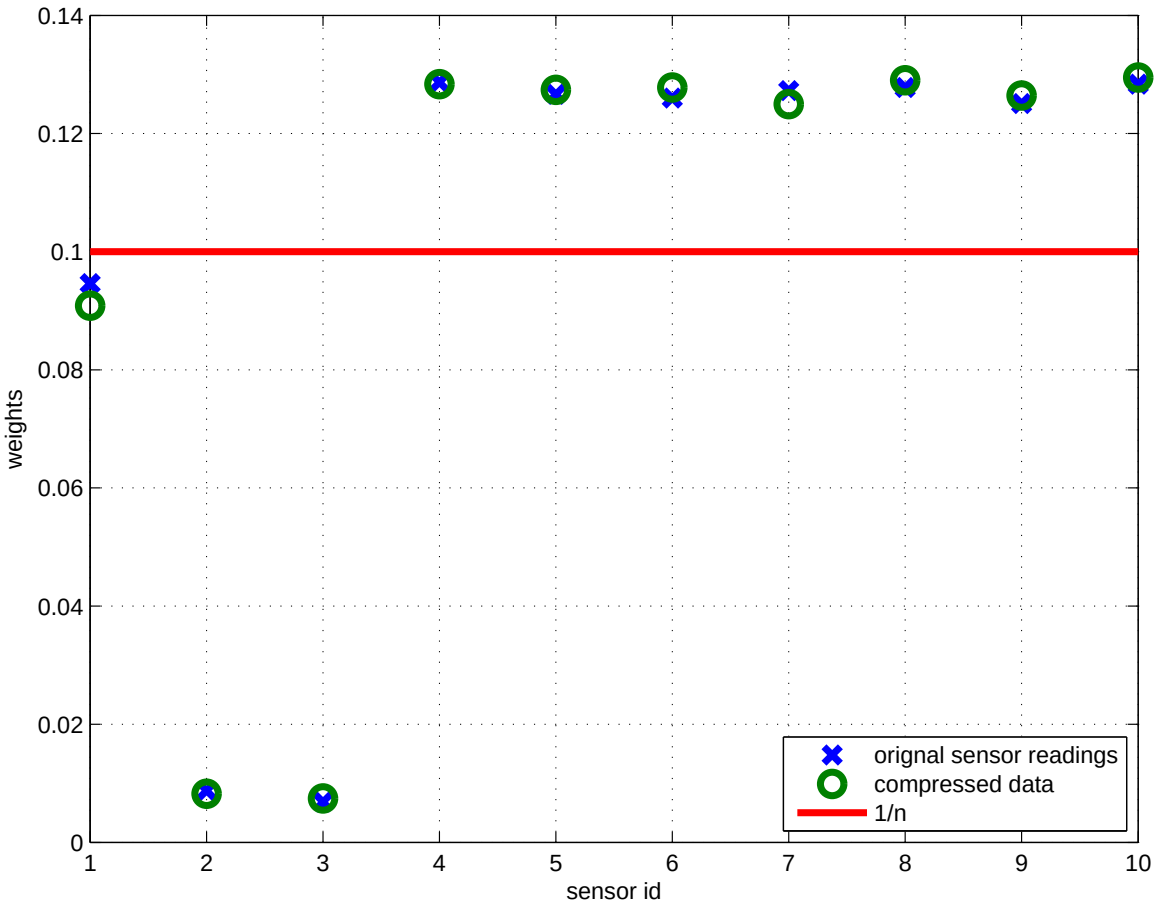

Figure 4.11: The weighs w_s from our robust averaging algorithm. The weights computed from the original sensor readings are shown in circles and those computed from the compressed data are shown in crosses. The line is at the level of $\frac{1}{10}$ which is the expected weight when no sensors are faulty.

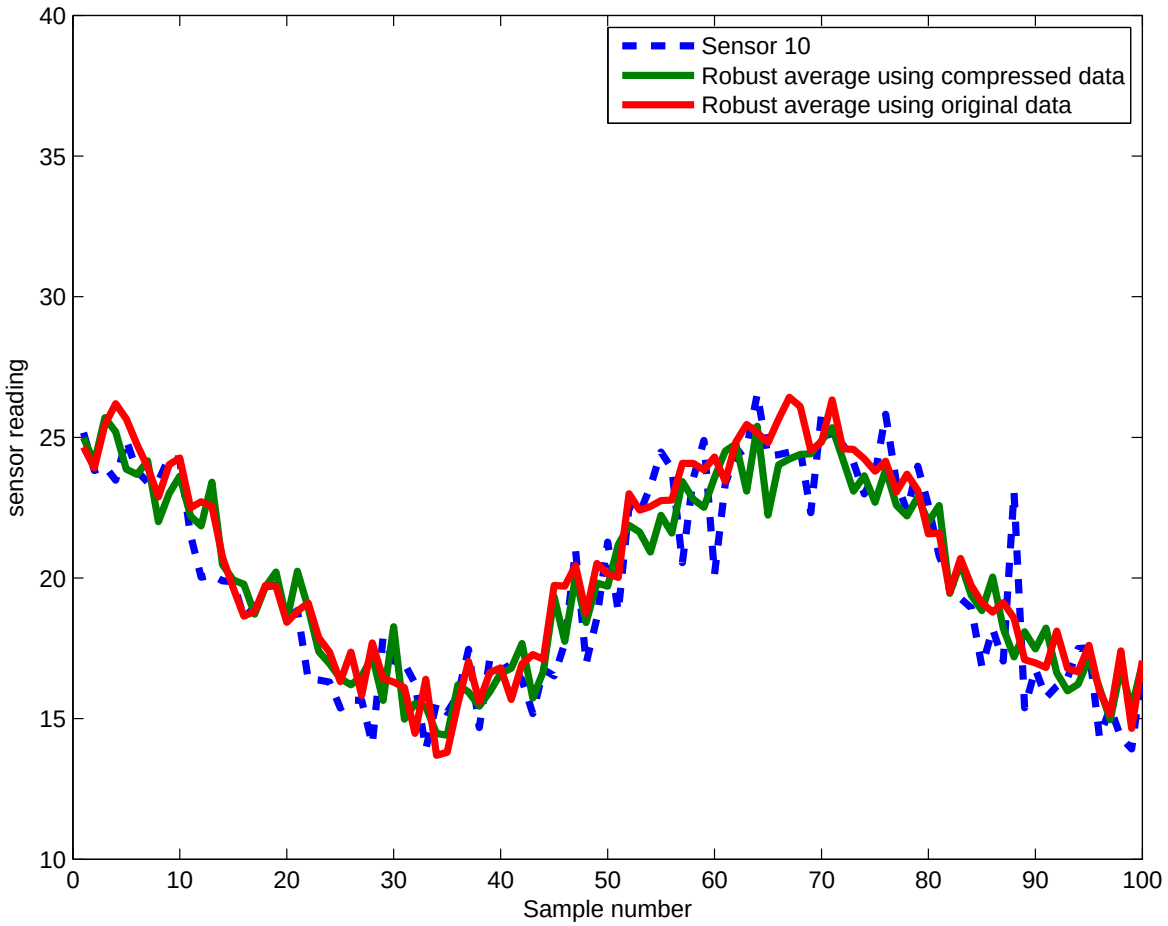

Figure 4.12: The dashed blue line show the readings from sensor 10, which is a working sensor. The red line shows the robust average computed using the original sensor reading. The green line shows the robust averaging computed using the compressed data.

5 Application to data collected from outdoor WSN deployments

5.1 The Belmont deployment

This section describes the results of applying our robust averaging algorithm to the data obtained from an outdoor wireless sensor network testbed operated by CSIRO in Belmont, central Queensland, Australia. The WSN consists of 32 sensors measuring temperature and a block of data for each sensor consists of 221 points.

Figures 5.1–5.4 show the temperature measurement of the 32 sensors over the measurement period. The correct trend is that the temperature readings should drop from (about) 31^{o}C to (about) 19^{o}C over the measurement period. Some of the sensors are clearly faulty. For example, sensor 3 is stuck at 31^{o}C throughout, sensor 10 is stuck at 28 ^{o}C and sensor 11 is stuck at 12 ^{o}C . Some sensors are only faulty for part of the time. For example, sensor 9 is stuck at 30^{o}C in the first half of the measurement period but works in the second half; both sensors 13 and 14 are stuck in the second half of the measurement period. There are 221 readings per sensors.

We apply our fault detection algorithm to the original sensor measurements, as well as to the compressed data with 88 projections with a Bernoulli distributed projection matrix. Figure 5.5 shows the weightings w_i for the sensors, where $i = 1, ..., 32$, obtained from using the original sensor readings as well as the compressed data. It can be seen that the two sets of weights are very close to each other. The solid line in figure 5.5 is at the level of $\frac{1}{32}$ which is the weighting to use when all sensors are working. Figure 5.5 shows that sensors 3, 9 and 10 have very low weightings, and they correspond to those three sensors that are stuck throughout the measurement period. Sensors 13 and 14 also receive low weightings because they are stuck for a good part of the measurement period. The same observation apply to sensors 5 and 9, which also have low weightings.

The fixed point iteration also converges very quickly. Figures 5.6 and 5.7 show the convergence of the weights w_s using the original sensor measurements as well as the compressed data. It can be seen that the weights converge to a stable value within 5 iterations.

Figure 5.8 shows the average temperature (in solid line) given by our fixed point iteration algorithm. The red curve shows the result obtained from applying the fixed point iteration to the original sensor measurements. The green curve shows the result obtained from applying the fixed point iteration to the compressed data followed by reconstruction. It can be seem that there is no loss in fidelity in using the compressed data. The figure also shows the robust average captures the trend of the working sensor.

Table 5.1 shows the values of the spectral norm of $\rho(\mathbf{F})$ (which determines the stability of the fixed point) and $\|\mathbf{G}\|_\infty$ (which gives the size of perturbation caused by using the compressed data). One pair of values is calculated from using the original sensor measurements. The other pair of values is calculated from using the compressed data. It can readily be seen that both the original data and the compressed data give similar results. Since the spectral norm $\rho(\mathbf{F})$ is less than 1, the fixed point is stable. In addition, $\|\mathbf{G}\|_\infty$ is very small, which means that the perturbation on the weights in using the compressed data is minimal. This is also confirmed earlier in Figure 5.5.

The bandwidth savings in using compressive sensing can also be computed. Assuming the sensors use a 10-bit analogue-to-digital converter, then sending all the sensor readings to the data fusion centre will mean each sensor needs to transmit 2210 bits of data. With compressive sensing, the number of bits of data that has to be sent by each sensor is $p + 1 + b\lceil \log_2(m) \rceil$ =1408 bits. This represents a 25% savings.

	$\rho(\mathbf{F})$	$\|\mathbf{G}\|_\infty$
Using original sensor readings	0.0982	0.0186
Using compressed data	0.0967	0.0185

Table 5.1: This table compares the values of $\rho(\mathbf{F})$ and $\|\mathbf{G}\|_\infty$ evaluated by using the original sensor readings against those computed by using the compressed data.

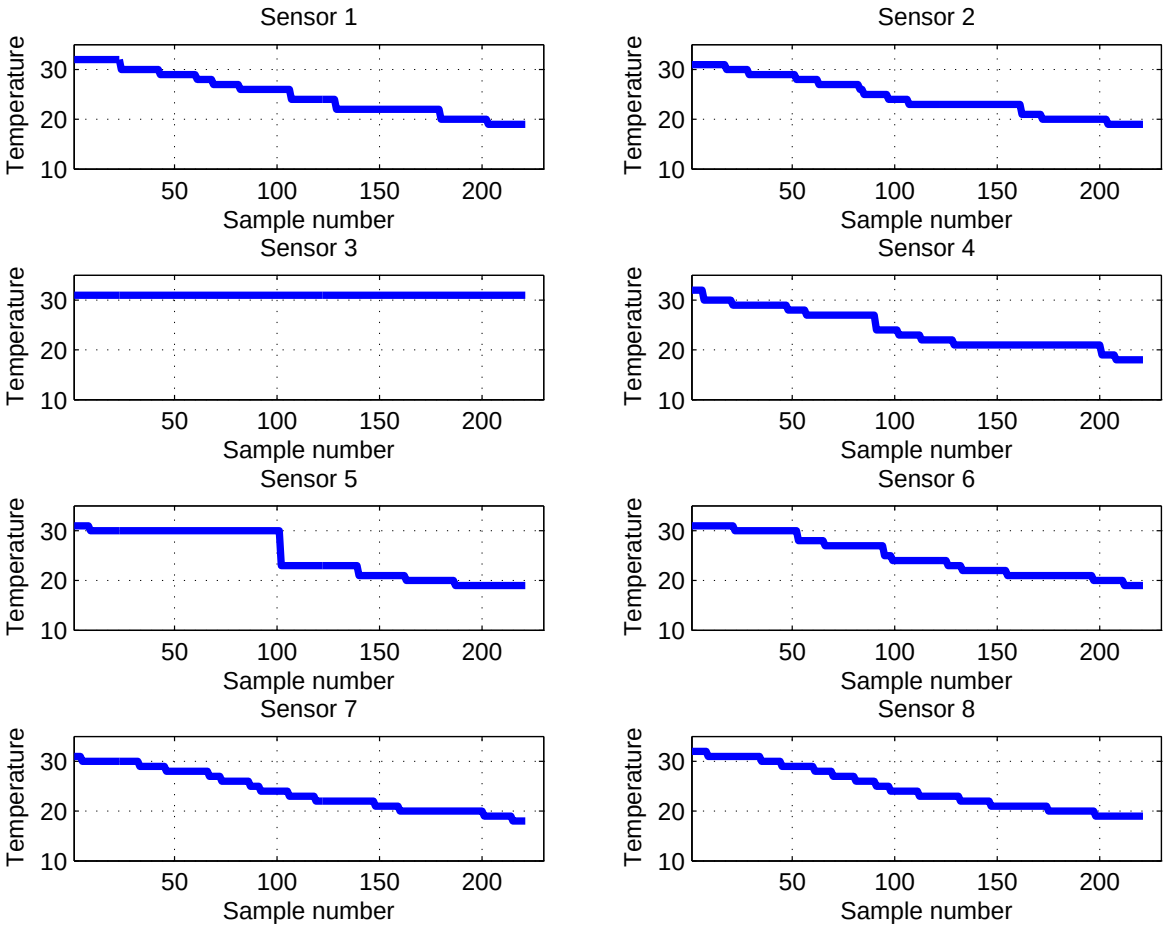

Figure 5.1: Sensor readings: Sensors 1–8.

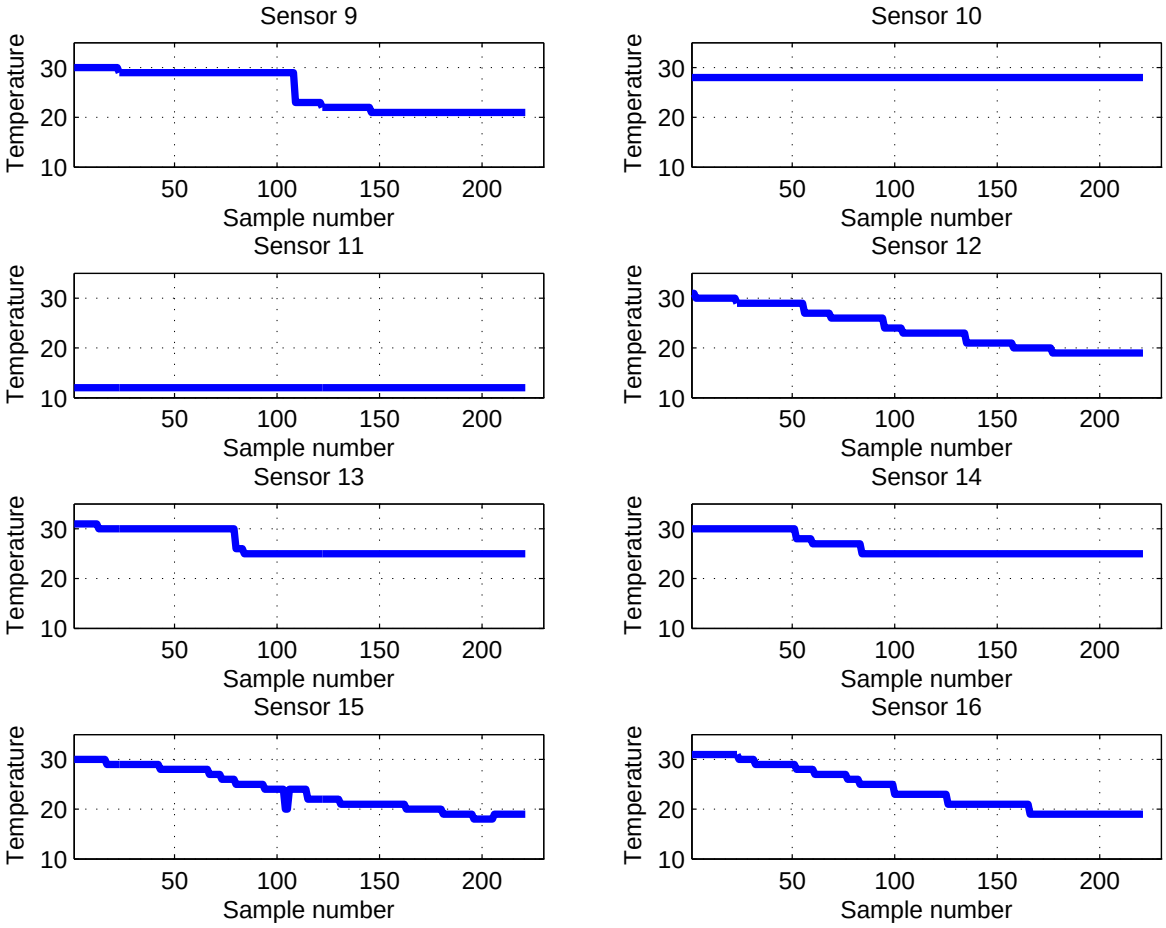

Figure 5.2: Sensor readings: Sensors 9–16.

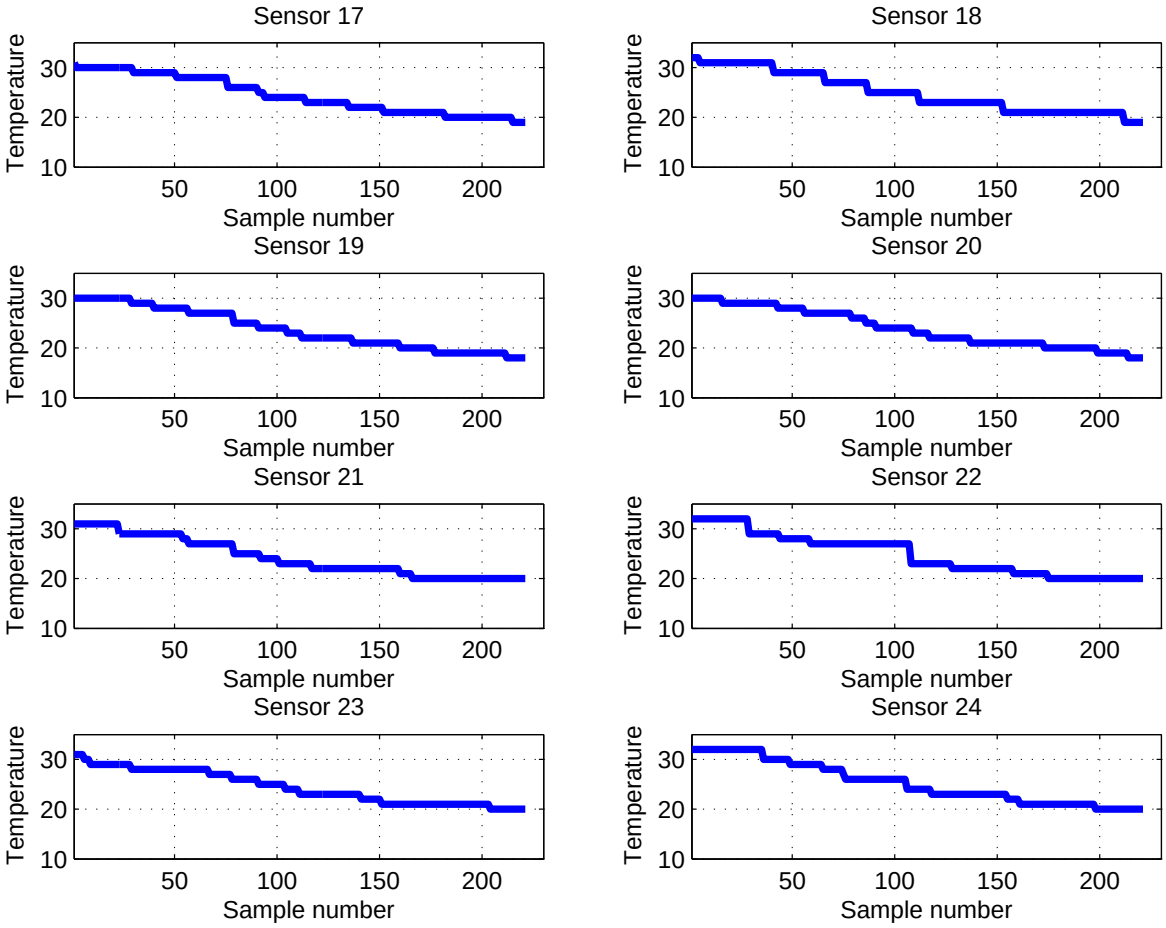

Figure 5.3: Sensor readings: Sensors 17–24.

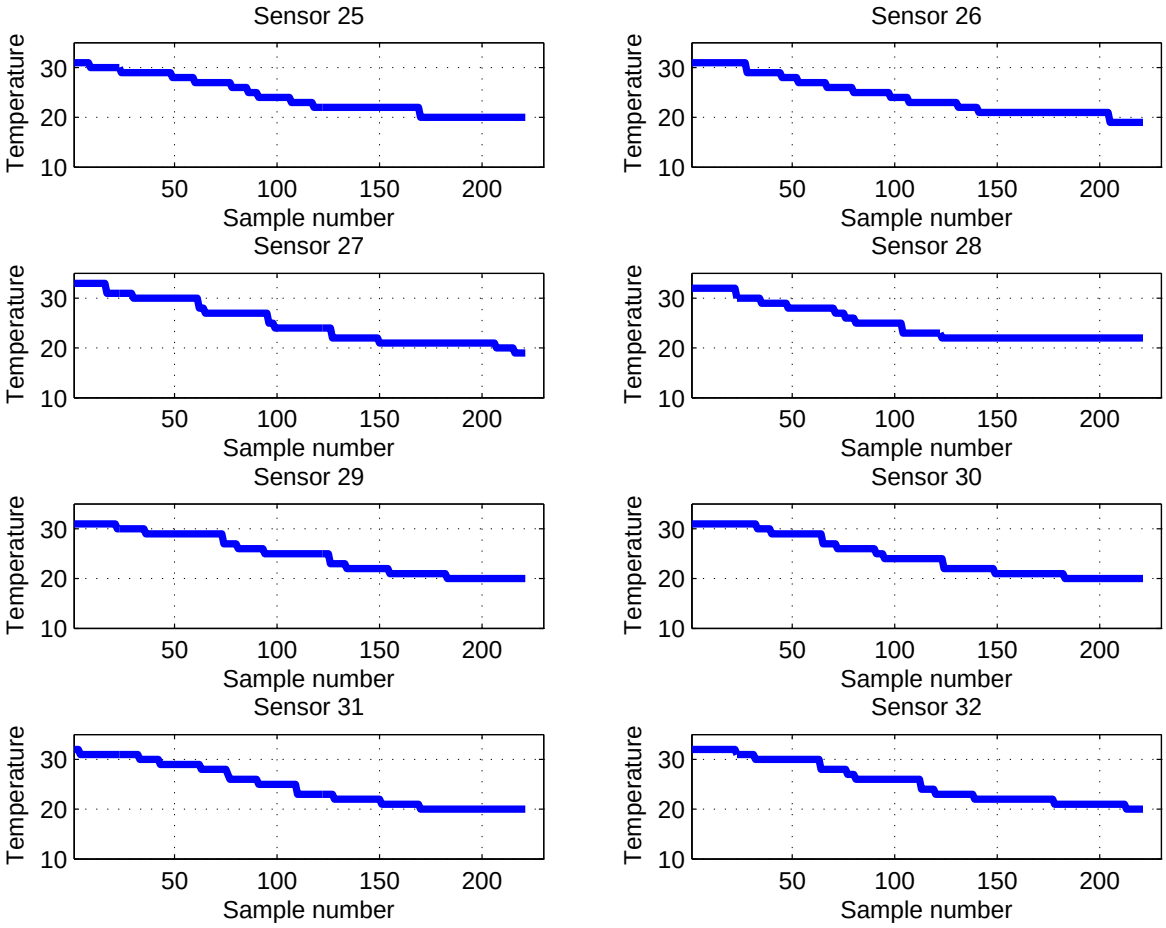

Figure 5.4: Sensor readings: Sensors 25–32.

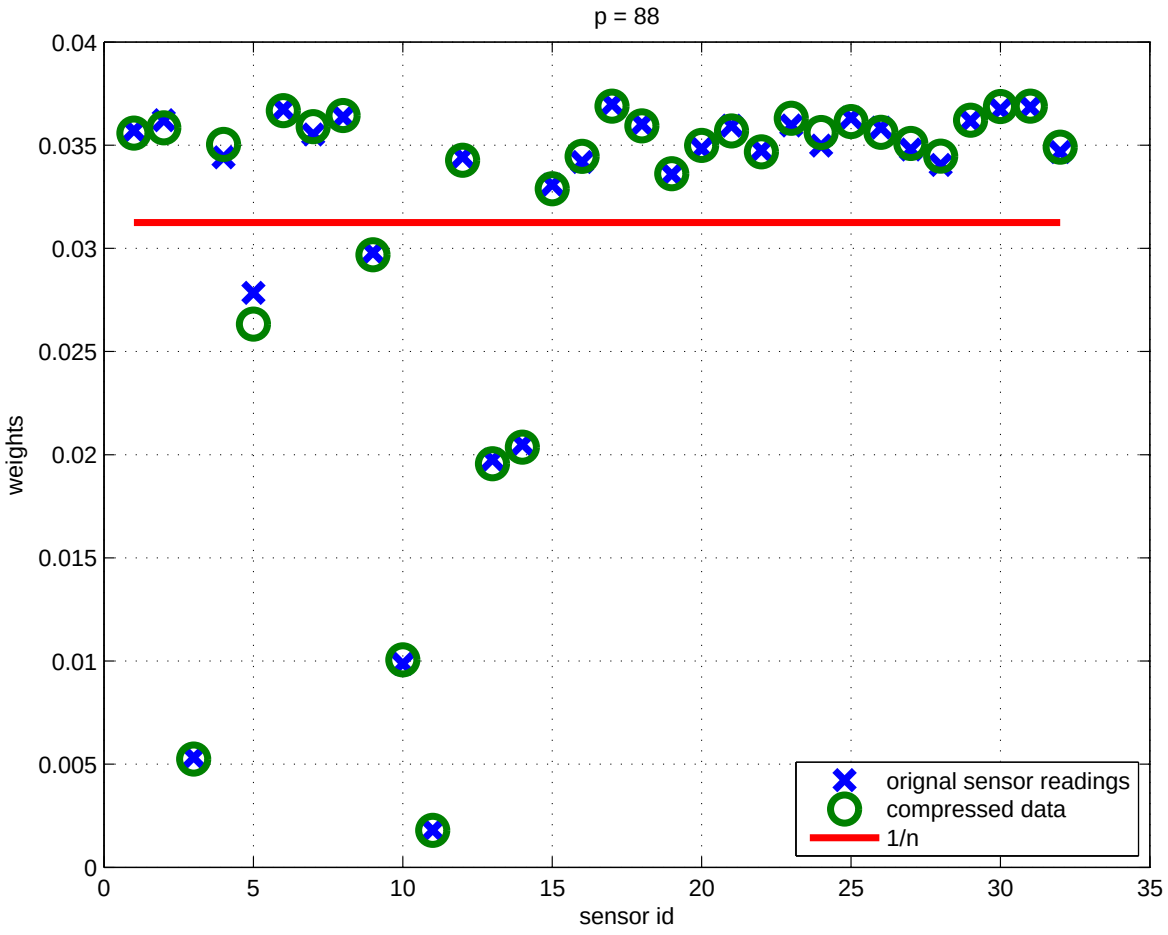

Figure 5.5: The final weightings are shown in circles (from original sensor readings) and crosses (from compressed data). The line is at the level of $\frac{1}{32}$ which is the expected weight when no sensors are faulty.

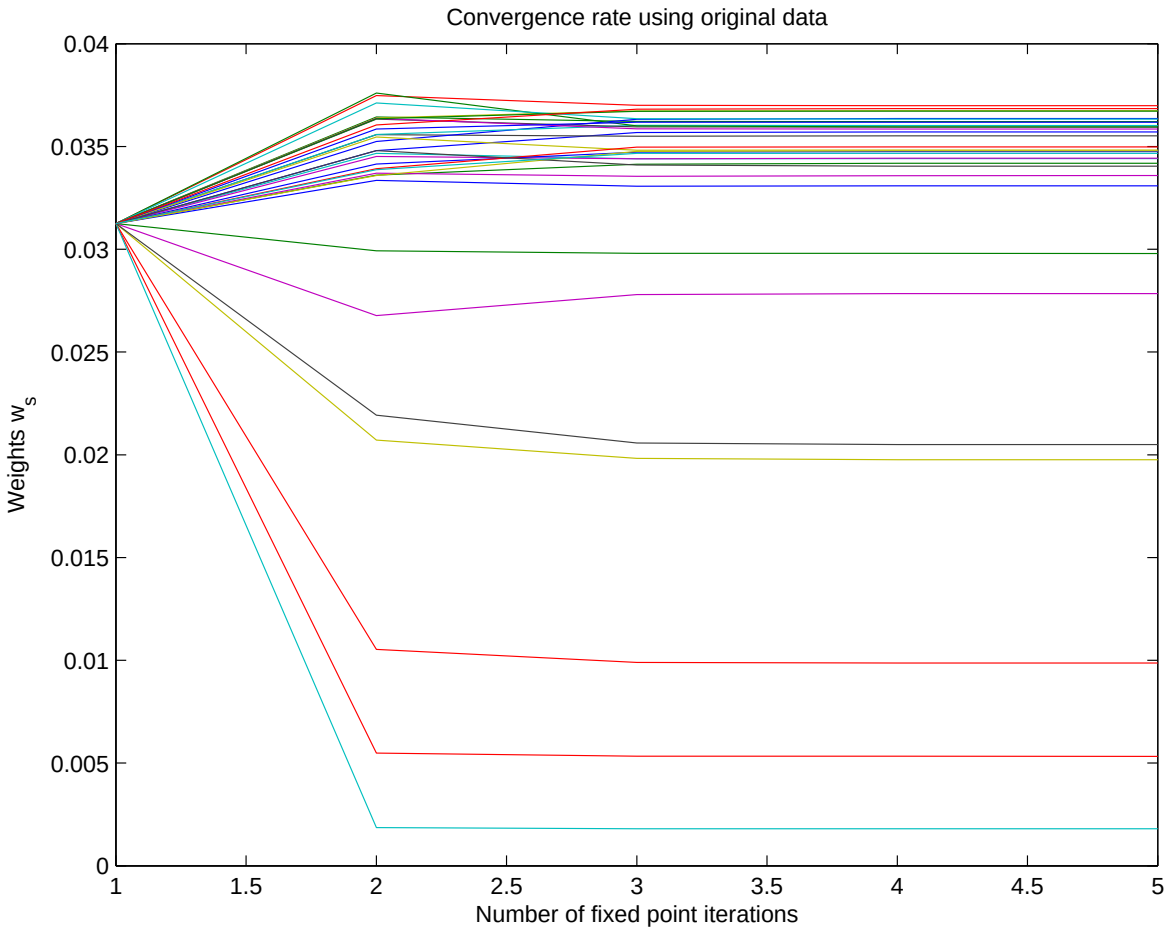

Figure 5.6: This figure shows the convergence of the weights w_s using the original sensor readings. The convergence is achieved within 5 iterations

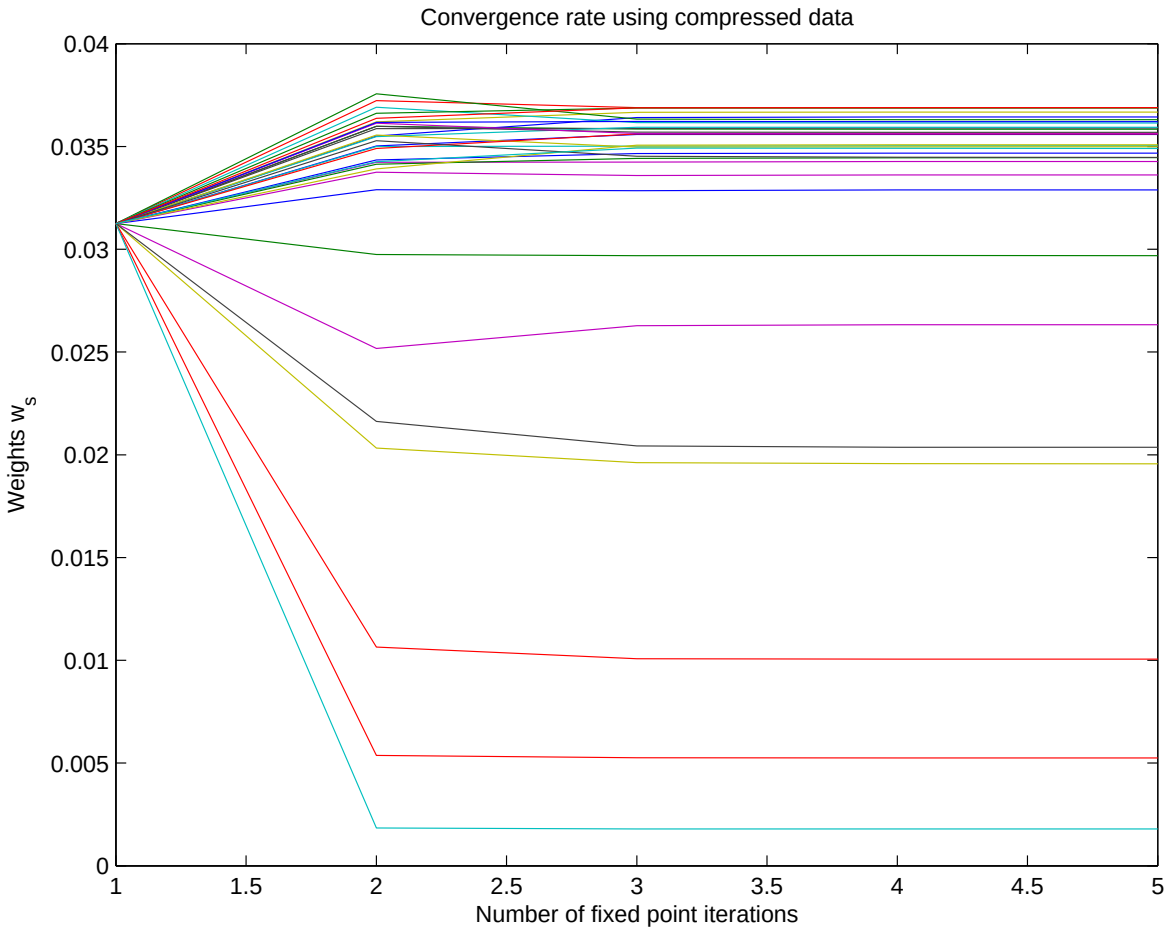

Figure 5.7: This figure shows the convergence of the weights w_s using the compressed data. The convergence is achieved within 5 iterations

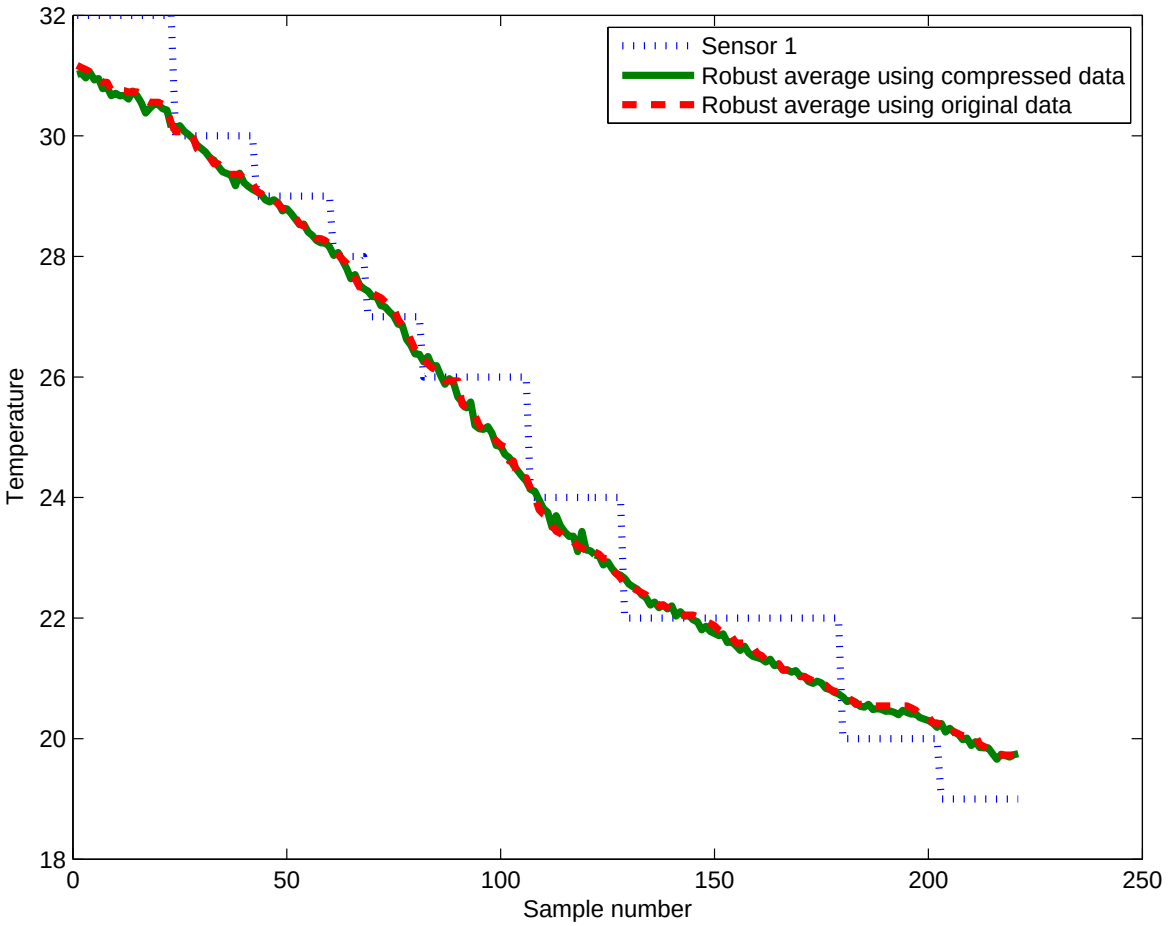

Figure 5.8: The figure shows the averages against the trend of one working sensor (the dashed blue line). The red curve shows the robust average computed by applying the fixed point iteration to the original data. The green curve shows the robust average by applying the fixed point iteration to the compressed data followed by reconstruction.

5.2 The QCAT deployment

This section describes the results of applying our robust averaging algorithm to the data obtained from an outdoor wireless sensor network testbed operated by CSIRO in their QCAT research facility in Brisbane, Australia. The WSN consists of 5 sensors measuring humidity and a block of data for each sensor consists of 400 points.

Figure 5.9 shows the humidity measured by 5 sensors over the measurement period. The first four sensors have the same trend but the fifth sensor shows completely erroneous measurements.

We apply our robust averaging algorithm to the original sensor measurements as well as to the compressed data with 160 projections using a Bernoulli distributed projection matrix. Figure 5.10 shows the weightings w_s for the sensors, where $s = 1, ..., 5$, obtained from using the original sensor readings as well as the compressed data. It can be seen that the two sets of weights are very close to each other. The solid line in figure 5.5 is at the level of $\frac{1}{5}$ which is the weighting to use when all sensors are working. Figure 5.10 shows that sensor 5 has a very low weighting and sensor 5 is the sensor that is giving erroneous measurements.

The fixed point iteration also converges very quickly. Figures 5.11 and 5.12 show the convergence of the weights w_s using the original sensor measurements as well as the compressed data. It can be seen that the weights converge to a stable value within 8 iterations.

Figure 5.13 shows the average humidity (in solid line) given by our fixed point iteration algorithm. The red curve shows the result obtained from applying the fixed point iteration to the original sensor measurements. The green curve shows the result obtained from applying the fixed point iteration to the compressed data followed by reconstruction. It can be seem that there is no loss in fidelity in using the compressed data. The figure also shows the robust average captures the trend of the working sensor.

Table 5.2 shows the values of the spectral norm $\rho(\mathbf{F})$ (which determines the stability of the fixed point) and $\|\mathbf{G}\|_\infty$ (which gives the size of perturbation caused by using the compressed data). One pair of values is calculated from using the original sensor measurements. The other pair of values is calculated from using the compressed data. It can readily be seen that both the original data and the compressed data give similar results. Since the spectral norm $\rho(\mathbf{F})$ is less than 1, the fixed point is stable. In addition, $\|\mathbf{G}\|_\infty$ is very small, which means that the perturbation on the weights in using the compressed data is minimal. This is also confirmed earlier in Figure 5.10.

We use the results of Proposition 1 to study the effect of large perturbation on the weights w_s. We first use the particular projection matrix that we have used in our experiment to generate a probability distribution of ϵ_s. By assuming that each ϵ_s is independently distributed, we compute the ∞-norm relative perturbation of the weights $\mathbf{w}$ given by the right-hand side of equation (3.30) for 10,000 sets of $\{\epsilon_s\}$ generated. Figure 5.14 shows the cumulative probability distribution of the bound of relative perturbation. It shows that there is a high probability that the relative perturbation is less than 0.02.

The bandwidth savings in using compressive sensing can also be computed. Assuming the sensors use a 10 bit analogue-to-digital converter, then sending all the sensor readings to the data fusion centre will mean each sensor needs to transmit $mb = 4000$ bits of data. With compressive sensing, the number of bits of data that has to be sent by each sensor is $p + 1 + b\lceil \log_2(m) \rceil$ =3200 bits. This represents a 20% savings.

The above results are obtained by using 160 projections per sensor. We investigate the effect of the number of projections per sensor on performance. We use three different performance metrics: (1) Relative error in reconstructing the robust average $\mathbf{r}$, expressed as a percentage, (2) Relative error in the estimation of the weights w_s, expressed as a percentage, (3) The percentage of bandwidth savings. In all cases, the reference is when all the data is sent to the data fusion centre. Table 5.3 shows the

	$\rho(\mathbf{F})$	$\|\mathbf{G}\|_\infty$
Using original sensor readings	0.1358	0.1894
Using compressed data	0.1427	0.1965

Table 5.2: This table compares the values of $\rho(\mathbf{F})$ and $\|\mathbf{G}\|_\infty$ evaluated by using the original sensor readings against those computed by using the compressed data. This table is for the QCAT deployment.

p	40	80	120	160
Metric 1	9.20	4.10	1.70	1.11
Metric 2	5.43	8.12	3.96	2.01
Metric 3	80	60	40	20

Table 5.3: This table shows the effect of the number of projections p per sensor on three performance metrics. (1) Relative error in reconstructing the robust average $\mathbf{r}$, expressed as percentage, (2) Relative error in the estimation of the weights w_s, expressed as percentage, (3) The percentage of bandwidth savings.This table is for the QCAT deployment.

results for 40 to 160 projections per sensors. If we are willing to accept a 5% error in reconstruction of robust average and weight estimation, then we can reduce the number of projections further.

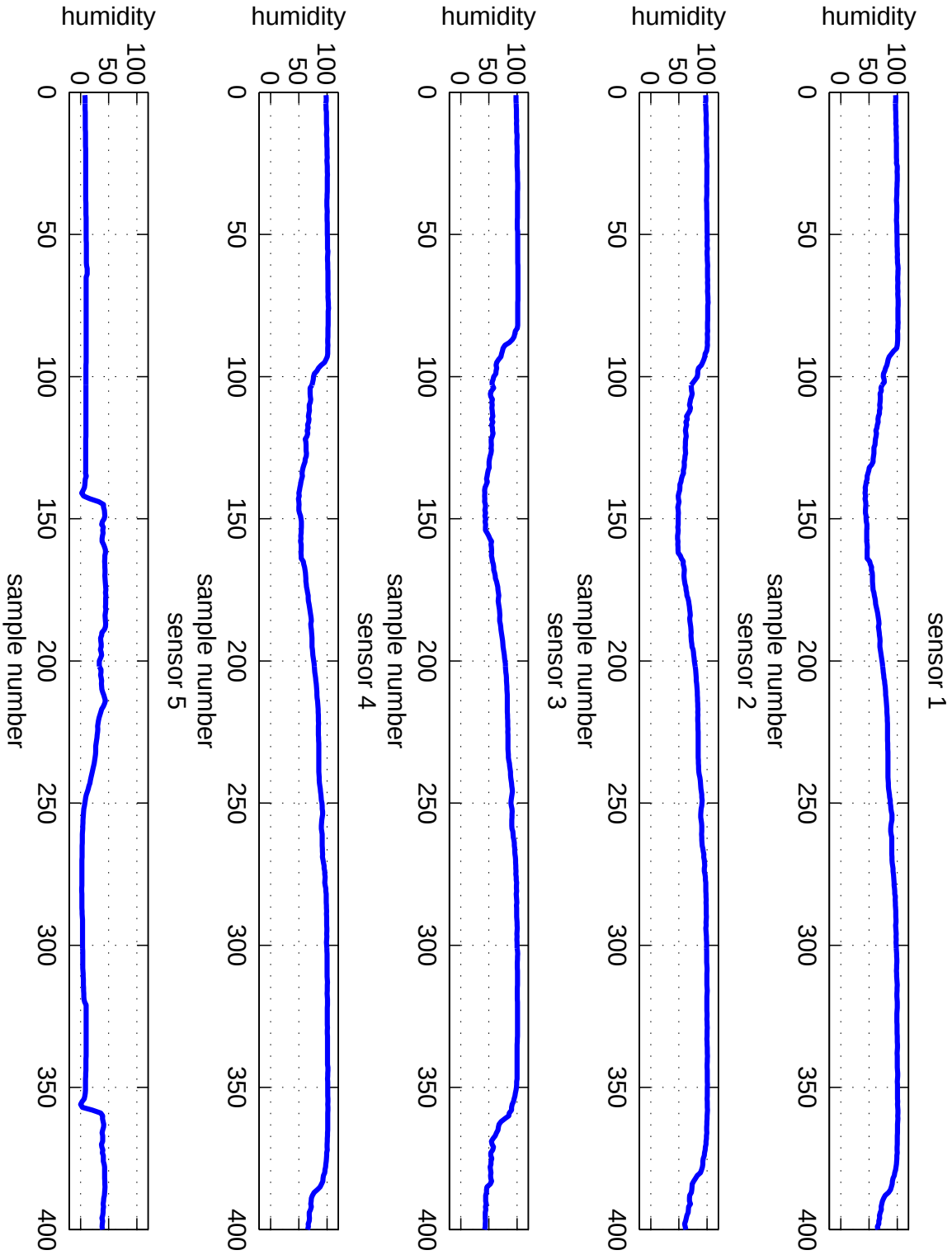

Figure 5.9: Humidity reading from the QCAT deployment. Note that the readings from sensor 5 is very different from the first 4 sensors. For ease of comparison, all plots use the same scale.

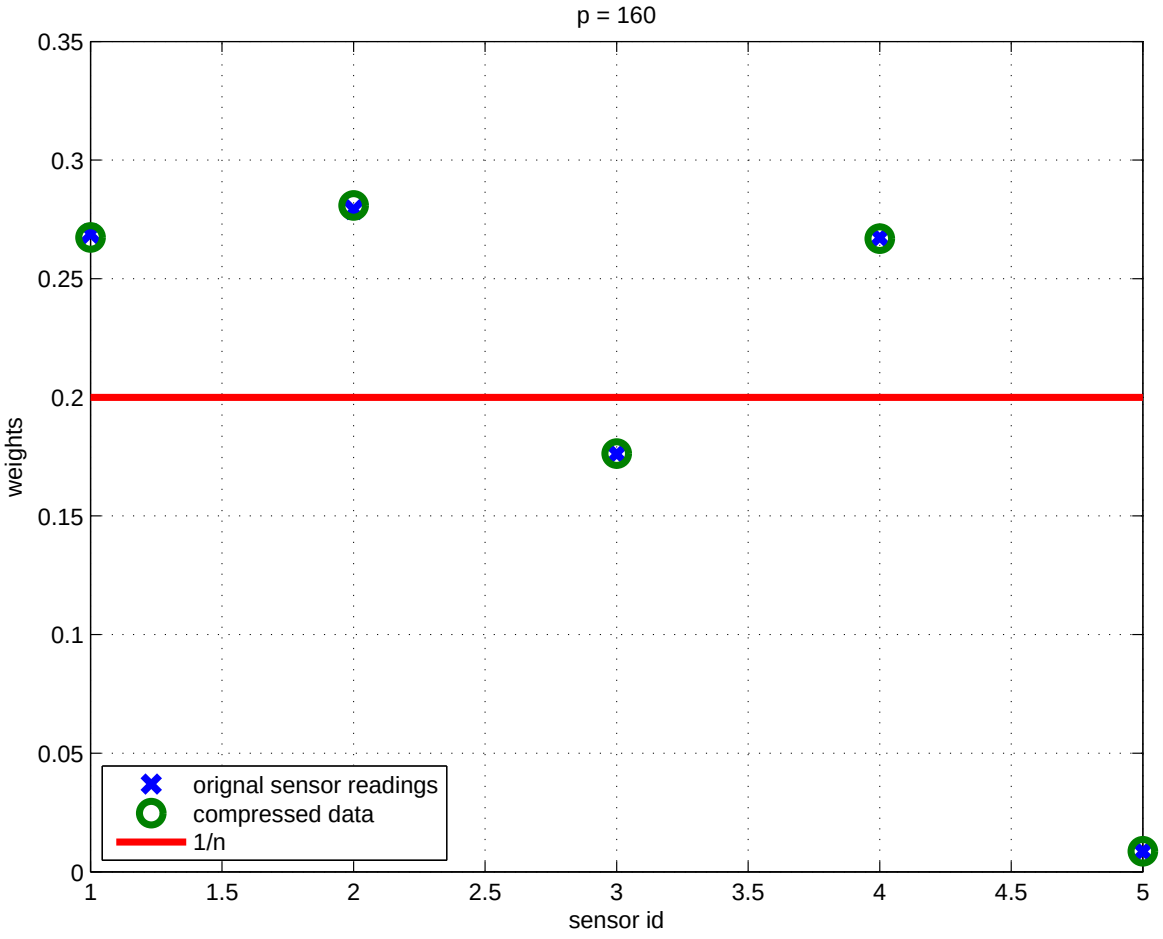

Figure 5.10: The final weightings are shown in circles (from original sensor readings) and crosses (from compressed data). The line is at the level of $\frac{1}{5}$ which is the expected weight when no sensors are faulty.

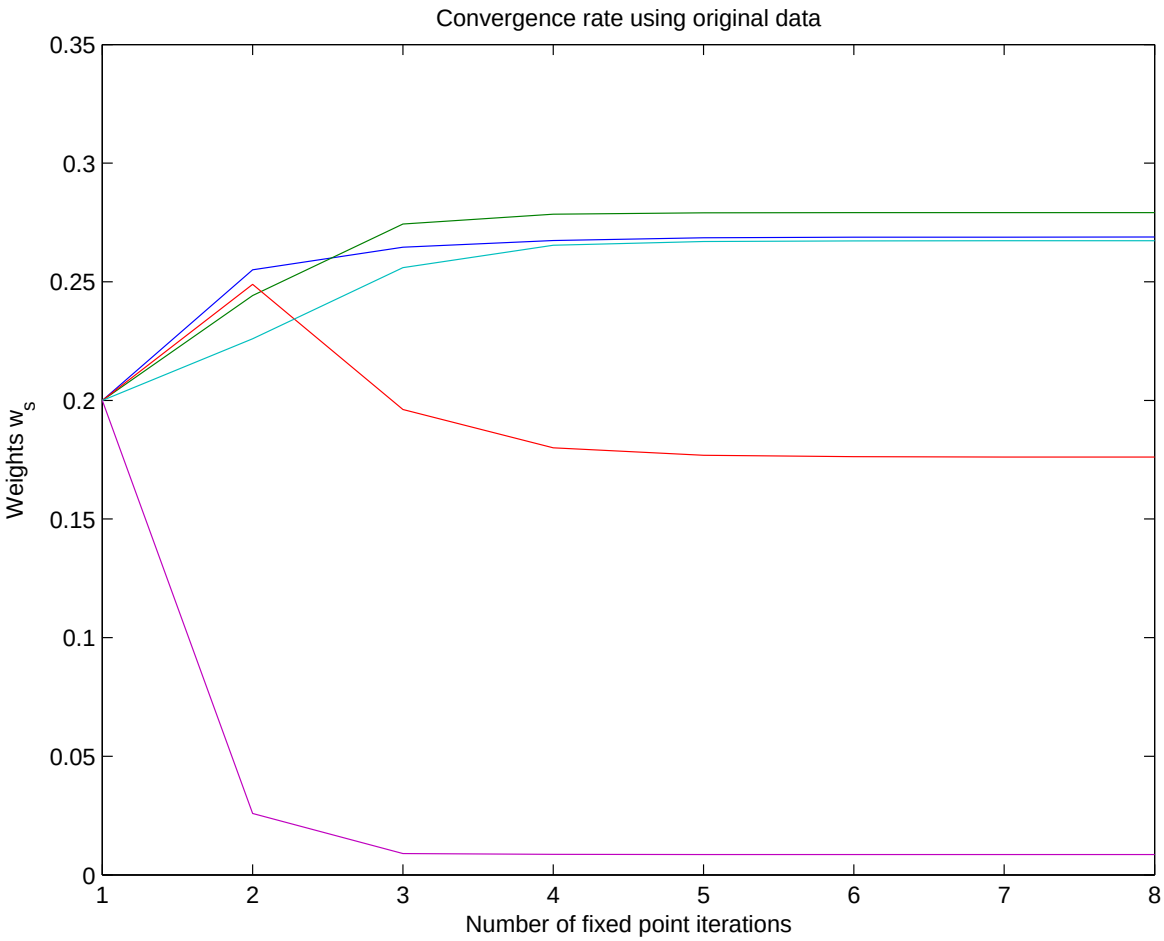

Figure 5.11: This figure shows the convergence of the weights w_s using the original sensor readings. The convergence is achieved within 8 iterations.

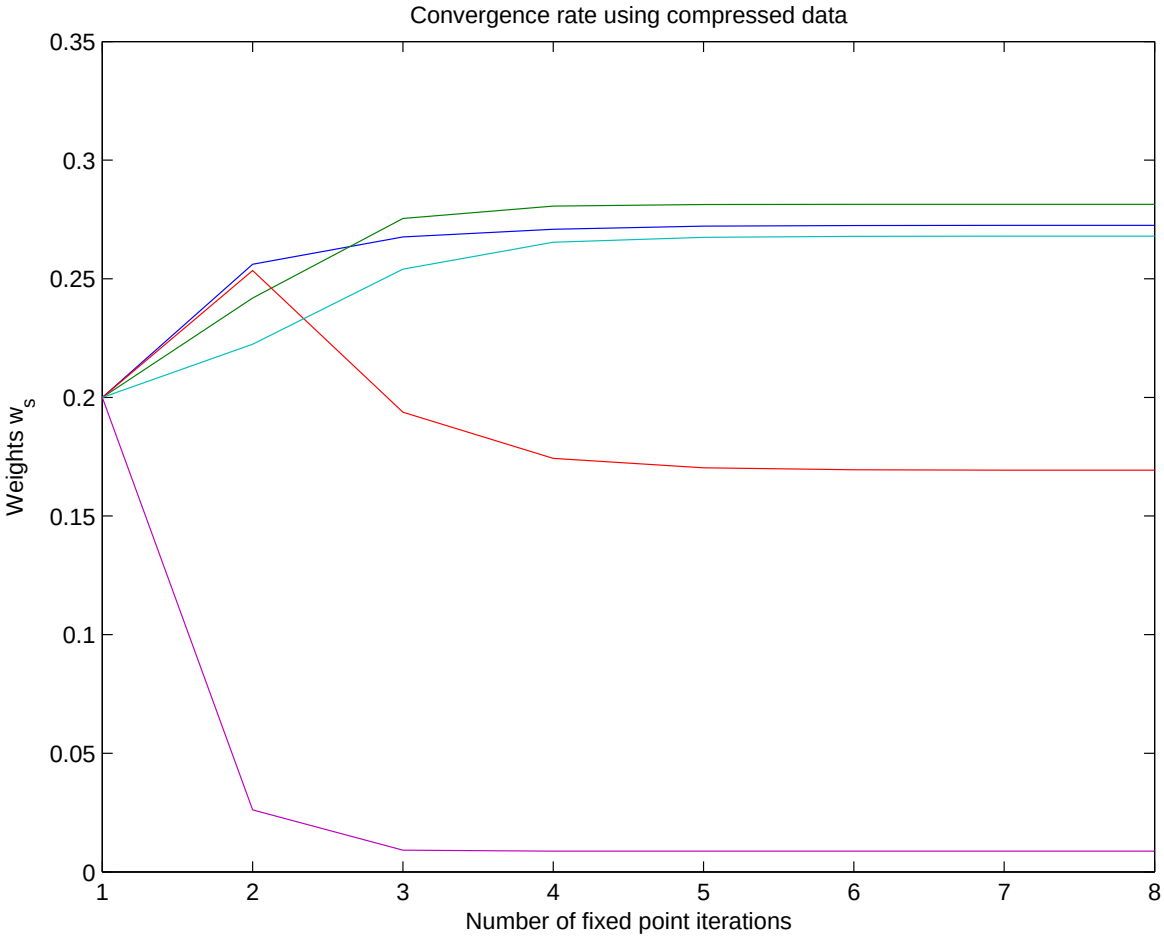

Figure 5.12: This figure shows the convergence of the weights w_s using the compressed data. The convergence is achieved within 8 iterations.

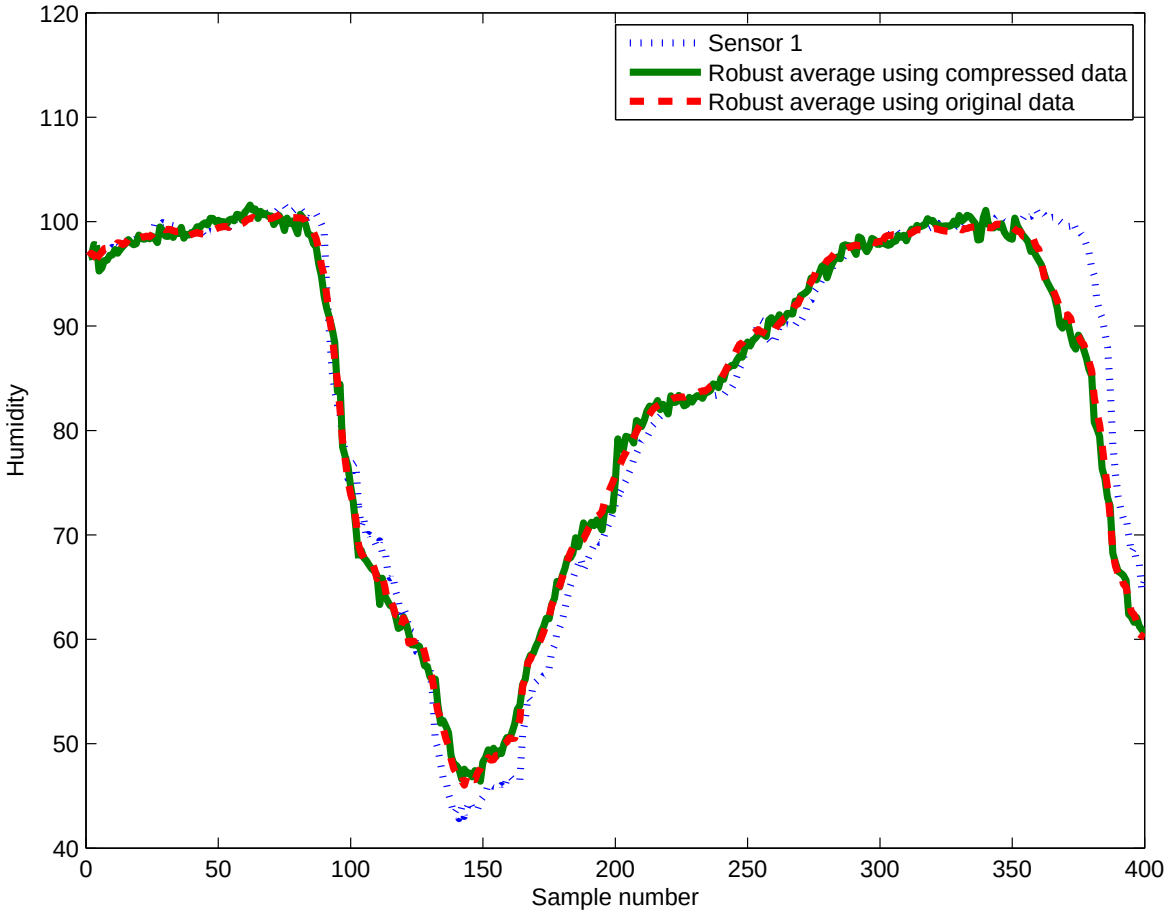

Figure 5.13: The figure shows the robust averages against the trend of one working sensor (the dashed blue line). The red curve shows the robust average computed by applying the fixed point iteration to the original data. The green curve shows the robust average by applying the fixed point iteration to the compressed data followed by reconstruction.

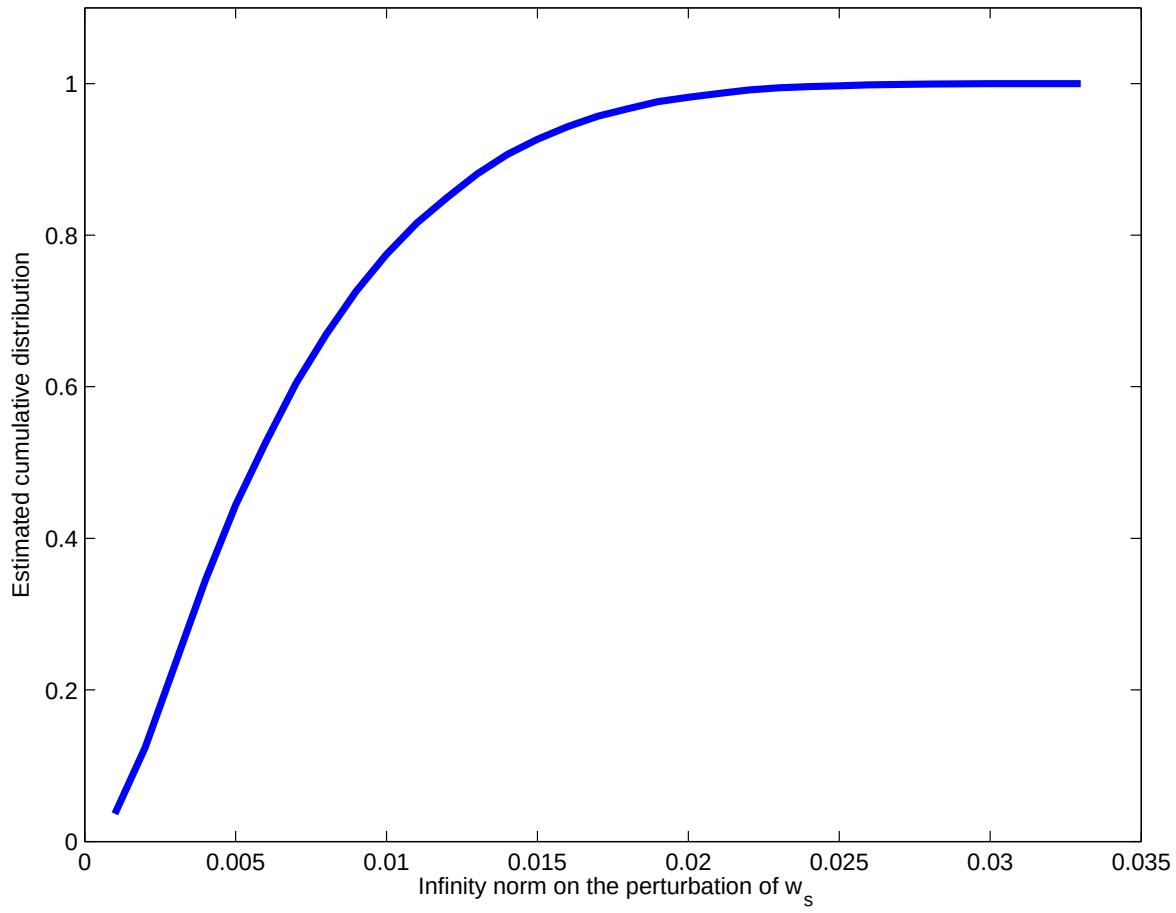

Figure 5.14: This figure shows the cumulative probability distribution of the bound of relative perturbation on the weight vector $\mathbf{w}$ obtained by Monte Carlo simulation.

Table 5.4: The performance of fault detection algorithm in Fleck3b sensor nodes.

Total number of projections	RAM (byte)	ROM (byte)	Computation time (ms)
250	1,338	6,092	616
300	1,538	6,292	734
350	1,738	6,492	852
400	1,938	6,692	890
450	2,138	6,892	1,088

5.3 Distributed implementation on wireless sensor nodes

We implemented and evaluated the proposed fault detection algorithm in a real world sensor platform. We use the Fleck3b sensor node platform [DHS+07], which features an eight MHz Atmel Amega1281 micro-controller with 8 KB Random Access Memory (RAM) and 128 KB flash programmable Read-Only Memory (ROM), as our hardware test environment. We use Fleck OS (FOS) [CS08] as our software test environment. FOS is a C-based cooperative multi-threaded operating system for WSNs. The fault detection algorithm is implemented as an application thread in FOS, and is waken up whenever the projection buffer is full.

We implemented our fixed-point iteration algorithm in C and ran it on the Fleck3b platform using different number of projections with the data set in Section 5.2. Table 5.4 shows the RAM and ROM usage, together with computation time, of proposed fault detection algorithm in Fleck3b sensor nodes against different number of total projections. Note that there are 5 sensors in this data set, therefore, if each sensor uses p projections, the total number of projections will be $5p$. As expected, RAM, ROM and computation time of the fault detection algorithm increased with the number of projections. Note that the increase is almost linear and this makes the fault detection algorithm a scalable solution in WSN. Table 5.4 also shows that the proposed fault detection algorithm is fairly affordable in resource-impoverished sensor platforms. For example, it takes a sensor node 800 seconds (recalled the sampling rate is one per ten seconds and there are five sensors) for a sensor to accumulate a total of 400 projections (= 80 projections per sensor). The required computation time is 890ms, which costs approximately 0.1% of micro-controller's duty cycle in that period.

6 Related work

This paper integrates random projections, Johnson-Lindenstrauss Lemma and compressive sensing to efficiently compute robust averages for WSNs. Here we discuss a number of related works.

There are a number of papers investigating how compressive sensing can be applied in WSNs. We can classify them according to whether projections are computed "temporally" or "spatially". In [DWBB06], "temporal" projections are used where each sensor computes projections of its own data and sends it to the data fusion centre. The work [DWBB06] also discusses how fault tolerance can be realised based on the framework discussed in Figure 1.1 where all signals are reconstructed at the data fusion centre. However, this paper adopts the framework in Figure 1.2 where only a minimal number of signals have to be reconstructed to achieve computation efficiency at the data fusion centre.

Other works in applying compressive sensing in WSNs views the spatial data from the sensors at each time instance as an image or snapshot, and compute projections of the sensor data for each snapshot, see for example [BHSN07, QMM+09, LPS09, CRH09, LWSC09]. All these works aim at

obtaining a sufficient accurate snapshot of the sensor field by using as little energy as possible. The work [BHSN07] realises this goal by using an additive radio channel to compute projections. The works in [QMM+09, LPS09, CRH09, LWSC09] compute projections by message passing.

There is a rich literature in fault tolerance in WSNs. The work [NRC+09] discusses the types of faults that commonly appear in WSNs. The papers [KI04, GBS08] uses a Bayesian approach to detect whether sensors are faulty. The work [Obs09] detect sensor faults by using neural networks. Instead of performing fault or outlier detection, this paper takes a *robust statistics* [Hub04] approach in computing a robust average of the sensor data. This avoids for the problem of false detection and false retention in fault detection [Hub04].

There is a rich literature in WSNs in using data aggregation and in-network processing to reduce the bandwidth requirements [KW05]. A commonly cited example is the computation of network averages [KW05], but without considering possible sensor faults. The work in this paper can also be viewed as in-network processing where each sensor compresses its own data by using a projection matrix.

This paper chooses to compute a robust average using an approximate maximum likelihood criterion. The robustness of maximum likelihood estimation, generally known as the M-estimator, is discussed in [Hub04]. We discuss the use of ℓ_1-norm robust estimator earlier in Section 3.4.

The problem of computing a robust average also appears in some other fields such as reputation ranking in the Internet. The paper [LMZY06] studies the statistical properties of using a maximum likelihood estimator (i.e. the $\lambda = 0$ case in Section 3.2) for reputation ranking. However, this estimator experiences convergence problem [dKD07], a fact which we also pointed out in Section 3.2. In this paper, we have proposed a new robust estimator which can work directly with compressed data. Note that this imposes a new requirement on our robust estimator, namely that the perturbation on the final result due to the use of compressed data should be limited. This is a new requirement and we study the perturbation due to using compressed data in Section 3.3.

The use of projection matrix as a dimensionality reduction method is fairly well known in the data mining and the machine learning communities, see [LHC06] and the references therein. There is also some recent effort in using compressed data (which is obtained from applying random projections to the original data) for classification and detection without first decompressing the data [HCN+06, DDW+07]. The framework depicted in Figure 1.2 opens up the possibility of using some of these algorithms that have been developed in the data mining or machine learning communities to WSNs, e.g. one can perform clustering using the compressed data and reconstruct the cluster centres with compressive sensing.

The connection between random projection matrix and norm preservation is the key idea behind the Johnson-Lindenstrauss Lemma. Furthermore, [BDDW08] shows the connection between Johnson-Lindenstrauss Lemma and compressive sensing. In this paper, we demonstrate how random projection matrix for dimensionality reduction, norm preservation property of Johnson-Lidenstrauss Lemma and compressive sensing can be integrated to efficiently compute robust averages in WSNs.

7 Conclusions

In this paper, we propose a method to compute a robust average of sensor measurements in a wireless sensor network in a bandwidth and computation efficient manner. Our method exploits compressive sensing for efficient data transmission. Furthermore, our method integrates norm-preserving property of random projection matrices and compressive sensing so that the data fusion centre can work directly with compressed data without first decompressing them. This means the data fusion centre

will only need to perform decompression once and this represents a great reduction in computation requirements. We have applied our algorithm to data obtained from two WSN deployments.

Bibliography

[Ach03] D Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, Jan 2003.

[ÅM08] Karl Johan Åström and Richard M. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton University Press, 2008.

[BDDW08] R Baraniuk, M Davenport, R DeVore, and M Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, Jan 2008.

[BHSN07] W Bajwa, J Haupt, A Sayeed, and R Nowak. Joint source–channel communication for distributed estimation in sensor networks. *Information Theory, IEEE Transactions on*, 53(10):3629 – 3653, Oct 2007.

[BJ05] Nirupama Bulusu and Sanjay Jha. *Wireless sensor network systems*. Artech, 2005.

[CRH09] Chun Tung Chou, Rajib Rana, and Wen Hu. Energy efficient information collection in wireless sensor networks using adaptive compressive sensing. *IEEE 34th Conference on Local Computer Networks (LCN 2009)*, 2009.

[CRT06] E Candes, J Romberg, and T Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489 – 509, Feb 2006.

[CS08] Peter Corke and Pavan Sikka. Demo abstract: FOS – a new operating system for sensor networks. In *Proceedings of Fifth European conference on wireless sensor networks (EWSN)*, 2008.

[CT05] E Candes and T Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203 – 4215, Dec 2005.

[CT06] E Candes and T Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406 – 5425, Dec 2006.

[CW08] E Candes and M Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21 – 30, Mar 2008.

[DDW+07] MA Davenport, MF Duarte, MB Wakin, JN Laska, D Takhar, KF Kelly, and RG Baraniuk. The smashed filter for compressive classification and target recognition. *Proceedings of SPIE*, 2007.

[DG03] S Dasgupta and A Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures and Algorithms*, Jan 2003.

[DHS+07] Tuan Le Dinh, Wen Hu, Pavan Sikka, Peter Corke, Leslie Overs, and Stephen Brosnan. Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network. *Local Computer Networks, Annual IEEE Conference on*, pages 799–806, 2007.

[dKD07] C de Kerchove and P Van Dooren. Iterative filtering for a dynamical reputation system. *Arxiv preprint arXiv:0711.3964*, Jan 2007.

[Don06] D Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289 – 1306, Apr 2006.

[DWBB06] Marco Duarte, Michael Wakin, Dror Baron, and Richard Baraniuk. Universal distributed sensing via random projections. *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, Apr 2006.

[GBS08] Saurabh Ganeriwal, Laura Balzano, and Mani Srivastava. Reputation-based framework for high integrity sensor networks. *Transactions on Sensor Networks*, 4(3), May 2008.

[HCN+06] J Haupt, R Castro, R Nowak, G Fudge, and A Yeh. Compressive sampling for signal classification. *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pages 1430 – 1434, Jan 2006.

[HJ90] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. CUP, 1990.

[HN06] J Haupt and R Nowak. Signal reconstruction from noisy random projections. *Information Theory, IEEE Transactions on*, 52(9):4036 – 4048, Sep 2006.

[Hub04] Peter J. Huber. *Robust statistics*. Wiley, 2004.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM*, 53(3), May 2006.

[KI04] B Krishnamachari and S Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *Computers, IEEE Transactions on*, 53(3):241 – 250, Jan 2004.

[KW05] Holger Karl and Andreas Willig. *Protocols and architectures for wireless sensor networks*. Wiley, Jan 2005.

[LHC06] Ping Li, Trevor Hastie, and Kenneth Church. Very sparse random projections. *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2006.

[LHC07] Ping Li, Trevor J Hastie, and Kenneth Church. Nonlinear estimators and tail bounds for dimension reduction in l1 using cauchy random projections. *Journal of Machine Learning Research*, 8:2497—2532, Aug 2007.

[Li07] Ping Li. Very sparse stable random projections for dimension reduction in l (0 ¡ 2) norm. *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2007.

[LMZY06] P Laureti, L Moret, Y Zhang, and Y Yu. Information filtering via iterative refinement. *Europhysics Letters*, Jan 2006.

[LPS09] S Lee, S Pattem, and M Sathiamoorthy. Spatially-localized compressed sensing and routing in multi-hop sensor networks. *3rd International Conference on Geosensor Networks (GSN)*, 2009.

[LWSC09] Chong Luo, Feng Wu, Jun Sun, and Chang Chen. Compressive data gathering for large-scale wireless sensor networks. *MobiCom '09: Proceedings of the 15th annual international conference on Mobile computing and networking*, Sep 2009.

[NRC+09] Kevin Ni, Nithya Ramanathan, Mohamed Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *Transactions on Sensor Networks*, 5(3), May 2009.

[Obs09] Oliver Obst. Poster abstract: Distributed fault detection using a recurrent neural network. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN 2009)*, pages 373–374, 2009.

[QMM+09] G Quer, R Masiero, D Munaretto, M Rossi, J Widmer, and M Zorzi. On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. *Information Theory and Applications Workshop (ITA 2009)*, 2009.

[TG07] J Tropp and A Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655 – 4666, Dec 2007.

[Whi82] H White. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society*, Jan 1982.

[Zam66] G Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *Automatic Control, IEEE Transactions on*, 11(2):228 – 238, Jan 1966.

A Expressions for Jacobian matrices

The Jacobian matrix $\mathbf{T_{wv}}$ is given by $\mathbf{T_{wv}} = \mathbf{J_1 J_2}$ where

$$[\mathbf{J_1}]_{ij} = \frac{\delta_{ij}}{\sum_{k=1}^{n} \tilde{w}_k} - \frac{\tilde{w}_i}{(\sum_{k=1}^{n} \tilde{w}_k)^2} \tag{A.1}$$

$$[\mathbf{J_2}]_{ij} = \frac{\frac{v_i}{(\sum_{k=1}^{n} v_k)^2} - \frac{\delta_{ij}}{\sum_{k=1}^{n} v_k}}{\frac{1}{(\frac{1}{\sum_{k=1}^{n} v_k} + \lambda)^2}} \tag{A.2}$$

where

$$\tilde{w}_s = \frac{1}{\frac{1}{\sum_{k=1}^{n} v_k} + \lambda}, \tag{A.3}$$

$[\bullet]_{ij}$ denotes the (i,j)-element of a matrix and $\delta_{ij} = 1$ if $i = j$ and is zero otherwise.

The Jacobian matrix $\mathbf{T_{vw}}$ is

$$[\mathbf{T_{vw}}]_{ij} = 2\mathbf{x_j}^T(\mathbf{r} - \mathbf{x_i}) \tag{A.4}$$

B Proof

By using equations (3.19) and (3.20), we have

$$\mathbf{w^1} = \mathbf{T_{vw}}(\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1})) \tag{B.1}$$

We will "center" this equation by using the following operations:

$$\begin{aligned}
& \mathbf{w^1} = \mathbf{T_{vw}}(\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1})) \\
\Leftrightarrow \quad & \mathbf{T_{vw}^{-1}}(\mathbf{w^1}) = \hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) \\
\Leftrightarrow \quad & \mathbf{T_{vw}^{-1}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1}) = \hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1}) - \mathbf{T_{wv}}(\mathbf{w^1}) \\
\Leftrightarrow \quad & (\mathbf{T_{vw}^{-1}} - \mathbf{T_{wv}})(\mathbf{w^1}) = (\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^1}) \\
\Leftrightarrow \quad & \mathbf{w^1} = (\mathbf{T_{vw}^{-1}} - \mathbf{T_{wv}})^{-1}(\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^1}) \\
\Leftrightarrow \quad & \mathbf{w^1} = \mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^1}))
\end{aligned}$$

Note that the Johnson-Lindenstrauss Lemma says that $\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1})$ appears as a perturbation around $\mathbf{T_{wv}}(\mathbf{w^1})$ (see equation (2.14)), therefore the aim of the above operation is to "centre" $\hat{\mathbf{T}}_{\mathbf{wv}}(\mathbf{w^1})$ around $\mathbf{T_{wv}}(\mathbf{w^1})$. The "centering" method has also been used in [Zam66] to obtain tighter stability bound in the presence of conic sector-bound non-linearity.

Consider the norm of $\mathbf{w^1} - \mathbf{w^0}$, we have

$$\begin{aligned}
& \|\mathbf{w^1} - \mathbf{w^0}\| \\
= \quad & \|\mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})) - \mathbf{w^0}\| && \text{(B.2)} \\
= \quad & \|\mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})) - \mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^0})) + \mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^0})) - \mathbf{w^0}\| && \text{(B.3)} \\
\leq \quad & \|\mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^1})) - \mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^0}))\| + \\
& \|\mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})(\mathbf{w^0})) - \mathbf{w^0})\| && \text{(B.4)} \\
\leq \quad & \|\mathbf{T}(\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T_{wv}})\|_L \|\mathbf{w^1} - \mathbf{w^0}\| + \|\mathbf{T}(\hat{\mathbf{T}}_{\mathbf{wv}}\mathbf{T_{wv}}) - \mathbf{I}\|\|\mathbf{w^0}\| && \text{(B.5)}
\end{aligned}$$

where $\mathbf{I}$ in equation (B.5) denotes the identity map. In the above derivation, equation (B.3) is obtained by adding and subtracting the term $\mathbf{T}((\hat{\mathbf{T}}_{\mathbf{wv}} - \mathbf{T}_{\mathbf{wv}})(\mathbf{w^0}))$ within the norm. Equation (B.4) is obtained by using the triangle inequality, and equation (B.5) is obtained from using the definition of the Lipschitz operator norm and operator norm.

Proposition 1 is now obtained by re-arranging the last inequality.