Pergamon

PII: S0098-1354(96)00000-0

Computers chem. Engng Vol. 22, No. 1-2, pp. 323–331, 1998
Copyright © 1997 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0098-1354/97 $17.00+0.00

Comparing software for interactive solution of systems of nonlinear algebraic equations

Mordechai Shacham[a], Neima Brauner[b] and Michael Pozin[b]

[a] Department of Chemical Engineering, Ben-Gurion University of the Negev,Beer-Sheva, 84105 Israel
[b] School of Engineering, Tel-Aviv University, Tel-Aviv, 69978 Israel

(Received 18 May 1994; revised 30 November 1995)

Abstract

Four general purpose interactive numerical solution packages: MAPLE, MATLAB, MATHEMATICA and POLYMATH are compared with regard to their performance in solving systems of nonlinear algebraic equations typical to chemical engineering applications.

Criteria for ease of learning, ease of use and user-friendliness of the packages are introduced. Benchmark problems are used to demonstrate that the selection of good initial estimates is still a critical issue in solving highly nonlinear problems with any of these programs. It is concluded that of the programs tested, the performance of MATHEMATICA and POLYMATH is the best for systems of NLEs, the interval of convergence of POLYMATH and MAPLE is the widest for a single NLE and POLYMATH possesses the most "user-friendly" features. © 1997 Elsevier Science Ltd

1. Introduction

Many practical problems in chemical engineering require the solution of systems of nonlinear algebraic equations (NLEs). Typical examples are chemical and phase equilibrium calculations and steady state material and energy balances. One of the authors (Shacham, 1985) published a comparison of software for solution of NLEs about ten years ago, in which the performance of the programs were compared on the basis of how many of the benchmark problems the program can solve and time required to reach the solution.

In the last ten years the development of interactive programs has progressed to the level where the equations serve as input for the programs. Contrary to the FORTRAN programs in use ten years ago, these programs include an editor as an integral part of the program, and there is no need to recompile and link the subroutines between runs. These programs are very convenient to use for small and medium scale problems, containing up to several tens of equations. When using such interactive programs, the criteria utilized previously by Shacham, 1985 for comparison between programs are no longer suitable and must be revised. For example, the number of iterations or the time required to reach the solution, is not important, as long as the user does not have to wait for the solution to appear on the screen. Thus, the user's time, not the computer or program development time, becomes the main consideration.

The dependability or robustness of the NLE solver program, defined as a function of the number of problems the program can solve from a given set of benchmarks, relative to other NLE solvers, is still very important. Boston *et al.*, 1993 predicted that by the year 2001, combinations of NLE solvers and expert systems will exist which will be able to solve any system of NLEs, provided that a solution exists. In Section 2, we try to assess how close the presently available interactive programs are from reaching this goal.

The present state of the art requires that the user provide an initial set of estimates for the unknowns. Depending on the level of the nonlinearity of the problem, selection of the initial estimates may be critical or non-critical. Some considerations for selecting initial estimates will be discussed in Section 3.

A very important consideration is "user-friendliness". The degree of "user-friendliness" is the key for converting the NLE solving programs from an expert's tool to a tool available and utilized by all engineers.

How can "user-friendliness" be measured and compared? Slaughter *et al.*, 1991 compared several NLE solvers for "ease of learning" and "ease of use" but the comparison was based on the subjective feelings of the evaluator. We believe that such comparison should be based upon objective criteria, such as flexible notation, online debugging, equation sorting, error detection, etc. In Section 4 we will introduce and demonstrate some of the features which, in our opinion, must exist in an NLE solver in order to be classified "user-friendly".

Four software packages were used for the comparison: MAPLE (Ellis *et al.*, 1992), MATHEMATICA (Wolfram, 1991), MATLAB (The Math Works, 1992) and POLYMATH 3.0 (Shacham and Cutlip, 1994). The PC DOS versions of MAPLE and POLYMATH were used and the Unix version of MATHEMATICA. Two versions of MATLAB were tested, MATLAB 3.5 PC DOS version and MATLAB 4.0 Unix version (or PC Windows). We will report only the results obtained using the Unix version, since its performance was observed to be superior.

All these packages are capable of doing much more than solve NLEs, but only this application was compared. Although there are several additional interactive packages that can be used for the same purpose (Slaughter *et al.*, 1991, Rosen, 1989) and most spread sheets can also be applied (Rosen and Adams, 1987), we have limited our study to only those packages where the solution is done on a single command, no programming or macros are required. The four packages studied were selected because they were available, but the criteria and benchmark problems presented in this paper can be applied to extend the comparision to additional software packages.

2. Performance of the 4 packages for a set of benchmark problems

In order to test the capability of the different programs to solve typical chemical engineering problems of different levels of nonlinearity and difficulty, we have used a set of 12 test problems. Five of them consist of a single implicit nonlinear equation and are taken from Shacham (1989) and seven additional problems, which include systems of equations, are taken from Shacham (1984). Several of the examples included in the benchmark set have been discussed extensively in the literature. The set includes a system of equations representing material and energy balances on a reactor from Carnahan *et al.*, 1969 and 2 chemical equilibrium problems from Hiebert, 1982. Some of the problems are badly scaled and possess singular points, or intervals in the vicinity of the solution where some functions are undefined. Several of them have multiple solutions, but only some of the solutions are physically feasible. More details for the benchmark problems can be found in the references Shacham, 1984, 1989.

Table 1 summarizes the performance of the packages for the benchmark set of single nonlinear equations. The equations in examples 3 and 5 are continuous with continuous derivatives, and the derivatives do not change sign over a wide interval near the root. None of the programs had any difficulty in converging from inside the interval shown in the table.

The function of Example 7 is undefined for $X \geq 0.8$. Nevertheless, POLYMATH and MAPLE converged from the entire feasible region: ($0 \leq X < 0.799$), but the interval of convergence of MATLAB and MATHEMATICA became very narrow ($\sim 0.64 < X < 0.79$).

The equation in Example 10 has two roots, and it is undefined for $X \leq 0$ or $X \geq 0.95$. POLYMATH and MAPLE found both solutions from the feasible interval ($0.01 \leq X \leq 0.99$), and MATLAB and MATHEMATICA converged to one root at a time, depending on the initial estimate specified. None of the programs had any difficulty in solving Example 13.

Table 2 shows the performance of the different programs for systems of NLEs. The example number given in the table is as it appears in Shacham (1984). All the examples (except No. 7) were solved from different initial estimates and two of them (Examples 8 and 10) were solved for different parameter sets. MAPLE could not be tested from different initial estimates, because it requires that the intervals be specified for all the unknowns within which the solution is expected to be located.

The performance of MATHEMATICA was the best for this set of test problems. It converged to a solution in 18 out of the 23 cases without any modification of the equations. In two cases it gave incorrect solutions without delivering any error message and in three cases

Table 1. Performance of the packages for a single nonlinear algebraic equation

example no.[a]	Root	Feasible region	Interval of convergence			
			POLYMATH	MAPLE	MATLAB	MATHEMATICA
3	T=551.77	$T>0$	500–800	500–800	500–800	500–800
5	T=4305.31	$T>0$	3000–5000	3000–5000	3000-5000	3000–5000
7	X=0.7574	$0 \leq X < 0.8$	0–0.799	0–0.799	0.64–0.76	0.63–0.79
10.1	X=0.5	$0 < X < 0.95$	0.01–0.94	0.01–0.94	0.28–0.79	0.23–0.94
10.2	X=0.03621	$0 < X < 0.55$	0.01–0.94	0.01–0.94	0.03–0.18	0.01–0.22
13	v=0.0757	$v \geq 0$	0–1	0–1	0–1	0–1

[a] Example number as it appears in Shacham (1989).

the solutions found were physically infeasible (negative mole fractions).

POLYMATH arrived at a solution in 16 out of the 23 cases and all the solutions reached were physically feasible. In two cases user intervention was required in order to arrive at the solution: In Example 7 there are 14 equations and the number of equations was reduced to 12 (the upper limit in POLYMATH) by solving two linear equations for two unknowns, and then substituting these values into the other equations. In Example 10 the solution for one of the parameter sets had to be restarted after introducing a partial solution as a new initial estimate (a built-in option in POLYMATH). In all cases, where POLYMATH did not converge to the solution, it was clearly indicated by an error message and display of intermediate results.

Since initial estimates cannot be changed for MAPLE, only 10 cases were solved by this program. It converged to a feasible solution in only 6 of the 10 cases. In two (out of the 6) cases, the equations had to be modified prior to arriving at the solution. In both Examples 7 and 8, the square of the functions had to be used in order to converge. In Example 7, there was a need to specify a very narrow interval around the solution as initial guess in order to achieve convergence. MAPLE gave error messages in most cases when it failed to converge, but in example 9 it failed without giving an error message.

MATLAB converged to the feasible solution in 9 out of the 23 cases. In the other cases it converged to a local minimum, without giving any indication that the solution found was not the root of the system of equations.

3. Considerations in selecting an initial estimate

All programs tested require specification of initial estimates for the unknowns by the user. The initial estimates may be provided as a single value for each one of the unknowns (POLYMATH, MATHEMATICA, MATLAB) or as an interval within which the solution is expected to be found (MAPLE, POLYMATH for single equation).

While it is always advisable to use initial estimates based on physical bounds, in most problems the selection of the initial estimates is not critical, as long as ranges of infeasible or absurd values are excluded (such as negative or greater than one for mole fraction, negative absolute temperature, etc.). However, the casual selection of initial estimates may not be good enough (for highly nonlinear problems). Some of the difficulties associated with selection of initial estimates are demonstrated by the following three examples.

The first example is a chemical equilibrium problem taken from Shacham and Cutlip, 1997. The equations for the example are shown in Appendix A and Table 3

Table 2. Performance of the programs for systems of NLEs

Example no.[a]	Parameter set	Initial guess	POLYMATH	MAPLE[b]	MATLAB	MATHEMATICA
2	–	1	+	+	+	+
2	–	2	+		—	*
5	–	1	–	+	—	–
5	–	2	+		—	–
5	–	3	–		+	+
5	–	4	+		+	+
6	–	1	+	+	+	+
6	–	2	+		+	+
7	–	1	–+	+	+	+
8	1	1	+	–+	—	+
8	1	2	+		—	*
8	2	1	+	–	—	+
8	2	2	+		—	*
8	3	1	+	–	—	+
8	3	2	–	—	—	–
9	–	1	+	—	—	+
9	–	2	–		—	+
9	–	3	–		—	—
9	–	4	–		—	+
10	1	1	+	–+	+	+
10	1	2	+		+	+
10	2	1	–+	–	+	—
10	2	2	–		—	+

[a] Example number, parameter set and initial guess as appear in Shacham (1984).

[b] MAPLE requires an interval defined as initial estimate, no specific point can be defined.

+ converged to a feasible solution.

– did not converge to solution with error message.

— did not converge to solution without error message.

* converged to an infeasible solution.

–+ converged to a feasible solution after modification.

Table 3. Performance of 3 packages for Example 1, from different initial estimates

Initial guess no.	C_D	C_X	C_Z	POLYMATH	MATLAB	MATHEMATICA
1	0.61	0.2	0.4	+	–	–
2	0.59	0.2	0.4	–	–	–
3	0.7	0.28	0.4	+	–	+
4	0.7	0.31	0.4	–	+	+
5	0.7	0.2	0.48	+	+	+
6	0.7	0.2	0.51	–	+	–
Solution	0.7053	0.1778	0.3740			

+ Converged to solution.
– Did not converge.

summarizes the solution and the convergence to the solution for this example as obtained from 6 different starting points. MAPLE was not included in this comparison since it requires specification of an interval instead of a single point.

It can be seen that even though all sets of initial estimates are very close to the solution, nevertheless, for some of them all of the programs fail to converge. The reason for this difficulty is that the concentration of C_C may become zero for different combinations of C_D, C_X and C_Z near the solution and none of the programs is able to cross this barrier of singularity. The problem can be made much less nonlinear and easier to solve by eliminating division by unknowns: f_1 can be multiplied by C_AC_B, f_2 by C_CC_B and f_3 by C_AC_X, to obtain the following modified set of equations:

$$\begin{aligned} &f_1(C_D, C_X, C_Z)=C_CC_D-1.06C_AC_B=0,\\ &f_2(C_D, C_X, C_Z)=C_XC_Y-2.63C_CC_B=0, \qquad (1)\\ &f_3(C_D, C_X, C_Z)=C_Z-5C_AC_X=0. \end{aligned}$$

In this form, the functions do not have any discontinuities. Indeed, all 4 programs converged to the right solution from a reasonable initial estimate, such as $C_D=C_X=C_Z=0$ (assuming no reaction takes place). The selection of initial estimates may be critical even for this formulation. Using POLYMATH with two different initial guesses, the program converged to two additional solutions, which are physically infeasible (some of the concentrations are negative). The initial estimates and additional solutions are shown in Table 4.

Example 2 is a system of equations representing steady state material and energy balances in a chemical reactor (taken from Fogler, 1974 and the equations are detailed in Shacham, 1984). The two unknowns in this equation set are X (the conversion) and T (the absolute temperature). This system of equations is difficult to solve because it is badly scaled and there are local minima with a singular Jacobian matrix in the region of interest.

The solution is at $X^*=0.53337$ and $T^*=1637.7$. The local minima are at $X=0.22484$; $T=1477.9$ and $X=0.016569$; $T=1368.7$. All programs had difficulty converging to the solution when the system is formulated as two implicit equations, regardless of the initial estimates used. They did converge after certain manipulations of the equations. POLYMATH and MATLAB required rescaling of the equations. With MAPLE, the square of the 2nd function had to be used and MATHEMATICA required close initial estimates and restriction of the unknowns within tight intervals. MATLAB converged in some cases to a local minimum.

The system can be modified to alleviate the problem by converting it to a single implicit equation in terms of X. Searching for a solution inside the interval $0 \le X \le 0.9$, POLYMATH and MAPLE found the solution without any difficulties. MATLAB and MATHEMATICA require a single initial estimate. When the initial estimate was set to $X \le 0.5$, both programs converged to incorrect solutions; MATLAB converged to $X=0.1972$ and MATHEMATICA to $X=0.1068$. Only with initial estimate in the interval $0.53337 > X > 0.5$ did these two programs converge to the correct solution.

Example 3 (taken from Henley and Rosen, 1969, see Appendix B) involves the calculation of the bubble point

Table 4. Initial estimates and solutions for the chemical equilibrium problem

Variable	Solution 1		Solution 2		Solution 3	
	Initial estimate	Solution	Initial estimate	Solution	Initial estimate	Solution
C_D	0	0.7053	1	0.0556	10	1.0702
C_X	0	0.1778	1	0.5972	10	−0.3225
C_Z	0	0.3740	1	1.0821	10	1.1304
C_A	—	0.4207	—	−0.3624	—	−0.7007
C_B	—	0.2429	—	−0.2348	—	0.808
C_C	—	0.1536	—	−1.6237	—	−0.3782
C_Y	—	0.5518	—	1.6793	—	0.2623

Table 5. Performance of the programs for Example 3

Initial estimate	X_{11}	X_{21}	X_{12}	X_{22}	POLYMATH	MATLAB	MATHEMATICA
1	0	1	1	0	+	+	+
2	0.05	0.95	1	0	+	+	+
3	0.1	0.9	1	0	–	+	–
4	0.2	0.8	1	0	–	+	–
5	0	1.0	0.5	0.5	+	+	+
6	0	1.0	0.3	0.7	–	+	–
Solution[a]	0.0227	0.9773	0.6867	0.3133			

[a] For β initial estimate 0.8; solution 0.7330.
For t initial estimate 80; solution 88.538.
+ Converged to the solution; – Did not converge.

temperature for a nonideal mixture. At the bubble point this mixture forms two liquid phases. Table 5 summarizes the performance of the different programs (except MAPLE) for this example with 6 different initial estimates. It can be seen that the convergence of all the programs is very much dependent on the initial estimate. A small change in the initial estimate for X_{11} can cause all the programs (except MATLAB) to fail. We could not find a formulation for this example which would render the system less sensitive to the initial estimate.

4. Program-user interface

In the past very little attention was paid to the man–machine interface of the equation solver program. In interactive programs, the user interface actually determines the extent to which the target users population can benefit from the software. Some general criteria for evaluating the user interface follow.

Menu based program control is considered more user friendly than **command based program control** because the options appear on the screen and the user does not have to remember the particular commands. This makes the program much easier to learn and to use. The advantages of menu based control are most obvious when the number of options available is relatively small. When the number of options increases, there is a need for several "levels" of menus. Only one level can be displayed on the screen at one time. In such cases, there is no obvious advantage to menu based control over command based control. From among the programs tested, POLYMATH provides menu based control, while MATLAB, MATHEMATICA, and MAPLE use command based control for solution of NLEs.

The notation and format used in equation entry will be demonstrated using the chemical equilibrium problem (Example 1). Figure 1 shows the equation and initial estimate input for the different programs.

POLYMATH required minimum alteration of the original problem statement. POLYMATH requires that only one of the unknowns will appear on the left-hand side of an equation and it must be unique. MAPLE also allows one to use the same notation as given in the problem definition. There are no special rules for the left-hand side of implicit equations: they are treated just as any ordinary variable. A union of these variables, as well as of the unknowns, must be defined before solution, and the initial estimate for a variable must be given as an interval. MATHEMATICA requires the definition of the functions in a form similar to that used in programming languages. The definition of a new variable (as a function of other variables) is in a statement which is similar to function statement in programming languages (like FORTRAN). On the left-hand side, the name of the new dependent variable is written, while the name of the independent variable(s) must appear inside brackets with an underline added to the end of the name. The format required by MATLAB is very similar to the structure of a subroutine in programming languages. The variables are transferred in and out of this subroutine in a one dimensional array form.

All of the programs contain certain **debugging aids**. They check for syntax errors after the equation is entered and the return key pressed. POLYMATH also displays the number of equations and variables that have been defined and lists the undefined variables. This feature, which can be very helpful in detecting mispelled variable names, does not exist in any of the other programs.

An example from Shacham and Cutlip (1997) will be used to compare the different programs with regard to **equation ordering and detection of implicit relationships**. This example includes a system of equations representing calculations of bubble and dew point temperatures for a nonideal binary mixture (see Appendix C). The equations are written in the order in which mathematical models are usually developed. Equations representing general principles are written first. The equations which relate the principal variables y_i to the unknown temperature follow in a logical order.The calculations, however, are carried out in the opposite direction.

Upon introducing the set of equations for the bubble point to POLYMATH, the program ordered the equations as shown in Fig. 2. The solution obtained is $T=82.845$ and the function value $f(T)=-0.16\times10^{-5}$. MATHEMATICA and MAPLE did not inform the user that the equations were reordered, but the correct solution was reached with no regard to the order in which the equations were entered. MATLAB found incorrect solutions and, apparently, calculates the equations exactly in the order they have been entered (using variables before values were assigned to them).

In order to calculate the dew point temperature for the same mixture, *a priori*, it appears that only the first five equations must be altered (as shown in Appendix C). POLYMATH is used to solve these five altered equa-

A. MAPLE

```
> ca0:=1.5; cb0:=1.5;
> eps:=0.05;

> ca:=ca0-cd-cz;
> cy:=cx+cz;
> cb:=cb0-cd-cy;
> cc:=cd-cy;

> f1:=cc*cd/(ca*cb)-1.06;
> f2:=cx*cy/(cc*cb)-2.63;
> f3:=cz/(ca*cx)-5;
> f:={f1,f2,f3}:
> var:={cd,cx,cz};
> Xinterval:={ cd=0..1,
cx=0.15..0.25, cz=0..1 };

> root:=fsolve(f,var,Xinterval);
```

B. MATHEMATICA

```
ca0=1.5;
cb0=1.5;
f1[cd_,cx_,cz_] := (cc[cd,cx,cz]*cd) /
            (ca[cd,cz]*cb[cd,cx,cz]) - 1.06;
f2[cd_,cx_,cz_] := cx*cy[cx,cz] /
            (cc[cd,cx,cz]*cb[cd,cx,cz]) - 2.63;
f3[cd_,cx_,cz_]  := cz/(ca[cd,cz]*cx)-5;
ca[cd_,cz_]      := ca0-cd-cz;
cy[cx_,cz_]      := cx+cz;
cb[cd_,cx_,cz_]  := cb0-cd-cy[cx,cz];
cc[cd_,cx_,cz_]  :=    cd-cy[cx,cz];

root=FindRoot [
{f1[cd,cx,cz], f2[cd,cx,cz],f3[cd,cx,cz]},
{cd,0.7}, {cx,0.2}, {cz,0.48},
MaxIterations->50 ];
```

C. MATLAB

```
global ca0; global cb0;
ca0=1.5; cb0=1.5;
cd0=0.7; cx0=0.2; cz0=0.48;
x0=[cd0 cx0 cz0]';

root=fsolve('cheq',x0)
%--------------------------------
function f=cheq(x);
global ca0; global cb0;
cd=x(1); cx=x(2); cz=x(3);

ca=ca0-cd-cz;
cy=cx+cz;
cb=cb0-cd-cy;
cc=cd-cy;

f(1)=(cc*cd)/(ca*cb)-1.06;
f(2)=cx*cy/(cc*cb)-2.63;
f(3)=cz/(ca*cx)-5;
```

D. POLYMATH

The equations:

```
 f(cd)=(cc*cd)/(ca*cb)-1.06
 f(cx)=cx*cy/(cc*cb)-2.63
 f(cz)=cz/(ca*cx)-5
 ca=ca0-cd-cz
 cb=cb0-cd-cy
 cc=cd-cy
 cb0=1.5
 cy=cx+cz
 ca0=1.5
```

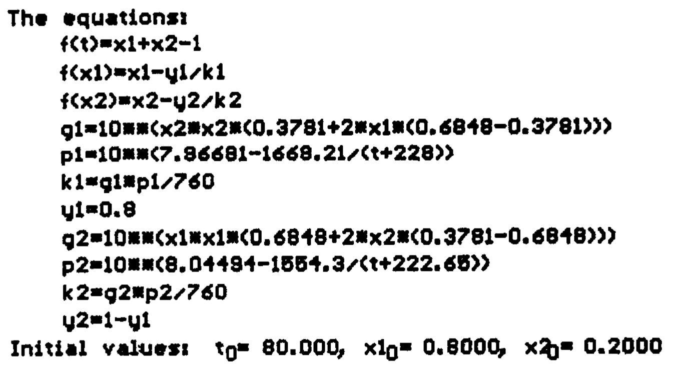
Initial values:

$cd_0=0.7$, $cx_0=0.2$, $cz_0=0.48$

Fig. 1. Equations and initial estimate input for example 1.

```
The equations:
    f(t)=y1+y2-1
    x1=0.80
    x2=1-x1
    g1=10**(x2*x2*(0.3781+2*x1*(0.6848-0.3781)))
    p1=10**(7.96681-1668.21/(t+228))
    k1=g1*p1/760
    y1=k1*x1
    g2=10**(x1*x1*(0.6848+2*x2*(0.3781-0.6848)))
    p2=10**(8.04494-1554.3/(t+222.65))
    k2=g2*p2/760
    y2=k2*x2
Search range: tmin= 80.000, tmax= 120.00
```

Fig. 2. Equations ordered by POLYMATH for calculation of bubble point temperature.



```
The equations:
     f(t)=x1+x2-1
     f(x1)=x1-y1/k1
     f(x2)=x2-y2/k2
     g1=10**(x2*x2*(0.3781+2*x1*(0.6848-0.3781)))
     p1=10**(7.86681-1668.21/(t+228))
     k1=g1*p1/760
     y1=0.8
     g2=10**(x1*x1*(0.6848+2*x2*(0.3781-0.6848)))
     p2=10**(8.04494-1554.3/(t+222.65))
     k2=g2*p2/760
     y2=1-y1
Initial values:  t0= 80.000,  x10= 0.8000,  x20= 0.2000
```

Fig. 3. POLYMATH equations input for calculation of the dew point temperature.

tions, an error message will indicate that some of the explicit equations require simultaneous solution. The set of equations must be rewritten as shown in Fig. 3 in order to calculate the dew point. In this case, three implicit equations are required, as opposed to the single implicit equation for calculating the bubble point temperature. The solution obtained using the set of equations in Fig. 3 are: $T=94.519$; $X_1=0.9748$ and $X_2=0.0252$.

Using the one implicit equation formulation, all other programs failed without being able to identify the reason for the failure. Using the 3-implicit equation formulation, MAPLE, MATHEMATICA and MATLAB all achieved the correct solution.

Once a solution is reached, **the root must be verified and multiple solutions detected**. For verification, the values of the unknown must be introduced into the functions to yield values close to zero. In POLYMATH the function value is displayed both graphically and numerically. In the other three programs the user must explicitly request both calculation and display of the variable or function values at the solution. For an inexperienced user this may mean accepting incorrect results.

For a system of equations all the programs will find only one root out of several possible solutions. For one nonlinear equation, POLYMATH will display and locate all the roots (up to 5 roots) which are inside the interval specified by the user, while the other programs will find only one solution from each initial guess.

5. Conclusions

The four packages tested are adequate for solving NLEs arising in chemical engineering in an interactive mode.

For solving one implicit nonlinear equation, POLYMATH and MAPLE had the widest convergence interval for the set of test problems that were used. For systems of nonlinear equations, MATHEMATICA reached the solution in the largest number of cases, closely followed by POLYMATH. It should be noted, however, that MATHEMATICA, MAPLE, and MATLAB converged occasionally to points which are not solutions of the system of equations, but local minima, without issuing any error message.

All the programs require the user to input initial estimates for the unknowns, and it has been shown that the selection of initial estimate can be critical for highly nonlinear problems. For such problems even an initial estimate close to the solution can lead to non-convergence, or convergence to a physically infeasible solution. In some cases, the problems can be transformed to make them less sensitive to the initial estimate used, by converting them to a single implicit equation or by eliminating division by unknowns. If the problem can be converted to a single implicit equation, POLYMATH and MAPLE will find at least one solution located inside the interval (in the case of multiple solutions, POLYMATH may find up to five solutions).

According to the criteria introduced in this study, POLYMATH is the most user friendly software for solving NLEs, among the programs tested. It has menu based program control, reorders the equations according to the calculation sequence, detects implicit relationship of variables, has debugging aids not available in the other programs, provides visual and numerical feedback regarding the accuracy of the solution and requires minimal alteration of notation and format of the equation set during input to the program. Introducing such features into the other programs can make them more user friendly, thereby making them more widely accepted by the engineering community and not only by specialists.

Appendix A:

Example 1, Chemical Equilibrium Problem (Shacham and Cutlip, 1997)

$$f_1(C_D, C_X, C_Z)=\frac{C_C C_D}{C_A C_B}-1.06=0,$$

$$f_2(C_D, C_X, C_Z)=\frac{C_X C_Y}{C_C C_B}-2.63=0,$$

$$f_3(C_D, C_X, C_Z)=\frac{C_Z}{C_A C_X}-5=0,$$

$$C_A=C_{AO}-C_D-C_Z,$$

$$C_B=C_{BO}-C_D-C_Y,$$
$$C_C=C_D-C_Y,$$
$$C_Y=C_X+C_Z,$$
$$C_{AO}=1.5,$$
$$C_{BO}=1.5.$$

Appendix B:

Example 3, Bubble and dew point temperature for a 20% (mol) isobutanol (1) and 80% water (2) mixture (Henley and Rosen, 1969)

$$f_1(\beta,x_{1,1},x_{2,1},x_{1,2},x_{2,2},t)=(x_{1,1}-x_{1,2})+(x_{2,1}-x_{2,2}),$$
$$f_2(\beta,x_{1,1},x_{2,1},x_{1,2},x_{2,2},t)=x_{1,1}-\frac{0.2}{\left[\beta+(1-\beta)\frac{k_{1,1}}{k_{1,2}}\right]},$$
$$f_3(\beta,x_{1,1},x_{2,1},x_{1,2},x_{2,2},t)=x_{1,2}-x_{11}\frac{k_{1,1}}{k_{1,2}},$$
$$f_4(\beta,x_{1,1},x_{2,1},x_{1,2},x_{2,2},t)=x_{2,1}-\frac{0.8}{\left[\beta+(1-\beta)\frac{k_{2,1}}{k_{1,1}}\right]},$$
$$f_5(\beta,x_{1,1},x_{2,1},x_{1,2},x_{2,2},t)=x_{2,2}-x_{2,1}\frac{k_{2,1}}{k_{2,2}},$$
$$f_6(\beta,x_{1,1},x_{2,1},x_{1,2},x_{2,2},t)=x_{1,1}(1-k_{1,1})+x_{2,1}(1-k_{2,1}),$$
$$\log P_1=7.62231-\frac{1417.09}{191.15+t},$$
$$\log P_2=8.10765-\frac{1750.29}{235+t},$$
$$\log\gamma_{1,1}=\frac{1.7x_{2,1}^2}{\left[\frac{1.7x_{1,1}}{0.7}+x_{2,1}\right]^2},$$
$$\log\gamma_{2,1}=\frac{0.7x_{1,1}^2}{\left[x_{1,1}+\frac{0.7x_{2,1}}{1.7}\right]^2},$$
$$\log\gamma_{1,2}=\frac{1.7x_{2,2}^2}{\left[\frac{1.7x_{1,2}}{0.7}+x_{2,2}\right]^2},$$
$$\log\gamma_{2,2}=\frac{0.7x_{1,2}^2}{\left[x_{1,2}+\frac{0.7x_{2,2}}{1.7}\right]^2},$$
$$k_{1,1}=\frac{\gamma_{1,1}P_1}{760};\ k_{2,1}=\frac{\gamma_{2,1}P_2}{760},$$
$$k_{1,2}=\frac{\gamma_{1,2}P_1}{760};\ k_{2,2}=\frac{\gamma_{2,2}P_2}{760}.$$

Appendix C:

Example 4, Bubble and dew point temperature for a 80% water (1) and 20% ethanol (2) mixture (Shacham and Cutlip, 1997)

Bubble Point	Dew Point
$f(t)=y_1+y_2-1=0$	$f(t)=x_1+x_2-1=0$
$y_1=k_1x_1$	$x_1=y_1/k_1$
$y_2=k_2x_2$	$x_2=y_2/k_2$
$x_1=0.8$	$y_1=0.8$
$x_2=0.2$	$y_2=0.2$

$$k_i=\gamma_iP_i/P \qquad i=1,2,$$

$$\log(P_1)=7.96681-1668.21ct(t+228.0),$$
$$\log(\gamma_1)=x_2^2[0.3781+2x_1(0.6848-0.3781)],$$
$$\log(P_2)=8.04494-1554.3/(t+222.65),$$
$$\log(\gamma_2)=x_1^2[0.6848+2x_2(0.3781-0.6848)].$$

References

Boston, J. F., Britt, H. I. and Tayyabkhan, M. T. (1993) Software: Tackling tougher tasks. *Chem. Eng. Progr.* **89**(11), 38–49.

Carnahan, B, Luther, H. A. and Wilkes, J. O. (1969) *Applied Numerical Methods*, p. 321. Wiley, New York.

Ellis, W. Jr., Johnson, E., Lodi, E. and Schwalbe, D. (1992) *MAPLE V flight manual*. Brooks/Cole Pub. Co., Pacific Grove, CA.

Fogler, H. S. (1974) *The Elements of Chemical Kinetics and Reactor Calculations (A Self-Paced Approach)*. Prentice-Hall, New Jersey.

Henley, E. J. and Rosen, E. M. (1969) *Material and Energy Balance Computation*. John Wiley, New York.

Rosen, E. M. (1989) Some experiences with TK SOLVER PLUS. *CAST Communications* **12**(2), 16

Rosen, E. M. and Adams, R. N. (1987) A review of spreadsheet usage in chemical engineering calculations. *Computers Chem. Engng* **11**(6), 723–736.

Hiebert, K. L. (1982) An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Trans. Math. Softw.* **8**(1), 5–20.

Shacham, M. (1984) Recent developments in solution techniques for systems of nonlinear equations. In *Proceedings of the Second International Conference on Foundations of Computer Aided Design*, eds Westerberg, A. W. and Chien, H. H. pp. 891-923, CACHE Publications, Ann Arbor, MI.

Shacham, M. (1985) Comparing software for the solution of systems of nonlinear algebraic equations arising in chemical engineering. *Computers Chem. Engng* **9**(2), 103–112.

Shacham, M. (1989) An improved memory method for solution of a nonlinear equation. *Chem. Eng. Sci.* **44**(7), 1495–1501.

Shacham, M. and Cutlip, M. B. (1994) *POLYMATH 3.0 Users' Manual*. CACHE Corporation, Austin, TX.

Shacham, M. and Cutlip, M. B. (1997) *Numerical Solution of Chemical Engineering Problems Using POLYMATH*. Prentice–Hall, Inc., Upper Saddle River, New Jersey.

Slaughter, J. M., Petersen, J. N. and Zollars, R. L. (1990) Use of PC based mathematics software in the undergraduate curriculum. *Chem. Eng.* Ed. **25**(1), 54–60.

MATLAB for Unix Computer's Users' Guide (1992) The Math Works, Inc., Natick, MA.

Wolfram, S. (1991) *MATHEMATICA, A System for Doing Mathematics by Computer*, 2nd ed. Addison Wesley, NY.