Journal of the Nigerian Association of Mathematical Physics, Volume 9 (November 2005)

A cubic interpolation algorithm for solving non-linear equations

P. U. Ohirhian
Department of Petroleum Engineering, University of Benin, Benin City, Nigeria

Abstract

A new Algorithm - based on cubic interpolation have been developed for solving non-linear algebraic equations. The Algorithm is derived from LaGrange's interpolation polynomial. The method discussed here is faster than the "Regular Falsi" which is based on linear interpolation. Since this new method does not involve derivatives, it does not have the problem of non-convergence associated with the Newton - Raphson method, when a small derivative is encountered. The efficiency of the cubic interpolation method was ascertained by practical testing with Leonardo's cubic equation and Dranchuk, Purivis and Robinson method for calculating the Gas Compressibility Factor (z) of Natural Gases. The Watfor computer program (based on the new method) that was used to solve the Dranchuk Purivis and Robinson equation is also included in this paper.

pp 317 - 324

1.0 Introduction

Several methods based on linear interpolation exist for solving non-linear equations. Ohirhian (1994) [5], developed several Algorithms based on quadratic interpolation for solving non-linear equations. The method he used for the development of Algorithms can be extended to cubic or even quartic of any of the standard interpolation polynomials.

In this work a cubic interpolation method developed from LaGrange's interpolation polynomial is presented. As a practical test, the new method, and two popular current methods used for solving non-linear equations (the "Regular Falsi" and the Newton-Raphson method) were used to solve a cubic equation attributed to Leonardo of Pisa (1225). The test shows that the new method requires fewer iterations then both the Regular Falsi and the Newton-Raphson methods to converge.

The cubic interpolation method was also used to solve the non-linear equation developed by Dranchuk, Purvis and Robinson (1974) for calculating the Compressibility Factor (z) of natural Gases.

2.0 Development of the Algorithm

The LaGrange's interpolation polynomial up to the third degree is:

$$y = \frac{y_0(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} + \frac{y_1(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} + \frac{y_2(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_3)} + \frac{y_3(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} \tag{2.1}$$

Interchanging the role of x and y and equating y to zero leads to:

$$x = -\frac{y_1 y_2 y_3 x_0}{(y_0-y_1)(y_0-y_1)(y_0-y_3)} - \frac{y_0 y_2 y_3 x_1}{(y_1-y_0)(y_1-y_2)(y_1-y_3)}$$

$$- \frac{y_0 y_1 y_3 x_2}{(y_2 - y_0)\ (y_2 - y_1)\ (y_2 - y_3)} \quad - \frac{y_0 y_1 y_2 x_3}{(y_3 - y_0)\ (y_3 - y_1)\ (y_3 - y_2)} \qquad (2.2)$$

Introduction of an iteration counter (k) in equation (2.2) leads to the final form of the new Algorithm.

$$x^{k+1} = - y_1^k y_2^k y_3^k x_0^k \,/ A^k - y_0^k y_2^k y_3^k x_1^k \,/ B^k - y_0^k y_1^k y_3^k x_2^k \,/ C^k - y_0^k y_1^k y_2^k x_3^k \,/ D^k \quad (2.3)$$

where $A^k = (y_0^k - y_1^k)(y_0^k - y_2^k)(y_0^k - y_3^k)$, $B^k = (y_1^k - y_0^k)(y_1^k - y_2^k)(y_1^k - y_3^k)$, $C^k = (y_2^k - y_0^k)(y_2^k - y_1^k)(y_2^k - y_3^k)$, $D^k = (y_3^3 - y_0^k)(y_3^k - y_1^k)(y_3^k - y_2^k)$. For the first iteration x_0 and x_3 must be available. Then x_1 and x_2 for the first iteration can be calculated as: $x_1 = x_0 + h$,, $x_2 = x_1 + h$, where, $h = (x_3 - x_0)/3$. For all other iterations:

If $\left[f(x_0^k).f(x^{k+1}) \text{ OR } f(x_1^k).f(x^{k+1}) \right] < 0$, $x_3^{k+1} = x_2^k$, $x_2^{k+1} = x$, $x_1^{k+1} = x^{k+1}$, $x_0^{k+1} = x_0^k$. If $\left[f(x_0^k).f(x^{k+1}) \text{ OR } f(x_1^k).f(x^{k+1}) \right] \geq 0$, $x_0^{k+1} = x_1^k$, $x_1^{k+1} = x_2^k$, $x_2^{k+1} = x^{k+1}$, $x_3^{k+1} = x_3^k$.

The convergence criterion is: $\left| f(x^{k+1}) \right| \leq \text{TOL}$ where TOL is a specified tolerance.

3.0 Testing of the New Algorithm

The first test performed on the new method was by solving Leonardo's cubic equation. Leonardo's equation is $x^3 + 2x^2 + 10x - 20 = 0$

Given that a root of this equation lies between 1 and 1.5, application of the new Algorithm produces the following results. The calculations were made by a Texas instrument (TI) 68 calculator with a TOL = 10^{-8}

3.1 Initial Values Required

$x_0 = 1.00000000$, $y_0 = -7.000000000$
$x_1 = 1.16666667$, $y_1 = -4.023148142$
$x_2 = 1.33333333$, $y_2 = -0.740740748$
$x_3 = 1.50000000$, $y_3 = 2.857000000$

3.2 Number 1 Iteration

$x_0 = 1.666666667$, $y_0 = -4.023148142$
$x_1 = 1.333333333$, $y_1 = -0.740740748$
$x_2 = 1.368689055$, $y_2 = -0.000401939$
$x_3 = 1.500000000$, $y_3 = 2.857000000$

3.3 Number 2 Iteration

$x_0 = 1.333333333$, $y_0 = -0.740740748$
$x_1 = 1.368789055$, $y_1 = -0.000401939$
$x_2 = 1.368808107$, $y_2 = -0.000000017$
$x_3 = 1.500000000$, $y_3 = 2.8570000000$

3.4 Number 3 Iteration

$x_0 = 1.368789055$, $y_0 = -0.000401939$
$x_1 = 1.368808107$, $y_1 = -0.000000017$
$x_2 = 1.368808108$, $y_2 = -0.000000004$
$x_3 = 1.500000000$, $y_3 = 2.857000000$

In the second test, the new method was used to solve the Dranchuk, Purvis and Robinson equation. The Dranchuk, Purvis and Robinson equation is:

$1 + (A_1 + A_2/\ Tr + A_3/Tr^3)x + (A_4 + A_5/Tr)x^2 + A_5A_6x^5/Tr + (A_7x^2/Tr^3)(1 + A_8x^2)\ \exp(-A_8x^2) - 0.27Pr/\ (xTr) = 0$ and $z = 0.27Pr/(xTr)$, where

$A_1 = 0.31506237$, $A_2 = -1.04670990$

$A_3 = -0.57832729$, $A_4 = 0.53530771$
$A_5 = -0.61232032$, $A_6 = -0.10488813$
$A_7 = 0.68157001$, $A_8 = 0.68446549$

For this equation it is known that:

If $Pr \le 8.0$, then $0.27Pr/(1.2Tr) \le x \le 0.2Pr/(0.25Tr)$
If $Pr > 8.0$, then $0.27Pr/(1.8Tr) \le x \le 0.27Pr/(0.95Tr)$

The solution to the Dranchuk, Purvis and Robinson [2] equation by the new method was done in a personal computer with a Watfor program. The program is shown together with the output in the appendix. A tolerance of 10^{-3} was used in the program. The values of z read from the output are shown in table 1. These values may be compared with corresponding values from table A-2 Katz, D.L, Cornell, C., Kobashj, K., Poettmann, F. H, Vary, J. A., Elenbaas, J. R., and C. F. Weinaug (1959) [3], also shown in Table 1.

	Pr	Tr	Z from output	Z from Katz et al
1.	1.65	1.05	0.294	0.251
2.	3.00	2.00	0.938	0.938
3.	3.20	1.10	0.485	0.463
4.	7.70	1.60	0.985	0.985
5.	9.50	2.80	1.157	1.150
6.	15.00	1.10	1.710	1.720

1

2 **TABLE 1:** COMPARISON Z VALUES FROM THE PROGAMME WITH Z VALUES FROM KATZ ET AL

4.0 **Comparison of the Cubic Method with Other Methods**

The Regular Falsi Algorithm is:

$$x^{k+1} = x_0^k - y_0^k \frac{(x_1^k - x_0^k)}{(y_1^k - y_o^k)} \tag{4.0}$$

if $\left[y_0^k f\left(x^{k+1}\right)\right] < 0$, $x_0^{k+1} = x^{k+1}, x_0^{k+1} = x_0^k$
if $\left[y_0^k f\left(x^{k+1}\right)\right] \ge 0$, $x_0^{k+1} = x^{k+1}, x_1^{k+1} = x_1^k$

Excluding y_2 and y_3 in the numerator, and ignoring the brackets involving y_2 and y_3 in the denominator, the first two terms of equation (2.2) break down to the "Regular Falsi". Thus the new Algorithm with more terms can be expected to converge more rapidly than the "Regular Falsi." Indeed, given the same condition that a root of Leonardo's equation lies between 1 and 1.5, it takes the "Regular Falsi" 7 iterations to converge.

The Newton- Raphson Algorithm is:

$$x^{K+1} = x^k - \frac{f^k(x)}{f^{1k}(x)} \tag{4.1}$$

This scheme converges rapidly. However, B.L. Meek and S. Fairthorne (1977) [4] have observed that the scheme fails to converge when $f^1(x)$ is small and can converge to a wrong value when an inflection point lies close to the root of a non-linear equation. The cubic interpolation method presented in this work does not have any derivative and can be expected to overcome the small derivative problem associated with the Newton-Raphson method.

The solution to Leonardo's equation by the Newton-Raphson method; given that a root lies close to 1.0 can be laid out as follows:

4.1 **Initial Values Required**

x = 1.000000000, f(x) = -7.000000000, f^{1} (x) = 17.000000000

4.2 **Number 1 Iteration**

x = 1.411764706, f(x) = 0.01148128, f^{1}(x) = 21.62629758

4.3 **Number 2 Iteration**

x = 1.369336471, f(x) = 0.011148128, f^{1}(x) = 21.10259300

4.4 **Number 3 Iteration**

x = 1.36880819, f(x) = 0.000001713, f^{1}(x) = 21.09614033

4.5 **Number 4 Iteration**

x = 1.368808108, f(x) = 0.000000004, f^{1}(x) = 21.09613934

The computations were made by a Texas Instrument 68 (Tl 68) calculator with a specified tolerance of 10^{-8}. Comparing this result with that from equation (2), we observe that it took equation (2) one iteration less than the Newton - Raphson method. Indeed, the new method would have converged after only 2 iterations if a tolerance of 10^{-7} was used, while the Newton-Raphson would have used some 4 iterations.

5.0 Conclusion

1. The Algorithm developed here prove superior to the "Regular Falsi" and the Newton-Raphson method.
2. The computer program included in the paper is an efficient package for those that need a solution to the Dranchuk, Purvis and Robinson *equation* for calculating the Compressibility Factor of Natural Gases.

Appendix

```
C *   SOLUTION TO DRANCHUK Z FACTOR EQUATION BY THE OHIRHIAN
C *   METHOD
C
C
C    MAIN PROGRAM
C
      OPEN(UNIT = 5,FILE = 'ZEDF.IN')
      OPEN(UNIT = 6,FILE = 'ZEDF.OUT')
  1   READ(5,11,ERR = 36)PR,TR
      WRITE(6,50)PR,TR
      Z = ZEDF(PR,TR)
 50   FORMAT(//2X,'PR = ',F10.6,3X,'TR = ',F10.6)
      WRITE(6,55) Z
 55   FORMAT(/3X,'Z = ',F8.3,//2X,'**********************************')
      GOTO 1
 36   STOP
 11  FORMAT(F5.3,F5.3)
     END
C
```

```
C
C * THIS FUNCTION OBTAINS A VALUE OF THE GAS COMPRESSIBILITY
C * FACTOR Z FROM THE EQUATION OF DRANCHUK BY USE OF THE CUBIC
C * INTERPO-LATION ALGORITHM. IN USING THE METHOD IT IS ASSUMED
C * THAT A ROOT OF F(X) =0 LIES BETWEEN X0 AND X3 AND THESE VALUES
C * MUST BE SUPPLIED INTHE MAIN PROGRAM OR IN THE SUBPROGRAM.
C * OTHER DATA NEEDED BY THE PROGRAM TO RUN ARE THR VALUE OF
C * THE TOLRANCE (TOL) NEEDED FOR CONVERGENCE TEST AND THE
C * MAXIMUM NUMBER OF ITRERATIONS (MAXIT) THE USER ALLOWS THE
C * ALGORITHM TO USE
C
      FUNCTION ZEDF(PR,TR)
C
      OPEN(UNIT = 6,FILE = 'ZEDF.OUT')
C
C * SUPPLY THE VALUES OF MAXIT AND TOL
C
      DATA MAXIT,TOL/20,0.0001/


C
C * THE FOLLOWING COEFFICIENTS CAME FROM THE DRANCHUK EQUATION
C * THEY ARE NOT GENERALLY NEEDE FOR THE CUBIC ALGORITHM TO RUN
C
      DATA A1,A2,A3/.31506237,-1.04670990,-.57832729/
      DATA A4,A5,A6/.53530771,-.61232032,-.10488813/
      DATA A7,A8/.68157001,.68446549/
C
C * SUPPLY IHE VALUES OF XO AND X3 NEEDED FOR THE ALGORITHM TO
C * START
C
       IF(PR.LE.8.0)GOTO 10
       X0 = 0.27*PR/(.95*TR)
       X3 = 0.27*PR/(1.80*TR)
C
C*     CALCULATE THE INTERMEDIATE VALUES X1 AND X2
C
       H = X3 - X0
       X1 = X0 + H/3.0
       X2 = X1 + H/3.0
        GOTO 12
  10  X0 = 0.27*PR/(0.25*TR)
        X3 = 0.27*PR/(1.20*TR)
        H = X3 - X0
        X1 = X0 + H/3.0
        X2 = X1 + H/3.0
C
C*  SET THE INITIAL VALUE OF THE ITERATION COUNTER(K) = 0
C
  12   K = 0
C
C
C * CALCULATE THE  INITIAL VALUES OF THE DEPENDENDENT VARIABLE(Y)
C * (THAT IS Y0,Y1,Y2AND Y3 ) THAT CORRESPOND TO X0,X1,X2 AND X3
C
C
```

```
C* CALCULATE Y0
C
      Y0 = 1. + (A1+A2/TR+A3/TR**3)*X0 + (A4 + A5/TR)*X0**2 + A5*A6*X0*
     X*5/TR + (A7*X0**2/TR**3)*(1.+A8*X0**2)*EXP(-A8*X0**2) - .27*PR/(X0
     X*TR)
C
C
C * CALCULATE Y1
C
      Y1 = 1. + (A1+A2/TR+A3/TR**3)*X1 + (A4 + A5/TR)*X1**2 + A5*A6*X1*
     X*5/TR + (A7*X1**2/TR**3)*(1.+A8*X1**2)*EXP(-A8*X1**2) - .27*PR/(X1
     X*TR)
C
C* CALCUKATE Y2

C
      Y2 = 1. + (A1+A2/TR+A3/TR**3)*X2 + (A4 + A5/TR)*X2**2 + A5*A6*X2*
     X*5/TR + (A7*X2**2/TR**3)*(1.+A8*X2**2)*EXP(-A8*X2**2) - .27*PR/(X2
     X*TR)



C
C* CALCULATE Y3
C
      Y3 = 1. + (A1+A2/TR+A3/TR**3)*X3 + (A4 + A5/TR)*X3**2 + A5*A6*X3*
     X*5/TR + (A7*X3**2/TR**3)*(1.+A8*X3**2)*EXP(-A8*X3**2) - .27*PR/(X3
     X*TR)
C
C * WRITE CA LCULATED VALUES OF (X0,Y0); (X1,Y1);(X2,Y2)AND (X3,Y3)
C
      WRITE(6,35)X0,Y0,X1,Y1,X2,Y2,X3,Y3
C
C * HERE, WE TEST THAT : IF THE INITIAL VALUES OF THE INDEPENDENT
C * VARIABLE (X) WERE SUCH THAT Y = F(X) BECAME APPROXIMATELY
C * EQUAL TO ZERO THEN EITHER X0,X1,X2 OR X3 IS A ROOT OF F(X) = 0
C * AND COMPUTATIONS END HERE
C
      IF(ABS(Y0).LE.TOL)GOTO 1
      IF(ABS(Y1).LE.TOL)GOTO 2
      IF(ABS(Y2).LE.TOL)GOTO 3
      IF(ABS(Y3).LE.TOL)GOTO 4
      GOTO 15
C
C * IN THE CASE OF THE DRANCHUK EQUATION ,ANOTHER VARIABLE Z
C * WHICH  IS DEPENDENT ON THE ROOT OF F(X)= 0 WAS WHAT IS NEEDED .
C * SO WE CAL CULATE IT HERE AND THE JOB IS FINISHED
C
  1   ZEDF = 0.27*PR/(X0*TR)
      RETURN
  2   ZEDF = 0.27*PR/(X1*TR)
      RETURN
  3   ZEDF = 0.27*PR/(X2*TR)
      RETURN
  4   ZEDF = 0.27*PR/(X3*TR)
      RETURN
```

```
C
C * BEGIN ITERATION IF NONE OF THE PREVIOUS VALUES OF X(THAT IS
C * X0, X1,X2,X3) WAS THE EXPECTED ROOT OF F(X) = 0
C * USE THE VALUES X0,X1,X2,X3 AND CORRESPONDING VALUES OF
C * Y0,Y1,Y2,Y3  TO OBTAIN A BETTER VALUE OF X BY USE OF THE CUBIC
C * INTER POLATION ALGORITHM
C
15   DO 25 K = 1,MAXIT
C
C * CALCULATE COEFICIENTS A,B,C,D NEEDED IN THE ALGORITHIM
C
     A = (Y0 - Y1)*(Y0 - Y2)*(Y0 - Y3)
     B = (Y1 - Y0)*(Y1 - Y2)*(Y1 - Y3)
     C = (Y2 - Y0)*(Y2 - Y1)*(Y2 - Y3)
     D = (Y3 - Y0)*(Y3 - Y1)*(Y3 - Y2)
C
C * WRITE THE CALCULATED VALUES OF THE COEFFICIETS A,B,C.
C
       WRITE(6,155)A,B,C,D
 155  FORMAT(3X,'A = ',E12.6,2X,'B = ',E12.6,2X,'C = ',E12.6,2X,'D = ',E
     X12.6)



C
C * IT IS HERE THAT THE BEVALUE OF X IS ACTUALLY CALCULATED
C
      X = -Y1*Y2*Y3*X0/A - Y0*Y2*Y3*X1/B - Y0*Y1*Y3*X2/C - Y0*Y1*Y2*X3/D
C
C * THE NEW VALUE OF Y CORRESPONDING TO THE NEW X IS CALCULATED
C * HRRE
C
     Y = 1. + (A1+A2/TR+A3/TR**3)*X + (A4 + A5/TR)*X**2 + A5*A6*X*
    X*5/TR + (A7*X**2/TR**3)*(1.+A8*X**2)*EXP(-A8*X**2) - .27*PR/(X
    X*TR)
C
C * WRITE THE NEW VALUES OF X AND Y
C
      WRITE(6,250) X, Y
  250 FORMAT(3X,'X = ',F10.6,4X,'Y = ',F10.6/)
C
C * CHECK THAT THE NEW ABSOLUTE VALUE OF THE NEW Y IS CLOSE TO
C * ZERO .  IF IT IS CLOSE TO ZERO THE NEW X IS THE EXPECTED ROOT OF
C * F(X) = 0  OTHERWISE OBTAIN A BETTER VALUE OF X BY USE OF THE
C * CUBIC ALGORITHM
C
       IF(ABS(Y).LE.TOL)GOTO 45
       IF(Y0*Y.LT.0.OR.Y1*Y.LT.0)GOTO 30
       X0 = X1
       Y0 = Y1
       X1 = X2
       Y1 = Y2
       X2 = X
       Y2 = Y
C
       WRITE(6,40)K,X0,Y0,X1,Y1,X2,Y2,X3,Y3
```

```
        GOTO 25
C
   30 X3 = X2
       Y3 = Y2
       X2= X1
       Y2 = Y1
       X1 = X
       Y1 = Y
       WRITE(6,40)K,X0,Y0,X1,Y1,X2,Y2,X3,Y3
C
  25   CONTINUE
       WRITE(6,60)
        RETURN
  45    ZEDF = 0.27*PR/(X*TR)
        RETURN
C
C
  35   FORMAT(/2X,'INITIAL VALUES REQUIRED'/2X,'X0 = ',F10.6,3X,'Y0 = ',
    XF10.6/2X,'X1 = ',F10.6,3X,'Y1 = ',F10.6,/2X,'X2 = ',F10.6,3X,'Y2 =
    X ',F10.6,/2X,'X3 = ',F10.6,3X,'Y3 = ',F10.6)
C
  40  FORMAT(2X,'NUMBER',1X,I3,1X,'ITERATION'/2X,'X0 = ',F10.6,3X,'Y0 =
    X',F10.6/2X,'X1 = ',F10.6,3X,'Y1 = ',F10.6,/2X,'X2 = ',F10.6,3X,'Y2
    X = ',F10.6,/2X,'X3 = ',F10.6,3X,'Y3 = ',F10.6)


C
  60  FORMAT(2X,'NO CONVERGENCE AFTER 20 ITERATIONS')
C
      END
```

References

[1] Carnhan, B. Luther, H.A. and Wilkes, J.O. 1969. Applied Numerical Methods. New York.
John Willey & Sons.

[2] Dranchuk, P.M, Purvis, R.A. and D.B. Robinson (1974) : “Cumputer Calculation of atural Gases Compressibility Factor using the Standing and Katz Correlation “.Institute of Petroleum IP 74-008

[3] Katz, D.L, Cornell ,C., Kobashj , K.., Poettmann , F.H, Vary , J. A. , Elenbaas ,J. R., and C.F., Weinaug (1959), 1959 Handbook of Natural Gas Engineering , New York., McGraw-Hill Book Company.

[4] Meek, B.L. and Faithorne, S. 1977. Using Computers. London. Ellis Horwood Publishers

[5] Ohirhian, P. U. 1994. "Iterative Solution of Non-Linear Equations", proceedings of SPE/IACMAG International Conference, Morgantown.

[6] Shied, F. 1968, Numerical Analysis, New York, McGraw-Hill Book Company.

[7] Spiegel, M.R. 1971. Finite Differences and Difference Equations, New York, McGraw-Hill Book Company