Adaptive Sampling in Wireless Sensor Networks for Air Monitoring System

Yongjae Jon
Abstract

Adaptive Sampling in Wireless Sensor Networks for Air Monitoring System

Yongjae Jon

In wireless sensor networks (WSN), it is important to use resources efficiently because sensors have limited resources such as battery life and computational power. In this thesis, we study the method which can save energy of air-monitoring sensor networks with respect of QoS (quality of service). From historical data, we observe that during certain time of the day, concentration of air pollutants has no radical change, from which we can conclude that applying high sampling rate uniformly all the time is not necessarily required. Our approach uses Kalman filter technique to eliminate the noise from the sensor measurements, and adjust the sampling interval based on the difference between the present and previous measurements. If the sampling interval is within the sampling interval range, we use the new sampling interval for the next measurement and if not, a central server assigns a new sampling interval and sampling interval range to the requesting sensor. This way, we can achieve adaptive sampling based on input characteristics so as to save energy of the sensor network and also to obtain proper accuracy of sensor measurements. We simulated our method with real measurement data with Matlab and finally implemented our method in the GreenIoT project to demonstrate the energy-efficiency and sensing-quality of our technique.
Acknowledgments

I would like to thank my supervisor Xiuming Liu for helping me solve problems during project and also for professional guidance. His advices and comments were really helpful to improve my project. I also should thank my reviewer Edith Ngai who helped me define the project and provided great support in writing thesis. Her beautiful smile and kindness encouraged me to overcome obstacles while the project was carried out. I am also grateful to Kjell Brunberg and Adrian Caragea at Upwis who gave me a chance to work with them and taught me lots of things about sensors and programming. Finally, I own my deepest gratitude to my family supporting me all the time and to my friends who stayed with me sharing happiness and sadness, and gave me power and courage.
Chapter 1

Introduction

1.1 Urban environmental monitoring

Environmental monitoring can be defined as the systematic sampling of air, water, soil, and biota in order to observe and study the environment, as well as to derive knowledge from this process [1]. Since things, such as to establish environmental baselines and to inform the public about environmental conditions, are very important issues around the globe, environmental monitoring gets a lot of attention from governments and communities. Especially, environmental monitoring is a critical process in cities to ensure public safety including the state of the national infrastructure, to set up continuous information services and to provide input for spatial decision support systems [2].

In recent years, economy and urbanization have grown fast and have caused serious problems like pollution particularly in urban areas. The quality of air, water and soil is getting worse and worse, and they are even containing materials harmful to human health. One of the most serious pollutions is air pollution, and in Uppsala, there is a project called “GreenIoT” which is basically an Internet-connected wireless sensor network that senses the air quality of Uppsala.

In this paper, I study the method to exploit resources, such as energy and network bandwidth, of the wireless sensor network of “GreenIoT” project.

1.2 Wireless sensor network

Wireless sensor network (WSN) is a set of sensors which are spatially distributed to monitor physical environments, such as temperature, humidity, air quality, etc., and configured to form a network [3].

WSNs consist of sensor nodes, sink node (gateway) and software. Sensors in the WSNs are called a “sensor node” or just a “node”. Each sensor node typically has a radio transceiver, a microcontroller and a power source, usually a battery. Sensor nodes monitor the environment and sends the measurement data (either directly or through other nodes) to the “sink node” which collects
the data from nodes wirelessly and transmits them to the central server. WSN is used in many industrial, military and consumer applications including environmental monitoring, military surveillance, medical care, tracking vehicles and so forth [4]. For example, in forest fire detection, sensor nodes strategically, randomly and densely deployed in a forest relay the exact origin of a fire before the fire is spread uncontrollable. Millions of sensor nodes can be deployed in a forest and collaborate with each other to perform distributed sensing and overcome obstacles like trees which can block signal propagation. Since sensor nodes are left unattended for years, they must have an effective proper power source such as high capacity batteries or solar cells.

WSNs have many advantages. It is obvious that we can avoid a lot of wiring so it is convenient to use. One of the benefits is that it makes able and easy to monitor harsh environments where human cannot approach or stay, such as near volcanoes. However, due to size and cost constraints, it has also disadvantages such as energy, memory, computational speed and communications bandwidth [3, 5]. In particular, energy limitation is the critical point because sensor nodes usually carry limited power sources and also it is inconvenient or impossible to replace or recharge batteries because of hugeness or hostile environments.

In this paper, the adaptive sampling method is introduced to a wireless sensor network which measures air pollutants in Uppsala to reduce the energy consumption of the sensor nodes.

1.3 Sampling and its impact on QoS of WSN

In signal processing, sampling is the reduction of a continuous signal to a discrete signal by taking samples among the original signal [6]. The figure (1.2) shows the sampling process. The continuous signal is a green line while the discrete samples are blue vertical lines.

A sample is a value at a point in time or any other dimension and if we get samples every T seconds, we call the sampling interval is T. We also define “sampling frequency” or “sampling rate” $f_s$ which is the average number of
samples obtained in one second which is represented as $f_s = 1/T$.
Since sensors send sample measurements of environment to the server, sampling rate affects the QoS (quality of service) of WSN. If we have a too low sampling rate, sensor measurements will be missing a lot of information so we cannot get the high-quality data. On the other hand, if we set the sampling rate too high, the measurements will be more accurate but there will be a lot of redundant traffic load on the communication channel between sensor nodes and central server and also computational power will be wasted, both of which result in a short life time of sensors. In short, there is a trade-off that gives users the option of prolonging network lifetime at the cost of lower throughput or higher transmission delay.
In this paper, we propose an adaptive sampling method to achieve high sensing quality as well as extend the network lifetime.
Chapter 2

System overview

2.1 GreenIoT project

The Internet of Things (IoT) is a recent communication paradigm in which physical objects of our life can communicate with one another by equipping microcontrollers, transceivers and proper protocol stacks, becoming an integral part of the Internet [8]. Smart homes are one of the practices. Smart homes have self-configured sensors and actuators which can be controlled remotely via the Internet, enabling various monitoring and control applications [9]. GreenIoT is also one of the IoT related projects in Sweden focusing on energy-efficient IoT technology for smart city [10]. In this project, we will find a method which uses green networking, smart sensing and cloud computing technologies to provide more interactive and approachable city planning (Figure(2.1)). We can apply this GreenIoT platform to many cases like environmental monitoring, transportation and home security.

IoT technology has been developed significantly in hardware, software and protocol design, however, there are still many challenges like how to extract useful information from enormous amount of data generated by devices. In this project, we achieve intelligent data management by using cloud computing, and energy-efficient and sustainable operations are attained by incorporating green networking and sensing techniques.

One major goal of the project is an integrated solution for an environmental sensing system, which enables experimentation with applications and services using open environmental data, in particular for sustainable urban and transportation planning (See Figure(2.2)) [10].

2.2 Upwis sensor

Upwis sensor, also called “U115”, is a sensor made at Upwis which is a company in Uppsala manufacturing sensors. In GreenIoT project, we deploy a lot of U115 sensors to monitor environments. U115 sensor board is shown in the Figure 2.3.
This sensor node has a Cortex M3 processor and local sensors for 3D acceleration, 3D gyro, 3D magnetic field/compass, humidity, temperature, noise/audio, barometric pressure, light, NO₂, CO and IR/proximity. This sensor is powered by external 4.1V LiPo cell which can be charged by USB or +5V, and the onboard RF transceiver and PCB antenna is for 2.4GHz.

Figure 2.4 is showing the block diagram of U115 and Table 2.1, 2.2 shows a technical specification of U115 sensor node.

### 2.3 6LoWPAN and MQTT

6LoWPAN (IPv6 over Low power Wireless Personal Area Network) is a protocol which enables IPv6 packets to be carried on top of IEEE 802.15.4 networks. 6LoWPAN is used to apply Internet Protocol to smallest devices and to let low-power devices such as sensors take part in Internet of Things [11]. Since IoT should reach an efficient support for global communications, access to services and information, discovery and look-up, it is necessary to enable a communication mechanism which allows global access to devices, sensors and smart objects [12]. Thus 6LoWPAN is the answer to the technological requirements due to its own advantages and attributes.

MQTT (Message Queuing Telemetry Transport) is an open publish-subscribe
Table 2.1: CPU board module of U115

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Technical specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>ST STM32L151VC 100pin LQFP</td>
</tr>
<tr>
<td>CPU speed</td>
<td>Max 72MHz, Min 32KHz or stopped</td>
</tr>
<tr>
<td>FLASH</td>
<td>128k</td>
</tr>
<tr>
<td>RAM</td>
<td>32k</td>
</tr>
<tr>
<td>Serial Interface</td>
<td>3 serial ports, 3 SPI ports, $I^2C$</td>
</tr>
<tr>
<td>Flash Update</td>
<td>via JTAG or Serial Port/Radio</td>
</tr>
<tr>
<td>Supply Voltage Charging Power</td>
<td>+5V DC, 200mA 5% from USB</td>
</tr>
<tr>
<td>Supply Voltage LiPo Cell</td>
<td>+2.7V to 4.1V DC</td>
</tr>
<tr>
<td>Current Consumption LiPo Cell</td>
<td>72MHz : 0.06A typ</td>
</tr>
<tr>
<td></td>
<td>32KHz : 2mA typ</td>
</tr>
<tr>
<td></td>
<td>0KHz : 10uAtyp</td>
</tr>
<tr>
<td>Temperature</td>
<td>Operation and Storage −40° to 85°C</td>
</tr>
<tr>
<td></td>
<td>Operation : 10% to 90%</td>
</tr>
<tr>
<td></td>
<td>Storage : 5% to 95% (non-condesing)</td>
</tr>
<tr>
<td>Humidity</td>
<td>56 X 44mm</td>
</tr>
<tr>
<td></td>
<td>≤ 10nm with CPU</td>
</tr>
<tr>
<td>Mechanical Dimensions</td>
<td></td>
</tr>
<tr>
<td>Thickness (Bottom + Top)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: On-board Sensors of U115

<table>
<thead>
<tr>
<th>Sensor types</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration Sensor</td>
<td>LSM9DS1</td>
</tr>
<tr>
<td>Magnetic Sensor</td>
<td>LSM9DS1</td>
</tr>
<tr>
<td>Gyro Sensor</td>
<td>LSM9DS1</td>
</tr>
<tr>
<td>Light Sensor</td>
<td>MAX44009</td>
</tr>
<tr>
<td>Humidity Sensor</td>
<td>SHT21</td>
</tr>
<tr>
<td>Pressure Sensor</td>
<td>MPL3115</td>
</tr>
<tr>
<td>Temperature Sensor</td>
<td>SHT21</td>
</tr>
<tr>
<td>Proximity Sensor</td>
<td>AMG8831</td>
</tr>
<tr>
<td>IR Image Sensor</td>
<td>AMG8831</td>
</tr>
<tr>
<td>Gas Sensor $NO_2$</td>
<td>MICS-4514</td>
</tr>
<tr>
<td>Gas Sensor CO</td>
<td>MICS-4514</td>
</tr>
</tbody>
</table>
Figure 2.2: Sensor data collected from different data sources are necessary to support smart city development.\cite{10}

protocol designed for constrained devices used in telemetry applications \cite{13}, MQTT supports the connections with remote locations with limited network bandwidth or with low data rate. Since publish-subscribe messaging pattern requires a message broker which distributes messages to clients based on the topic of a message, MQTT also needs a MQTT broker. MQTT is widely used in wireless networks that has some latency which is caused by bandwidth constraints or unreliable connections.

In GreenIoT project, sensors monitor the environment and send the measurements to a server via 6LoWPAN Router by using MQTT protocol.
Figure 2.3: Upwis sensor node: U115

Figure 2.4: Block diagram of U115
Chapter 3

Related works

In wireless sensor networks, sensor nodes are generally powered by battery with a limited energy budget. Moreover, since too many nodes could be deployed over a wide area or deployed in a hostile or harsh environment such as around the volcanoes and in the deep sea, it could be impossible or inconvenient to recharge or replace the battery when they run out. Therefore, how to reduce the energy consumption of sensor nodes has been a critical issue in order to extend the lifetime of the network to fulfill the application requirements [14]. Many researchers proposed new MAC protocols for WSNs to achieve energy conservation [15, 16, 17]. In [15], an adaptive MAC protocol is presented to guarantee the pre-configured network lifetime while reducing the end-to-end latency. In order to accomplish this, the protocol introduces the adaptive duty cycle which is adjusted based on the ratio of the remaining energy to the initial energy and the pre-configured lifetime. Thus if a sensor node has a lot of energy, it wakes up frequently so that relaying data can be fastened, and on the other hand, a sensor node with low energy has a long sleep-mode. This way the sensor node can run out of energy around the pre-configured lifetime.

There are other studies to save energy of sensor network by using good routing protocols [18, 19, 20, 21, 22]. [18] argued that it is impossible to derive optimal routing in wireless sensor network, and proposed a new routing method which works based on energy histogram and traffic flow. Finally several techniques are compared and it was found that good routing can increase the network lifetime. Ambient backscatter method proposed in [23] let sensors use TV signals and other source of RF signals as both the source of power and the means of communication. This design can avoid the expensive process of generating radio waves by using backscatter communication which is more power-efficient than traditional radio communication. Although this method does not require a dedicated power infrastructure and batteries, it is only available where ambient RF signals exist.

There has been studies about participatory sensing which coordinates mobile phone-based sensing with stationary sensors [24, 25, 26, 27]. In [24], a collaborative sensing paradigm is proposed which exploits mobile phones and station-
ary sensors to optimize the sensing. In order to obtain better sensing quality while reducing the energy consumption in stationary sensors, it enables the mobile phones and stationary sensors to complement each other. It gives mobile phones a higher priority to perform sensing which means stationary sensors perform sensing if the required sensing rate is not reached after mobile phone sensing, so in the area where many mobile phones exist we can save energy of wireless sensors as well as provide good quality of service.

In [28], on-demand network flooding method is proposed which supports on-demand multi-hop flooding with end-to-end latencies of tens of milliseconds, while dissipating less than ten microwatts during periods of inactivity. This paper argues that although previous studies are for disseminating periodic events through a network in a multihop topology, in reality, many events are not periodic but occur only on rare events. Thus they demonstrate a new low-power protocol design which performs on-demand wake-up of a multi-hop network using low-complexity radio hardware.

In addition to studies above, there has been a lot of researches about adaptive sampling in sensor networks. The use of adaptive sampling and bandwidth management in sensor networks has been well-motivated in [29, 30, 31, 7, 32]. In [30], they propose a method called “Backcasting” which consists of two steps. In the first step, called “preview” step, they use only subset of sensors to estimate the environment. According to the result of the first step, it is determined that how many extra sensors are needed in order to achieve a desired level of accuracy. Then in the second step, called “refinement” step, additional sensors are activated. Thus fewer sensors are used to obtain enough sensing accuracy so energy is saved compared to dense, non-adaptive sampling. However, this method is available when a sensor network covers a wide area and consists of a lot of sensors. In addition, this method tries to save energy by reducing the number of sensors activated, not by decreasing the energy consumption in each sensor.

One adaptive sampling method which uses Kalman filter is described in [7]. Here they use Kalman filter to estimate the environment in video surveillance application and if estimation and measurement is quite different, it increases the sampling rate. This method is good but it can change the sampling rate only when the Kalman filter estimation is much different from the measurement. Thus, it could be a good way for applying to tracking systems but not for air monitoring.
Chapter 4

Problem formulation

4.1 Motivation

In WSN, energy conservation is very important since sensor nodes are operating with their own limited battery and if large number of sensors are deployed over wide area, it is very difficult to replace the batteries of all sensors. In order to maintain a sensor network for a long time, we have to maximize battery lifetime of sensor nodes.

In sensor nodes, energy is consumed in many ways such as transmitting and receiving the data, sensing, data processing, etc. Among all these, data transmission is very expensive in terms of energy consumption, while data processing consumes significantly less [33]. Therefore, we can save considerable energy by reducing the number of transmission in sensor networks.

In GreenIoT project, sensors monitor air pollutants and send the data to the sink node periodically all the time. If the air quality is same for certain period, transmission during that time can be omitted, and from historical data, we found that there are some periods when air pollutants have almost same or at least, no radical change. The figure (4.1), (4.2), (4.3) show the nitrogen dioxide ($NO_2$) in several areas in Sweden on Feb 2, March 4, April 7 in 2015. As we can see in the graphs, the amount of $NO_2$ in the air is quite stable between 0.am and 5.am and has no big variance from 9.pm to 12.pm. This means we do not have to send the data as often as we do from 16.pm to 20.pm during which we have a lot of large fluctuations, and thus we can save energy for the transmissions we can omit in those time periods.

4.2 Challenges

Adaptive sampling has been developed by several researchers so far and gained a lot of progress in theoretical and practical field. However, adaptive sampling based on input characteristic in air monitoring is still not that sophisticated since it has some challenging problems.
One challenge is the accuracy of raw sensor measurements. Adaptive sampling determines the sampling interval based on the measurement values from the sensors so we can easily fall into wrong results if sensors are not quite accurate. Another challenge is that air pollutants such as nitrogen dioxide or ozone have different properties. Our method adjusts the sampling interval based on input characteristic which is the characteristic of the air pollutants. Since every air pollutant has its own attribute, the adaptive sampling method should work differently with different air pollutants in order to achieve our goal.

4.3 Solution

The goal of this project is to save energy of the sensors while not degrading the sensing quality by adjusting the sampling rate. There are several ways to do this but what we study here is an adaptive sampling based on the input characteristics. Specifically, if the input measurement is changing rapidly (both increasing and decreasing), we raise the sampling rate to observe the change of air pollutants more correctly. If input is same or changed a little, we keep the low sampling rate.

In our project, sensors have “sampling interval (SI)” and “sampling interval range (SIR)” and they can modify the SI independently only within the SIR. If modified SI is out of the SIR, the sensor requests the new SIR from the server. Here is a detailed description of the method. First, we calculate the difference between the present measurement and the previous measurements to find whether the air pollutants have large fluctuations or quite stable. Small difference means that there is no big change in air pollutants while big differ-
ence shows that there must be some unexpected events such as heavy traffic and fire which we have to observe carefully. Next, we adjust the SI with a certain formula which represents the relationship between difference and SI. Then, we check if modified SI is within the SIR. If the new SI belongs to the current SIR, we use that SI and if not, sensor asks the server to assign a new proper SIR and SI. When the server gets the request from the sensor, it allocates a new SIR to the requesting sensor according to the available bandwidth, network contention and sensor priority. This way, we can detect the moments when there are unexpected events, and monitor them more carefully than usual by adjusting the sampling rate based on input characteristics which leads to reduced transmission. Therefore, we can save energy as well as achieve high quality of service.

However, there is one more problem - “sensor measurement noise”. Sensors have certain noises and especially noise of gas sensor measurements is not negligible. Since our adaptive sampling method can be very sensitive to the input values, noises above certain amount can result in wrong SIR request to the server which is energy-consuming transmission. Thus we have to eliminate the noise as much as possible before we calculate the difference between the present and the past measurements. As a solution for this, we choose Kalman filter to remove the noise from the sensor measurements.

Figure 4.2: NO2 in several cities in Sweden, 2015-3-4
Figure 4.3: NO2 in several cities in Sweden, 2015-4-7
Chapter 5

Approach

5.1 Workflow of adaptive sampling method

Figure 5.1 shows the work flow of our adaptive sampling method. After every sampling interval, the sensor reads the measurement which includes some noise. Then we use extended Kalman filter to remove the noise from the measurement so as a result, we can get less noisy measurement. With these measurements, we calculate a new sampling interval for next measurement. If the new sampling interval belongs to current sampling interval range, we use it but if not, the sensor requests the new sampling interval range from the server. Then the server gives a proper sampling interval range to the requesting sensor considering the sensor priority, the available bandwidth and network contention.
5.2 Kalman filter

In 1960, R. E. Kalman developed the Kalman filter as a recursive solution to the discrete-data linear filtering problem [34]. Since then, it has been applied in many fields such as data smoothing, object tracking and process estimation. The traditional Kalman filter is a linear algorithm that estimates the internal state of a system based on a prediction/correction paradigm. Here is a brief introduction to the Kalman filter’s mathematical basis, for more details refer [35].

Kalman filter tries to estimate the state $x$ of a discrete-time controlled process whose system model is represented in the form of the following equations

$$x_k = Ax_{k-1} + Bu_k + w_{k-1},$$  \hspace{1cm} (5.1)

with a measurement $z$ that is

$$z_k = Hx_k + v_k.$$ \hspace{1cm} (5.2)

where

- $w_k$: the process noise
- $v_k$: the measurement noise
- $u_k$: the zero-mean white random noise process
- $A$: state transition matrix relating $x_{k-1}$ to $x_k$
- $B$: coefficient matrix of $u_k$
- $H$: matrix relating system state and measurement vector
- $k$: discrete time index

Now we define $x_{k-}$ as a priori state estimate at step $k$ provided that we know the process prior to step $k$, and also define $x_{k+}$ to be a posteriori state estimate at step $k$ given measurement $z_k$. Then a priori and a posteriori estimate error can be defined as $e_{k-} = x_k - x_{k-}$ and $e_k = x_k - x_{k+}$.

The a priori and a posteriori estimate error covariances are then

$$P_{k-} = E[e_{k-}e_{k-}^T].$$ (5.3)

$$P_{k} = E[e_ke_k^T].$$ (5.4)

Here, we describe $x_k$ as a linear combination of an a priori estimate $x_{k-}$ and
a weighted difference between an actual measurement \( z_k \) and a measurement prediction \( Hx_{k-} \) like below.

\[
x_k = x_{k-} + K(z_k - Hx_{k-})
\]  

(5.5)

\( K \) in the above equation is “innovation” or the “residual” and can be calculated by

\[
K_k = P_{k-}H^T(HP_{k-}H^T + R)^{-1}
\]  

(5.6)

The Kalman filter operates by using feedback control, that is, first the filter estimates a process at some time and then use real measurements as a feedback. Thus Kalman filter is divided into two, one is “time update equations” and another one is “measurement update equations”. In time update equations, the filter estimates the future state and error covariance. On the other hand, measurement update equations provide time update equations with feedback. The algorithm is described in the figure 5.2.

Figure 5.3 shows the complete figure of the Kalman filter algorithm.

5.2.1 Extended Kalman filter

Sensor measurements are noisy data. Our adaptive sampling method determines the next sampling rate based on the difference between the present measurement and the previous ones. Thus if there is a large noise, it will affect the result, and since the method is very sensitive to the small change of the measurements, we cannot ignore the noise. Therefore, the noise must be eliminated, and Kalman filter is very powerful when it comes to the noisy systems. In this project, we apply the extended Kalman filter to eliminate the noise from the raw sensor measurement data because air pollution is non-linear.

Here is a brief mathematical overview of extended Kalman filter. First of all,
we have to remove the noise from the raw sensor measurements by using
extended Kalman filter. Extended Kalman Filter is same as Kalman filter but it
is applied to non-linear systems, and since air pollution is non-linear, we use
extended Kalman filter in this project. Extended Kalman filter has two steps:
the prediction step (where the next state of the system is predicted given the
previous measurement data) and the update step (where the current state of
the system is estimated given the measurement at that time).

Prediction:

\[
\begin{align*}
\hat{x}_k^- & = f(x_{k-1}^-, u_k, 0) \\
P_k^- & = A_k P_{k-1} A_k^T + W_k Q_k W_k^T 
\end{align*}
\] (5.7)

Update:

\[
\begin{align*}
K_k & = P_k^- H_k^T (H_k P_k^- H_k^T + V_k P_k V_k^T)^{-1} \\
x_k & = x_k^- + K_k (Z_k - h(x_k^-, 0)) \\
P_k & = (1 - K_k H_k) P_k^-
\end{align*}
\] (5.10)

- \(x_k\) : the state
- \(Z_k\) : the measurement
- \(u_k\) : the zero-mean white random noise process
- \(f()\) : the dynamic model function
- \(h()\) : the measurement model function
- \(K_k\) : Kalman gain

If you want more detail, please refer [34].

In this project, we model the air pollutant by using AR method. AR (autoregresive) approach has been applied to modeling the time series of air pollutants
in several cities and the AR model presented how the current measurement depended on the previous measurements [36]. We assume that air pollutant is the Markov process which is a particular type of stochastic process where only the present value of a variable is relevant for predicting the future and the past history of the variable is irrelevant. Environmental data profiles are usually assumed to follow a Markov process [37]. We use AR(1) model and by using Yule-walker method we calculate the parameters for AR(1) model for this project.

5.3 Calculating the new sampling interval

After Kalman filtering, we should calculate the new sampling interval based on input characteristic, specifically the difference between current measurement and previous measurements.

The variance of the air pollutants measurement is calculated as

\[ \text{dif}(k) = \sqrt{(m(k) - m(k-1))^2} \] (5.12)

- \( \text{dif}(k) \): the difference between kth and the (k-1)th measurement
- \( m(k) \): the kth measurement

The reason we take square of the difference is to eliminate the negative values so that we can increase the sampling rate when the air pollutant is increasing rapidly as well as decreasing significantly. Finally we take square root to get the absolute value of the difference.

Now what we do is dividing the difference by the sampling interval. What we should notice here is that the difference does not represent the air pollutant variance by means of absolute. Here the sensor gets the measurement after the sampling interval at a certain point and we are constantly changing the sampling interval. Therefore, if the sampling interval is big, the difference could be big because the period of change is long. As a result, we should evaluate the variance of air pollutants with differences per minute which is a gradient of the differences. The gradient is calculated by

\[ \text{graddif}(k) = \frac{\text{dif}(k)}{SI} \] (5.13)

- \( SI \): sampling interval at the kth measurement

Our method has a sliding window of size \( W \) that holds the last \( W \) values of the difference gradients. The reason is that in order to adjust the sampling rate based on input characteristics, we should consider not only the measurement value at one moment but also the trend of the past measurements. Thus if \( W = 3 \), the average difference is calculated by

\[ \text{avdif} = w(k) \cdot x + w(k-1) \cdot y + w(k-2) \cdot z \] (5.14)

- \( x, y, z \): user parameters which are constrained with \( x + y + z = 1 \) and \( x > y > z \)
Table 5.1: SIR when \( b = 2 \), \( SI_{\text{max}} = 15 \)

<table>
<thead>
<tr>
<th>SIR</th>
<th>SI</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-2</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2-4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4-6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6-8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8-10</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>10-12</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>12-14</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>14-15</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

- **avdif**: parameter which is calculated by considering differences and trends

Equation (5.14) makes the current measurement have higher priority and weight. Now we should adjust the sampling interval with this average difference and there is a equation as below.

\[
SI_{\text{new}} = SI_{\text{now}} - (avdif \times c - a \times b)
\]  
(5.15)

- \( SI_{\text{new}} \): new sampling interval
- \( SI_{\text{now}} \): current sampling interval
- \( a \): parameter for every SIR
- \( b \): range of SIR
- \( c \): parameter which controls the sensitivity of the adaptive sampling method

Here, “a”, “b” and “c” are user parameters which can be determined by user according to the target air pollutant. We can get the optimal values for “a”, “b” and “c” by doing simulations with target air pollutant dataset.

Descriptions about the user parameters are here. First of all, “b”, the range of SIR, is important because if “b” is big, new SI is likely to be within the current SIR and if not, new SI can be easily out of the present SIR so the sensor should request the new SIR to the server more often. Parameter “a” is kind of an index for every SIR ranges. You can get the idea about “a” in the Table 5.1. The reason why we introduce “a” is to avoid the situation that SI keeps decreasing or increasing when the air pollutants is constantly changing.

User parameter “c” which controls the sensitivity of the adaptive sampling is also important because the air pollutant measurements have different characteristics. The figure (5.4), (5.5), (5.6) show that \( NO_2, Ozone, CO \) on March 12, 2015. As we can see, their attributes are very different, especially carbon monoxide has a low average value and few fluctuations. In addition, the fluctuation range is much smaller than other air pollutants which means that “avdif” in equation
(5.15) should be small. Therefore, in order to get proper adaptive sampling results, we should increase the parameter “c” and this is why “c” is called sensitivity parameter. Finally, we check that new sampling interval is in the current sampling range with equation (5.16).

\[(SI_{last} - b/2) < SI_{new} < (SI_{last} + b/2)\]  

(5.16)

If \(SI_{new}\) satisfies the equation (5.16), we use the \(SI_{new}\) as a new sampling interval and keep the current sampling interval range; otherwise, the sensor requests the new sampling interval range from the server. Then the server assigns the new sampling interval range and the sampling interval to the requesting sensor according to the channel bandwidth and the priority, and the sensor uses them as a new sampling interval and sampling interval range.
Figure 5.5: Ozone on March 12

Figure 5.6: CO on March 12
Chapter 6

Experiments and evaluation

6.1 Simulation with Matlab

Here we present the simulation results of our adaptive sampling method with Matlab. First, we used the NO$_2$ data hourly measured on March 13th, 2014 in Essingeleden from the “SLB analys” (See Figure(6.1)). Next, based on this

![Hourly measured data](image)

**Figure 6.1**: NO$_2$ data measured on March 13th, 2014 in Essingeleden
measurement data we produced minutely measured \( NO_2 \) data by interpolating certain values with Gaussian distributed noise (See Figure(6.2)).

With this minutely measured data, we compared the uniform sampling method and the adaptive sampling method by calculating the number of transmissions and mean error percentage. First, we simulated two uniform sampling methods under the condition that uniform sampling interval is 10 and 15 respectively. The result is shown in the Table 6.1. As you can see in Table 6.1, mean error is inversely proportional to the transmission numbers. Next we simulated the adaptive sampling method under the condition that maximum sampling interval is 15, \( b = 2 \) and \( c = 5 \) and the result is shown in Table 6.2.

In Table 6.2, requiring transmission is the number of transmissions which are used to request new sampling interval from the server. Thus among 105 transmissions only 101 transmissions represent real measurements. Since the maximum sampling interval is 15, adaptive sampling method measured more

Table 6.1: uniform sampling results

<table>
<thead>
<tr>
<th>sampling interval</th>
<th>mean error percentage</th>
<th>transmission numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>8.4</td>
<td>144</td>
</tr>
<tr>
<td>15</td>
<td>9.33</td>
<td>96</td>
</tr>
</tbody>
</table>

Figure 6.2: minutely measured \( NO_2 \) data
Table 6.2: adaptive sampling results

<table>
<thead>
<tr>
<th>$S_{max}$</th>
<th>mean error percentage</th>
<th>transmission numbers</th>
<th>requiring transmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>9</td>
<td>105</td>
<td>4</td>
</tr>
</tbody>
</table>

than 15-min uniform sampling method and the mean error is smaller which means it is more accurate. However, accuracy of the adaptive sampling is not better than 10-min uniform sampling since its number of measurements is far smaller than uniform method. Yet the transmission is reduced almost one third which means that the sensor saves energy a lot. In one word, there is a “trade-off” here. If good quality is required, you should spend much energy, and if energy conservation is important, you should take a risk that the sensing quality can be bad. Therefore this result can be considered reasonable and from this result, we can see that adaptive sampling can save a lot of energy while providing decent accuracy.

Here is a figure which shows the sampling intervals during measurements. As

![The relations between SI and advf](image)

Figure 6.3: Sampling Intervals and Average Differences

we can see in the figure (6.3), if average difference gets bigger, sampling interval becomes smaller. In other words, if there is a big change in air pollutants, the sensor measures the air pollutants more often. This result proves that our code works fine and our method can control the sampling rate according to the input
characteristic.

Then we change the parameters “b” and “c” to see how the parameters affect the adaptive sampling method. The results can be shown in the Table 6.3 and Table 6.4. The parameter “b” is a width of each sampling interval range so if “b” is small, new sampling interval can be easily out of the range and require new SIR from the server often. Thus when b = 1, the transmission number is bigger than when b = 2 and same thing happens for other values of “b”. You can notice that although “b” is big, transmission numbers are not that much big. The reason is that b = 2 is already big enough that new sampling intervals are likely to be within current SIR.

The parameter “c” represents the sensitivity of the adaptive sampling method so the bigger the “c”, the more sensitive to input change the sensor is. Then the new sampling interval can be easily changed by small change in measurement and number of transmissions can be larger. This is proved in the Table 6.4. As “c” grows bigger, transmission numbers also rise and especially requiring transmissions increases. Figure (6.4) and Figure (6.5) can illustrate why this happens. Figure (6.4) is when c = 1 and Figure (6.5) is when c = 10. We can find that when c = 10, new sampling intervals are affected a lot by average difference of measurements.

Table 6.3: adaptive sampling results with different b

<table>
<thead>
<tr>
<th>b</th>
<th>c</th>
<th>mean error percentage</th>
<th>transmission numbers</th>
<th>requiring transmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>8.8</td>
<td>127</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>9</td>
<td>105</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>9.09</td>
<td>105</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>8.95</td>
<td>103</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8.9</td>
<td>103</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6.4: adaptive sampling results with different c

<table>
<thead>
<tr>
<th>b</th>
<th>c</th>
<th>mean error percentage</th>
<th>transmission numbers</th>
<th>requiring transmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>9.31</td>
<td>98</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>9.1</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>8.94</td>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>9.05</td>
<td>104</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>9</td>
<td>105</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>9.15</td>
<td>114</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>8.78</td>
<td>121</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>9.81</td>
<td>126</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9.23</td>
<td>128</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8.92</td>
<td>129</td>
<td>22</td>
</tr>
</tbody>
</table>
6.2 Implementation with Upwis sensors

I implemented the adaptive sampling method to a Upwis U115 sensor node and here the result is described below.

U115 sensor node is described in Section 2-2 in detail. It measures the air pollutants as well as many things such as humidity, acceleration, temperature and light. After measuring these, U115 sends the measurement data to the IBM cloud server via 6LoWPAN network router by using MQTT messaging protocol. In this test, I introduced my adaptive sampling method to the light sensor because light is easier to control than air pollutants which are normally poisonous and harmful to health.

The light sensor has a timer and whenever the timer expires, it senses the light and sends it to the server. Therefore, I implemented my method by setting the timer to the result of my adaptive sampling method function which is a new sampling interval. In order to adjust the sampling interval based on the input characteristic, I stored the previous measurement values in an array so that adaptive sampling function can calculate the new sampling interval.

After uploading the new firmware to U115 sensor node, I tested my method by changing the light around the sensor. Simply I put U115 node near the window where there is enough sunshine and after a short time, moved it to the shadow.
Finally I hold the sensor in my hand in order to block the light. These processes repeated several times during test.

As mentioned above, the sensor senses the light and sends the measurement to the server via 6LoWPAN network router by using MQTT protocol. Then I receive the data from the server on the terminal window and the data includes upload time and the degrees of light in Lux. Therefore, by comparing the upload time and the degrees of light, I can check whether my method is working correctly or not.

The results shown in Table 6.5 demonstrate that our adaptive sampling method is working correctly (These measurements are only small part of the whole data). Here I initialized the light as 0 and assigned the maximum sampling interval to 60 seconds. In order to simplify the analysis, I set the sliding window size to 1 which means I calculate the difference between only current and the very previous measurements. In addition, since the embedded systems cannot deal with floating numbers, floating digits are rounded. As can be seen in Table 6.5, at starting point, light is 910lx and uploading time is 23s. Second measurement says light is same and the measurement was uploaded 45 seconds after first measurement. Originally second measurement should be uploaded 60 seconds after the first one but because of the difference of light, sampling interval of the sensor became 15 seconds smaller. We can also find that the method is working
good by analyzing the third measurement. Since there was no change in degrees of light between second and third measurement points, the third measurement was done after 60 seconds. We can be sure that the adaptive sampling method successfully adjusts the sampling rate based on the input characteristic with the rest of the data.

Table 6.5: Results with light sensor of U115

<table>
<thead>
<tr>
<th>No</th>
<th>Light(Lux)</th>
<th>Upload Time(second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>910</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>910</td>
<td>68</td>
</tr>
<tr>
<td>4</td>
<td>911</td>
<td>128</td>
</tr>
<tr>
<td>5</td>
<td>634</td>
<td>188</td>
</tr>
<tr>
<td>6</td>
<td>634</td>
<td>244</td>
</tr>
<tr>
<td>7</td>
<td>627</td>
<td>304</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>364</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>414</td>
</tr>
</tbody>
</table>
Chapter 7

Discussion, future work and conclusion

7.1 Discussion and future work

Now we are going to discuss the simulation results and the real sensor test results. According to the results in Chapter 6, we can find that adaptive sampling method can save the energy as well as provide decent sensing quality. One of interesting things of this adaptive sampling method is that it has user parameter “b” and “c” which let us adjust the method itself to the target gas. If the target air pollutant has big changes, it is better to have a big “b” and set “c” small. On the other hand, if gas has a small changes, we should increase the “c” to make the method sensitive to the change.

This method also has some drawbacks we should mention. First, with the tables about simulation, we can find that although the transmission numbers are big, mean error percentages are also large or at least not much small. We found two reasons for these bizarre results. First, we made a dataset by interpolating some values with gaussian noise so the results are different every time. Another reason is that if the measurements are normally done at peak points, the error can be small but if not, there can be a large difference in spite of many measurements. Shortly, the mean error is easily influenced by the characteristic of the measurement dataset.

Another issue is that this method determines the next sampling period by considering present and several previous measurements so it cannot respond to the unexpected events quickly. Thus unexpected events which happen shorter than the sampling interval can be missed. In order to avoid such cases, we should have an appropriate maximum sampling interval. Fortunately, speaking of air pollutants, it is hard to find such short-period happening but we should consider this matter. Another method to overcome this shortcoming can be to predict the next measurement value to calculate the new sampling interval. However, no matter what it is, it is impossible to predict unexpected events such as ex-
plosion or fire correctly so this has also some limitations.

One drawback of this method can be that if the sensor includes several local sensors (which has several sensors such as a humidity, temperature, gas, light, etc.), it cannot be used. Such sensors send the data collected from all local sensors to the server at once so if I use this method to gas sensors, other local sensors like light and humidity can do unnecessary monitoring which causes energy waste.

Finally, we found that if there are too many big fluctuations in the air pollutants, adaptive sampling can use more energy than uniform sampling because much energy should be used for transmissions of requiring new sampling interval range. In reality, it is hard to find too many peaks in air pollutants but if we set the parameter $b$ too small or set the parameter $c$ too big, then new sampling interval can be easily out of the SIR and this kind of problem can happen.

As we discussed above, this adaptive sampling method is not complete and still has many things to improve. One is the air pollutants modeling for Kalman filter. Since air modeling determines the performance of the Kalman filter, it is very important to improve this. There has been many studies to model the air pollutants and all of them are not perfect because of the nonlinearity and random property of the air pollutants. In this paper, AR model is used but it does not perfectly represent the characteristic of the air pollutants yet.

Another problem is that the code for server part. In this paper, we simulated the server part, not ran the process for new sampling interval range request in the server side.

\section{Conclusion}

In wireless sensor network, energy conservation is very important and there has been many studies to solve this problem. However, some of them save energy while lowering the sensing quality and some have restrictions on area or time. In addition, most of them are not for wireless sensor network for air monitoring. In this thesis, we study the adaptive sampling method to save energy of the sensors as well as maintain the good sensing quality especially for air monitoring wireless sensor network. We found that there are certain pattern and similarity in the everyday air measurements which let us reduce the transmission which is the big part of energy consumption in sensors. We decided that adaptive sampling method is appropriate to achieve our goal and developed the method to adapt the sampling interval of sensors based on the input characteristic which is the gradient of change of air pollutants.

We simulated the method with Matlab with minutely measured dataset derived from the real hourly measured data. The outcome proves that adaptive sampling method decreases the number of transmissions significantly while providing quite fine quality. Finally we tested our method with real sensors used in GreenIoT project and demonstrated the effectiveness and flexibility of the method.
Appendix A

Matlab code

```matlab
% Initializations for Kalman filter
P_k_prev = 1;
x_kprev_hat = 0;
x_k_hat = [];

% Time update equations
A = 0.9991;  % AR(1) model parameter
W = 1;

% Measurement update equations
H = 1;
V = 1;

% Measurement model --- hourly model
z_k = [19.04; 17.76; 18.26; 18.12; 21.39; 47.76; 84.48; 81.43; 71.39; 64.17; 65.77; 67.77; 70.25; 73.08; 59.52; 53.54; 48.51; 57.99; 35.27; 27.01; 20.33; 14.96; 11.38; 9.379];

% Interpolation --- per-minute model = new_z_k
rate = 60;
i = 1 : 1 : length(z_k);
j = z_k(i);
k = 0 : 1/rate : length(z_k);
new_z_k = interp1(i,j,k,"spline") + normrnd(0,3,[1,rate*length(z_k)+1]);

% Measurement noise
R = 0.0001;  % R = 0.000001 for a very low value
% Process noise covariance
Q = 1e-5;

% past value's weight  (x+y+z = 1, x>y>z)
x = 0.6;
y = 0.25;
z = 0.15;

% si_new = si_current - (err*c - a*b)
a = 0;
```
b = 2;
c = 10;

% uniform sampling interval
rate = 15;

% sampling interval
si_ini = 15; % initial sampling interval
si_max = si_ini + b/2;
si_min = 1;
si_new(1) = si_ini;

% AR method window
w = zeros(1,3);
v = zeros(1,2);

% variables initialization
h = 2;
k = 1;
trans_adap = 0; % number of transmissions in adaptive sampling
trans_req = 0; % number of transmissions requiring new SIR
adap_samp = []; % measurements in adaptive sampling
interp_adap_samp = []; % interpolated adaptive sampling measurements

% adaptive sampling interval
while k < length(new_z_k)
adap_samp = cat(2, adap_samp, new_z_k(k));
% interpolation for adaptive sampling
if k < 3 % first 2 measurements go directly into interp_adap_samp
interp_adap_samp = cat(2, interp_adap_samp, adap_samp(k));
else
j = length(adap_samp); % number of elements in adap_samp
dif = (adap_samp(j) - adap_samp(j-1)) / add_time;
for i = 1 : add_time
added = adap_samp(j-1) + i*dif;
interp_adap_samp = cat(2, interp_adap_samp, added);
end
end

% Extended Kalman Filter
% Time update equations
x_k_hat_minus = A*x_kprev_hat; % a priori estimate
P_k_minus = A*P_k prev + Q; % a priori estimate error covariance

% Measurement update equations
Kk = P_k_minus/(P_k_minus + R); % Kalman gain
x_kprev_hat = x_k_hat_minus + Kk*(new_z_k(k) - x_k_hat_minus); % a priori estimate
x_k_hat = cat(2, x_k_hat, x_kprev_hat); % a posteriori estimate
P_k prev = (1 - Kk)*P_k_minus; % a posteriori estimate error covariance
P(k) = P_k prev;

% Sampling interval adaptation
if k == 1
k = k + 1;
add_time = 1;
elseif k > 1
% difference between two consecutive measurements
Err(h) = abs(x_k^\hat{(h)} - x_k^\hat{(h-1)});
% difference per minute
Err_min(h) = Err(h) / si_new(h-1);
% updating AR window
w(3) = w(2);
w(2) = w(1);
w(1) = Err_min(h);
% past-considered error
Av_err = w(1) * x + w(2) * y + w(3) * z;
% new sampling interval
si_new(h) = si_max - a*b - b/2 - (Av_err * c - a*b);
% assigning a new sampling internal range
if (si_new(h) < (si_max - (a+1)*b)) && (si_new(h) > 0)
    a = floor((si_max - si_new(h)) / b);
    si_new(h) = si_max - a*b - b/2;
    trans_adap = trans_adap + 1;
    trans_req = trans_req + 1;
elseif (si_new(h) > (si_max - a*b)) && (si_new(h) > 0)
    a = floor((si_max - si_new(h)) / b);
    si_new(h) = si_max - a*b - b/2;
    trans_adap = trans_adap + 1;
    trans_req = trans_req + 1;
elseif si_new(h) < 0
    si_new(h) = si_min;
end
add_time = round(si_new(h)); % time difference between two consecutive measurements
k = k + add_time;
h = h + 1;
trans_adap = trans_adap + 1;
end
end

% ------------------- uniform sampling interval every 10 min -------------------
k = 1;
uni_samp = []; % measurements in uniform sampling
trans_uni = 0; % number of transmissions in uniform sampling
while k < length(new_z_k)
    uni_samp = cat(2, uni_samp, new_z_k(k));
k = k + rate;
trans_uni = trans_uni + 1;
end

% ------- interpolation for uniform sampling measurements -------
interp_uni_samp = []; % interpolated uniform sampling measurements
for k = 1 : (length(uni_samp) - 1)
dif = (uni_samp(k+1) - uni_samp(k)) / rate;
for i = 0 : (rate - 1)
    added = uni_samp(k) + i*dif;
    interp_uni_samp = cat(2, interp_uni_samp, added);
end
end
% ------- calculating difference between real, uniform and adaptive ---
sum_dif_per_uni = 0;%sum of all percentage difference in uniform sampling
sum_dif_per_adap = 0;
sum_dif Uni = 0;%sum of all difference in uniform sampling
sum_dif adap = 0;
if length(interp_adap samp) > length(interp_uni samp)
i_max = length(interp_uni samp);
else
i_max = length(interp_adap samp);
end
for i=1:i_max
dif Uni = abs(new_z_k(i) - interp_uni samp(i));
sum_dif Uni = sum_dif Uni + dif Uni;
dif adap = abs(new_z_k(i) - interp_adap samp(i));
sum_dif adap = sum_dif adap + dif adap;
dif percent Uni = abs(new_z_k(i) - interp_uni samp(i))/new_z_k(i);
sum_dif percent Uni = sum_dif percent Uni + dif percent Uni;
dif percent adap = abs(new_z_k(i) - interp_adap samp(i))/new_z_k(i);
sum_dif percent adap = sum_dif percent adap + dif percent adap;
end
tav_dif_per_uni = sum_dif_per_uni / i_max *100;
tav_dif_per_adap = sum_dif_per_adap / i_max *100;
av_dif Uni = sum_dif Uni / i_max;
av_dif adap = sum_dif adap / i_max;

%------Plotting-------------------------------
figure;
plot(z_k);
figure;
plot(err_min, 'rx-');
hold on;
plot(si_new, 'gx-');

%------Yule-Walker method-------------------------------
function [a,sig2]=yulewalker(y,n)
   % [a,sig2]=yulewalker(y,n);
   % y -> the data vector
   % n -> AR model order
   % a <- the AR coefficient vector estimate
   % sig2 <- the white noise variance estimate
   y=y(:);
   N=length(y); % data length
   if (N < n)
     disp('Error: the AR model order is greater than the data length.');
     return
   end
   % compute the standard biased ACS estimate {\(r(0)\ r(1)\ r(2)\ ...\ r(n)\})
r=zeros(n+1,1);
   for i = 0 : n,
r(i+1) = y(1:N-i)' * y(i+1:N) / N;
end

% form the Toeplitz covariance matrix
Rn = toeplitz(conj(r(1:n)));

% compute the AR coefficients
a = -Rn \ r(2:n+1);

% compute the noise variance
sig2 = real(r(1)+a.' * conj(r(2:n+1)));

a = [1; a];
Bibliography


