

# Minimizing Relative Error using Numerical Results of Three Term Conjugate Gradient Method

B. Baluch<sup>1</sup> and Z. Salleh<sup>2</sup>

<sup>1,2</sup>Department of Mathematics, Faculty of Ocean Engineering Technology and Informatics,  
 Universiti Malaysia Terengganu, 21030 Kuala Nerus, Terengganu, Malaysia ;  
 Email : <sup>1</sup>srkbaluch@gmail.com, <sup>2</sup>zabidin@umt.edu.my

## ABSTRACT

*In this paper, a new approach related to relative error is shown. Previous research on three term conjugate gradient methods shows that they are numerically efficient but not many researcher use them to show anything new. Recently, we have proposed two efficient three term conjugate gradient method namely BZAU and BZA methods. In these methods it has been shown that these methods are numerically efficient in terms of number of iteration, CPU time and function evaluation. Relative error has been employed on the result of these methods to see which of the methods has less relative error. Furthermore, we have shown that our proposed three term conjugate gradient methods relative error is less as compared to the CG methods by which these methods was compared.*

**Keywords:** Three term conjugate gradient method, relative error, BZAU method, BZA method, large-scale.

**2010 Mathematics Subject Classification:** 90C06, 62-07, 65K05

## 1. INTRODUCTION

In optimization three term conjugate gradient (CG) methods are successful and famous for solving the large scale unconstrained optimization problems. These methods are simple and require less storage requirement. Generally three term CG methods generates a sequence  $(x_k)$  for minimizing large scale unconstrained optimization problem, defined by

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, \dots$$

where  $\alpha_k > 0$  is a general line search and  $d_k$  is a search direction.

Alhawarat and Salleh (2017), Salleh and Alhawarat (2016) and Alhawarat et al. (2017; 2015) proposed efficient CG and hybrid CG methods that fulfill the required global convergence properties. To improve the existing methods, a three term CG technique has been introduced. Several different researchers have suggested various modifications to the three term CG method. Beale (1972) and Nazareth (1977) proposed CG methods based on three terms. Furthermore, reports by McGuire and Wolfe (1973), Deng and Li (1995), Zhang et al. (2006 ; 2007), Cheng (2007), Al-Bayati and Sharif (2010), Zhang Xiao and Wei (2009), Andrei (2011 ; 2013a ; 2013b), Sugiki et al. (2012), Narushima et al. (2011), Babaie-Kafaki

and Ghanbari (2014), Al-Baali et al. (2015), Sun and Liu (2015) and Baluch et al. (2017 ; 2018) discussed the global convergence and numerical results of modified three term CG methods.

However, despite many researchers have been working on three term CG methods, not yet many of them have showed the numerical efficiency benefit in a sense of minimizing any real life formula. Therefore, in this paper, we present benefit by utilizing numerical results of the three term CG methods in terms of number of iteration, CPU time and function evaluation.

Recently, we have proposed two new three term CG methods namely BZAU (Baluch et al., 2017) and BZA (Baluch et al., 2018). The search direction of both these methods are given below,

$$d_k = \begin{cases} -g_k & \text{if } k = 0, \\ -g_k + \beta_k^{BZAU} d_{k-1} - \theta_k^{BZAU} y_{k-1} & \text{if } k \geq 1, \end{cases}$$

and

$$d_k = \begin{cases} -g_k & \text{if } k = 0, \\ -g_k + \beta_k^{BZA} d_{k-1} - \theta_k^{BZA} y_{k-1} & \text{if } k \geq 1. \end{cases}$$

Both of the above methods are efficient as they satisfy the sufficient descent property, independent of any line search used. The global convergence properties for convex and nonconvex functions are proven using Wolfe-Powell line search. Numerically these methods are established well and are tested thoroughly on large scale unconstrained optimization problems in which these methods shows a promising result.

In this paper, the numerical result of both of the methods are used to show the minimization of relative error (RE).

## 2. MINIMIZING RELATIVE ERROR

Generally, many experiments in different field of science like Chemistry, Physics, Biology and Mathematics are used to determine different types of data. Mostly the experiment contains accepted values. During conduction of these experiments, there is a need to make the comparison between the measured values and the accepted values for purpose to determine how error-free the experiment is, so there is need of the term named as error. The error is the deviation of the measured values from accepted values. Generally, the error is of two types, one is absolute error and the second one is a relative error.

Absolute error is the difference between the magnitudes of the accepted value with the measured value. Absolute error is used to find the difference when both the values are in the actual same unit, like kg, cm, grams, metre etc. On the other hand, the relative error is similar like absolute error and the only difference is that the absolute error is divided by accepted value. The unconstrained optimization is used here, so relative error is adopted to check the error. The relative error is also used to check the uncertainty in the data.

The formula of relative error is given and is denoted by RE,

$$RE = \frac{|measured\ value - accepted\ value|}{accepted\ value}.$$

The basic task of three term CG method is to minimize the large-scale unconstrained optimization problems, therefore the relative error is minimized as a new approach by using three term CG method results,

$$\min(RE),$$

Total fifteen large scale unconstrained optimization problems are selected due to the reason their dimensions are the same. The relative error (RE) for both of the three term CG methods BZAU and BZA are given.

### 3. RESULT AND DISCUSSION

This section is divided into two subsection one is for three term BZAU method and the second one is for three term BZA method.

#### 3.1. Relative Error of Three Term BZAU Method

In the three term BZAU method (Baluch et al., 2017), the comparison of BZAU method was made with TMPRP1 method. Hence, the values of BZAU is taken as measured values and the values of TMPRP1 are taken as accepted values and the formula is given below,

$$b1 = RE = \frac{|measured\ value(BZAU) - accepted\ value(TMPRP1)|}{accepted\ value(TMPRP1)},$$

As TMPRP1 method is taken from Sun and Liu (2015), the relative error (RE) for this paper is also calculated. To compare the relative error (RE) with the BZAU method (Baluch et al., 2017) relative error (RE), TMPRP1 method values is taken as measured values and values of CG descent and DTPRP method are taken as accepted values. The formulas are given below:

$$b2 = RE = \frac{|measured\ value(TMPRP1) - accepted\ value(CG\ Descent)|}{accepted\ value(CG\ Descent)},$$

$$b3 = RE = \frac{|measured\ value(TMPRP1) - accepted\ value(DTPRP)|}{accepted\ value(DTPRP)}.$$

**Note:** In  $b1$  the values of BZAU and TMPRP1 are taken from (Baluch et al., 2017). The values used in  $b2$  and  $b3$  are taken from Sun and Liu (2015).

Table 1: Relative error based on number of iteration for  $b1$ ,  $b2$  and  $b3$ .

| Problem                           | $n$  | BZAU | TMPRP1 | $b1$   | TMPRP1 | CG<br>Descent | DTPRP | $b2$   | $b3$   |
|-----------------------------------|------|------|--------|--------|--------|---------------|-------|--------|--------|
| Extended white and Holst function | 1000 | 31   | 32     | 0.0312 | 48     | 101           | 71    | 0.5247 | 0.3239 |
| Extended Rosen function           | 5000 | 28   | 21     | 0.3333 | 44     | 119           | 54    | 0.6302 | 0.1851 |
| NONDIA Shano(78)                  | 5000 | 14   | 16     | 0.125  | 30     | 60            | 209   | 0.5    | 0.8564 |
| Diagonal 4 function               | 5000 | 2    | 2      | 0      | 48     | 128           | 55    | 0.625  | 0.1272 |
| Diagonal 2 function               | 100  | 964  | 968    | 0.0041 | 780    | 104           | 780   | 6.5    | 0      |

|                                |      |      |      |        |     |     |     |        |        |
|--------------------------------|------|------|------|--------|-----|-----|-----|--------|--------|
| HIMMELBH                       | 5000 | 9    | 8    | 0.125  | 18  | 23  | 16  | 0.2173 | 0.125  |
| Hager                          | 100  | 25   | 25   | 0      | 44  | 32  | 34  | 0.375  | 0.2941 |
| Extended Maratos function      | 1000 | 62   | 31   | 1      | 66  | 132 | 136 | 0.5    | 0.5147 |
| SINCOS                         | 5000 | 8    | 12   | 0.3333 | 26  | 22  | 52  | 0.1818 | 0.5    |
| EDENSCH                        | 1000 | 3090 | 3094 | 0.0012 | 40  | 35  | 49  | 0.1428 | 0.136  |
| Quadratic QF1 function         | 5000 | 426  | 426  | 0      | 731 | 790 | 882 | 0.0746 | 0.1712 |
| Extended Quadratic penalty QP2 | 5000 | 61   | 86   | 0.2906 | 64  | 143 | 78  | 0.5524 | 0.1794 |
| Perturbed Quadratic            | 5000 | 425  | 425  | 0      | 760 | 959 | 774 | 0.2075 | 0.0180 |
| Generalized Quartic GQ1        | 5000 | 11   | 12   | 0.0833 | 21  | 18  | 36  | 0.1666 | 0.4166 |
| Extronsb                       | 5000 | 0    | 0    | 0      | 39  | 159 | 43  | 0.7547 | 0.0930 |

In Table 1,  $b_1$  shows 86.6% less relative error (RE) compared with  $b_2$  and having 73.3% less relative error compared with  $b_3$ . So  $b_1$  is efficient in minimizing the relative error. All of these percentages are calculated for the Table 1 to 3 by the formula

$$\text{Percentage} = \frac{\text{how many values of } b_1 \text{ is less than } b_2}{\text{total number of functions}} \times 100\%.$$

Further the same formula is used to calculate the percentage when  $b_1$  is compared with  $b_3$ .

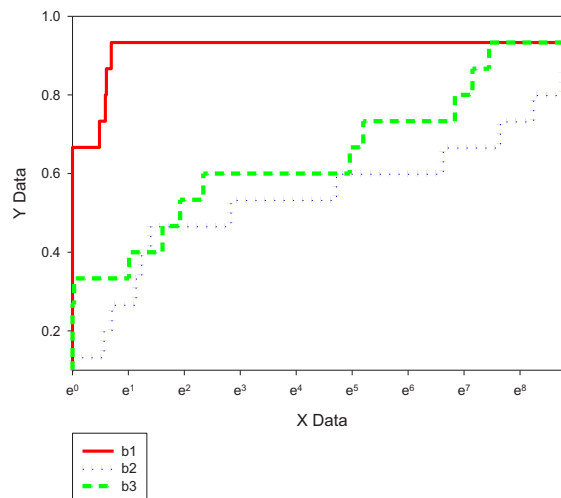


**Figure 1.** Relative error of number of iteration.

In Figure 1, the top curve is the method that shows which of the above method has less relative error (RE) in terms of minimum. The relative error (RE) of  $b_1$  is minimized more than other two methods. So from here it can be concluded that the performance of  $b_1$  is better than  $b_2$  and  $b_3$  in terms of minimum.

Table 2: Relative error based on CPU time for  $b_1$ ,  $b_2$  and  $b_3$ .

| Problem                           | $n$  | BZAU    | TMPRP1  | $b_1$   | TMPRP1 | CG<br>Descent | DTPRP  | $b_2$  | $b_3$  |
|-----------------------------------|------|---------|---------|---------|--------|---------------|--------|--------|--------|
| Extended white and Holst function | 1000 | 1.845   | 1.4398  | 0.28142 | 1.3594 | 3.3438        | 2.0625 | 0.5934 | 0.3408 |
| Extended Rosen function           | 5000 | 2.0617  | 1.1303  | 0.82402 | 1.7344 | 3.6875        | 1.8281 | 0.5296 | 0.0512 |
| NONDIA Shano(78)                  | 5000 | 1.4958  | 1.5309  | 0.02292 | 1.8906 | 3.7812        | 10.734 | 0.5    | 0.8238 |
| Diagonal 4 function               | 5000 | 0.6138  | 0.623   | 0.01476 | 1.3906 | 3.6406        | 1.1719 | 0.6180 | 0.1866 |
| Diagonal 2 function               | 100  | 12.4522 | 12.5155 | 0.0050  | 0.5938 | 0.1875        | 0.5313 | 2.1669 | 0.1176 |
| HIMMELBH                          | 5000 | 1.1324  | 1.7183  | 0.34097 | 0.7969 | 1.1406        | 0.6719 | 0.3013 | 0.1860 |
| Hager                             | 100  | 1.2086  | 0.9215  | 0.3115  | 0.3281 | 0.1563        | 0.2188 | 1.0991 | 0.4995 |
| Extended Maratos function         | 1000 | 3.0034  | 1.5017  | 1       | 0.6563 | 1.3126        | 1.2344 | 0.5    | 0.4683 |
| SINCOS                            | 5000 | 1.1289  | 1.2905  | 0.1252  | 1.5313 | 1.1719        | 3.0626 | 0.3066 | 0.5    |
| EDENSCH                           | 1000 | 135.232 | 131.676 | 0.0270  | 1.7188 | 1.1875        | 3.0938 | 0.4474 | 0.4444 |
| Quadratic QF1 function            | 5000 | 10.7051 | 11.6042 | 0.0774  | 19.843 | 20.187        | 22.390 | 0.0170 | 0.1137 |
| Extended Quadratic penalty QP2    | 5000 | 6.627   | 9.3083  | 0.2880  | 5.171  | 13.921        | 6.4063 | 0.6285 | 0.1926 |
| Perturbed Quadratic               | 5000 | 11.039  | 10.9626 | 0.0069  | 42.781 | 52.906        | 43.906 | 0.1913 | 0.0256 |
| Generalized Quartic GQ1           | 5000 | 0.987   | 1.219   | 0.1903  | 0.7500 | 1.0469        | 1.8594 | 0.2835 | 0.5966 |
| Extronsb                          | 5000 | 0.5188  | 0.5168  | 0.0038  | 1.7500 | 14.8750       | 2.3281 | 0.8823 | 0.2483 |

In Table 2, CPU time shows the relative error (RE) of  $b_1$  is 66.6% less as compared to  $b_2$  and 73.3% less as compared to  $b_3$ .

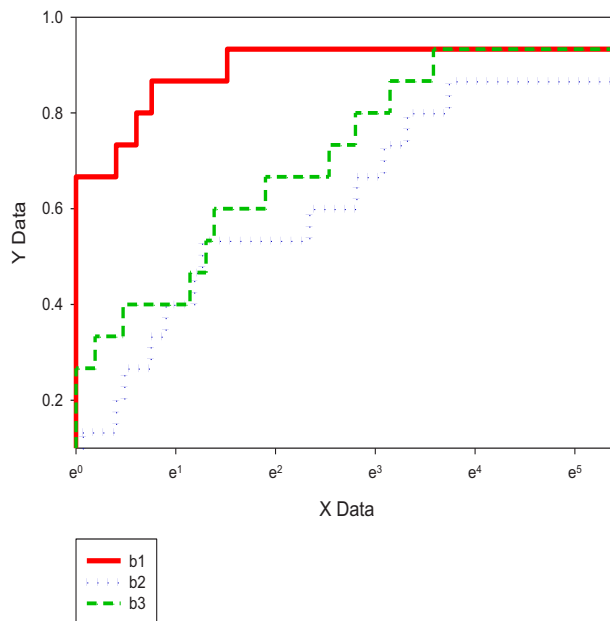


**Figure 2.** Relative error of CPU time.

Figure 2 shows the performance of  $b1$  is better than  $b2$  and  $b3$  in terms of minimum. The relative error (RE) of  $b1$  is minimized more than other two methods. So from here, it can be concluded that relative error (RE) of  $b1$  minimizes more in terms of CPU time.

*Table 3:* Relative error based on function evaluation for  $b1$ ,  $b2$  and  $b3$ .

| Problem                           | $n$  | BZAU | TMPRP1 | $b1$    | TMPRP1 | CG   | DTPRP | $b2$   | $b3$   |
|-----------------------------------|------|------|--------|---------|--------|------|-------|--------|--------|
| Descent                           |      |      |        |         |        |      |       |        |        |
| Extended white and Holst function | 1000 | 191  | 160    | 0.1937  | 874    | 2321 | 1474  | 0.6234 | 0.4070 |
| Extended Rosen function           | 5000 | 141  | 88     | 0.6022  | 868    | 2195 | 956   | 0.6045 | 0.0920 |
| NONDIA Shano(78)                  | 5000 | 69   | 73     | 0.05479 | 626    | 1252 | 4327  | 0.5    | 0.8553 |
| Diagonal 4 function               | 5000 | 6    | 6      | 0       | 815    | 2383 | 673   | 0.6579 | 0.2109 |
| Diagonal 2 function               | 100  | 1930 | 1938   | 0.0041  | 781    | 341  | 781   | 1.2903 | 0      |
| HIMMELBH                          | 5000 | 67   | 65     | 0.0307  | 103    | 224  | 91    | 0.5401 | 0.1318 |
| Hager                             | 100  | 125  | 76     | 0.6447  | 640    | 251  | 403   | 1.5498 | 0.5880 |
| Extended Maratos function         | 1000 | 362  | 181    | 1       | 1121   | 2242 | 2206  | 0.5    | 0.4918 |
| SINCOS                            | 5000 | 30   | 38     | 0.2105  | 250    | 132  | 500   | 0.8939 | 0.5    |
| EDENSCH                           | 1000 | 6185 | 6193   | 0.0012  | 615    | 450  | 1273  | 0.3666 | 0.5168 |

|                                         |      |      |      |        |       |       |       |        |        |
|-----------------------------------------|------|------|------|--------|-------|-------|-------|--------|--------|
| Quadratic<br>QF1<br>function            | 5000 | 1278 | 1278 | 0      | 12453 | 13180 | 14508 | 0.0551 | 0.1416 |
| Extended<br>Quadratic<br>penalty<br>QP2 | 5000 | 678  | 970  | 0.3010 | 988   | 2919  | 1170  | 0.6615 | 0.1555 |
| Perturbed<br>Quadratic                  | 5000 | 1275 | 1275 | 0      | 16744 | 22230 | 17831 | 0.2467 | 0.0609 |
| Generalized<br>Quartic<br>GQ1           | 5000 | 35   | 39   | 0.1025 | 159   | 342   | 777   | 0.5350 | 0.7953 |
| Extronsb                                | 5000 | 334  | 0    | 0      | 741   | 7756  | 1010  | 0.9044 | 0.2663 |

In Table 3, on the basis of function evaluation  $b1$  shows 93.3% less relative error (RE) compared to  $b2$  and  $b1$  shows 66.6% less relative error (RE) compared to  $b3$ .

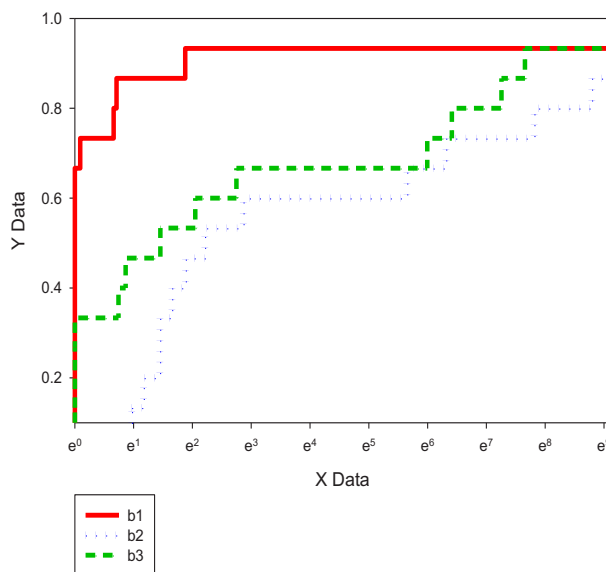


**Figure 3.** Relative error of function evaluation.

Figure 3 shows the performance of  $b1$  is better than  $b2$  and  $b3$  in terms of minimum. The relative Error (RE) of  $b1$  reduces more in terms of function evaluation. So from here, it can be concluded that relative error (RE) of  $b1$  is less than  $b2$  and  $b3$ .

It can also be concluded that  $b1$  relative error (RE) is less than other two  $b2$  and  $b3$  in all the cases such as the number of iteration, CPU time and function evaluation.

### 3.2. Relative Error of Three Term BZA Method

In the three term BZA method (Baluch et al., 2018), the comparison of three term BZA method was made with MTTHS and DHS method. As the DHS method by Dai and Wen (2012) have different dimensions and very less number of test functions available, so the relative error (RE) of three term

BZA method is compared with MTTHS method only which have the same dimension. The values for BZA method are taken as measured values and the values for MTTHS method are taken as accepted values.

$$b4 = RE = \frac{|measured\ value(BZA) - accepted\ value(MTTHS)|}{accepted\ value(MTTHS)}.$$

The MTTHS method is taken from Zhang et al. (2007), in which it was compared with CG Descent, LBFGS and PRP+ method. Therefore the relative error (RE) for this paper is also calculated and compared in the end with the three term BZA method (Baluch et al., 2018) relative error (RE).

$$b5 = RE = \frac{|measured\ value(MTTHS) - accepted\ value(CG\ Descent)|}{accepted\ value(CG\ Descent)},$$

$$b6 = RE = \frac{|measured\ value(MTTHS) - accepted\ value(LBFGS)|}{accepted\ value(LBFGS)},$$

$$b7 = RE = \frac{|measured\ value(MTTHS) - accepted\ value(PRP+)|}{accepted\ value(PRP+)}.$$

**Note:** The values used in  $b4$  are taken from three term BZA method results given in (Baluch et al., 2018). The values used in  $b5$ ,  $b6$  and  $b7$  are taken from Zhang et al. (2007).

Table 4: Relative error based on number of iteration for  $b4$ ,  $b5$ ,  $b6$  and  $b7$ .

| Problem                        | n     | BZA  | MTTHS | $b4$   | MTTHS | CG<br>Descent | LBFGS | PRP+ | $b5$   | $b6$   | $b7$   |
|--------------------------------|-------|------|-------|--------|-------|---------------|-------|------|--------|--------|--------|
| DIXMAANA                       | 6000  | 8    | 8     | 0      | 7     | 9             | 11    | 8    | 0.2222 | 0.3636 | 0.125  |
| DIXMAANB                       | 6000  | 9    | 8     | 0.1    | 8     | 9             | 11    | 7    | 0.1111 | 0.2727 | 0.1428 |
| DIXMAANC                       | 6000  | 8    | 7     | 0.1    | 9     | 10            | 12    | 7    | 0.1    | 0.25   | 0.2857 |
| DIXMAANE                       | 6000  | 333  | 331   | 0.0060 | 306   | 302           | 345   | 307  | 0.0132 | 0.1130 | 0.0032 |
| NONDIA<br>Shano(78)            | 10000 | 15   | 16    | 0.0625 | 9     | 9             | 22    | 6    | 0      | 0.5909 | 0.5    |
| Generalized<br>Quartic<br>GQ1  | 10000 | 10   | 11    | 0.0909 | 34    | 35            | 52    | 16   | 0.0285 | 0.3461 | 1.125  |
| Generalized<br>Quartic<br>GQ2  | 5000  | 25   | 26    | 0.0384 | 33    | 33            | 49    | 17   | 0      | 0.3265 | 0.9411 |
| Power<br>function              | 1000  | 1565 | 4612  | 0.6606 | 348   | 116           | 137   | 114  | 2      | 1.5401 | 2.0526 |
| DQDRTIC<br>(Cute)              | 5000  | 5    | 14    | 0.6428 | 14    | 7             | 13    | 5    | 1      | 0.0769 | 1.8    |
|                                | 10000 | 5    | 14    | 0.6428 | 12    | 7             | 13    | 5    | 0.7142 | 0.0769 | 1.4    |
| EDENSCH<br>function            | 1000  | 3089 | 3094  | 0.0016 | 31    | 31            | 28    | 25   | 0      | 0.1071 | 0.24   |
|                                | 5000  | 3089 | 3094  | 0.0016 | 34    | 34            | 31    | 23   | 0      | 0.0967 | 0.4782 |
| SINQUAD<br>(Cute)              | 5000  | 411  | 822   | 0.5    | 93    | 46            | 30    | 22   | 1.0217 | 2.1    | 3.2272 |
|                                | 10000 | 417  | 834   | 0.5    | 380   | 56            | 28    | 27   | 5.7857 | 12.571 | 13.074 |
| Extended<br>Wood               | 1000  | 101  | 146   | 0.3082 | 227   | 125           | 88    | 26   | 0.816  | 1.5795 | 7.7307 |
|                                | 10000 | 89   | 213   | 0.5821 | 524   | 187           | 87    | 232  | 1.8021 | 5.0229 | 1.2586 |
| ARGLINB                        | 50    | 1    | 1     | 0      | 3     | 3             | 3     | 4    | 0      | 0      | 0.3333 |
| Extended<br>Hiebert            | 200   | 901  | 1802  | 0.5    | 76    | 40            | 59    | 15   | 0.9    | 0.2881 | 4.0666 |
| Extended<br>Powell<br>function | 5000  | 306  | 2213  | 0.8617 | 396   | 162           | 38    | 148  | 1.4444 | 9.4210 | 1.6756 |

In Table 4,  $b_4$  shows 68.42% less relative error (RE) than  $b_5$ . Further, the relative error (RE) of  $b_4$  is 84.21% less than  $b_6$  and 94.73% less than  $b_7$ . All of these percentages are calculated for the Table 4 to 6 by the formula

$$\text{Percentage} = \frac{\text{how many values of } b_4 \text{ is less than } b_5}{\text{total number of functions}} \times 100.$$

Further the same formula is used to calculate the percentage in which  $b_4$  is compared with  $b_6$  and  $b_4$  is compared with  $b_7$ .

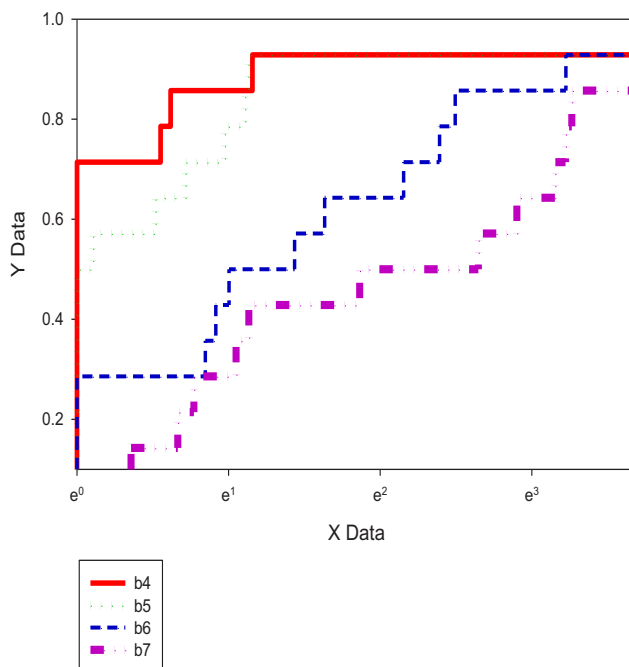


**Figure 4.** Relative error of number of iteration.

Figure 4 shows the performance of  $b_4$  is better than all in terms of minimum, the top curve is the method that shows which of the above method has less relative error (RE),  $b_5$  is close to  $b_4$ , still,  $b_4$  performance is better. On the other hand,  $b_4$  outclass  $b_6$  and  $b_7$  totally. Therefore relative error (RE) of  $b_4$  is less than  $b_5, b_6$  and  $b_7$ .

**Table 5:** Relative error based on CPU time for  $b_4, b_5, b_6$  and  $b_7$ .

| Problem             | $n$       | BZA        | MTTH<br>S | $b_4$      | MTTH<br>S | CG<br>Descen<br>t | LBFG<br>S | PRP<br>+ | $b_5$      | $b_6$      | $b_7$      |
|---------------------|-----------|------------|-----------|------------|-----------|-------------------|-----------|----------|------------|------------|------------|
| DIXMAANA            | 6000      | 1.850<br>4 | 1.8623    | 0.006<br>3 | 0.14      | 0.15              | 0.16      | 0.19     | 0.066<br>6 | 0.125      | 0.263<br>1 |
| DIXMAANB            | 6000      | 2.036<br>2 | 1.8966    | 0.073<br>6 | 0.14      | 0.15              | 0.17      | 0.18     | 0.066<br>6 | 0.176<br>4 | 0.222<br>2 |
| DIXMAANC            | 6000      | 2.173<br>6 | 2.0093    | 0.081<br>7 | 0.15      | 0.16              | 0.18      | 0.20     | 0.062<br>5 | 0.166<br>6 | 0.25       |
| DIXMAANE            | 6000      | 83.09<br>5 | 81.226    | 0.023<br>0 | 1.86      | 1.77              | 2.87      | 2.85     | 0.050<br>8 | 0.351<br>9 | 0.347<br>3 |
| NONDIA<br>Shano(78) | 1000<br>0 | 6.211<br>3 | 6.3412    | 0.020<br>4 | 0.28      | 0.30              | 0.46      | 0.36     | 0.066<br>6 | 0.391<br>3 | 0.222<br>2 |

|                          |       |        |        |        |      |      |      |      |        |        |        |
|--------------------------|-------|--------|--------|--------|------|------|------|------|--------|--------|--------|
| Generalized Quartic GQ1  | 10000 | 3.0317 | 3.4464 | 0.1203 | 0.27 | 0.27 | 0.58 | 0.30 | 0      | 0.5344 | 0.1    |
| Generalized Quartic GQ2  | 5000  | 2.7548 | 2.9409 | 0.0632 | 0.14 | 0.13 | 0.27 | 0.15 | 0.0769 | 0.4814 | 0.0666 |
| Power function           | 1000  | 11.567 | 33.526 | 0.6549 | 0.09 | 0.04 | 0.10 | 0.03 | 1.25   | 0.1    | 2      |
| DQDRTIC (Cute)           | 5000  | 0.8084 | 1.6324 | 0.5050 | 0.19 | 0.14 | 0.18 | 0.16 | 0.3571 | 0.0555 | 0.1875 |
|                          | 10000 | 1.5979 | 3.7862 | 0.5779 | 0.37 | 0.32 | 0.41 | 0.33 | 0.1562 | 0.0975 | 0.1212 |
| EDENSCH function         | 1000  | 17.626 | 17.651 | 0.0014 | 0.07 | 0.06 | 0.07 | 0.14 | 0.1666 | 0      | 0.5    |
|                          | 5000  | 198.22 | 198.85 | 0.0031 | 0.40 | 0.37 | 0.80 | 0.80 | 0.0810 | 0.5    | 0.5    |
| SINQUAD (Cute)           | 5000  | 198.61 | 397.23 | 0.5    | 1.21 | 0.81 | 2.42 | 2.42 | 0.4938 | 0.5    | 0.5    |
|                          | 10000 | 493.68 | 987.36 | 0.5    | 8.65 | 2.12 | 17.3 | 17.3 | 3.0801 | 0.5    | 0.5    |
| Extended Wood            | 1000  | 1.7151 | 1.7443 | 0.0167 | 0.16 | 0.10 | 0.12 | 0.05 | 0.6    | 0.3333 | 2.2    |
|                          | 10000 | 31.880 | 57.843 | 0.4488 | 4.06 | 1.76 | 1.27 | 2.57 | 1.3068 | 2.1968 | 0.5797 |
| ARGLINB                  | 50    | 0.3832 | 0.3863 | 0.0080 | 0.01 | 0.01 | 0.01 | 0.02 | 0      | 0      | 0.5    |
| Extended Hiebert         | 200   | 23.007 | 46.014 | 0.5    | 0.94 | 0.62 | 0.65 | 0.48 | 0.5161 | 0.4461 | 0.9583 |
| Extended Powell function | 5000  | 48     | 330    | 0.8545 | 1.06 | 0.46 | 0.22 | 0.22 | 1.3043 | 3.8181 | 0.7966 |

In Table 5, the CPU time shows the relative error (RE) of  $b_4$  is 63.15% less than  $b_5$ . Moreover, the relative error (RE) of  $b_4$  is 63.15% less as compared with  $b_6$  and 78.94% less as compared with  $b_7$ .

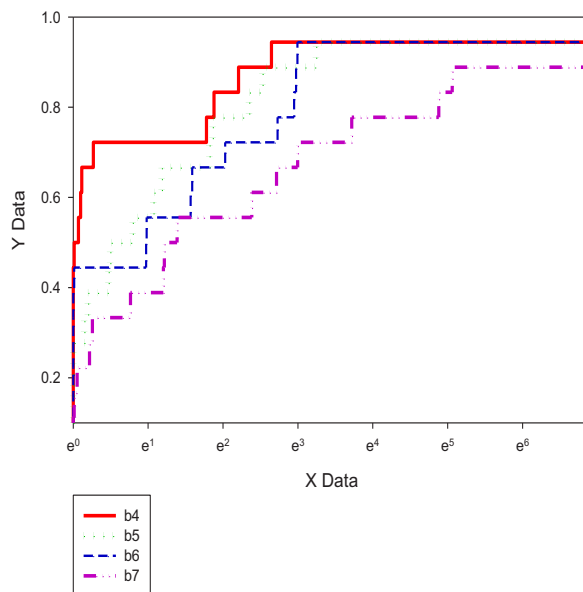


Figure 5. Relative error of CPU time.

Figure 5 shows the performance of  $b_4$  is better than all in terms of minimum,  $b_5$  and  $b_6$  is near to  $b_4$ , still  $b_4$  performance is better. On the other hand  $b_4$  outclass  $b_7$  totally. Therefore relative error (RE) of  $b_4$  is less than  $b_5$ ,  $b_6$  and  $b_7$  in terms of CPU time.

Table 6: Relative error based on function evaluation for  $b_4$ ,  $b_5$ ,  $b_6$  and  $b_7$ .

| Problem                        | $n$   | BZA         | MTTHS   | $b_4$  | MTTHS | CG<br>Descen<br>t | LBFG<br>S | PR<br>P+ | $b_5$  | $b_6$  | $b_7$  |
|--------------------------------|-------|-------------|---------|--------|-------|-------------------|-----------|----------|--------|--------|--------|
| DIXMAANA                       | 6000  | 26          | 26      | 0      | 15    | 19                | 13        | 24       | 0.2105 | 0.1538 | 0.375  |
| DIXMAANB                       | 6000  | 28          | 25      | 0.12   | 17    | 19                | 13        | 23       | 0.1052 | 0.3076 | 0.2608 |
| DIXMAANC                       | 6000  | 28          | 25      | 0.12   | 19    | 21                | 14        | 23       | 0.0952 | 0.3571 | 0.1739 |
| DIXMAANE                       | 6000  | 1351        | 1323    | 0.0211 | 613   | 605               | 355       | 620      | 0.0132 | 0.7267 | 0.0112 |
| NONDIA<br>Shano(78)            | 10000 | 76          | 79      | 0.0379 | 21    | 22                | 27        | 26       | 0.0454 | 0.2222 | 0.1923 |
| Generalized<br>Quartic GQ1     | 10000 | 32          | 35      | 0.0857 | 69    | 71                | 60        | 69       | 0.0281 | 0.15   | 0      |
| Generalized<br>Quartic GQ2     | 5000  | 84          | 89      | 0.0561 | 67    | 67                | 57        | 66       | 0      | 0.1754 | 0.0151 |
| Power<br>function              | 1000  | 4695        | 13836   | 0.6606 | 697   | 233               | 139       | 236      | 1.991  | 4.0143 | 1.9533 |
| DQDRTIC<br>(Cute)              | 5000  | 15          | 42      | 0.6428 | 29    | 15                | 21        | 15       | 0.9333 | 0.3809 | 0.9333 |
|                                | 10000 | 15          | 42      | 0.6428 | 25    | 15                | 21        | 15       | 0.6666 | 0.1904 | 0.6666 |
| EDENSCH<br>function            | 1000  | 6183        | 6193    | 0.0016 | 56    | 60                | 32        | 60       | 0.0666 | 0.75   | 0.0967 |
|                                | 5000  | 6183        | 6193    | 0.0016 | 57    | 61                | 38        | 62       | 0.0655 | 0.5    | 0.0806 |
| SINQUAD<br>(Cute)              | 5000  | 2425        | 4850    | 0.5    | 179   | 111               | 41        | 61       | 0.6126 | 3.3658 | 0.6592 |
|                                | 10000 | 2232        | 4464    | 0.5    | 614   | 121               | 44        | 72       | 4.0743 | 12.954 | 7.5277 |
| Extended<br>Wood               | 1000  | 504         | 529     | 0.0472 | 486   | 293               | 123       | 67       | 0.6587 | 2.9512 | 6.2537 |
|                                | 10000 | 455         | 765     | 0.4052 | 1070  | 426               | 112       | 487      | 1.5117 | 8.5535 | 1.1971 |
| ARGLINB                        | 50    | 3           | 3       | 0      | 7     | 7                 | 4         | 12       | 0      | 0.75   | 0.4166 |
| Extended<br>Hiebert            | 200   | 9611        | 19222   | 0.5    | 153   | 81                | 62        | 38       | 0.8888 | 1.4677 | 3.0263 |
| Extended<br>Powell<br>function | 5000  | 12687<br>50 | 8562500 | 0.8167 | 797   | 332               | 40        | 346      | 1.4006 | 18.925 | 1.3034 |

In Table 6, the relative error (RE) of  $b_4$  is 73.68% less as compared to  $b_5$ . Further the relative error (RE) of  $b_4$  is 89.47% less than  $b_6$  and 78.94% less than  $b_7$ .

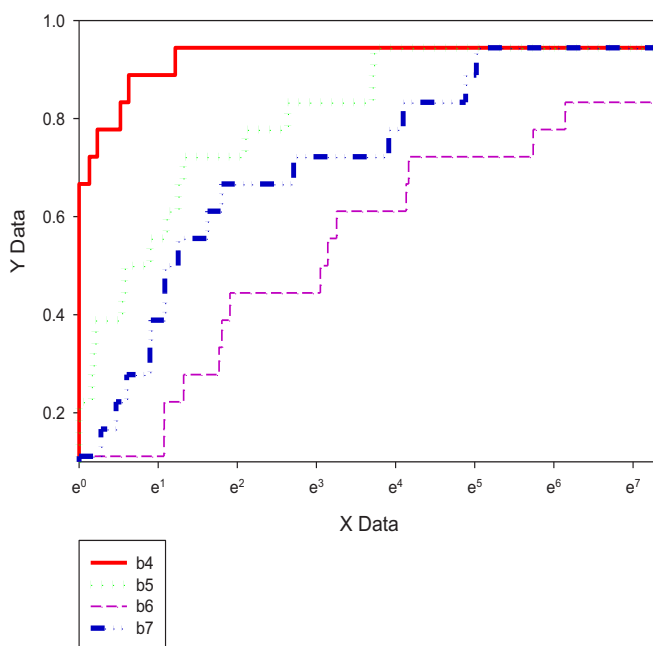


**Figure 6.** Relative error of function evaluation.

Figure 6 shows the performance of  $b_4$  is better than all in terms of minimum,  $b_5$  and  $b_6$  slightly compete with  $b_4$ , still  $b_4$  performance is better than  $b_5$  and  $b_6$ . On the other hand,  $b_4$  outclass  $b_7$  totally. Therefore relative error (RE) of  $b_4$  is less as compared to  $b_5$ ,  $b_6$  and  $b_7$ . So from all the cases such as number of iteration, CPU time and function evaluation, it shows that  $b_4$  has a less relative error (RE) than all others. For Figures 1 to 6 we adopted the performance profiles given by Dolan and Moré (2002). The Sigma plot software was used to plot graph the data.

#### 4. DISCUSSION AND CONCLUSION

In this paper, the relative error has been employed as a new approach on the numerical results of three term Conjugate Gradient method. The idea was to use the numerical results and to minimize relative error. For BZAU method, the BZAU method was compared with TMPRP1 method, next the TMPRP1 method was compared with CG Descent and then DTRP method to get  $b_1$ ,  $b_2$  and  $b_3$ , respectively. While for BZA method, the BZA method was compared with MTTHS method, and then the MTTHS method was compared with CG Descent, LBFGS and PRP+ methods to obtain  $b_4$ ,  $b_5$ ,  $b_6$  and  $b_7$ , respectively.

After applying the relative error on the numerical results of BZAU and BZA methods, in terms of number of iteration, CPU time and function evaluation, the results indicate that our proposed methods namely BZAU and BZA methods have less relative error as compared to the methods by which these methods was compared as mentioned above.

## 5. REFERENCES

- Alhawarat, A., Salleh, Z., 2017, Modification of nonlinear conjugate gradient method with weak Wolfe-Powell line search. *Abstr. Appl. Anal.* 2017, Article ID 7238134, 6 pages.
- Alhawarat, A., Salleh, Z., Mamat, M., Rivaie, M., 2017, An efficient modified Polak-Ribière-Polyak conjugate gradient method with global convergence properties. *Optim. Methods Softw.* **32**(6), 1299-1312.
- Alhawarat, A., Mamat, M., Rivaie, M., Salleh, Z., 2015, An efficient hybrid conjugate gradient method with the strong Wolfe-Powell line search. *Math. Probl. Eng.* 2015, Article ID 103517, 7 pages.
- Al-Baali, M., Narushima, Y., Yabe, H., 2015, A family of three term conjugate gradient methods with sufficient descent property for unconstrained optimization. *Comput. Optim. Appl.* **60**(1), 89-110.
- Al-Bayati, A.Y., Sharif, W.H., 2010, A new three-term conjugate gradient method for unconstrained optimization. *Can. J. Sci. Eng. Math.* **1**(5), 108-124.
- Andrei, N., 2011, A modified Polak-Ribière-Polyak conjugate gradient algorithm for unconstrained optimization. *Optimization* **60**, 1457-1471.
- Andrei, N., 2013a, On three-term conjugate gradient algorithms for unconstrained optimization. *Appl. Math. Comput.* **219**, 6316-6327.
- Andrei, N., 2013b, A simple three-term conjugate gradient algorithm for unconstrained optimization. *J. Comput. Appl. Math.* **241**, 19-29.
- Babaie-Kafaki, S., Ghanbari, R., 2014, Two modified three term conjugate gradient methods with sufficient descent property. *Optimization Letters* **8**(8), 2285-2297.
- Baluch, B., Salleh, Z., Alhawarat, A., Roslan, U.A.M., 2017, A new modified three-term conjugate gradient method with sufficient descent property and its global convergence. *J. Math.* Article ID 2715854, 12 pages.
- Baluch, B., Salleh, Z., Alhawarat, A., 2018, A New Modified Three-Term Hestenes–Stiefel Conjugate Gradient Method with Sufficient Descent Property and Its Global Convergence. *Journal of Optimization.* Article ID 5057096, 13 pages.
- Beale, E.M.L., 1972, *A derivative of conjugate gradients*. In: Lootsma, F.A (ed.) Numerical Methods for Nonlinear Optimization, pp. 39-43. Academic, London.
- Cheng, W., 2007, A two-term PRP-based descent method. *Numer. Funct. Anal. Optim.* **28**, 1217-1230.
- Dai, Z.F., Wen, F.H., 2012, Another improved Wei-Yao-Liu nonlinear conjugate gradient method with sufficient descent property. *Appl. Math. Comput.* **218**, 7421-7430.
- Deng, N.Y., Li, Z., 1995, Global convergence of three terms conjugate gradient methods. *Optim. Method Softw.* **4**, 273-282.
- Dolan, E.D., Moré, J.J., 2002, Benchmarking optimization software with performance profiles. *Math. Program* **91**, 201-213.
- McGuire, M.F., Wolfe, P., 1973, *Evaluating a Restart Procedure for Conjugate Gradients*. Report RC-4382, IBM Research Center, Yorktown Heights.
- Nazareth, L., 1977, A conjugate direction algorithm without line search. *J. Optim. Theor. Appl.* **23**, 373-387.
- Narushima, Y., Yabe, H., Ford, J.A., 2011, A three-term conjugate gradient method with sufficient descent property for unconstrained optimization. *SIAM J. Optim.* **21**, 212-230.

Salleh, Z., Alhawarat, A., 2016, An efficient modification of the Hestenes-Stiefel nonlinear conjugate gradient method with restart property. *J. Inequal. Appl.* 2016:**110**  
DOI 10.1186/s13660-016-1049-5.

Sun, M., Liu, J., 2015, Three modified Polak-Ribière-Polyak conjugate gradient methods with sufficient descent property. *J. Inequal. Appl.* 2015(1), 1-14.

Sugiki, K., Narushima, Y., Yabe, H., 2012, Globally convergent three-term conjugate gradient methods that use secant conditions and generate descent search directions for unconstrained optimization. *J. Optim. Theory Appl.* **153**(3), 733-757.

Zhang, L., Zhou, W., Li, D.H., 2006, A descent modified Polak-Ribière-Polyak conjugate gradient method and its global convergence. *IMA J. Numer. Anal.* **26**, 629-640.

Zhang, L., Zhou, W., Li, D.H., 2007, Some descent three-term conjugate gradient methods and their global convergence. *Optim. Methods Softw.* **22**, 697-711.

Zhang, J., Xiao, Y., Wei, Z., 2009, Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained optimization. *Math. Probl. Eng.* 2009, Article ID 243290, 16 pages.