Dhaka Univ. J. Sci. 62(2): 95-102, 2014 (July)

# A Study on 1-D Simplex Search Algorithm with its Numerical Experiments Through Computer Algebra

**H. K. Das, Tapash Saha**, and **M. Babul Hasan***

*Department of Mathematics, Dhaka University, Dhaka-1000, Bangladesh*

(Received: 14 April 2013; Accepted: 20 March 2014)

**Abstract**

Computer techniques have been developed to solve 1-D NLP problems. The 1-D simplex search algorithm was studied and a code corresponding to the modified phase-0 has been developed. The 1-D two phase methods and modified Phase-0 method have been compared with those reported by others. The efficiency of computer techniques and algorithms have been demonstrated with a number of numerical examples.

**Keywords:** Non Linear Programming,1-D, Two Phase, Simplex Search, Computer Algebra.

## I. Introduction

Unconstrained problems with nonlinear objective function has many applications. In addition, Numerical experiments and developing codes are very important in the current research of OR. This is often viewed as a discipline in itself. However, because of difficulty of analyzing non-linear calculations, the vast majority of questions that are important to the performance of optimization algorithms. Also, 1-D simplex search method which was originally proposed by Powell[1] and subsequently modified by Nelder and Mead[2] is one of most widely known multidimensional direct search methods. In 1987, Choo et al.[3], were able to show that it converges to the optimal solution if the function to be maximized is unimodal.

In 1989, Xue[4] showed that it can be made to behave either as the Golden Section search or the binary search by assigning certain parameters. In 1965, Nelder et al.[2] simplex search was developed on the concept of reflection, expansion and contraction through the entire search phase. However, Choo et al.[3] modified the Nelder et al.[2] simplex search by dividing it into two phases where the contraction step is eliminated during the first phase, called phase 0. We must point out, however, there are a strong relation between Phase 0 and the Nelder et al.[2] simplex search.

Das et al.[5] developed a computer technique incorporated with Golden section and Gradient search methods which is very powerful for the given optimal interval but if a proper optimal interval is not chosen it should face a complexity to find the optimal solution like Choo et al.[3]. In the current research, the complexity of the Choo et al.[3] and Das et al.[5] has been removed by an uniform algorithm of the Choo et al.[3] and as proposed by Ayode[6] modified phase-0.

## II. One Dimensional Method of NLP

In this section, we discuss the 1-D simplex search methods for finding the optimum solutions and also present two flow charts reported by Choo et al. in two phase method.

*1-D Simplex Search*

In this section, we summarize the method of Choo et al.[3]'s 1-D simplex search. This method was proposed by Choo et al.[3.] in 1987.

i. This method can start with any two points and converges to the optimum solution.

ii. With appropriate parameters, this algorithm can be made to behave equivalent to some of the most efficient 1-D search methods.

*1-D Simplex Search[1]*

In this section, we summarize the Powel's 1-Dsimplex search method which is given as follows:

i. Reviews some of the most successful methods for unconstrained, constrained and non differentiable optimization calculations.

ii. Particular attention was given to the contribution of theoretical analysis.

*Golden Section and Fibonacci Search[7]*

In this section, we summarize the method of *Golden Section* search method. In 1953 *Golden section* and *Fibonacci search* was developed by Kiefer[7] which is given as follows:

i. Find the extremum of a unimodal function over an interval without using derivatives.

ii. Golden section narrows the range of values and it is based on the golden ratio.

iii. If the interval is not optimal then the method will fail and needs more iteration.

*Nelder and Mead Method[2]*

In 1965, Nelder et al.[2] proposed a method which is discussed as follows:

i. It was designed for unconstrained optimization without using gradients.

ii. The operation of this method is to rescale on the local behavior of the function by using four basic procedures:1. Reflection, 2. Expansion, 3. Contraction and 4. Shrinking

## III. One Dimensional Unconstrained NLP

In this section, we first modify an algorithm for solving 1-D simplex search and then develop a code for solving 1-D NLP problems.

*Flow Chart of Phase 0*

In this section, we present Phase-0 flowchart corresponding to the Choo et al.[3] algorithm. The parameters $\alpha$ and $\delta$ determine the step sizes where $\alpha + \delta \geq 1$.

* Author for correspondence. *e-mail :* babulhasan@yahoo.com

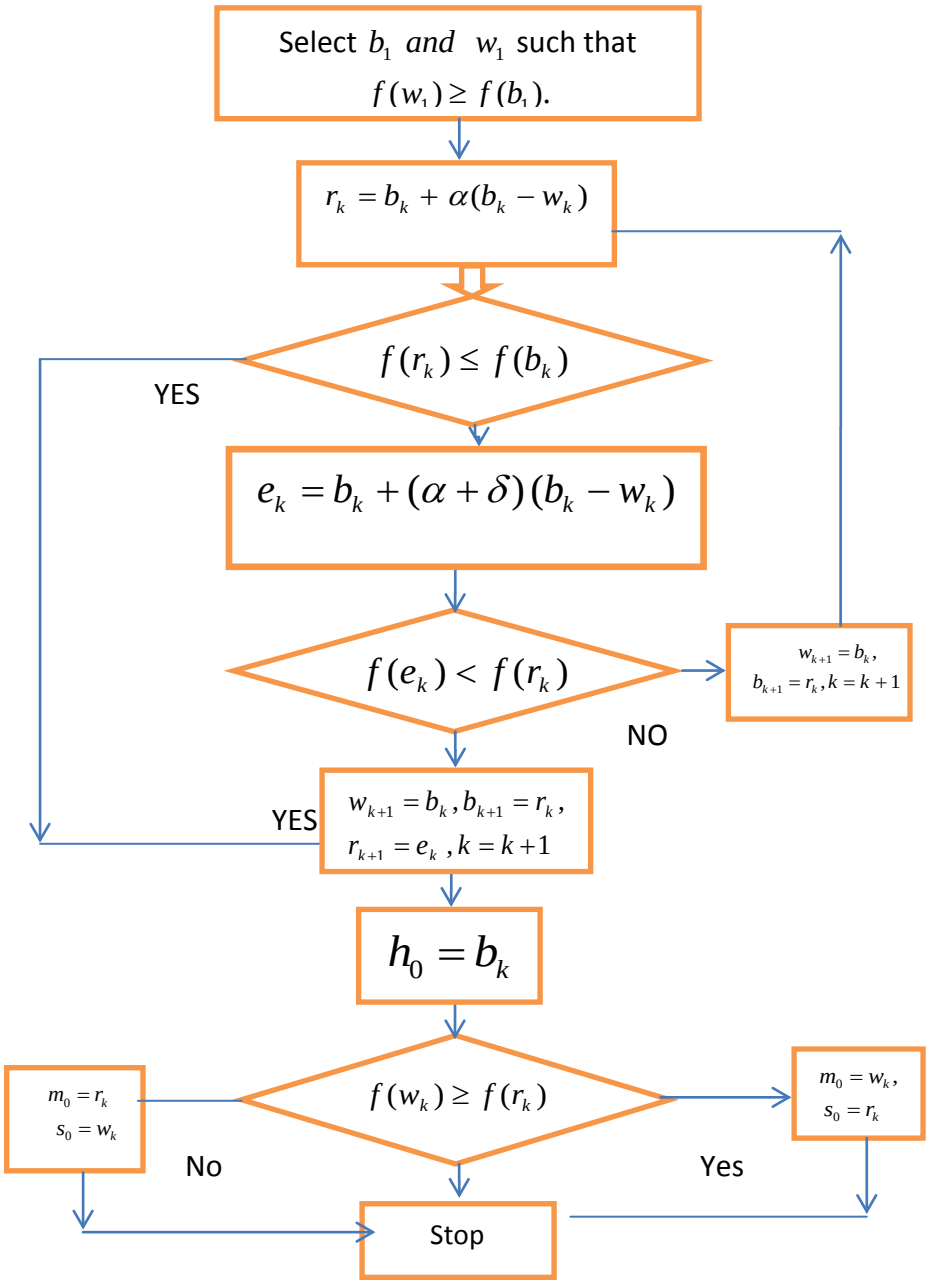


**Fig. 1**. Phase 0 Flow chart

*Flow Chart of Phase 1*

In this section, we present Phase 1 flowchart corresponding to the algorithm of Choo – Kim[3] Phase-1. The parameters $\alpha, \beta$ and $\delta$ determines the step sizes of reflection, contraction, and expansion respectively. The flowchart of Phase 1 is shown in the following figure.

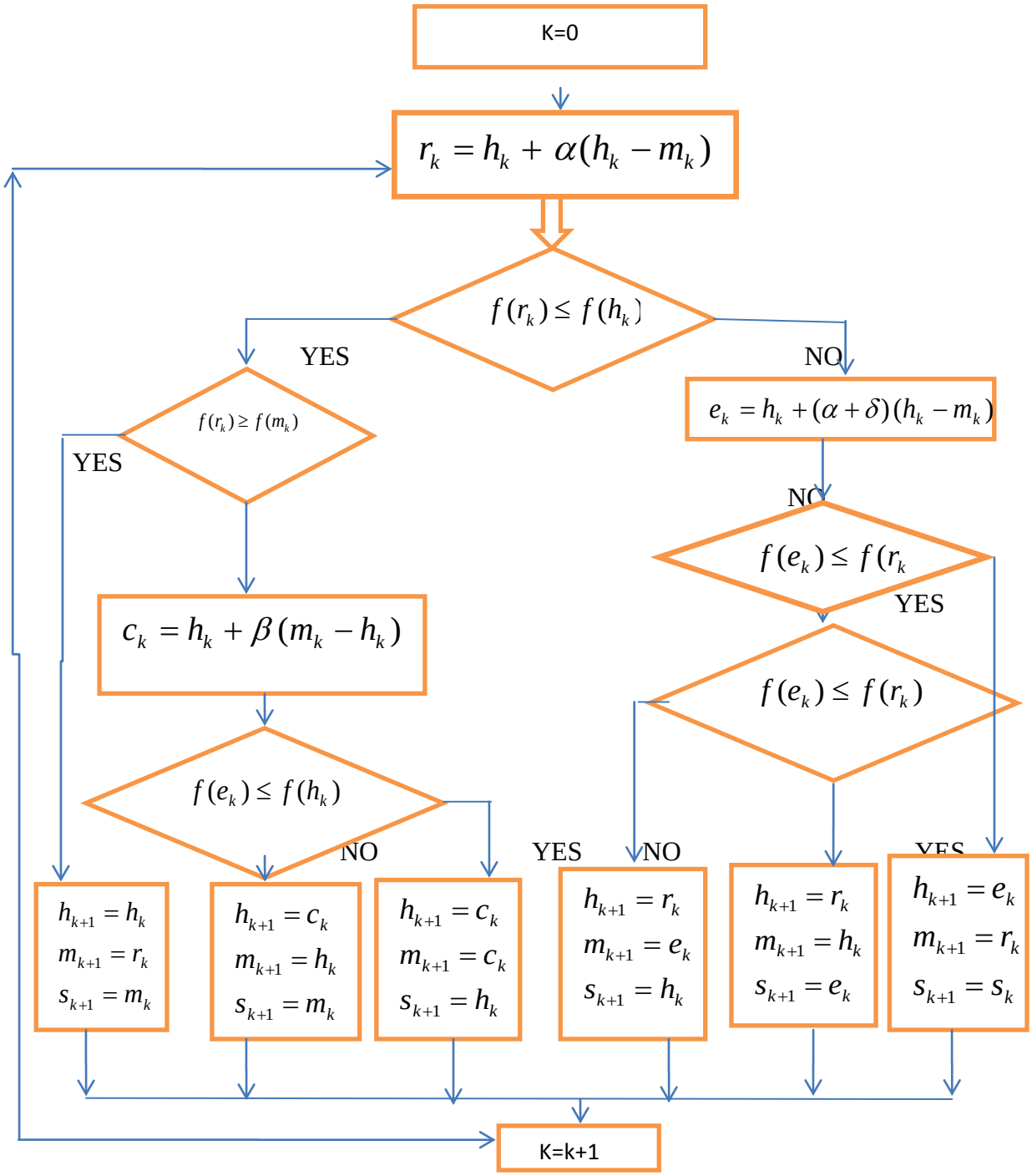


**Fig. 2.** Phase 1 Flowchart

*Modified Algorithm*

In this section, we present a unique algorithm for solving unconstrained 1-D NLP problems.

*Step 1:* If number of variables 1 then go to the following Step.

*Step 2*: Input the initial interval and check that is it optimal interval or not.

*Step 3:* Finding the optimum interval using the following Sub Steps.

*Sub Step1:* Select two points $b_1$ *and* $w_1$ such that $f(w_1) \ge f(b_1)$. Set k=1.

*Sub Step 2:* Contraction:

$c = b_1 + \beta(w_1 - b_1)$. If $f(c) \le f(b_1)$ then set $m = b_1, h = c, s = w_1$ and stop. Otherwise go to Sub step 3.

*Sub Step 3:*

Reflection: $r_k = b_k + \alpha(b_k - w_k)$.

*Sub Step 4:* If $f(r_k) \geq f(b_k)$ then go to *Sub Step 5.*

Expansion: $e_k = b_k + (\alpha + \delta)(w_k + b_k)$.

If $f(e_k) > f(r_k)$ then set

$w_{k+1} = b_k, b_{k+1} = r_k, r_{k+1} = e_k, k = k+1$ and go to *Sub Step 5*. Otherwise set $w_{k+1} = b_k, b_{k+1} = r_k, k = k+1$ and go to *Sub Step 3*.

*Sub Step 5:* Set $h = b_k$ If $f(w_k) \leq f(r_k)$ then set

$m = w_k, s = r_k$ and stop and go to *Step 4*. Otherwise set

$m = r_k, s = w_k$ and go to the step 3.

*Step 4:* To find the optimal solution $\bar{x}$ of the NLP we use the value $m = r_k, s = w_k$ and follow the following Sub-Steps.

*Sub Step 1:* Set k=0 and Go to *Sub*-Step 2.

*Sub Step 2:*

Reflection: let $r_k = h_k + \alpha(h_k - m_k)$. Go to *Sub*-Step 3.

*Sub Step 3:* If $f(r_k) \leq f(h_k)$ then go to *Sub* Step 5.

Expansion: $e_k = h_k + (\alpha + \delta)(h_k - m_k)$. If

$f(e_k) \leq f(r_k)$ then go to *Sub Step 4*. Otherwise let

$h_{k+1} = e_k, m_{k+1} = r_k, s_{k+1} = s_k, k = k+1$ and go to *Sub Step 2*.

*Sub Step 4:* If $f(e_k) \leq f(h_k)$ then let

$h_{k+1} = r_k, m_{k+1} = h_k, s_{k+1} = e_k, k = k+1$ and go to *Sub Step 2*. Otherwise, we let

$h_{k+1} = r_k, m_{k+1} = e_k, s_{k+1} = h_k, k = k+1$ and go to *Sub Step 2*.

*Sub Step 5:* Contraction: if $f(r_k) \geq f(m_k)$ then let

$h_{k+1} = h_k, m_{k+1} = r_k, s_{k+1} = m_k, k = k+1$ and go to *Sub*-Step 2. Otherwise, we let $c_k = h_k + \beta(m_k - h_k)$. If

$f(c_k) \leq f(h_k)$ then let

$h_{k+1} = h_k, m_{k+1} = c_k, s_{k+1} = r_k, k = k+1$ and go to *Sub-Step 2*. Otherwise, we let

$h_{k+1} = c_k, m_{k+1} = h_k, s_{k+1} = m_k \; k = k+1$ and go to *Sub Step 2*.

*Sub Step 6:* Stop and find optimal solution.

## IV. Program Organization

In this section, we develop computer codes implementing the Choo et al.[3] 1-D simplex search method for solving unconstrained NLP problems by programming language Mathematica[9].

*Program Organization*

In this section, we present the programming code for the Choo et al.[3] algorithm. In this sense, we present Phase 0 code and then we present Phase 1 code.

*Program of Phase 0 Algorithm*

In this section, we develop the code corresponding to the Fig-1: Phase-0 flowchart of Choo et al. Phase 0 algorithm.

```
PHASE[0_]:=Module[{},
f[x_]:=Evaluate[Input["objective function"]];
 w1=Input["Enter t4he input"];
 b1=Input["Enter the input"];
α=0.2;δ=1;β=.1;  n=100;
If[f[b1]≤f[w1], b[1]=b1;w[1]=w1;
c=b[1]+β (w[1]-b[1]);
If[f[c]≤f[b[1]],
m=b[1];h=c;s=w[1],
Do[r[i]=b[i]+α (b[i]-w[i]);
If[f[b[i]]≤f[r[i]],h=b[i];
If[f[w[i]]≤f[r[i]],m=w[i];Print[i];s=r[i],m=r[i];Print[i];s=w[i
],e[i]=b[i]+(α+δ) (b[i]-w[i]);
If[f[r[i]]<f[e[i]],w[i+1]=b[i];
b[i+1]=r[i];r[i+1]=e[i];
h=b[i+1];If[f[w[i+1]]≤f[r[i+1]],
m=w[i+1];Print[i];s=r[i+1],m=r[i+1];Print[i];s=w[i+1]],
w[i+1]=b[i];b[i+1]=e[i];
m=w[i+1];s=b[i+1];
 ]],{i,1,n}]],Print["jfdgfdg"]]
 Print[m]; Print[s]; Print[h];  ]
```

*Program of Phase 1 Algorithm*

In this section, we develop the code corresponding to the Fig-2: Phase-1 flowchart of Choo et al. Phase 1 algorithm.

```
PHASE[1_]:=Module[ {},
f[x_]:=Evaluate[Input["objective function"]];
   m1=Input["Enter t4he input"];
   h1=Input["Enter the input"];
α=0.2;δ=1;β=.1;    n=1000;
h[1]=h1;m[1]=m1;
Do[r[i]=h[i]+α (h[i]-m[i]);
If[f[h[i]]≤f[r[i]],
If[f[r[i]]≤f[m[i]],
m[i+1]=r[i];s[i+1]=m[i];h[i+1]=h[i],
c[i]=h[i]+ β (m[i]-h[i]);If[f[c[i]]≥f[h[i]],
m[i+1]=c[i];s[i+1]=r[i];h[i+1]=h[i],
m[i+1]=h[i];s[i+1]=m[i];h[i+1]=c[i]]],
e[i]=h[i]+(α+δ) (h[i]-m[i]);
If[f[e[i]]≥f[r[i]],
If[f[e[i]]≥f[h[i]],m[i+1]=h[i];
s[i+1]=e[i];h[i+1]=r[i],m[i+1]=e[i];
s[i+1]=h[i];h[i+1]=r[i]],
m[i+1]=r[i];s[i+1]=s[i];
h[i+1]=e[i]] ],{i,1,n}]
Print[m[n+1]] ;  Print[s[n]]; Print[h[n+1]] ]
```

*Program of Golden Section Algorithm*

In this section, we develop the code corresponding to the Golden Section algorithm.

```
BA[GOLDEN_]:=Module[{},
a=Input["Enter  Largest Left hand point for Golden Section method i.e. a"];
b=Input["Enter Largest Right hand point for Golden Section method i.e. b"];
ε=Input["Enter  Length of uncertainity"]
g[x_]=Evaluate[Input["Objective function"]];
Print["Initial Interval=[",a,",",b,"]"];    r=0.618;
pts=FindRoot[(b-a)*r^k□ε,{k,1}];
   m=Ceiling[k]/.pts;
Do[Label[st];x1=b-r*(b-a);
Label[th]; x2=a+r*(b-a);
If[g[x1]<g[x2],
Print["Iteration Number =",k," and it's interval of uncertainity is:"];
Print["[",x1,",", b,"]"];a=x1;Continue[th],
Print["Iteration Number =",k," and it's interval of uncertainity is:"];
Print["[",a,",",x2,"]"];b=x2;Continue[st]],{k,1,m}]]
```

*Program of Our Modified Algorithm*

In this section, we developed a generalized code corresponding to the modified algorithm.

```
MODIFIEDPHASE[O_]:=Module[{},
f[x_]:=Evaluate[Input["objective function"]];
   w1=Input["Enter t4he input"];
   b1=Input["Enter the input"];
α=0.2;δ=1;β=.1;    n=100;
If[f[b1]≤f[w1],  b[1]=b1;w[1]=w1;
c=b[1]+β (w[1]-b[1]);
If[f[c]≤f[b[1]],
m=b[1];h=c;s=w[1],
Do[r[i]=b[i]+α (b[i]-w[i]);
If[f[b[i]]≤f[r[i]],h=b[i];
If[f[w[i]]≤f[r[i]],
m=w[i];Print[i];s=r[i],m=r[i];Print[i];s=w[i],
e[i]=b[i]+(α+δ) (b[i]-w[i]);
If[f[r[i]]<f[e[i]],
w[i+1]=b[i];b[i+1]=r[i];r[i+1]=e[i];
 h=b[i+1];    If[f[w[i+1]]≤f[r[i+1]],
m=w[i+1];Print[i];s=r[i+1],m=r[i+1];Print[i];s=w[i+1]],w[i+1]=b[i]; b[i+1]=e[i];
m=w[i+1];  s=b[i+1];  ] ],{i,1,n}]],Print["jfdgfdg"]
]Print[m];Print[s];Print[h];]
PHASE[0_]:=Module[{},
f[x_]:=Evaluate[Input["objective function"]];
  w1=Input["Enter t4he input"];
  b1=Input["Enter the input"];
α=0.2;δ=1;β=.1;  n=100;
If[f[b1]≤f[w1], b[1]=b1;w[1]=w1;
c=b[1]+β (w[1]-b[1]);
If[f[c]≤f[b[1]],
m=b[1];h=c;s=w[1],
Do[r[i]=b[i]+α (b[i]-w[i]);
If[f[b[i]]≤f[r[i]],h=b[i];
If[f[w[i]]≤f[r[i]],
m=w[i];Print[i];s=r[i],m=r[i];Print[i];s=w[i],
e[i]=b[i]+(α+δ) (b[i]-w[i]);
If[f[r[i]]<f[e[i]],
w[i+1]=b[i];b[i+1]=r[i];r[i+1]=e[i];
h=b[i+1];If[f[w[i+1]]≤f[r[i+1]],
m=w[i+1];Print[i];s=r[i+1],m=r[i+1];Print[i];s=w[i+1]],w[i+1]=b[i];b[i+1]=e[i];
m=w[i+1];s=b[i+1]; ]],{i,1,n}]],Print["jfdgfdg"]]
 Print[m];
 Print[s];
 Print[h];  ]
PHASE[1_]:=Module[ {},
f[x_]:=Evaluate[Input["objective function"]];
m1=Input["Enter t4he input"];
h1=Input["Enter the input"];
α=0.2;δ=1;β=.1;   n=1000;
h[1]=h1;m[1]=m1;
Do[r[i]=h[i]+α (h[i]-m[i]);
If[f[h[i]]≤f[r[i]],If[f[r[i]]≤f[m[i]],
m[i+1]=r[i];s[i+1]=m[i];h[i+1]=h[i],
c[i]=h[i]+ β (m[i]-h[i]);
If[f[c[i]]≥f[h[i]],
m[i+1]=c[i];s[i+1]=r[i];h[i+1]=h[i],
m[i+1]=h[i];s[i+1]=m[i];h[i+1]=c[i] ]],
e[i]=h[i]+(α+δ) (h[i]-m[i]);
If[f[e[i]]≥f[r[i]],
If[f[e[i]]≥f[h[i]],m[i+1]=h[i];s[i+1]=e[i];h[i+1]=r[i],m[i+1]=e[i];
s[i+1]=h[i];h[i+1]=r[i]],
m[i+1]=r[i];s[i+1]=s[i];
h[i+1]=e[i]] ],{i,1,n}]
Print[m[n+1]] ;  Print[s[n]]; Print[h[n+1]] ]
```

*Program of Golden Section Algorithm*

In this section, we develop the code corresponding to the Golden Section algorithm.

```
BA[GOLDEN_]:=Module[{},
a=Input["Enter  Largest Left hand point for Golden Section method i.e. a"];  b=Input["Enter Largest Right hand point for Golden Section method i.e. b"];
ε=Input["Enter  Length of uncertainity"]
g[x_]=Evaluate[Input["Objective function"]];
Print["Initial Interval=[",a,",",b,"]"];
r=0.618;pts=FindRoot[(b-a)*r^k□ε,{k,1}];
m=Ceiling[k]/.pts;
Do[Label[st];x1=b-r*(b-a);
Label[th]; x2=a+r*(b-a);
If[g[x1]<g[x2],
Print["Iteration Number =",k," and it's interval of uncertainity is:"];
Print["[",x1,",", b,"]"];a=x1;Continue[th],
Print["Iteration Number =",k," and it's interval of uncertainity is:"];
Print["[",a,",",x2,"]"];b=x2;Continue[st]],{k,1,m}]
```

*Input and Output System*

In this section, we have shown run file Local kernel box to get the results. When we run the prescribed program and complete the required statement, we must press "Enter" individually for each required statement. Finally, we will get the desired results is the following way.

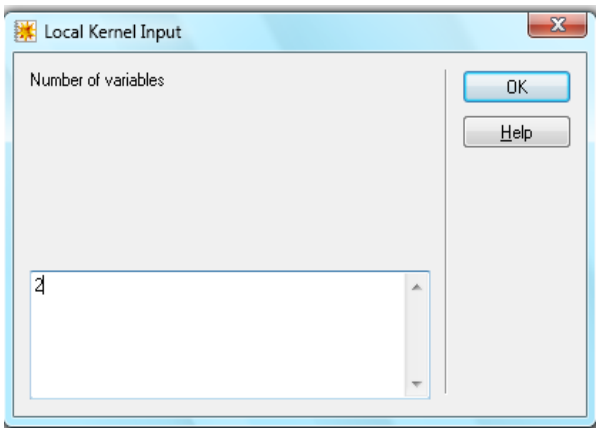


**Fig. 3.** Local Kernel Input Box

## V. Numerical Experiments

In this section, we present a number of numerical examples to show the efficiency of our computer techniques.

*Test Problem Number (TPN)*

*Example: 1 This numerical example is taken from Bazaraa et al*[8].

Find the extreme points of the function $62x^2 - 28x - 4$ ? Arbitrarily, we choose the initial interval $-12 \leq x \leq 5$.

*Example 2:* This numerical example is taken from Bazaraa et al[8].

Find the optimal solution to Minimize the function $(x - 2)4$. Arbitrarily we choose the interval $-4 \leq x \leq 4$.

*Example 3:* This numerical example is taken from Bazaraa et al[8].

Find the optimal solution to Minimize the function $4x^3 + 3x^4$. Arbitrary we choose the interval $-4 \leq x \leq 2$.

*Example 4:* This numerical example is taken from Das &Hasan[5].

Suppose that the function to be maximized is $f(x) = 12x - 3x^4 - 2x^6$. Arbitrary we choose the initial interval $-80 \leq x \leq 100$.

*Example 5:* This numerical example is taken from Das et al.[5].

Find the optimal solution to Max $x^2 + 2x$ . S/t.$-3 \leq x \leq 5$ within an initial length of 0.8.

## VI. Results and Discussion

In the present section, we present the comparison and discussion of the Choo et al.[3] methods and *Golden Section* methods and with their limitations. We also present the Ayoade[6] modified phase 0 with its effectiveness.

*Convergence of Different Methods*

In this section, we present the convergence of our algorithm using the test problems in Section 5. We also present the convergence of Choo et al. Phase 0, Phase 1 using computer algebra.

In the previous tables, we can say that, our 1-D simplex search is a powerful technique compared to the method of *Golden Section Search* and Choo et al. method. But if we use phase 0 or our modified phase-0 before using *Golden Section* then it will give good accuracy whereas *Golden section* method fails. It should be mentioned that *Golden Section Search* starts with the optimal point $\bar{x}$ ,that must lie between a and b that is, $\bar{x} \in [a, b]$ . If we consider, $\bar{x} \notin [a, b]$ "Golden Section Search" cannot be applied. While 1-D simplex search can be applied not only considering $\bar{x} \in [a, b]$ but also $\bar{x} \in [a, b]$ . From the above comparison and the computational results, we can say that our method is still one of the most robust and efficient method for solving 1-D NLP problems compared to the Choo et al.Phase-0 and Phase-1and Golden Section method.

**Table 1. Accuracy test with Different methods**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Accuracy test of the test Problem No-1 | | | | | | | | |
| Data | Phase-0 | N | Phase-1 | N | Modified Phase0 | N | Phase-1 | N |
| Input | PHASE [0] | 1 | PHASE [1] | 33 | MODIFIEDPHASE[O] | 1 | PHASE [1] | 21 |
| Output+ Solution | [5, 8.4] | | 0.225807 | | [3.3,5] | | 0.225807 | |
| Total Iteration of phase 0 & Phase 1 is 34 | | | | | Total Iteration of Modified & Phase 1 is 22 | | | |
| The comparison from TPN-1 show that modified phase-0 is stronger than phase-0. | | | | | | | | |
| Golden Section method Failed because the initial interval is not optimal. | | | | | | | | |
| Accuracy test of the test Problem No-2 | | | | | | | | |
| Data | Phase-0 | N | Phase-1 | N | Modified Phase0 | N | Phase-1 | N |
| Input | PHASE [0] | 1 | PHASE [1] | 14 | MODIFIEDPHASE[O] | 1 | PHASE [1] | 13 |
| Output+ Solution | [1, 5.6] | | 2 | | [3.2,4] | | 2 | |
| Total Iteration of phase 0 & Phase 1 is 15 | | | | | Total Iteration of Modified & Phase 1 is 14 | | | |
| The comparison from TPN-1 show that modified phase-0 is stronger than phase-0. | | | | | | | | |
| Golden Section method Failed because the initial interval is not optimal. | | | | | | | | |
| Accuracy test of the test Problem No-3 | | | | | | | | |
| Data | Phase-0 | N | Phase-1 | N | Modified Phase0 | N | Phase-1 | N |
| Input | PHASE [0] | 2 | PHASE [1] | 19 | MODIFIEDPHASE[O] | 2 | PHASE [1] | 19 |
| Output+ Solution | [-2, 0.88 | | -1 | | [-2, 0.88] | | -1 | |
| Total Iteration of phase 0 & Phase 1 is 21 | | | | | Total Iteration of Modified & Phase 1 is 21 | | | |
| The comparison from TPN-1 show that modified phase-0 and Phase-0results is coinciding. | | | | | | | | |
| Golden Section method Failed because the initial interval is not optimal. | | | | | | | | |

## VII. Conclusion

In this paper, we developed computer techniques to verify Choo and Kim's method for solving 1-D NLP problems. We also constructed codes corresponding to the Ayoade's modified phase-0. Finally, we compared the 1-D two phase methods and Ayoade's modified Phase-0 with the Golden section method. We demonstrated the efficiency of our computer techniques with a number of numerical examples.

## References


1. Powell, M. J. D., 1986. Convergence properties of algorithms for nonlinear optimization, *SIAM*, Rev.
2. Nelder, J. A, and, R. Meal, 1965. A Simplex method for function Minimization, Computer Journal, **7**, 308-313.
3. Choo E. & Kim, 1987. One Dimensional Simplex search, Computer and Operations Research, **14**, 47-54.
4. Xue G., 1989. One the convergence of one dimensional simplex search, Computer & Operations Research, **16**, 113-116.
5. Das H. K., & M. B. Hasan, 2013. A Generalized Computer technique for solving unconstrained NLP problems, Dhaka Univ. J. Sci, **61(**1), 75-80.
6. Ayoade K., 1991. Numerical Experiments with 1-D Non-linear Simplex search, Computers & Operations Research, **18**, 497-506.
7. Kiefer, J., 1957. Optimum sequential search and approximation methods under minimum regularity assumptions, J. Soc. Indust. Appl. Math., **5(3),** 105-136.
8. Bazaraa M.S., H. D. Sherali and Shetty, 2003. NLP theory and Applications, Canada., 343-462.
9. Wolfram, S., Mathematica, 2001. Addision-Wesly, Company, New York., 201-250
10. Winston, W. L., 1994. Application and algorithms, Duxbury press, U.S.A., 610-695.
11. Don, E ., 2001. Theory and Problems of Mathmatica, Schaum's Outline Series., 70-80.