# A Real Time Multivariate Robust Regression Based Flood Prediction Model Using Polynomial Approximation for Wireless Sensor Network Based Flood Forecasting Systems

Victor Seal[1], Arnab Raha[1], Shovan Maity[1], Souvik Kr. Mitra[1], Amitava Mukherjee[2], and Mrinal Kanti Naskar[1]

[1] Advanced Digital and Embedded Systems Laboratory, Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700032, India
[2] IBM India Private limited, Kolkata 700091, India
{arnabraha1989,shovanju35,souvikmitra.ju}@gmail.com,
{victor.seal,mrinalnaskar}@yahoo.co.in.
amitava.mukherjee@in.ibm.com,

**Abstract.** The paper introduces a statistical model to be used in wireless sensor network (WSN) for forecasting floods in rivers using simple and uncomplicated calculations and provide a reliable and timely warning to the people who may be affected. The statistical process used for this real time prediction uses linear robust multiple variable regression method to provide simplicity in cost and feature, and yet efficiency is speed, power consumption and prediction accuracy which is the prime goal of any design algorithm. This model is theoretically independent of the number of parameters, which may be varied according to practical needs. When increasing, the water level trend is approximated using a polynomial and its nature is used to predict when the water level may cross the flood line in future. We have simulated the comparison of predicted water level with the actual level in a time interval, around and below the flood line. The accuracy of prediction above flood line is of no value in real life and but a data above flood line is shown in our simulation results for the sake of continuity and logical justification of the algorithm.

**Keywords:** Flood forecasting, robust regression, WSN, polynomial fitting, multi-square weight minimization, event, query.

## 1 Introduction

Natural disasters like floods, hurricanes, etc. tremendously affect the lives of the poor and under privileged. A lot of work has been done to develop systems which help to minimize the damage through early disaster predictions. As the network has to be deployed in the rural areas, there is a severe limitation of resources.

Nowadays, WSN based systems are usually used in prediction models. WSNs are low power, low cost, multi-hopping systems, and independent of external service providers that can form an extendable network without line of sight coverage; but have self-healing data paths. WSNs can be deployed more or less homogeneously in a geographical region using a two-tiered approach having clusters of short distance communicating nodes together with some nodes capable of communication over a wider range. WSN nodes communicate only with neighbouring nodes to reduce the transmission power and losses, thus eliminating the need for expensive repeaters and transmitters used in traditional telemetry systems. Every node in a WSN acts as a data acquisition device, a data router and a data aggregator. This architecture maximizes the redundancy and consequently the reliability of the entire flash- flood monitoring system. The independence from 3rd party providers and the absence of infrastructure requirements – as those needed in cellular based telemetry systems allow a WSN to be deployed quickly. They allow online, self-calibration of the prediction model.

Two types of model may be created: First, a centralized model where computation occurs at the central node only. It needs simpler components as terminal nodes don't need any hardware for computational purposes. Second is a distributed model with computations at several levels instead of only one computing node as in the previous model. This model is more immune to errors but cost ineffective. Combining these two to get a get an optimal balance of cost and accuracy is the best idea for a practical system.

## 2 Related Work

In most of the data-driven statistical flood-prediction models, details of the topography, soil composition, and land cover, along with meteorological conditions and hydro-meteorological quantities such as soil moisture are required [3]. In the development of these rainfall-runoff models, on-going work covers a range of models from lumped to spatially distributed variations [5,6]. Most of the algorithms which come under the same category as our subject requires a set of tests for calibration and require repeated re-calibrations [1,7].Some algorithms even restrict the number of parameters to a fixed quantity thereby making their model a rigid one.

## 3 Proposed Work

**WSN System Architecture**

Our proposed model comprises sensors (which sense and collect the data relevant for calculations), some nodes referred to as computational nodes (that have large processing powers and implement our proposed prediction algorithm) and a manned central monitoring office (which verifies the results with the available online information, implements a centralized version of the prediction algorithm as a redundancy mechanism, issues alerts and initiates evacuation procedures). Different types of sensors are required to sense water discharge from dam, rainfall, humidity, temperature, etc. The computational nodes possess powerful CPUs required to implement the algorithm. Computational nodes transmit the prediction results to the monitoring node. Inter-node communication eliminates errors. The work of the central node is not our concern. However it is important to include a manned central node in this whole process to raise the alarm and co-ordinate evacuation measures if needed.

The next important aspect is to minimize the effect of a node failure while connecting the computational nodes to the central. Intermediate nodes (INs) have to be deployed to ensure this connectivity in case the central does not fall within the communication range of all the nodes. Given below is the entire picture of the WSN at site is given. As we can see the river has been broken into several monitoring zones. In each zone, a sensor node collects data and sends them to its computational node. Data collection and localized prediction takes place at each computational node. The results are then shared between all the computational nodes themselves and also to the central.

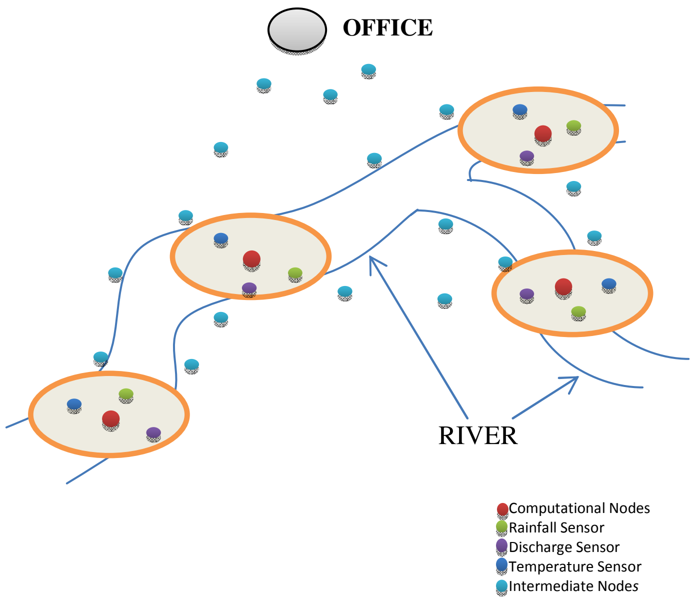

**Fig. 1.** Wireless Sensor Nodes Deployment Scheme

Our prediction algorithm is compatible with a 2 layered network architecture comprising a number of sensor nodes collecting the data needed for prediction. Data is sent to the single computation node (or administrative office) which bears the responsibility of most of the computations and predictions (online and offline). It has great computational power and processing capability. With data collected online dynamically, forecasts are made using our suggested algorithm and results communicated to the local administrative office which raises alarm if needed.
Sensors take data as decided by one or more of these three parameters:

*a)Time*- The administrator decides a time interval after which a reading is to be taken, accordingly a sensor takes the time driven data as input.

*b )Event*-Event refers to sudden change in a parameter affecting the desired output. Suppose the next time driven data is slated to be taken after an hour. But a dam starts discharging water at such a rate that flood may occur within the next 30 minutes. A hardware interrupt has to override the time driven mechanism to get an instant event driven data and make a reliable prediction before the flood occurs.

*c)Query*-Suppose an administrator wants to see the immediate level and get an instantaneous prediction even when it's not time to take a reading. A hardware interrupt is used get the instant query driven data. It is quite similar to the event driven case, the only difference being that this interrupt is provided by a human unlike natural parameters that trigger the event triggering interrupt.

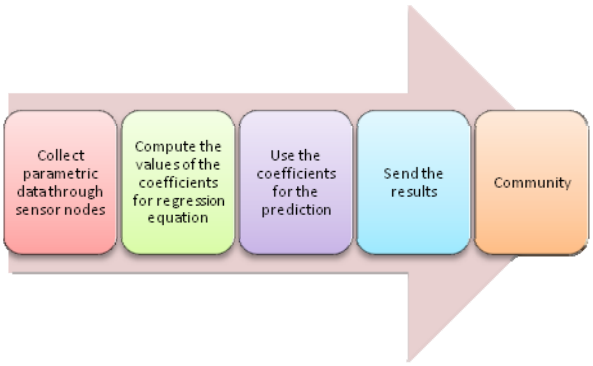
Collect parametric datathrough sensor nodes
Compute the values of the coefficients for regression equation
Use the coefficients for the prediction
Send the results
Community

**Fig. 2.** Block-wise depiction of data sent from sensors to the community

Our algorithm handles all these issues. It operates using a time driven mechanism and checks for hardware interrupt at every instant of wait interval (time before next reading) and forfeits its wait state if there is an interrupt.

The advantage of such a method is that it can incorporate a wide number of parameters upon which the river water-level varies providing us with a more precise prediction and reducing the possibility of false alarms. For simplicity, we, in prior, choose a threshold level for river water such that if the predicted value is more than that level an alarm can be raised. Choosing such a threshold level is very critical and important in our case as it may lead to unnecessary false alarms.

Suppose that the water level is close to the flood line but remains constant, just below it for a long time .instead of sounding a continuous alarm, we suggest reducing the sampling interval then so that the system sensitivity increases and it sounds an alarm only if needed. Even the intensity of flood may be determined based on the predicted value. To make the prediction faster and more accurate, we also need to logically decide the necessity of a parameter in predicting the output.

## Our Floodprediction Model

We provide the algorithm of the main program and follow it up with the part wise algorithms of the different functions used in it.

**Program Algorithm**

**Input:** Past data tables: Y=water level, R= rainfall, D=Discharge, $X_i$ ,i=1,2…= other parameters affecting water level; Present value of parameters. SC=Storage capacity (in number of readings) of node. S=stored regression coefficient algorithm, W=its weight. P=present coefficient matrix.

Initialise: Set counter=1, weight=0, initialise all matrices to zeroes.

**Output:** Predicted value of Y.

**Step 1.** Read past data tables: Y=water level, R= rainfall, D=Discharge, $X_i$, i=1,2…=other parameters affecting water level.

**Step 2.** Compute/re-compute linear regression model (i.e. coefficient matrix P) relating Y with the above parameters using the robust fit function.

**Step 3.** Find the weighted mean of P and S (weight of P=counter value now, weight of S=W).This is our final regression model.

**Step 4.** Input the present value of parameters and predict corresponding value of Y based on the final regression model.

**Step 5.** Decide time interval before next reading using the time multiplier function.

**Step 6.** Maintain a table of water levels v/s time and see if the final trend(most recent reading to back in time) is increasing or not.

**Step 7.** If water level trend is not increasing, flood is not expected. Go to step 10.

If increasing, use Polynomial Fit function to approximate the trend v/s time (from present value to the 1st local minima reached going back in time) and generate a prediction model for it.

**Step 8.** Put in values of time in future in progressive order into the prediction model above and predict future values of water level.

**Step 9.** If water level stays below flood line up to a defined reliability period of our prediction model, predict flood not possible.

If water level exceeds the flood line within the defined reliability period, predict flood possibility.

**Step 10.** Recalibrate time interval of step 3 based on this prediction.

**Step 11.** Input present value of water level. Increment counter by 1.

**Step 12:**If counter value is less than SC, append all present values to the end of their respective tables in step1(Y, D, R, $X_i$) . Then go to step 2.

Else, store the coefficient matrix and increment its weight (W) by SC. Then clear all the data and output tables, reset counter to one to ready it for taking inputs.

**Step 13:** Enter a wait state; waiting time= recalibrated time interval (step 10)

**Step 14:** Check-is the system still in wait state?

If yes, go to step 15.

Else go to step 2.

**Step 15:** Check- is there any hardware interrupt (event/query driven data acquisition demand) now?

If yes, go to step 2 immediately.

Else go to step 14.

**Polynomial Fit function:** We approximate the increasing water level by a second order polynomial function to predict future values i.e. $y=a_1x^2+a_2x+a_3$. Where,

x is a time index starting from the moment water level starts rising;

y=corresponding water level at that time.

Tabulating for several values $(x_i,y_i)$, the solution is given in form:
$Y_{nx1}=X_{nx3}A_{3x1}$ (n=no. of readings) where X is the Vandermonde matrix whose elements are given by
$X_{i,j}=x_i^{3-j}$ (i=1,2,3;j=1,2,3) i.e. simply stated, the elements of each row I are $x_i^2,x_i$ ,1 respectively.
In Y=XA we know Y,X; solving for A gives the coefficient matrix $[a_1\ a_2\ a_3\ ]'$ above.
To solve,
$X^TY=X^TXA$
Or, $(X^TX)^{-1}\ X^TY=A$.
Now we have the 2nd order polynomial equation $y=a_1x^2+a_2x+a_3$ ready to use.
Going by the present nature of the curve, we predict its future trend and see if it goes over the flood line within the reliability period of our future prediction and predict accordingly.
The 2nd order polynomial function ensures simplicity of calculation over larger polynomial.

**Robust Fit function:** The main disadvantage of ordinary least-squares fitting is its sensitivity to outliers. Squaring the residuals magnifies the effects of these extreme data points and hence they have a large influence on the fit. To minimize the influence of outliers, we fit our data using robust least-squares regression.

Robust fitting with multi-square weights (explained later) uses an iteratively reweighted least-squares algorithm, and follows this procedure:
1. First, we set the model by weighted least squares using multi-square weight minimization.
2. Then, we calculate the residuals (after adjusting them) and give them a standard value. These residuals are given by
$r_{adjusted}=r_i/(sqrt(i-h_i)$
$r_i$ are the usual least-squares residuals and $h_i$ are *leverages* that adjust the residuals by down-weighting high-leverage data points, which have a large effect on the least-squares fit. The standardized adjusted residuals(r) are given by

$$r=r_{adjusted}/K$$

*K* is a tuning constant, K=4.685, *s*=robust variance given by *MaD*/0.6745 where *MaD*=median absolute deviation of the residuals. Mathematically,
$MaD=median_i\ (|X_i-median_j(X_j)|)$
1. We find the robust weights as a function of *r*. The multi-square weights are given by

$$w_i= \begin{cases} (1-(r_i)^2)^2 & \text{if } |r_i|<1 \\ 0 & \text{if } |r_i|>=1 \end{cases}$$

2. If the curve converges within a defined tolerable range, our work is done. Otherwise, we perform the next iteration of the fitting procedure by returning to the first step.

**Multi-square weight minimization:** By extension of definition of a bi-squared number, a multi-square number is the sum of squares of two or more numbers. This method is used to minimize a weighted sum of squares. The weight given to each data point depends on how far it is from the fitted line.

$y=a_1+a_2x_1+a_3x_2+\ldots+a_{m+1}x_m$ is a general form of m variable regression. For n sets of values {y(i) ,$x_1$(i) ,$x_2$(i) ,…,$x_m$(i) }, i=1,2,3…n; we have n equations:
$y(i)=a_1+a_2x_1(i)+a_3x_2(i)+\ldots a_{n-1}x_n(i)$,
which in matrix form is $Y_{nx1}=X_{nxm}A_{mx1}$ where X matrix has 1st column of1s,2nd column of $x_1$ values (i.e. 1st row of 1st of n readings,…ith row=ith reading of $x_1$),3rd column of $x_2$ ,ith column of $x_{i-1}$ and the last column (m+1th column) of $x_m$ values.

The central point of multi-square weight minimization is, unlike simple least square minimization, we do not give same value/weight to the equation $y=a_1+a_2x_1+a_3x_2+\ldots+a_{m+1}x_m$ for every value of i.Let $m_y$ denote mean of $y_i$ .The weight assigned to the ith equation is determined by the value of $(y_i-m_y)^2$. We have used the Andrews function for our procedure but any of the weight assigning functions as suggested in figure2 of [1]may be tried out. For region specific application, the user may try out different functions to decide upon the best predictor for a particular set of data.

In matrix form,
WY=WXA. Where the weight matrix must be nxn, with elements $w_{i,i}$=weight assigned to $y_i$ ;all elements except lead diagonal elements=0.
We have to solve for the coefficient matrix A;
we know W,Y and X.
Solving for A:
$X^TWY=X^TWXA$
Or, $(X^TWX)^{-1}\ X^TWY=A$.
This coefficient matrix A is used in the next robust approximation.

This method is alterable based on available resources and memory space. We can easily limit the number of iterations to enhance system simplicity by just adding a counter to the algorithm and checking it at the end of every fit. As approximation system improves, the system needed to support it becomes more complicated. For high end systems, non-linear regression models may even be used. For simple sensors, just 2 or 3 iterations of the above algorithm is sufficient to provide a reasonably good approximation over simple least square fit taking the system simplicity into account.

**Time Multiplier function:** This predicts the time interval in minutes between two successive readings. This is a worst case predictor that tells how soon, or how late the next reading must be taken to predict flood successfully based on two parameters.

a) Difference in water level between subsequent readings-Say the water level has increased by an amount equal to 90% of flood line between two subsequent readings. Our predictor assumes the increase is not uniform over the interval, but may have happened in the last few moments. Thus it asks for another reading in the minimum possible time to see the gradient and make a better prediction. (Note: Flood is not predicted just if the parameter value is high, we only ask for another reading faster for accurate prediction.)
b) Present water level-Say the water level has reached 90% of flood line. A little rain/discharge any moment can trigger a flood. So it is best to take readings as soon as possible and make accurate predictions.

Continuing the thought process, we assign different time intervals for subsequent readings as logical from the measured/calculated values of these parameters.

We define a *time set* containing different time intervals at which the sensor nodes can take readings. Based on the above discussion, the time multiplier function decides which value of the time set is to be taken for a particular calculation.

This function decides the time interval based on a worst case scenario but it doesn't affect prediction of flood. A worst case flood predictor would generate numerous false alarms, hence is undesirable.

**Recalibration of time interval:** Suppose the next reading is slated to be after 2 hours. But our algorithm predicts a flood within an hour from now. The time interval must be changed to be at least less than the time at which flood is predicted. *If a flood is predicted before the minimum time interval in the time set, an alarm must be issued.*

**Consideration for node storage capacity:** Our algorithm stores the regression coefficient values and frees the system memory by removing parameter values whenever there is a constraint on system memory. Effectively, nothing is lost as the single coefficient matrix is the result desirable from such enormous amounts of parameter data. A new coefficient matrix is constructed using the new inputs. A resultant coefficient matrix calculated using the weighted mean of these two, each matrix assigned a weight in proportion to the number of readings used in it.

**Improvements over [1]:**

*i) Time adjustment factor*- [1] does not talk about forecast time interval for next reading/flood prediction. But we have introduced a logical time predictor to decide the next time for taking a reading. Say flood may occur after a week at the present rate of rainfall. It is impractical to ask for data every 5 minutes now. Our algorithm handles these issues.

*ii)Improvement of flood dependency model on its parameters*- [1] suggests to incorporate prediction error values as a parameter along with rainfall, etc. (other parameters on which flood level depends) to recalibrate the regression based model and predict a flood level with better accuracy. Treating error as a parameter affecting water level is a poor technique. Also, the parameters will get considerably worsened in presence of any outliers. We append the present input values to our data tables to do the same recalibration using robust techniques quite insensitive to outliers but have nearly same simplicity.

*iii) Use of weighted/Robust linear model*- [1] suggests smoothing the data using a low pass filter. Arguably, a much better and formal technique is to assign weights to each data and compute accordingly. Its compatibility with low end systems has already been discussed.

*iv) Consideration for node storage capacity*- As discussed, our algorithm effectively uses all the past data though the actual data is deleted when it exceeds system memory capacity. [1] does not talk about system memory constraints.

*v) Independency w.r.t number of parameters*- Theoretically, our algorithm can handle any number of parameters and data elements, the only restriction being memory and/or available resources.

## 4 Simulation Results

Our simulation was conducted in MATLAB. Predicted water-levels were extremely close to actual measurements. Only once the river water level crossed threshold signalling the onset of flood as in Fig. 3. False alarms were not generated in any predicted value proving the reliability of our scheme. We simulated the regression model for water level with reference to two parameters-rainfall and discharge from dam. Fig. 4 denotes the percentage errors in predicted water level at different time instants compared to the actual water level measured at those instants.

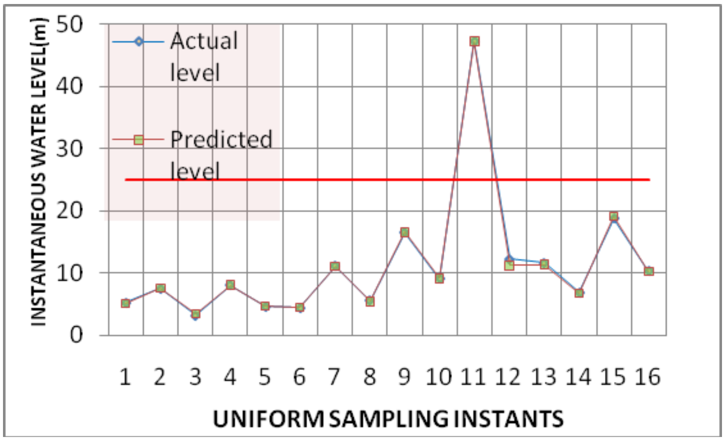

**Fig. 3.** Plot showing the measured and predicted values of instantaneous water levels at different instants

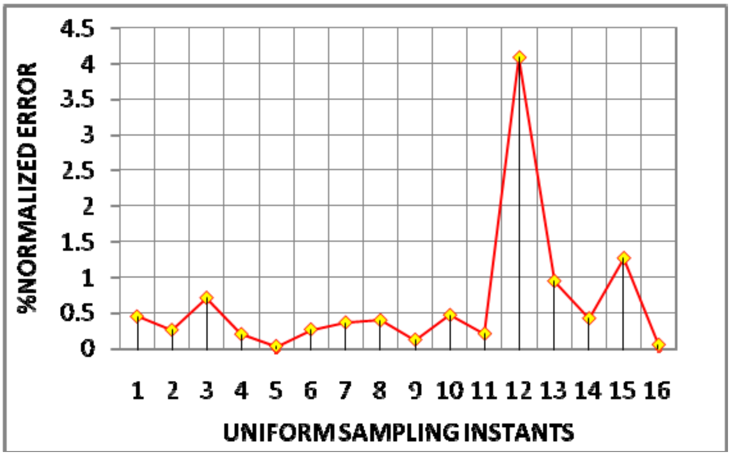

**Fig. 4.** Plot of Percentage Error in Predicted Instantaneous Water Level w.r.t. Actual Measured Water Level at different time instants (% error normalized with respect to the flood line)

## 5 Conclusion and Future Work

Our algorithm shows very accurate prediction results in forecasting critical parameters for flood prediction when the data collected are accurate. Even if some data entered may be wrong (due to sensor failure at that instant or loss of data in transmission, etc), the robust fit procedure (or a few iterations of multi-square weighted minimization for very simple nodes) is enough to provide a reasonably accurate data by assigning algorithm outlines a novel way to include the results from those data in calculations though the actual data is deleted. The advantages of our algorithm over present models are evident from the comparison with [1] and simulation results. Although real hardware implementation is going to be the next major step in developing this algorithm further, we can also improve upon accurate time prediction by manipulating the prediction algorithm to suit specific rivers. Future work involves performing field tests to observe the communication process between the nodes and the real-time implementation of the distributed prediction algorithm in situ.

## References

1. Basha, E.A., Ravela, S., Rus, D.: Model-Based Monitoring for Early Warning Flood Detection. In: SenSys 2008, Raleigh, North Carolina, USA, November 5-7 (2008)
2. Lawson, C., Keats, J.B., Montgomery, D.C.: Comparison of Robust and Least-Squares Regression in Computer-Generated Probability Plots. IEEE Transactions on Reliability 46(1) (March 1997)
3. Ivanov, Y., Vivoni, E.R., Bras, R.L., Entekhabi, D.: Preserving high-resolution surface and rainfall data in operational-scale basin hydrology: a fully-distributed physically-based approach. Journal of Hydrology 298, 80–111 (2004)
4. Reed, S., Koren, V., Smith, M., Zhang, Z., Moreda, F., Seo, D.-J., Participants, D.: Overall distributed model intercomparison project results. Journal of Hydrology 298, 27–60 (2004)
5. Smith, M.B., Seo, D.-J., Koren, V.I., Reed, S.M., Zhang, Z., Duan, Q., Moreda, F., Cong, S.: The distributed model intercomparison project (DMIP): motivation and experiment design. Journal of Hydrology 298, 4–26 (2004)
6. Box, G.E.P., Jenkins, G.M.: Time series analysis: forecasting and control. Holden-Day, San Francisco (1976)
7. Andrews, D.F.: A Robust Method For Multiple Linear Regression. Technometrics 16, 125–127 (1974)
8. Ramsay, O.: A comparative study of several robust estimates of slope, intercept, and scale in linear regression. J. Amer. Statistical Assoc. 72, 608–615 (1977)
9. MATLAB 7.11.0, Help menu. Copyright 1984-2010. The Mathworks, Inc. (2010b)