Volume 2 ▪ Issue 2 ▪ August 2011

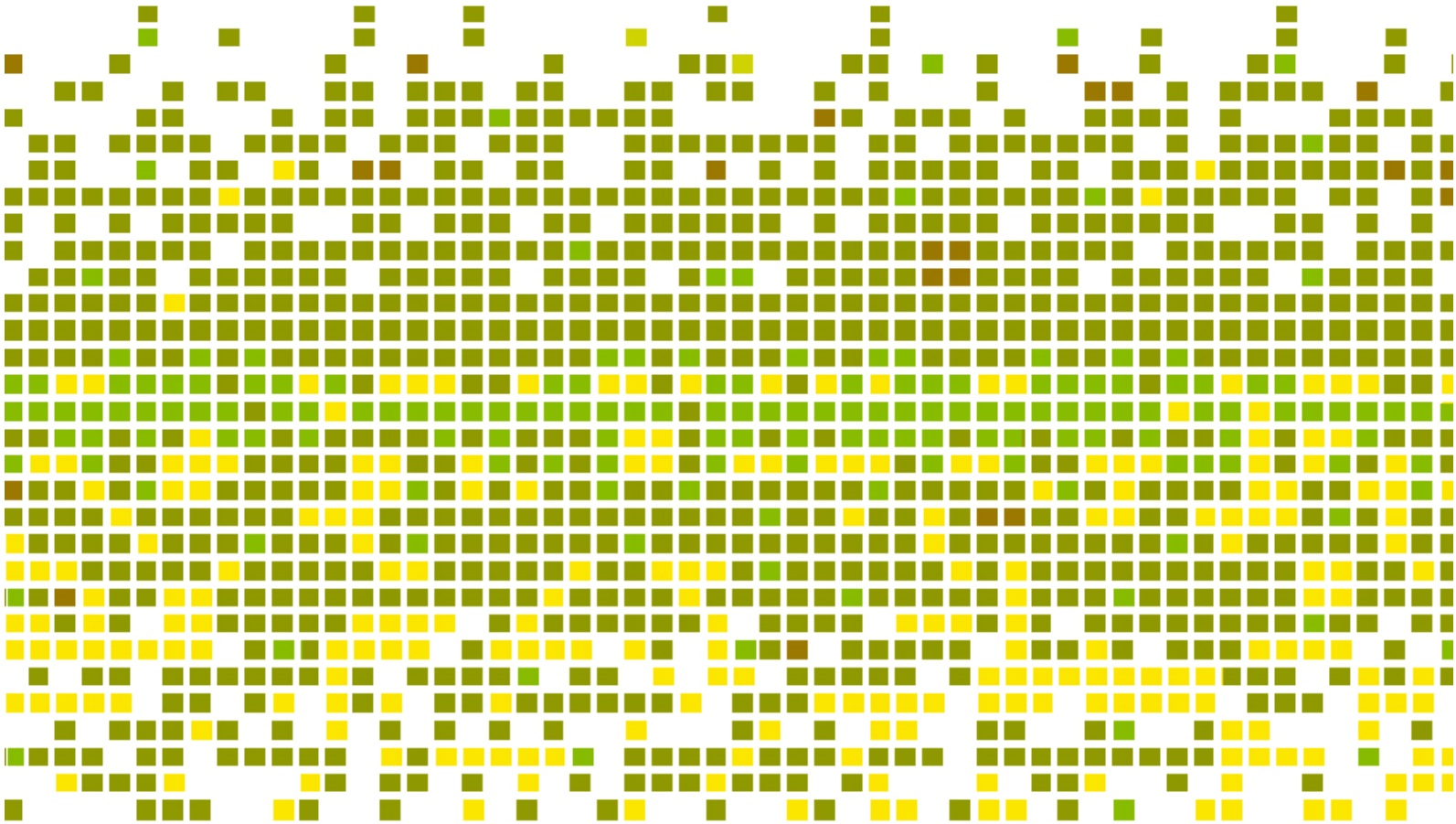
Editor-in-Chief
Professor. HALIL R. OZ

INTERNATIONAL JOURNAL OF

EXPERIMENTAL ALGORITHMS (IJEA)

ISSN : 2180-1282

Publication Frequency: 6 Issues / Year



CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF EXPERIMENTAL ALGORITHMS (IJEA)

VOLUME 2, ISSUE 2, 2011

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180-1282

International Journal of Experimental Algorithms (IJEA) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJEA Journal is a part of CSC Publishers

Computer Science Journals

http://www.cscjournals.org

INTERNATIONAL JOURNAL OF EXPERIMENTAL ALGORITHMS (IJEA)

Book: Volume 2, Issue 2, August 2011
Publishing Date: 31-08-2011
ISSN (Online): 1985-4129

IJEA Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2011

EDITORIAL BOARD

TABLE OF CONTENTS

Volume 2, Issue 2, August 2011

Pages

Mathematical Derivation of Annuity Interest Rate and its Application

K.A.Fayed *karamfayed_1@hotmail.com*
Ph.D.From Dept. of applied Mathematics and Computing,
Cranfield University, UK.
Faculty of commerce/Dept. of applied Statistics and Computing,
Port Said University, Port Fouad, Egypt.

Abstract

A fundamental task in business for investor or borrower is to know the interest rate of an annuity. In this type of problem, the size of each periodic payment(R), the term(n), and the amount(Sn) or the present value of the annuity(An) are usually given.However, a direct equation representing the Annuity Interest Rate(i) is not available, since an approximate value of the Annuity Interest Rate is obtained by interpolation methodbased on table showing (Sn/R) values. This paper emphasizes the real time computational problem for Annuity interest rate. It has therefore been important to derive an equation for computing the Annuity Interest rate. The evaluation of error analysis has been discussed. The new algorithm saved computational energy by approximately 99.9%than that of the tabulated one.

Keywords: Investment Mathematics, Statistical Toolbox**,** MATLAB Programming.

1. INTRODUCTION

There are many situations in which both businesses and individuals would be faced with either receiving or paying a constant amount for a length of period. When a firm faces a stream of constant payments on a bank loan for a period of time, we call that stream of cash flows an annuity.

An annuity is a series of periodic payments, usually made in equal amounts. The payments are computed by the compound interest method[1] and are made at equal intervals of time. Individual investors may make constant payments on their home or car loans, or invest a fixed amount year after year to save for their retirement. Any financial contract that calls for equally spaced and level cash flows over a finite number of periods is called an annuity. If the cash flow payments continue forever, the contract is called perpetuity. Constant cash flows that occur at the end of each period are called ordinary annuities.

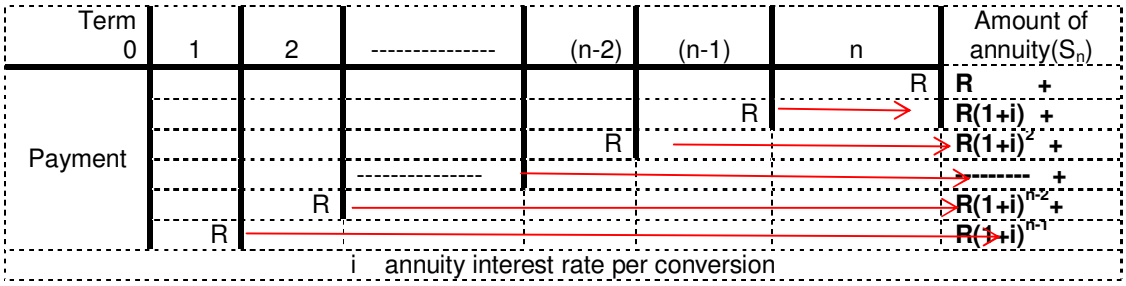
2. THE AMOUNT OF AN ANNUITY

In Business, the amount of an annuity is the final value at the end of the term of the annuity. To derive the formula for the amount of an ordinary annuity,
let:

R is the size of each regular payment.
i is the interest rate per conversion period.
n is the number of payments during the term of an annuity.
S_n is the amount of an ordinary annuity.

Then:

The amount of an ordinary annuity is given by:



$$S_n = R + R(1+i) + R(1+i)^2 + \cdots + R(1+i)^{n-2} + R(1+i)^{n-1} \quad (1)$$

Multiply both sides by (1+i), we have:

$$S_n(1+i) = R(1+i) + R(1+i)^2 + \cdots + R(1+i)^{n-1} + R(1+i)^n \quad (2)$$

Subtracting Eq.(2) from Eq.(1), we get:

$$S_n(1+i) - S_n = R(1+i)^n - R$$

$$S_n(1+i-1) = R[(1+i)^n - 1]$$

$$S_n = R\frac{[(1+i)^n - 1]}{i} \quad (3)$$

3. THE PRESENT VALUE OF AN ANNUITY

The present value of an annuity is the value at the beginning of the term of the annuity. The present value of an annuity can be derived by the same way to get the following formula:

$$A_n = R\frac{[1-(1+i)^{-n}]}{i} \quad (4)$$

Where:

An is the present value of an ordinary annuity.

4. ANNUITY INTEREST RATE PER CONVERSION(i)

The annuity equation (Eq.3 or Eq.4)can also be used to the find the interest rate or discount rate for an annuity.To determine an accurate valueof the Annuity interest rate instead of using a trial-and-error approach, we need to solve the equation for the unknown value i as follow:

a) When the Amount is Known (S_n)

(1) Two_Term Simplification

To find the annuity interest rate when the amount is known, use the Eq.(3) as follow:

$$\frac{S_n}{R}i = (1+i)^n - 1 \quad (5)$$

From eq.(5), the term$(1+i)^n$can be simplified using binomial theorem, since it can obtain the binomial series which is valid for any real number $n\ if\ |x| < 1$ as follow:

$$(1+x)^n = 1 + nx + \frac{n(n-1)}{2!}x^2 + \frac{n(n-1)(n-2)}{3!}x^3 + \cdots] \quad (6)$$

The term$(1+i)^n$can be rewritten in the following form:

By replacing$x$ $by\ i$, we have:

$$(1+i)^n = 1 + ni + \frac{n(n-1)}{2!}i^2 + \frac{n(n-1)(n-2)}{3!}i^3 + \cdots] \quad (7)$$

From eq.(5) & the two ith term expansion of eq.(7), We have:

$$\frac{S_n}{R}i = (1+i)^n - 1$$

$$= -1 + (1+i)^n$$

$$= -1 + 1 + ni + \frac{n(n-1)}{2!}i^2$$

$$= ni + \frac{n(n-1)}{2}i^2$$

Dividing both sides by *i*, we get:

$$\frac{S_n}{R} = n + \frac{n(n-1)}{2}i$$

$$2\left(\frac{S_n}{R}-n\right)=n(n-1)i$$

$$\therefore i=\frac{2\left(\frac{S_n}{R}-n\right)}{n(n-1)} \quad ,\forall \quad \frac{S_n}{R}>n \qquad (8)$$

Therefore:
Equation(8) represents the annuity interest rate equation for computing*i*after the two i[th]term expansion.

(2) Three_Term Simplification

From eq.(5) & the threei[th] term expansion of eq.(7), We have:

$$\frac{S_n}{R}i-(1+i)^n-1$$
$$=-1+(1+i)^n$$
$$=-1+1+n\,i+\frac{n(n-1)}{2!}i^2+\frac{n(n-1)(n-2)}{3!}i^3$$
$$=n\,i+\frac{n(n-1)}{2}i^2+\frac{n(n-1)(n-2)}{6}i^3$$

Dividing both sides by *i*, we get:

$$\frac{S_n}{R}=n+\frac{n(n-1)}{2}i+\frac{n(n-1)(n-2)}{6}i^2$$
$$\left(n-\frac{S_n}{R}\right)+\frac{n(n-1)}{2}i+\frac{n(n-1)(n-2)}{6}i^2-0$$
$$n(n-1)(n-2)i^2+3n(n-1)i+6\left(n-\frac{S_n}{R}\right)=0$$
$$n(n-1)(n-2)i^2+3n(n-1)i+6\left(n-\frac{S_n}{R}\right)=0$$
$$i^2+\frac{3}{(n-2)}i+\frac{6\left(n-\frac{S_n}{R}\right)}{n(n-1)(n-2)}=0$$

Solving the above quadratic equation for *i*, we get:

$$i=\frac{-\frac{3}{n-2}+\sqrt{(\frac{3}{n-2})^2-4\frac{6\left(n-\frac{S_n}{R}\right)}{n(n-1)(n-2)}}}{2}$$

Simplifying the above equation, we get:

$$\therefore i=\frac{1}{2(n-2)}\left[\left(9-\frac{24(n-2)\left(n-\frac{S_n}{R}\right)}{n(n-1)}\right)^{\frac{1}{2}}-3\right] \quad ,\forall \quad \frac{S_n}{R}>n \qquad (9)$$

Therefore, equation(9) represents the annuity interest rate equation for computing *i* after the threei[th] term expansion.

b) When the Present Value is Known (A_n)

(1) Two_Term Simplification:

Using Eq.(4), we can get the following formulae:

$$\frac{A_n}{R}i=1-(1+i)^{-n} \qquad (10)$$

From eq.(10) & the two i[th] term expansion of eq.(7), We have:

$$\frac{A_n}{R}i=1-\left[1+(-n)\,i+\frac{[(-n)(-n+1)]}{2!}i^2\right]$$
$$=1-\left[1-n\,i+\frac{n(n-1)}{2}i^2\right]$$
$$=1-1+n\,i-\frac{n(n-1)}{2}i^2$$

$$= n\,i - \frac{n(n-1)}{2} i^2$$

Dividing both sides by *i*, we get:

$$\frac{A_n}{R} = n - \frac{n(n-1)}{2} i$$

$$2\left(n - \frac{A_n}{R}\right) = n(n-1)i$$

$$\therefore i = \frac{2\left(n - \frac{A_n}{R}\right)}{n(n-1)} \quad ,\forall \quad \frac{A_n}{R} < n \qquad (11)$$

Therefore, equation(11) represents the annuity interest rate equation for computing *i* after the two i[th] term expansion.

(2) Three_Term Simplification

From eq.(10) & the three i[th] term expansion of eq.(7), We have:

$$\frac{A_n}{R} i = 1 - (1+i)^{-n}$$

$$= 1 - \left[1 + (-n)\,i + \frac{(-n)(-n+1)}{2!} i^2 + \frac{(-n)(-n+1)(-n+2)}{3!} i^3\right]$$

$$= 1 - \left[1 - n\,i + \frac{n(n-1)}{2} i^2 - \frac{n(n-1)(n-2)}{6} i^3\right]$$

$$= n\,i - \frac{n(n-1)}{2} i^2 + \frac{n(n-1)(n-2)}{6} i^3$$

Dividing both sides by *i*, we get:

$$\frac{A_n}{R} - n - \frac{n(n-1)}{2} i + \frac{n(n-1)(n-2)}{6} i^2$$

$$\left(n - \frac{A_n}{R}\right) - \frac{n(n-1)}{2} i + \frac{n(n-1)(n-2)}{6} i^2 = 0$$

$$n(n-1)(n-2)i^2 - 3n(n-1)i + 6\left(n - \frac{A_n}{R}\right) = 0$$

$$i^2 - \frac{3}{(n-2)} i + \frac{6\left(n - \frac{A_n}{R}\right)}{n(n-1)(n-2)} = 0$$

Solving the above quadratic equation for *i*, we get:

$$i = \frac{\frac{3}{n-2} - \sqrt{\left(\frac{-3}{n-2}\right)^2 - 4\frac{6\left(n - \frac{A_n}{R}\right)}{n(n-1)(n-2)}}}{2}$$

Simplifying the above equation, we get:

$$\therefore i = \frac{1}{2(n-2)}\left[3 - \left(9 - \frac{24(n-2)\left(n - \frac{A_n}{R}\right)}{n(n-1)}\right)^{\frac{1}{2}}\right] \quad ,\forall \quad \frac{A_n}{R} < n \qquad (12)$$

Therefore, equation(12) represents the annuity interest rate equation for computing *i* after the three i[th] term expansion.

5. CALCULATION OF ANNUITY INTEREST RATE

a) Tabulated Annuity Interest Rate

(1) Known Amount:

Table_1 includes selection of annuity interest rate used in the investment market. The ratio in Table_1 has been computed for given values of conversion period(n) and the corresponding annuity interest rate. This ratio is used back to extract the annuity interest rate(i_tabulated) from Tables given in [1].

i% exact	Time period (n)			
	n=10		n=20	
	i_tabulated	S_n/R	i_tabulated	S_n/R
0.2	0.249	10.0904816840387	0.248	20.3845990093093
0.4	0.416	10.1819335047275	0.416	20.7785540890338
0.6	0.624	10.2743656882306	0.584	21.1821069182341
0.8	0.872	10.3677885591048	0.868	21.5955054350601
1	1.000	10.4622125411205	1.000	22.0190039947967
1.2	1.247	10.5576481580867	1.133	22.4528635317327
1.4	1.493	10.6541060346834	1.268	22.8973517249426
1.6	1.623	10.7515968972984	1.515	23.3527431680687
1.8	1.868	10.8501315748704	1.758	23.8193195431968
2	2.000	10.9497209997379	2.000	24.2973697989177
2.2	2.244	11.0503762084931	2.238	24.7871903326693
2.4	2.488	11.1521083428429	2.475	25.2890851774580
2.6	2.511	11.2549286504744	2.525	25.8033661930578
2.8	2.756	11.3588484859271	2.764	26.3303532617892
3	3.000	11.4638793114707	3.000	26.8703744889805
3.2	3.242	11.5700326979890	3.233	27.4237664082190
3.4	3.483	11.6773203258690	3.464	27.9908741914986
3.6	3.516	11.7857539858976	3.536	28.5720518643747
3.8	3.758	11.8953455801620	3.769	29.1676625262402
4	4.000	12.0061071229586	4.000	29.7780785758355
4.2	4.037	12.1180507417060	4.084	30.4036819421117
4.4	4.479	12.2311886778653	4.453	31.0448643205664
4.6	4.520	12.3455332878658	4.547	31.7020274151745
4.8	4.953	12.4610970440374	4.895	32.3755831860388
5	5.000	12.5778925355488	5.000	33.0659541028884

TABLE 1: Computing the ratio S_n/R and tabulated annuity interest rate

(2) Known Present Value

Similarly, Table_2 computes the ratio for given values of conversion period(n) and the corresponding annuity interest rate. This ratio is used back to extract the annuity interest rate(i_tabulated) from Tables given in [1].

i% exact	Time period (n)			
	n=10		n=20	
	i_tabulated	A_n/R	i_tabulated	A_n/R
0.2	0.251	9.89087431187258	0.251	19.5860898344387
0.4	0.332	9.78347474743335	0.331	19.1840839823320
0.6	0.626	9.67776811620015	0.627	18.7935810581347
0.8	0.748	9.57372195913692	0.746	18.4141947010670
1	1.000	9.47130453070169	1.000	18.0455529662705
1.2	1.253	9.37048478137687	1.256	17.6872977422976
1.4	1.373	9.27123234066807	1.372	17.3390841937310
1.6	1.764	9.17351750055746	1.776	17.0005802277864
1.8	1.745	9.07731119939899	1.742	16.6714659838048
2	2.000	8.98258500624224	2.000	16.3514333445971
2.2	2.256	8.88931110557294	2.261	16.0401854686493
2.4	2.513	8.79746228245785	2.524	15.7374363422453
2.6	2.487	8.70701190808258	2.477	15.4429103506104
2.8	2.743	8.61793392567109	2.737	15.1563418672197
3	3.000	8.53020283677584	3.000	14.8774748604555
3.2	3.258	8.44379368792813	3.265	14.6060625168388
3.4	3.518	8.35868205763838	3.532	14.3418668800934
3.6	3.778	8.27484404373630	3.801	14.0846585053389
3.8	4.040	8.19225625104152	4.072	13.8342161277393
4	4.000	8.11089577935504	4.000	13.5903263449677
4.2	3.961	8.03074021176281	3.930	13.3527833128750
4.4	4.522	7.95176760324237	4.539	13.1213884537818
4.6	4.478	7.87395646956413	4.461	12.8959501768371
4.8	5.049	7.79728577647907	5.086	12.6762836099142
5	5.000	7.72173492918482	5.000	12.4622103425400

TABLE 2: Computing the ratio A_n/R and tabulated annuity interest rate

b) Simplified Annuity Interest Rate

(1) Known Amount

Table_3 computes annuity interest rate derived in Eq.(9), Eq.(10), Eq.(11), and Eq.(12) respectively. These Calculations have been computed for given values of S_n/R (Table_3_a) or A_n/R (Table_3_b) in addition to different conversion period(n).

Figure_1 shows the tabulated annuity interest rate, the exact annuity interest and the simplified one against different annuity interest rate. This figure indicates that the simplified annuity interest rate moves smoothly without any abrupt change or fluctuations. On the other hand, the tabulated annuity interest rate moves irregularly along with different interest rate. This variation reverses a wide range of errors associated with the tabulated calculation of annuity interest rate.

i% exact	i_Tabulated	Mathematical formula		Absolute Relative Error(ARE)%		
		i_2_Term	i_3_Term	Tabulated	2_Term	3_Term
0.2	0.249	0.2010	0.20000	24.718	0.53	0.001
0.4	0.416	0.4042	0.40002	4.088	1.07	0.007
0.6	0.624	0.6097	0.60009	4.048	1.61	0.016
0.8	0.872	0.8173	0.8002	9.003	2.16	0.028
1	1.000	1.0271	1.0004	1.07e-08	2.71	0.044
1.2	1.247	1.2392	1.2007	3.930	3.26	0.064
1.4	1.493	1.4535	1.4012	6.656	3.82	0.086
1.6	1.623	1.6702	1.6017	1.446	4.38	0.112
1.8	1.868	1.8891	1.8025	3.810	4.95	0.140
2	2.000	2.1104	2.0034	0.00019	5.52	0.172
2.2	2.244	2.3341	2.2045	2.038	6.09	0.207
2.4	2.488	2.5602	2.4058	3.690	6.67	0.245
2.8	2.756	3.0196	2.8092	1.559	7.84	0.329
3	3.000	3.2530	3.0112	0.00069	8.43	0.375
3.2	3.242	3.4889	3.2135	1.328	9.03	0.424
3.4	3.483	3.7273	3.4161	2.467	9.62	0.475
3.6	3.516	3.9683	3.6190	2.326	10.23	0.529
3.8	3.758	4.2118	3.8222	1.086	10.83	0.586
4	4.000	4.4580	4.0258	0.000059	11.45	0.645
4.2	4.037	4.7067	4.2297	3.873	12.06	0.707
4.4	4.479	4.9581	4.4339	1.798	12.68	0.771
4.6	4.520	5.2122	4.6385	1.717	13.31	0.837
4.8	4.953	5.4691	4.8435	3.200	13.93	0.906
5	5.000	5.7286	5.0488	0.00073	14.57	0.977

TABLE 3: a)Annuity Interest Rate at known S_n/R and time period n=10

i% exact	i_Tabulated	Mathematical formula		Absolute Relative Error(ARE)%		
		i_2_Term	i_3_Term	Tabulated	2_Term	3_Term
0.2	0.248	0.20	0.20	24.40	1.21	0.01
0.4	0.416	0.40	0.40	4.00	2.44	0.04
0.6	0.584	0.62	0.60	2.62	3.69	0.09
0.8	0.868	0.83	0.80	8.58	4.96	0.15
1	1.000	1.06	1.00	0.00	6.26	0.24
1.2	1.133	1.29	1.20	5.56	7.58	0.33
1.4	1.268	1.52	1.41	9.39	8.92	0.45
1.6	1.515	1.76	1.60	5.32	10.28	0.58
1.8	1.758	2.01	1.81	2.29	11.67	0.72
2	2.000	2.26	2.017	0.00	13.08	0.88
2.2	2.238	2.51	2.22	1.76	14.52	1.05
2.4	2.475	2.78	2.43	3.12	15.98	1.23
2.8	2.764	3.33	2.84	1.29	18.99	1.64
3	3.000	3.61	3.05	0.00	20.53	1.86
3.2	3.233	3.90	3.26	1.05	22.10	2.09
3.4	3.464	4.20	3.47	1.89	23.69	2.33
3.6	3.536	4.51	3.69	1.77	25.32	2.59
3.8	3.769	4.82	3.90	0.80	26.97	2.86
4	4.000	5.14	4.12	0.00	28.65	3.13
4.2	4.084	5.47	4.34	2.76	30.37	3.42
4.4	4.453	5.81	4.56	1.21	32.11	3.72
4.6	4.547	6.15	4.78	1.14	33.89	4.02
4.8	4.895	6.51	5.01	1.99	35.69	4.34
5	5.000	6.87	5.23	0.00	37.53	4.67

TABLE 3: b)Annuity Interest Rate at known S_n/R and time period n=20

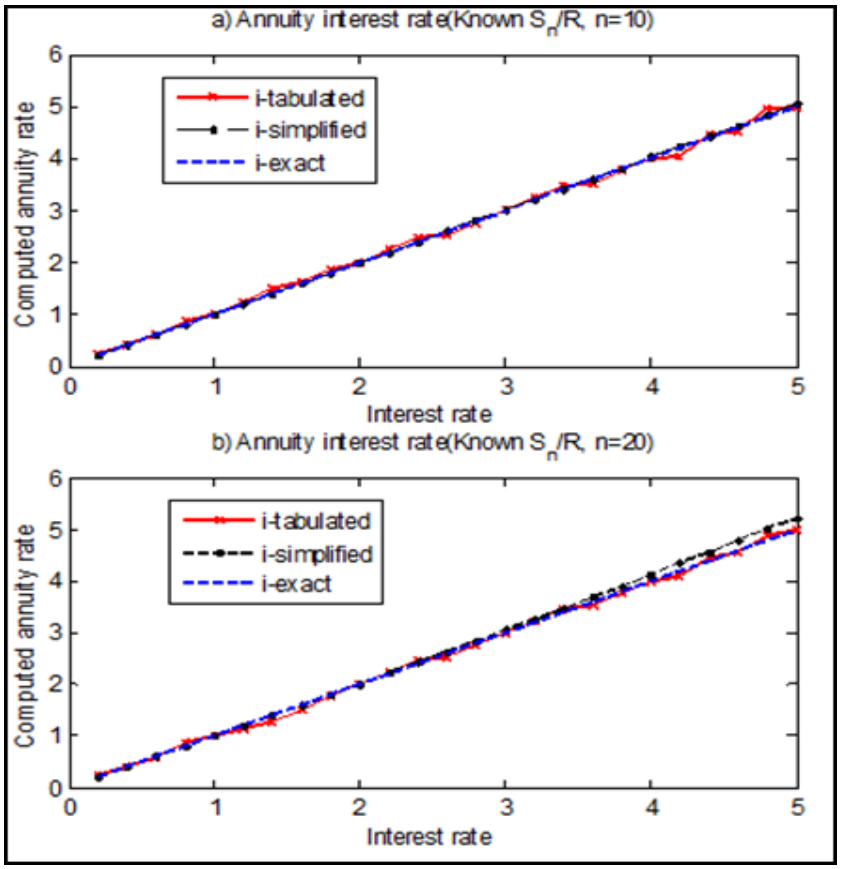


FIGURE 1: Tabulated and Simplified annuity interest rate (known S_n/R)

(2) Known Present Value:

The tabulated annuity interest rate and the simplified one are shown in Table_4. These Calculations have been computed for given values of A_n/R in addition to different conversion period(n).

Figure_2 shows the tabulated annuity interest rate, the exact annuity interest and the simplified one against different annuity interest rate for known values of A_n/R. This figure indicates that the simplified annuity interest rate moves smoothly without any abrupt change or fluctuations. On the other hand, the tabulated annuity interest rate moves irregularly along with different interest rate. This variation reverses a wide range of errors associated with the tabulated calculation of annuity interest rate.

i% exact	i_Tabulated	Mathematical formula		Absolute Relative Error(ARE)%		
		i_2_Term	i_3_Term	Tabulated	2_Term	3_Term
0.2	0.251	0.243	0.244	25.343	21.251	22.045
0.4	0.332	0.481	0.488	16.968	20.292	21.876
0.6	0.626	0.716	0.730	4.309	19.345	21.715
0.8	0.748	0.947	0.973	6.502	18.411	21.563
1	1.000	1.175	1.214	0.000	17.488	21.419
1.2	1.253	1.399	1.455	4.445	16.577	21.284
1.4	1.373	1.619	1.696	1.915	15.677	21.158
1.6	1.764	1.837	1.937	10.244	14.789	21.040
1.8	1.745	2.050	2.177	3.033	13.912	20.932
2	2.000	2.261	2.417	0.001	13.046	20.833
2.2	2.256	2.468	2.656	2.539	12.191	20.744
2.4	2.513	2.672	2.896	4.708	11.346	20.664
2.8	2.743	3.071	3.375	2.037	9.688	20.536
3	3.000	3.266	3.615	0.000	8.874	20.488
3.2	3.258	3.458	3.854	1.822	8.070	20.450
3.4	3.518	3.647	4.094	3.462	7.276	20.424
3.6	3.778	3.834	4.335	4.955	6.491	20.410
3.8	4.040	4.017	4.575	6.320	5.716	20.407
4	4.000	4.198	4.817	0.001	4.950	20.417
4.2	3.961	4.376	5.058	5.702	4.194	20.440
4.4	4.522	4.552	5.301	2.778	3.446	20.477
4.6	4.478	4.725	5.544	2.653	2.707	20.527
4.8	5.049	4.895	5.788	5.186	1.978	20.592
5	5.000	5.063	6.034	0.000	1.256	20.672

TABLE 4: a)Annuity Interest Rate at known $A_n/R$and time period n=10

i% exact	i_Tabulated	Mathematical formula		Absolute Relative Error(ARE)%		
		i_2_Term	i_3_Term	Tabulated	2_Term	3_Term
0.2	0.251	0.218	0.221	25.652	8.924	10.386
0.4	0.331	0.429	0.441	17.239	7.357	10.276
0.6	0.627	0.635	0.661	4.434	5.826	10.198
0.8	0.746	0.835	0.881	6.725	4.329	10.154
1	1.000	1.029	1.101	0.000	2.866	10.145
1.2	1.256	1.217	1.322	4.688	1.434	10.174
1.4	1.372	1.400	1.543	2.028	0.034	10.243
1.6	1.776	1.579	1.766	10.993	1.335	10.357
1.8	1.742	1.752	1.989	3.250	2.674	10.517
2	2.000	1.920	2.215	0.000	3.985	10.728
2.2	2.261	2.084	2.442	2.763	5.268	10.995
2.4	2.524	2.243	2.672	5.158	6.523	11.323
2.8	2.737	2.549	3.141	2.243	8.954	12.193
3	3.000	2.696	3.383	0.001	10.131	12.752
3.2	3.265	2.839	3.629	2.029	11.284	13.411
3.4	3.532	2.978	3.882	3.879	12.413	14.185
3.6	3.801	3.113	4.143	5.579	13.518	15.095
3.8	4.072	3.245	4.414	7.152	14.601	16.166
4	4.000	3.374	4.698	0.000	15.662	17.438
4.2	3.930	3.499	4.996	6.426	16.702	18.960
4.4	4.539	3.620	5.316	3.165	17.720	20.813
4.6	4.461	3.739	5.663	3.016	18.718	23.119
4.8	5.086	3.855	6.053	5.956	19.696	26.098
5	5.000	3.967	6.510	0.000	20.655	30.206

TABLE 4: b)Annuity Interest Rate at known $A_n/R$and time period n=20

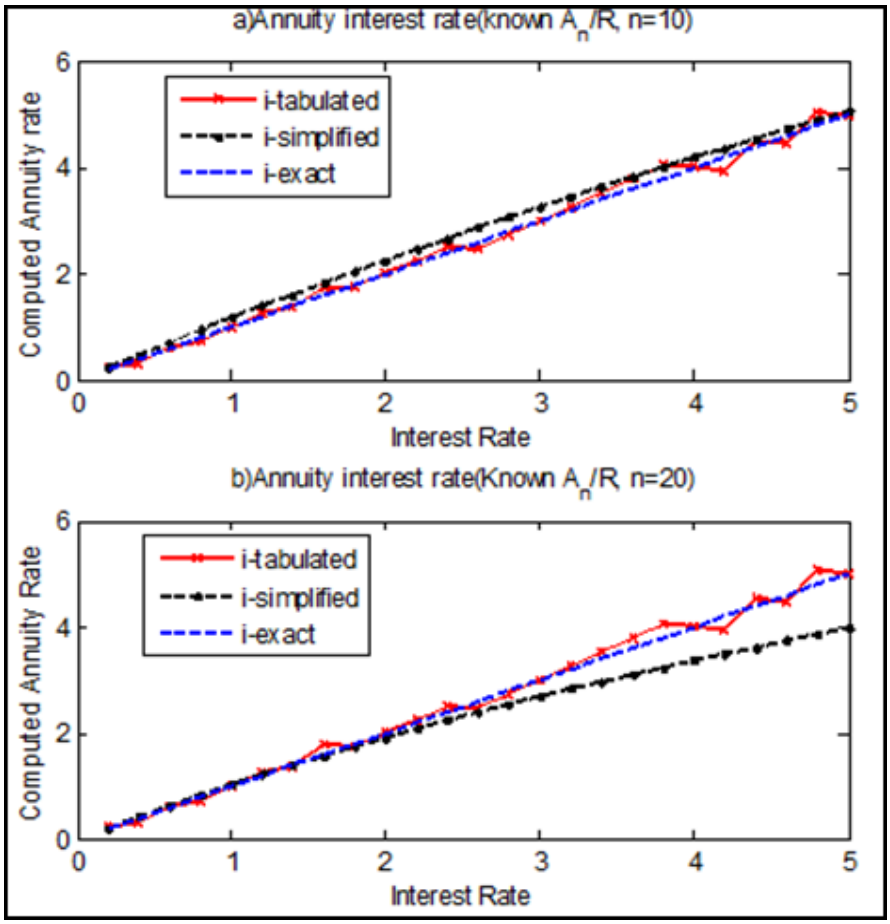


FIGURE 2: Tabulated and Simplified annuity interest rate (known $\mathbf{A_n/R}$)

6.ERROR ANALYSIS OF ANNUITY INTEREST RATE

The percentage absolute relative error(ARE) between the exact and simplified Annuity Interest Rate is given by:

$$ARE = \left|\frac{exact - simplified}{exact}\right| * 100 \qquad (13)$$

Table_3 and Table_4 show the percentage Absolute Relative Error(ARE)of the tabulated and simplified annuity interest rate respectively. These tables compute the error associated with the annuity interest rate at known amount and present value respectively.Figure_3 shows the variation. Therefore, computing the annuity interest rateusing Eq.(9) ,three_Term_simplification, is recommended especially when theamount is known and lower conversion time period(n).

Similarly, Figure_4 indicates that the annuity interest rate using Eq.(11) ,two_Term_simplification, is recommended especially when the present value is known and lower conversion time period(n).

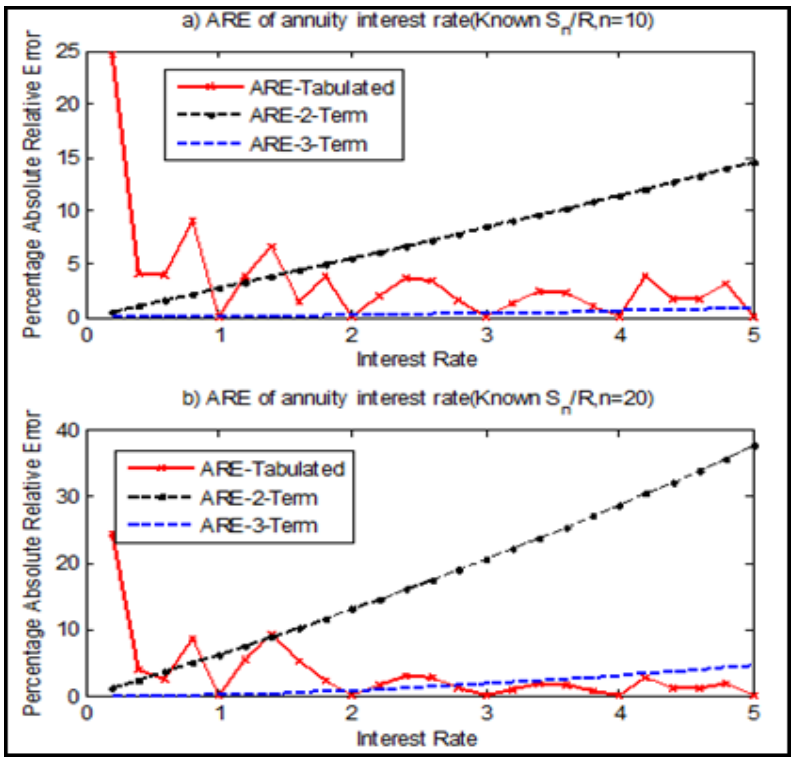


FIGURE 3: Percentage absolute relative error of annuity interest rate (known S_n/R)

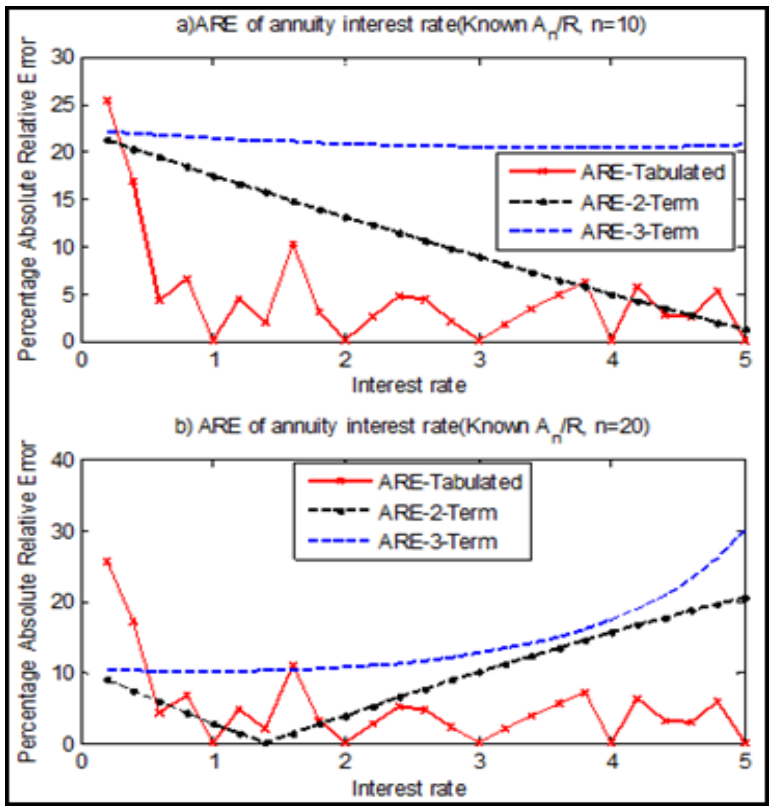


FIGURE 4: Percentage absolute relative error of annuity interest rate (known A_n/R)

The percentage reduction in Relative Error between the tabulated technique and the simplified one of the annuity interest rate is given by:

$$\delta E = \left(\frac{ARE\ of\ tabulated\ rate - ARE\ of simplified\ rate}{ARE\ tabulated\ rate}\right) * 100 \qquad (14)$$

Where:

δE isthe percentage reduction in Relative Error between the tabulated technique and the simplified one.If δE is positive values, Error reduction will occur using simplified technique. If it is negative values, Error reduction will occur using tabulated technique. otherwise, there is no error reduction.

Table_5 shows the percentage reduction in Relative Error between the tabulated technique and the simplified one for different interest rate.

The squared error(Er) between the exact values and the computed annuity interest rate is given by:

$$E_i = (exact - computed)^2$$
$$= (ARE * exact)^2 \qquad (15)$$

i% exact	Known S_n/R & n=10			Known A_n/R & n=10		
	Error reduction (%)	Squared error(E_i)		Error reduction (%)	Squared error(E_i)	
		Tabulated	3_Term		Tabulated	2_Term
0.2	99.99	24.44	1.4E-07	16.15	25.69	18.06
0.4	99.82	2.67	8.6E-06	-19.59	46.07	65.88
0.6	99.60	5.90	9.7E-05	-348.99	6.68	134.72
0.8	99.68	51.89	5.4E-04	-183.16	27.05	216.93
1.2	98.37	22.24	5.9E-03	-272.92	28.45	395.70
1.4	98.70	86.84	1.5E-02	-718.45	7.19	481.73
1.6	92.24	5.36	3.2E-02	-44.37	268.65	559.93
1.8	96.30	47.05	6.4E-02	-358.70	29.80	627.10
2.2	89.82	20.12	2.1E-01	-380.08	31.21	719.30
2.4	93.35	78.46	3.5E-01	-141.01	127.66	741.51
2.6	91.60	78.29	5.5E-01	-142.18	127.35	746.97
2.8	78.88	19.06	8.5E-01	-375.70	32.52	735.81
3.2	68.08	18.08	1.8E+00	-342.99	33.98	666.86
3.4	80.72	70.38	2.6E+00	-110.15	138.56	611.94
3.6	77.23	70.15	3.6E+00	-31.00	318.18	546.06
3.8	46.04	17.05	5.0E+00	9.56	576.82	471.79
4.2	81.74	264.72	8.8E+00	26.45	573.52	310.23
4.4	57.11	62.60	1.2E+01	-24.06	149.37	229.91
4.6	51.22	62.42	1.5E+01	-2.05	148.92	155.11
4.8	71.67	235.95	1.9E+01	61.87	619.73	90.10
Mean	83.61 %	_____	_____	_____	_____	_____

TABLE 5: Squared error and Error reduction for computing Annuity Interest Rate

If the integer values of annuity interest rate in Table_5(integer values of i_exact) is excluded due to very small errors associated with it. This indicates that the 3_Term simplified technique of known amount gives reduction in ARE by approximately 83.61% compared to the tabulated technique. Figure_5.a shows the variation. On the other hand, the tabulated technique of known present value is appropriate compared to the simplified one. This led us to implement a correction factor which will be discussed in the coming paper.Figure_5.b shows the variation.

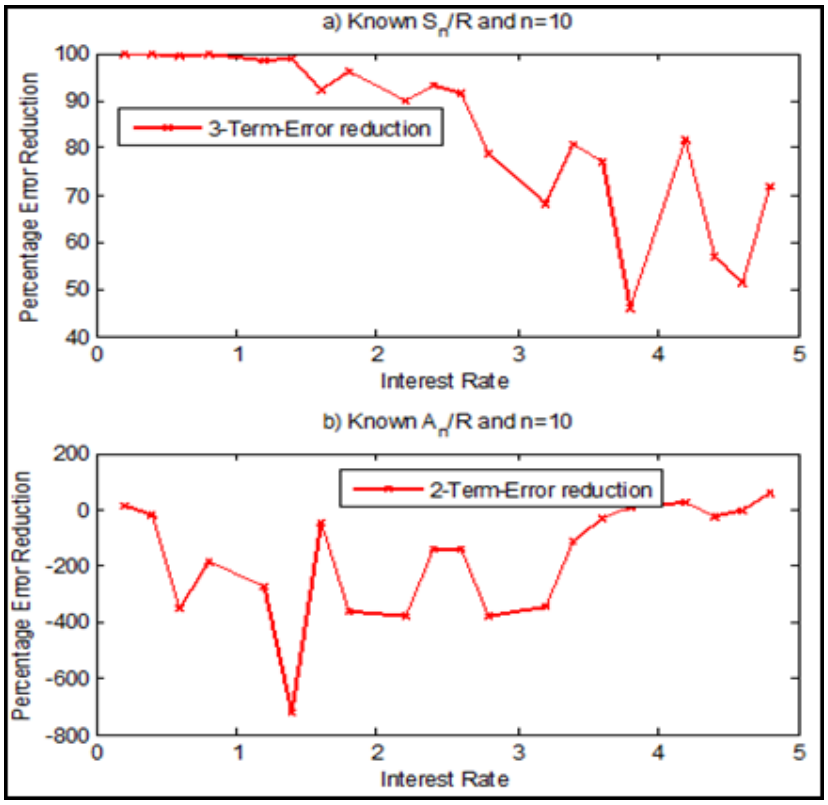


FIGURE 5: Percentage error reduction of annuity interest rate.

7. PROCESSING TIME OF THE ANNUITY INTEREST RATE

The processing time required for Computing the annuity interest rateis executed by LaptopDELL-inspiron-1520.Table_6 indicates that the average processing time required for computing the annuity interest rateusing the tabulated and the simplified technique.

i% exact	Known S_n/R & n=10		Known A_n/R & n=10	
	CPU time (Second)		CPU time (Second)	
	Tabulated	3_Term	Tabulated	2_Term
1.6	63.5	2.6137e-007	65	5.4142e-007

TABLE 6: Average CPU time of Annuity Interest Rate

8. MATLAB PROGRAMMING:

A complete program can be obtained by writing directly to the author[2].

9. COMPUTATIONAL ENERGY OF THE ANNUITY INTEREST RATE

Computing the computational energy for annuity interest rate requires the determination of conversion period(n), the square error(Ei), and the average processing time(CPU time). Therefore, consider the conversion period(n) represents the resistance, the square error is measured in [volts]2, and the CPU time in second. Then, the computational energy per conversion period is given by:

$$CE = \frac{E_i * t}{n} \qquad (16)$$

Where:

CE is the computational energy per conversion period.
$E_i$is the i^{th} square error.
t isthe averageCPU time.

n is the conversion period.

The computational energy saved by the simplified technique compared to the tabulated one is given by:

$$\delta CE = \frac{CE_T - CE_s}{CE_T} * 100 \qquad (17)$$

Where:

δCE is the relative computational energy saved by the simplified technique.
CE_T is the computational energy for the Tabulated method.
CE_s is the computational energy for the simplified technique.

Table_7 shows the computational energy(CE) for each technique. This table indicates that the simplified technique saved computational energy by approximately 99.9% compared to the tabulated one. Figure(6) shows the variation in computational energy required for calculation.

i_exact %	Known S_n/R & n=10		
	CE_Tabulated	CE_3_Term	CE_saved by 3_Term
0.2	155.20	3.58E-15	100
0.4	16.98	2.26E-13	99.9999999999987
0.6	37.48	2.53E-12	99.9999999999933
0.8	329.47	1.40E-11	99.9999999999958
1.2	141.24	1.55E-10	99.9999999998906
1.4	551.42	3.84E-10	99.9999999999304
1.6	34.03	8.42E-10	99.9999999975242
1.8	298.74	1.68E-09	99.9999999994367
2.2	127.76	5.45E-09	99.9999999957343
2.4	498.21	9.06E-09	99.9999999981822
2.6	497.11	1.44E-08	99.9999999970962
2.8	121.01	2.22E-08	99.9999999816483
3.2	114.84	4.82E-08	99.9999999580682
3.4	446.94	6.84E-08	99.9999999847050
3.6	445.43	9.51E-08	99.9999999786590
3.8	108.29	1.30E-07	99.9999998801563
4.2	1680.97	2.31E-07	99.9999999862822
4.4	397.53	3.01E-07	99.9999999242805
4.6	396.35	3.88E-07	99.9999999020713
4.8	1498.30	4.95E-07	99.9999999669690
Mean	----------	----------	**99.99**

TABLE 7: Computational Energy of Annuity Interest Rate(CE)

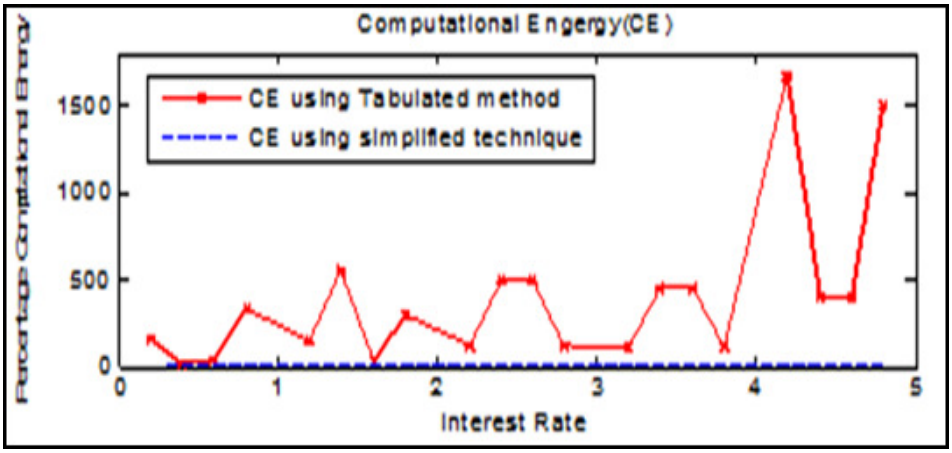


FIGURE 6: Percentage Computational Energy of annuity interest rate.

10. CONCLUSIONS

A new mathematical formula is derived here to compute an approximate value of the Annuity Interest Rate. It can be implemented using simple calculator, to save time and to avoid systematic errors associated with tables, and to calculate missing values of i in tables. Since other solutions depend on a trial-and-error approach.

A new algorithm has been derived for fast evaluation of the annuity interest rate. As a result the new technique offered four advantages over the tabulated one:

(1) It drastically reduces the average CPU time required for calculating the annuity interest rate.
(2) It drastically reduces the absolute relative error(ARE) for calculating the annuity interest rateby 83.61%compared to the current one.
(3) It gives minimum square error compared to the current tabulated method.
(4) It has lowest computational energy.

The aforementioned features are combined in a mathematical formula to describe the system performance. This formula is called the computational energy. A quantitative study has been carried out to compute the computational energy for each technique. The results show that the simplified technique saved computational energy by 99.9% compared to the current one. A correction factor will be discussed in the next paper.

11. REFERENCES

[1] Shao and Shao, “Mathematics for management and finance”, eighth edition, 1998.

[2] Email: karamfayed_1@hotmail.com

Distance Sort

Krishna Mohan Ankala *krishna.ankala@gmail.com*
Assoc.Prof, Dept of Computer science,
Ucek, JNTU Kakinada.

Hari Krishna Gurram, *harikrishna553@gmail.com*
M.Tech (CS),
Ucek, JNTU Kakinada.

Shanmukha Rao Kummari *kshanmuk@gmail.com*
M.Tech(CS),
Ucek, JNTU Kakinada.

Abstract

One of the fundamental issues in computer science is ordering a list of items. Although there is a number of sorting algorithms, sorting problem has attracted a great deal of research, because efficient sorting is important to optimize the use of other algorithms. This paper presents a new sorting algorithm which runs faster by decreasing the number of comparisons by taking some extra memory. In this algorithm we are using lists to sort the elements. This algorithm was analyzed, implemented and tested and the results are promising for a random data.

Keywords: Distance Sort, Distance

1. INTRODUCTION

Today real world getting tremendous amounts of data from various sources like data warehouse, data marts etc. To search for particular information we need to arrange this data in a sensible order. Many years ago, it was estimated that more than half the time on commercial computers was spent in sorting. Fortunately variety of sorting algorithms came into existence with different techniques [1].

Many algorithms are well known for sorting the unordered lists. Most important of them are merge sort, heap sort, shell sort and quick sort etc. [2]. As stated in [3], sorting has been considered as a fundamental problem in the study of algorithms, that due to many reasons:

- The need to sort the information is inherent in many applications.
- Algorithms often use sorting as a key subroutine.
- Many engineering issues come to the fore when implementing sorting algorithms.
- In algorithm design, there are many essential techniques represented in the body of sorting algorithms.
-

Sorting algorithms plays a vital role in various indexing techniques used in data warehousing, and daily transactions in online Transactional processing (OLTP). Efficient sorting is important to optimize the use of other sorting algorithms that require sorted lists correctly.

Sorting algorithms can be classified by:

- Computational complexity (best, average and worst behavior) of element comparisons in terms list size n. For a typical sorting algorithm best case, average case and worst case is O(n log n), example merge sort.
- Number of swaps
- Stability : A sorting algorithm is stable if whenever there are two records X and Y, with the same key and X appearing before Y in original list, X will be appear before Y in the sorted list.
- Usage of memory

In this paper, a new sorting algorithm (Distance sort) is proposed; here the basic idea is by taking a random sample of elements in the input we calculate the distance first, which is used to sort the elements in the given input. Here, we are taking the sample size as one percent of the total size of the input. As compared to other sorting algorithms this algorithm takes more memory, To restrict it from using very huge memory we are restricting the size of the list to 1.5 * n in sorting the data with n elements.

Section 2 presents the concept of Distance sorting algorithm and its pseudo code. Section 3 shows the implementation results for various sizes of random input. Finally, the conclusion was presented in section 4.

2: DISTANCE SORT

2.1 Concept

This algorithm works efficiently on random data by calculating the approximate position of the element. The main logic presented here is by calculating the approximate position we are able to minimize the number of comparisons, obviously the efficiency of the algorithm is increased.

2.2 Pseudocode

In pseudocode**,** the distance sort algorithm might be expressed as,

```
function sort ( input, size )
1.  var sizeOfSample := size / 100
2.  average( input )
3.  var maximum, minimum
4.  var distance := getDistance()
5.  if distance := 0
6.      distance := 1
7.  end if
8.  findMaxMin(input)
9.  marker = maximum + 1
10. var approxEle := getApproxEle() + 1
11. var constraintSize  = 1.5 * size
12. if approxEle > constraintSize
13.    approxEle := constraintSize
14. end if
15. Node in[approxEle]
16. initializeNode()
17. for i:=0 to size do
18.    var x := input[i]
19.   var position := ( x – minimum) /       distance
20. If position > approxEle
21.    Position := approxEle
22. end if
23. if in[position].element := marker
24.   in[position].element := x
25. end if
26. else
27.     if ( in[position].element >= x )
28.        Node temp
29.        temp.element = x
30.        temp.next = in[position]
31.        in[position] = temp
32.    end if
33.    else
34.      var flag := 1
35.      Node temp1 := in[position]
```

```
36.     Node temp := in[position]
37.      while (temp1.element < input[i])
38.          temp := temp1
39.          temp1 := temp1.next
40.          if( temp1 := null )
41.             temp1.element := x
42.             flag := 0
43.             break
44.          end if
45.       end while
46.    if flag := 0
47.       Node temp2
48.       temp2.element := input[i]
49.       temp.next := temp2
50.    end if
51.    else
52.     temp2.element := input[i]
53.    temp2.next := temp1
54.   temp.next := temp2
55.  end else
56.   end else
57.  end for
58.  var counter := 0
59.  for i:=0 to approxEle
60.     Node temp3 := in[i]
61.     while ( temp3.element != 0 )
62.         input[counter]:= temp3.element
63.        counter := counter + 1
64.        temp3 := temp3.next
65.        if temp3 := null
66.            break
67.       end if
68.       if (counter :=  (size -1) )
69.           break
70.      end if
71.    end while
72.    if ( counter := ( size -1 ) )
73.      Break
74.     end if
75.   end for
76.  end sort
```

Line 1, declares a variable sizeOfSample, which calculates the one percent of total input elements. By using this variable we are going ti calculate the approximate distance between the elements.
In line2, we called the function, *average*, the pseudocode for that function is given below.

```
function average(input)
1.  var counter := 0
2.  for counter 0 to sizeOfSample
3.    randPos := rand( size )
4.   average :=average +  input[randPos]
5.  End LOOP
6.  average = average/sizeOfSample
7.  end average
```

In line 4 of sort method we are calling the getDistance method which is used to calculate the approximation distance between the random elements, the pseudocode for the getDistance method is given below.

```
function getDistance()
```

```
1.  var dist =  (2*average)/ size
2.  if(dist < 0 )
3.     dist = -1 * dist
4.  Return dist
5.  end getDistance
```

In line8 of sort method we are calling the *findMaxMin* method, which is used to calculate the maximum and minimum values in the input elements.

```
function findMaxMin()
1.  var counter := 0
2.  maximum := input[0]
3.  minimum := input[0]
4.  for counter 1 to size
5.     if ( minimum > input[counter] )
6.        minimum := input[counter]
7.    end if
8.     if(maximum < input[counter] )
9.        maximum := input[counter]
10.   end if
11. end for
12. end findMaxMin
```

In line 9 of sort method we are calling the *getApproxEle* method which is used to calculate the approximate number of Nodes used to sort the given data. The pseudocode for the *getApproxEle* method is given below.

```
function getApproxEle()
1.  return (maximum – minimum)/distance
2.  end getApproxEle
```

Lines 10, 11, 12, 13 are used to constrain the number of nodes to sort the input elements. Line 14 declares an array *in* of type Node. The pseudo code for the structure Node is given below.

```
1.  struct Node
2.     int element
3.     Node next
4.  End struct Node
```

In line 15 of *sort* method we are calculating the *initializeNode* method which is used to initialize the nodes. The pseudocode for the initializeNode is given below.

```
Function initializeNode()
1.  var counter := 0
2.  for counter 0 to approxEle
3.     in[counter].element := marker
4.     In[counter].next := null
5.  end for
6.  end initializeNode
```

In line 18, we are calculating the approximate position for each input element, by calculating the approximate position we are going to reduce the number of comparisons. Lines 19 to 70 sort the elements by comparing from the approximation position.

3: IMPLEMENTATION RESULTS

Number Of Elements	Time Taken	Total comparisons
1000000	335	1428360
2000000	725	2856095
3000000	1105	4282827
4000000	1492	5708973
5000000	1874	7135946
6000000	2300	8558582
7000000	2666	9983612

TABLE 1: Best case scenario for Distance Sort (Time in milliseconds)

When there are very less number of duplicates in the input then this algorithm works in best case. If there are more duplicates in the input then this algorithm goes into the worst case.

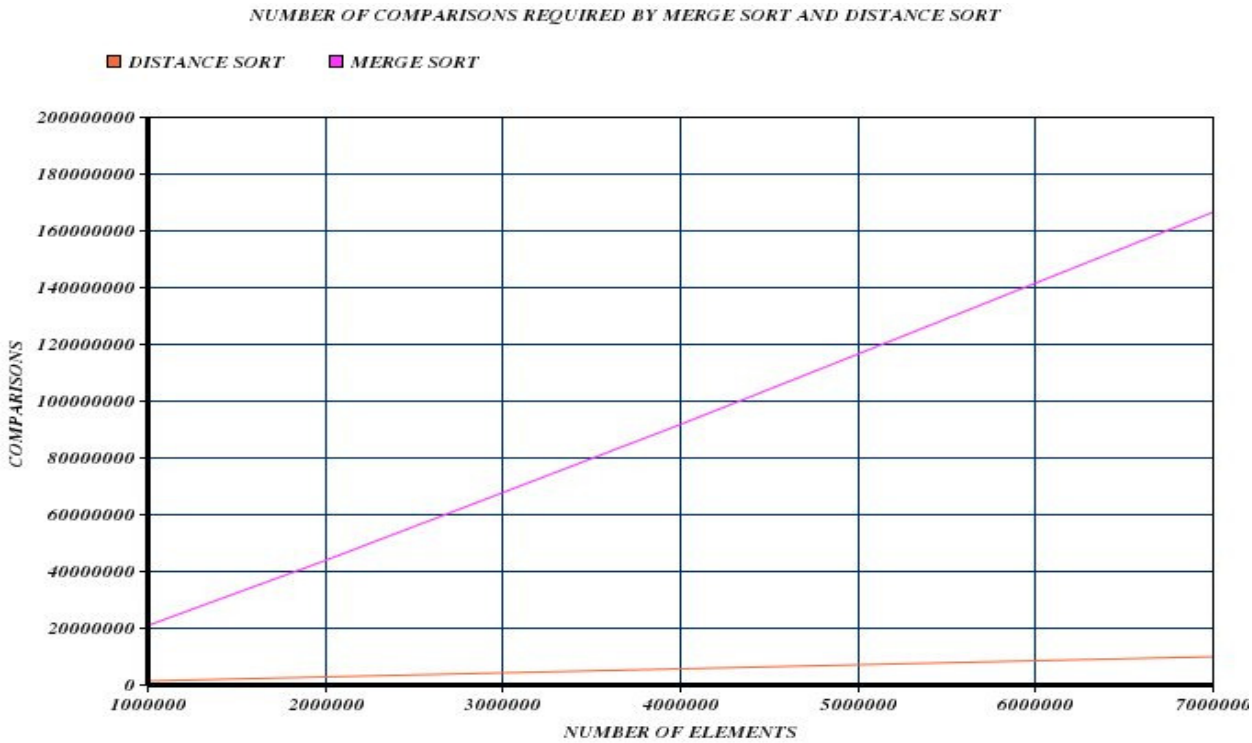


FIGURE 1: Best case comparison of Distance sort Vs Merge Sort.

Fig 1 shows that in best case distance sort work far better than the merge sort, Since in best case the position of an element is found approximately equal to the actual position. The best case occurs for the distance sort only when there are less number of duplicates.

Since we are finding the input element position approximately by the distance (we calculated from the average value), If the distance is calculated appropriately, then this algorithm works in best case, if the distance is not appropriate then this algorithm goes into worst case.

4: CONCLUSION

This distance sorting algorithm works very fast when there are very less number of duplicates in the input data and this algorithm totally depends on the distance value we are calculating to find out approximate position of the element to reduce comparisons.

5: REFERENCES

[1] Kruse R., and Ryba A., *Data Structures and Program Design in C++*, Prentice Hall, 1999.

[2] Shahzad B. and Afzal M., "Enhanced ShellSorting Algorithm," *Computer Journal of Enformatika*, vol. 21, no. 6, pp. 66-70, 2007.

[3] Cormen T., Leiserson C., Rivest R., and Stein C., *Introduction to Algorithms*, McGraw Hill, 2001.

[4] Aho A., Hopcroft J., and Ullman J., *The Design and Analysis of Computer Algorithms*, Addison Wesley, 1974.

[5] Astrachanm O., *Bubble Sort: An Archaeological Algorithmic Analysis*, Duk University, 2003.

[6] Bell D., "The Principles of Sorting," *Computer Journal of the Association for Computing Machinery*, vol. 1, no. 2, pp. 71-77, 1958.

[7] Box R. and Lacey S., "A Fast Easy Sort,"*Computer Journal of Byte Magazine*, vol. 16,no. 4, pp. 315-315, 1991.

.

[8] Deitel H. and Deitel P., *C++ How to Program*, Prentice Hall, 2001.

[9] Friend E., "Sorting on Electronic ComputerSystems," *Computer Journal of ACM*, vol. 3, no. 2, pp. 134-168, 1956.

[10] Knuth D., *The Art of Computer Programming*,Addison Wesley, 1998.

[11] Ledley R., *Programming and Utilizing Digital Computers*, McGraw Hill, 1962.

[12] Levitin A., *Introduction to the Design andAnalysis of Algorithms*, Addison Wesley, 2007.

[13] Nyhoff L., *An Introduction to Data Structures*, Nyhoff Publishers, Amsterdam, 2005.

[14] Organick E., *A FORTRAN Primer*, AddisonWesley, 1963.

[15] Pratt V., *Shellsort and Sorting Networks*,Garland Publishers, 1979.

[16] Sedgewick R., "Analysis of Shellsort andRelated Algorithms," *in Proceedings of the 4th Annual European Symposium on Algorithms*,pp. 1-11, 1996.

[17] Seward H., "Information Sorting in theApplication of Electronic Digital Computers to Business Operations," *Masters Thesis*, 1954.

[18] Shell D., "A High Speed Sorting Procedure,"*Computer Journal of Communications of the ACM*, vol. 2, no. 7, pp. 30-32, 1959.

[19] Thorup M., "Randomized Sorting in O(n log logn) Time and Linear Space Using Addition,Shift, and Bit Wise Boolean Operations,"*Computer Journal of Algorithms*, vol. 42, no. 2,pp. 205-230, 2002.

INSTRUCTIONS TO CONTRIBUTORS

Experimental Algorithmics studies algorithms and data structures by joining experimental studies with the more traditional theoretical analyses. With this regard, the aim of The International Journal of Experimental Algorithms (IJEA) is (1) to stimulate research in algorithms based upon implementation and experimentation; in particular, to encourage testing, evaluation and reuse of complex theoretical algorithms and data structures; and (2) to distribute programs and testbeds throughout the research community and to provide a repository of useful programs and packages to both researchers and practitioners. IJEA is a high-quality, refereed, archival journal devoted to the study of algorithms and data structures through a combination of experimentation and classical analysis and design techniques. IJEA contributions are also in the area of test generation and result assessment as applied to algorithms.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJEA.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJEA appears in more focused issues. Besides normal publications, IJEA intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJEA LIST OF TOPICS

The realm of International Journal of Experimental Algorithms (IJEA) extends, but not limited, to the following:

- Algorithm Engineering
- Algorithmic Code
- Algorithmic Engineering
- Algorithmic Network Analysis
- Analysis of Algorithms
- Approximation Techniques
- Cache Oblivious algorithm
- Combinatorial Optimization
- Combinatorial Structures and Graphs
- Computational Biology
- Computational Geometry
- Computational Learning Theory
- Computational Optimization
- Data Structures
- Distributed and Parallel Algorithms
- Dynamic Graph Algorithms
- Experimental Techniques and Statistics
- Graph Manipulation
- Heuristics
- Mathematical Programming For Algorithms
- Metaheuristic Methodologies
- Network Design
- Parallel Processing
- Randomized Techniques in Algorithms
- Routing and Scheduling
- Searching and Sorting
- Topological Accuracy
- Visualization Code
- VLSI Design
- Graphics

CALL FOR PAPERS

Volume: 3 - **Issue:** 1 - February 2012

i. Paper Submission: November 30, 2011 **ii. Author Notification:** January01, 2012

iii. Issue Publication: January / February 2012

CONTACT INFORMATION

Computer Science Journals Sdn BhD
B-5-8 Plaza Mont Kiara, Mont Kiara
50480, Kuala Lumpur, MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org

COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA