LETTER

The Software Reliability Model Using Hybrid Model of Fractals and ARIMA

Yong CAO†, Student Member and Qingxin ZHU†, Nonmember

SUMMARY The software reliability is the ability of the software to perform its required function under stated conditions for a stated period of time. In this paper, a hybrid methodology that combines both ARIMA and fractal models is proposed to take advantage of unique strength of ARIMA and fractal in linear and nonlinear modeling. Based on the experiments performed on the software reliability data obtained from literatures, it is observed that our method is effective through comparison with other methods and a new idea for the research of the software failure mechanism is presented.

Key words: software reliability forecasting, fractal, ARIMA models, software failures

1. Introduction

Software reliability is going to become a highly visible and important field. Therefore, software reliability forecasting is a problem of increasing importance for many critical applications and failure analysis is an important part of the research of software reliability. An underlying assumption of these models is that software failures occur randomly in time. It was mainly treated as random and statistical problem.

Recently, along with the development of prediction theory, support vector machines and artificial neural networks have been applied in forecasting software reliability [1], [2]. Some forecasting techniques have been developed, each one with its particular advantages and disadvantages compared to other approaches. This motivates the study of hybrid model combining different techniques and their respective strengths. Different forecasting models can complement each other in capturing patterns of data sets, and both theoretical and empirical studies have concluded that a combination of forecast outperforms individual forecasting models [3], [4], [7]. Terui and Dijk [4] presented a linear and nonlinear time series model for forecasting the US monthly employment rate and production indices. Their results demonstrated that the combined forecasts outperformed the individual forecasts.

Time series data often contain both linear and nonlinear patterns. The ARIMA (autoregressive integrated moving average processes) models are the most general class of models for forecasting a time series which can be stationaryized by transformations such as differencing and logging. When modeling linear and stationary time series, the researcher frequently chooses ARIMA models because of their high performance and robustness. Fractal model has good performance in nonlinear patterns which brought good effect [5]. A hybrid ARIMA and fractal model is capable of exploiting their strengths respectively.

2. Hybrid Model in Prediction of Software Failure

2.1 Fractal Model

The term fractal, which means broken or irregular fragments. Fractals are mathematical or natural objects that are made of parts similar to the whole in certain ways. It belongs to geometrical category. Time series also follow the laws of fractal geometry. According to [5] self-similarity exists in time series and we may investigate the relationship between software failures and fractal. Cao and Zhu have applied it to forecasting software failures and provided software prediction model based on fractal. Please see [5] for a detailed exposition.

2.2 Run Test

The runs test can be used to decide if a data set is from a stationary random process. A run is defined as a series of positive values or negative values. The number of positive values or negative values is the length of the run. In a stationary random data set, the probability that the ith value Yi is larger or smaller than the mean value follows a binomial distribution, which forms the basis of the runs test. The first step in the runs test is to compute the sequential differences Yi − Yi−1. Positive values indicate the difference values greater than or Equal to 0 and negative values indicate the difference values less than 0. Let N1 = (the number of positive values) and N2 = (the number of negative values). N = N1 + N2 and UN represents the total number of runs. We adopt hypothesis test to decide if the time series is stationary.

Hypothesize: H0 : \{yt, t = 1, 2, . . . , N\} is stationary random series. When the significance level is 5% and N1 ≤ 15 and N2 ≤ 15, given the upper limit rv and the lower limit rl and if UN ≥ rv or UN ≤ rl, we reject H0. Otherwise H0 is accepted.

If N1 > 15 or N2 > 15, UN follows the normal distribution N(μ, σ2), where μ = \frac{2N1N2}{N} + 1 and σ = \frac{2N1N2(N2N1-1)}{N(N-1)}. Let Z = \frac{UN-μ}{σ} and N follows N(0, 1). At the 5% significance level, when |Z| ≤ 1.96 the initial hypothesis is accepted.
2.3 ARIMA Model

Introduced by Box and Jenkins, the ARIMA model has been one of the most popular approaches to forecasting. In an ARIMA model, the future value of a variable is supposed to be a linear combination of past values and past errors, expressed as follows:

\[ y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \]  

(1)

where \( y_t \) is the actual value and \( \varepsilon_t \) is the random error at time \( t \); \( \phi_i \) and \( \theta_j \) are the coefficients; \( p \) and \( q \) are integers that are often referred to as autoregressive and moving average polynomials, respectively.

The autoregressive part AR(\( p \)) models a time series as a linear function of \( p \) previous observations in order to predict the current one. The moving average part MA(\( q \)) determines the moving average of the series with a time window of size \( q \). Finally, the ARIMA process (\( p, d, q \)) is based on a series that has been differentiated \( d \) times, with \( p \) autoregressive terms and \( q \) moving average terms.

2.4 The Hybrid Model

The software failures can not easily be captured. Therefore, a hybrid strategy that has both linear and nonlinear modeling abilities is a good alternative for forecasting software failures. Both the ARIMA and the fractal models have different capabilities to capture data characteristics in linear and nonlinear domains, so the hybrid model proposed in this investigation is composed of the ARIMA component and the fractal component. Thus, the hybrid model can model linear and nonlinear patterns with improved overall forecasting performance. We compute the logarithms of the time series to make the curve of series smooth and stationary. The hybrid model \( (E_t) \) composed of linear and nonlinear components can be represented as follows:

\[ \ln(E_t) = \ln(A_t) + \ln(F_t) \]  

(2)

where \( A_t \) is the linear part and \( F_t \) is the nonlinear part of the hybrid model. Both \( A_t \) and \( F_t \) are estimated from the data set. \( F_t \) is the forecasting value of \( F_t \) and we estimate it through fractal model. Let \( \varepsilon_t \) represent the residual at time \( t \) as obtained from the fractal model. Then

\[ \varepsilon_t = \ln(E_t) - \ln(\hat{F}_t) \]  

(3)

The residuals are modeled by the ARIMA and can be represented as follows:

\[ \varepsilon_t = f(\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-n}) + \Delta_t \]  

(4)

where \( f \) is a linear function modeled by the ARIMA and \( \Delta_t \) is the random error. Therefore, the combined forecast is

\[ \hat{E}_t = (\hat{F}_t)\exp(\hat{\varepsilon}_t) = (\hat{F}_t)(\hat{A}_t) \]  

(5)

where \( \hat{\varepsilon}_t \) is the forecasting value of Eq. (4) and \( \hat{A}_t \) is the forecasting value of \( A_t \).

In this investigation, at first we adopt fractal model to forecast failure time, and for the sake of accuracy we compute the difference of the logarithm of the forecasting time minus the logarithm of actual time to obtain a series of differences. We apply run-test method to the difference series to check the stationary of the series. Basically the ARIMA model has three phases: model identification, parameter estimation, and diagnostic checking. We adopt autocorrelation and partial autocorrelation analysis to select an appropriate model from AR(\( p \)), MA(\( p \)), ARMA(\( p, q \)) and ARIMA(\( p, d, q \)). After the model is selected we use BIC (Bayesian Information Criterion) to determine the order of the model. Please see [6] for a detailed exposition.

**Algorithm 1:**

Initialization: Suppose the size of slide window \( m, l = 1 \), the size of training set \( n, A \) is a array of corresponding failure time of the \( i \)th failure, \( D \) is a array of the error series, \( F \) is a array of forecasting value of fractal model and \( E \) is a array of forecasting value of hybrid model;

```
for i = 1 to m + l - 1 {
    B(i) = log(A(i));
    C(i) = log(i);
    for i = m + l - 1 to n{
        (1) According to Eq. (5) in literature [5] and method of linear regression, compute the slope of linear regression in the slide window \( b = d = \frac{1}{l} \) and constant \( a = \log(s) = -dlog(C); \)
        (2) Make a prediction Prediction \( \hat{F}_t \) of next point out of the slide window using the above a and b;
        (3) Add \( \hat{F}_t \) to F;
        (4) Add the practical failure time \( t_i \) of the next point to A;
        (5) Compute the error \( \ln(t_i) - \ln(\hat{F}_t) \);
        (6) Add the error to D;
        \}
    l + +;
    B(m + l - 1) = log(A(m + l - 1));
    C(m + l - 1) = log(m + l - 1);}
Repeat
Repeat steps (1), (2), (3), (4), (5), (6) to compute the nonlinear part of prediction \( \hat{F}_t \) of point \( m + l - 1 \) using fractal model;
    Determine the stationarity of the error series \( D; \)
    Identify the model of ARIMA;
    Determine the order of model;
    Compute the linear part of prediction \( \hat{A}_t \) using ARIMA model and error series \( D; \)
    Make a total prediction \( \hat{E}_t = (\hat{A}_t)(\hat{F}_t) \) and add \( \hat{E}_t \) to E;
    l + +;
    B(m + l - 1) = log(A(m + l - 1));
    C(m + l - 1) = log(m + l - 1);}
```
Table 1  The NTDS set of software failure time series, and from left to right the time in each cell denotes the cumulate time of the ith software failure, \(i = 1, 2, \ldots\).

<table>
<thead>
<tr>
<th>9</th>
<th>21</th>
<th>32</th>
<th>36</th>
<th>43</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>63</td>
<td>70</td>
<td>71</td>
<td>77</td>
<td>78</td>
<td>87</td>
</tr>
<tr>
<td>91</td>
<td>92</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>104</td>
<td>105</td>
</tr>
<tr>
<td>116</td>
<td>149</td>
<td>156</td>
<td>247</td>
<td>249</td>
<td>250</td>
<td>337</td>
</tr>
<tr>
<td>384</td>
<td>396</td>
<td>405</td>
<td>540</td>
<td>798</td>
<td>814</td>
<td>849</td>
</tr>
</tbody>
</table>

Fig. 1  Prediction errors of different models of Musa’s data set 1. HMFE stands for hybrid model forecasting errors, AKFE stands for adaptive Kalman filter model forecasting errors, and FE stands for fractal model forecasting errors. Sliding-window size \(m = 5\) and ARIMA(0, 0, 1).

Fig. 2  Prediction errors of different models of Musa’s data set 2. HMFE stands for hybrid model forecasting errors, AKFE stands for adaptive Kalman filter model forecasting errors, AFE stands for ARIMA model forecasting errors, and FE stands for fractal model forecasting errors. Sliding-window size \(m = 3\) and ARIMA(1, 0, 1).

Fig. 3  Prediction errors of different models of NTDS data set. HMFE stands for hybrid model forecasting errors, AKFE stands for adaptive Kalman filter model forecasting errors, AFE stands for ARIMA model forecasting errors, and FE stands for fractal model forecasting errors. Sliding-window size \(m = 3\) and ARIMA(1, 1, 1).

Table 2  Prediction results of different models of Musa’s data set 1, 2 and NTDS. AK stands for adaptive Kalman filter, HM stands for Hybrid Model.

<table>
<thead>
<tr>
<th></th>
<th>Error</th>
<th>HM</th>
<th>Fractal</th>
<th>ARIMA</th>
<th>AK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa 1</td>
<td>MAE</td>
<td>0.0181</td>
<td>0.0271</td>
<td>0.0432</td>
<td>0.0425</td>
</tr>
<tr>
<td></td>
<td>NRMSE</td>
<td>0.0255</td>
<td>0.0312</td>
<td>0.0493</td>
<td>0.0481</td>
</tr>
<tr>
<td>Musa 2</td>
<td>MAE</td>
<td>0.0437</td>
<td>0.0574</td>
<td>0.0718</td>
<td>0.0635</td>
</tr>
<tr>
<td></td>
<td>NRMSE</td>
<td>0.0528</td>
<td>0.0645</td>
<td>0.0824</td>
<td>0.0702</td>
</tr>
<tr>
<td>NTDS</td>
<td>MAE</td>
<td>0.0492</td>
<td>0.0625</td>
<td>0.1019</td>
<td>0.0947</td>
</tr>
<tr>
<td></td>
<td>NRMSE</td>
<td>0.0741</td>
<td>0.0939</td>
<td>0.1349</td>
<td>0.1203</td>
</tr>
</tbody>
</table>

4. Conclusion

This paper proposed a model of obtaining more accurate predictions by combining fractal and ARIMA. A comparison is made between this approach and other methods for predicting the failure time using actual data sets from...
The superior forecasting ability of the proposed model is due to the following two causes. First, good self-similarity exists in software failure time series. Second, good linearity exists in the residual series which actual failure data minus prediction data using fractal. In the future, some other factors which affect the software reliability can be considered in the model to predict software reliability to improve forecasting accuracy.

### References


