Enhancing Computer-Based Problem Solving Skills with a Combination of Software Packages

Mordechai Shacham
Dept. of Chemical Engineering, Ben Gurion University of the Negev
Beer-Sheva 84105, Israel
e-mail: shacham@bgumail.bgu.ac.il

Michael B. Cutlip
Dept. of Chemical Engineering, University of Connecticut
Storrs, CT 06269, USA
e-mail: michael.cutlip@uconn.edu

Introduction

It is generally recognized that computer-based (or computer-enhanced) problem solving (CBPS) can be a very important or possibly the most important application of the computer in engineering education and practice. However, the penetration of CBPS in the various engineering disciplines has been disappointingly slow and of limited extent. (Kantor and Edgar[2], Jones[3]) The mistaken premise that one software package is appropriate for all problem solving is possibly one of the major impediments to CBPS. There are selected engineering courses where the computer is used as a sophisticated calculator that can solve differential and implicit algebraic equations and also perform regressions on experimental data. In other cases, the graphical and the database management capabilities of the software are important. Still other courses require considerable programming abilities.

We have long advocated that the most educational benefit can be gained by the integration of several software packages throughout the curriculum (Shacham and Cutlip[4]). There is a need for a numerical problem solver that accepts the model equations close to their mathematical forms and provides their numerical solution with very minimal user intervention. Spreadsheets are used widely for organization and presentation of information in tabular and graphical forms and for database management with related operations. Software packages that support programming (such as MATLAB, for example) are needed to implement algorithms are required in advanced programming and numerical analysis courses.

In our experience, most students find it rather difficult to convert mathematical models of even medium complexity to the format required by spreadsheets and by programming languages. This initial difficulty may prevent them from using spreadsheets and programming languages in an advanced level. This barrier to the extensive use of CBPS can be minimized if the software can be utilized for converting the equations from their mathematical form into the format required by spreadsheets and programming languages.

In this paper we will describe a numerical software package called POLYMATH[A] that we have developed over the last 20 years that accepts the equations close to their mathematical forms and provides their numerical solutions. The latest version of this software also automatically outputs the equations in the format acceptable to a spreadsheet program, Excel[B], and a software package that supports programming, MATLAB[C]. This automated generation of completely operational Excel and MATLAB programs greatly enhances the teaching of computing and numerical while simultaneously introducing the students to widely used software packages. The potential application of POLYMATH with these recently developed capabilities will be demonstrated by several examples in engineering education.

Numerical Methods Courses

Courses involving the fundamentals of "Numerical Methods" are taught in most engineering disciplines. These courses can consider the fundamental algorithms involved in numerical problem solving and can also provide engineering students with many good examples that illustrate the use of several software packages for effective CBPS. The examples provided in most textbooks of numerical methods are often unrelated to the problems that have to be solved in the student's particular field. The use of realistic problems in a student's engineering discipline can increase their motivation in numerical problem solving and develop his/her ability to select the appropriate numerical method to be used with the particular problem under consideration.

In this paper we will demonstrate, using two examples, the potential benefits of the use of the POLYMATH package to code and debug the mathematical model of the problem in hand. The POLYMATH model can then be easily converted to an Excel worksheet or to a MATLAB function. These packages can then be used to carry out the repetitive and/or iterative calculations required by the particular numerical method.

Example 1 - Calculation of the Flow Rate in a Pipeline Using Successive Substitution

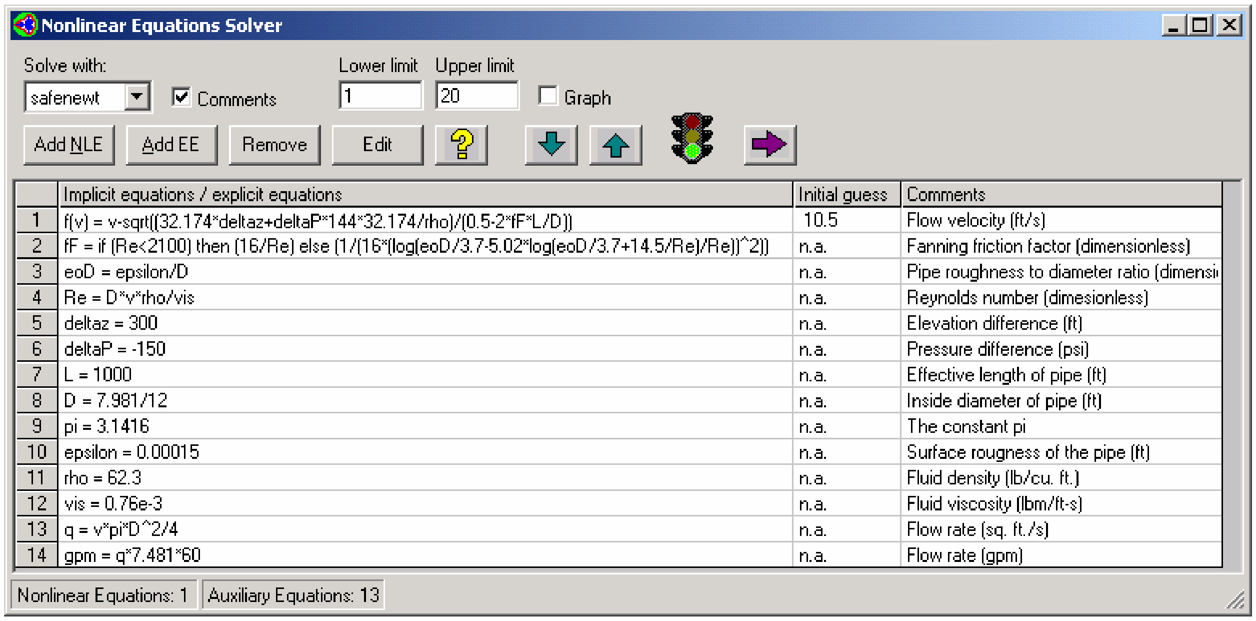
This example is based on Problem 5.10 presented by Cutlip and Shacham[1]. The detailed problem statement is shown in Appendix A. The problem involves calculation of flow velocity and flow rate in a pipeline configuration for a large number of combinations of pipe lengths and diameters. The results should be presented in tabular and graphical forms. The solution for one set of pipe diameter and length values involves solution of a nonlinear equation, the general mechanical energy balance on an incompressible fluid, where the friction factor is function of the Reynold's

[A] POLYMATH is copyrighted by M. Shacham, M. B. Cutlip and M. Elly (http://www.polymath-software.com/)
[B] Excel is a trademark of Microsoft Corporation (http://www.microsoft.com)
[C] MATLAB is a trademark of The MathWorks, Inc. (http://www.mathworks.com/)

number (thus the flow velocity). The successive substitution method has long been used for solving this type of problems (even when graphical solution techniques were used), and it is known that this technique converges very fast for flow velocity calculations. The successive substitution method can be conveniently programmed with Excel, but the direct input of the model equations into Excel can be a tedious and error-prone process. A better approach for students is to first enter the equations into POLYMATH that requires minimal changes in the naming of the variables, solve the equations for one sets of parameter values, and then compare the results with the solution provided in the problem statement. After this is completed, an option within POLYMATH can be used to convert the set of equations into a spreadsheet within Excel that can immediately be used to also obtain a problem solution.

The equations, as entered into the POLYMATH program, are shown in Figure 1. Students find the "user friendly" features of POLYMATH very helpful while entering the equations: The notation used in the equation entry is almost the same as in the problem definition (except that no Greek characters can be used). POLYMATH issues warnings for undefined variables so that errors such as using the letter o in the variable name (like in eoD) in one equation and the number 0 in another equation can be easily detected. The needed equations can be entered in the same order as they appear in the problem definition even if the calculation order must be different since POLYMATH reorders the equations when the calculations are made. For example, the necessary calculation for the friction factor, f_F, first involves ε/D and then Reynolds number, *Re*. However POLYMATH allows direct entry of the equations in the same order as they are defined in the problem statement.

Common mistakes in entering the model equations typically involve inconsistency in the units used (i.e. inches instead of feet), wrong sign of the pressure or elevation difference, and inappropriate use of parenthesis in Equations (A-1) and (A-4). POLYMATH enables much easier detection of errors in the model equations because the model debugging stage is clearly separated from the numerical solution stage.

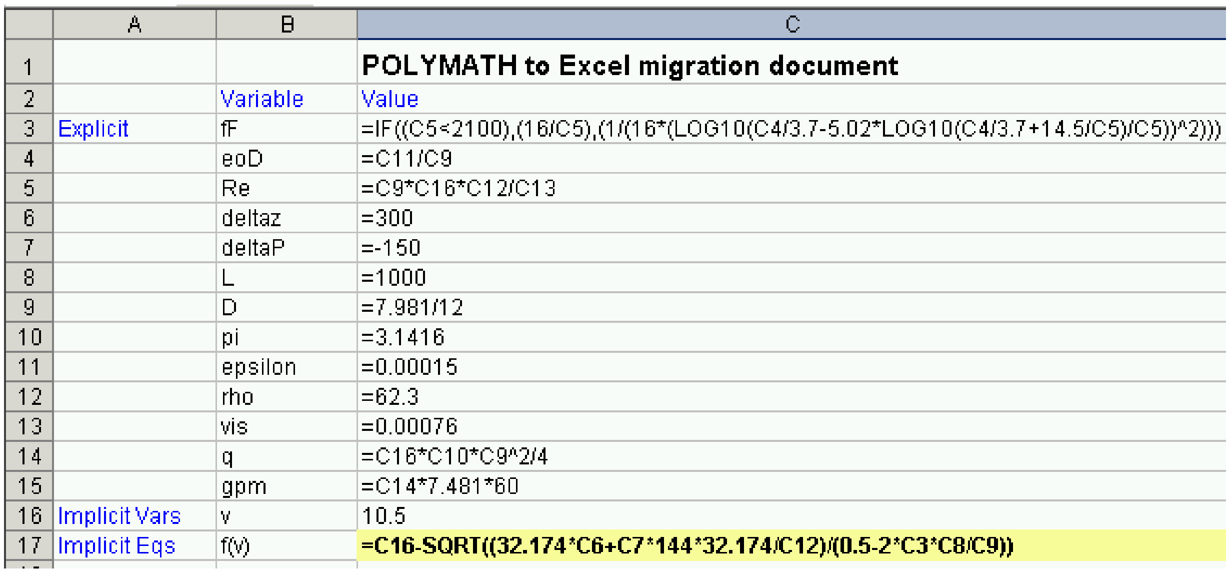


	Implicit equations / explicit equations	Initial guess	Comments
1	f(v) = v-sqrt((32.174*deltaz+deltaP*144*32.174/rho)/(0.5-2*fF*L/D))	10.5	Flow velocity (ft/s)
2	fF = if (Re<2100) then (16/Re) else (1/(16*(log(eoD/3.7-5.02*log(eoD/3.7+14.5/Re)/Re))^2))	n.a.	Fanning friction factor (dimensionless)
3	eoD = epsilon/D	n.a.	Pipe roughness to diameter ratio (dimensi
4	Re = D*v*rho/vis	n.a.	Reynolds number (dimesionless)
5	deltaz = 300	n.a.	Elevation difference (ft)
6	deltaP = -150	n.a.	Pressure difference (psi)
7	L = 1000	n.a.	Effective length of pipe (ft)
8	D = 7.981/12	n.a.	Inside diameter of pipe (ft)
9	pi = 3.1416	n.a.	The constant pi
10	epsilon = 0.00015	n.a.	Surface rougness of the pipe (ft)
11	rho = 62.3	n.a.	Fluid density (lb/cu. ft.)
12	vis = 0.76e-3	n.a.	Fluid viscosity (lbm/ft-s)
13	q = v*pi*D^2/4	n.a.	Flow rate (sq. ft./s)
14	gpm = q*7.481*60	n.a.	Flow rate (gpm)

Nonlinear Equations: 1 Auxiliary Equations: 13

Figure 1 - POLYMATH Equation Entry for Example 1

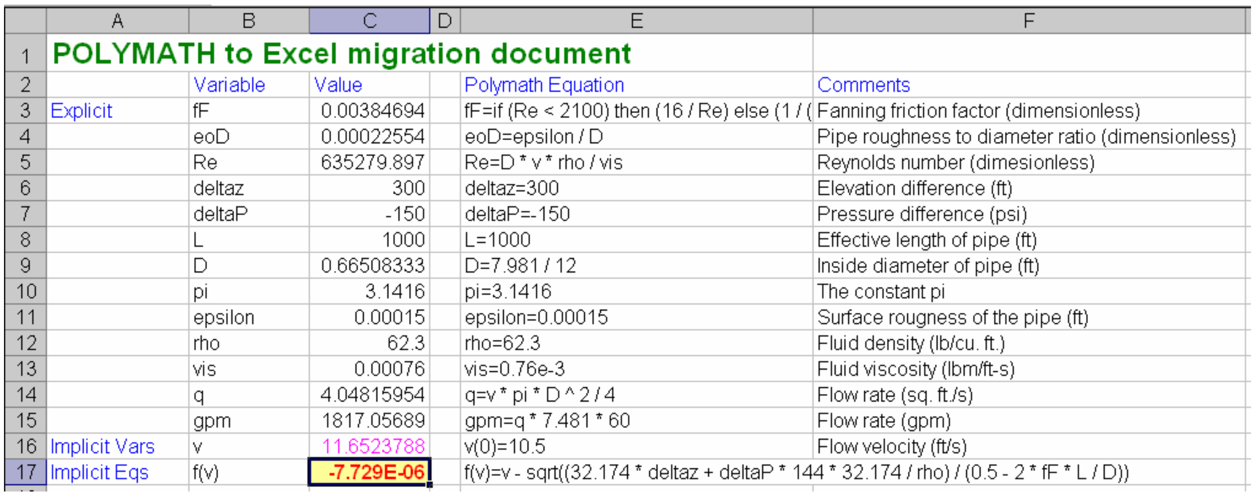
After the correct solution as given in the problem statement is obtained, the model equation set can be converted to an Excel worksheet using a single command within POLYMATH. Part of the Excel worksheet generated is shown in Figure 2 where the variable cell calculations are indicated.



	A	B	C
1			**POLYMATH to Excel migration document**
2		Variable	Value
3	Explicit	fF	=IF((C5<2100),(16/C5),(1/(16*(LOG10(C4/3.7-5.02*LOG10(C4/3.7+14.5/C5)/C5))^2)))
4		eoD	=C11/C9
5		Re	=C9*C16*C12/C13
6		deltaz	=300
7		deltaP	=-150
8		L	=1000
9		D	=7.981/12
10		pi	=3.1416
11		epsilon	=0.00015
12		rho	=62.3
13		vis	=0.00076
14		q	=C16*C10*C9^2/4
15		gpm	=C14*7.481*60
16	Implicit Vars	v	10.5
17	Implicit Eqs	f(v)	**=C16-SQRT((32.174*C6+C7*144*32.174/C12)/(0.5-2*C3*C8/C9))**

Figure 2 - POLYMATH Equations Converted to Excel Formulas for Example 1

The variable names are translated to cell addresses, intrinsic function names are changed as necessary, and the syntax of the *if* statement is changed. The equations are rearranged in a form that appropriate for solving the equation using the *goal seek* or *solver* tools available within Excel. The complete worksheet with the solution obtained using *goal seek* is shown in Figure 3.

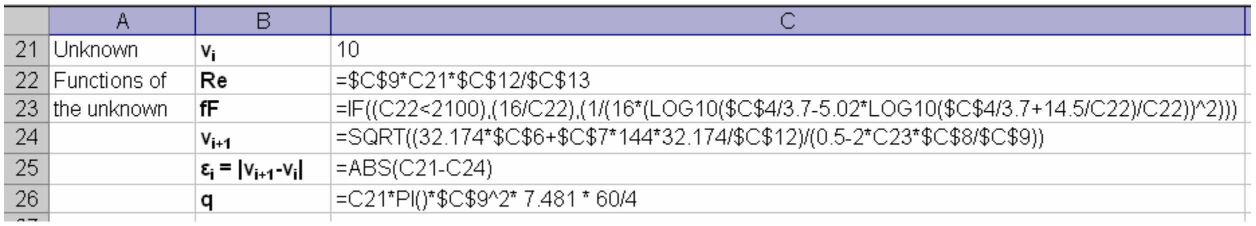


	A	B	C	D	E	F
1	**POLYMATH to Excel migration document**					
2		Variable	Value		Polymath Equation	Comments
3	Explicit	fF	0.00384694		fF=if (Re < 2100) then (16 / Re) else (1 / (	Fanning friction factor (dimensionless)
4		eoD	0.00022554		eoD=epsilon / D	Pipe roughness to diameter ratio (dimensionless)
5		Re	635279.897		Re=D * v * rho / vis	Reynolds number (dimesionless)
6		deltaz	300		deltaz=300	Elevation difference (ft)
7		deltaP	-150		deltaP=-150	Pressure difference (psi)
8		L	1000		L=1000	Effective length of pipe (ft)
9		D	0.66508333		D=7.981 / 12	Inside diameter of pipe (ft)
10		pi	3.1416		pi=3.1416	The constant pi
11		epsilon	0.00015		epsilon=0.00015	Surface rougness of the pipe (ft)
12		rho	62.3		rho=62.3	Fluid density (lb/cu. ft.)
13		vis	0.00076		vis=0.76e-3	Fluid viscosity (lbm/ft-s)
14		q	4.04815954		q=v * pi * D ^ 2 / 4	Flow rate (sq. ft./s)
15		gpm	1817.05689		gpm=q * 7.481 * 60	Flow rate (gpm)
16	Implicit Vars	v	11.6523788		v(0)=10.5	Flow velocity (ft/s)
17	Implicit Eqs	f(v)	-7.729E-06		f(v)=v - sqrt((32.174 * deltaz + deltaP * 144 * 32.174 / rho) / (0.5 - 2 * fF * L / D))	

Figure 3 - Excel Worksheet with Numerical Results and Documentation for Example 1

The numerical results are identical to those obtained by POLYMATH, and thus the correctness of the Excel solution has been verified. The variable names in column C, the POLYMATH equations in column E, and the variable descriptions in column F provide complete documentation for the Excel formulas in column C.

The successive substitution method can be implemented by revising the equations that are functions of the unknown flow velocity *v* and adding Equation (A-6) for estimating the error in the current value of v_i as shown below.



	A	B	C
21	Unknown	v_i	10
22	Functions of	**Re**	=$C$9*C21*$C$12/$C$13
23	the unknown	**fF**	=IF((C22<2100),(16/C22),(1/(16*(LOG10($C$4/3.7-5.02*LOG10($C$4/3.7+14.5/C22)/C22))^2)))
24		v_{i+1}	=SQRT((32.174*$C$6+$C$7*144*32.174/$C$12)/(0.5-2*C23*$C$8/$C$9))
25		$\varepsilon_i = \lvert v_{i+1}-v_i \rvert$	=ABS(C21-C24)
26		**q**	=C21*PI()*$C$9^2* 7.481 * 60/4

Arranging the variables in consecutive columns, copying and pasting them in consecutive rows and substituting v_{i+1} in the cell that contains v_i starting iteration No. 1 yields the desired solution as obtained by the successive substitution method (see Table 1).

Table 1 - Successive Substitution Iterations for Example 1

Iteration. No. i	v_i	Re	f_F	v_{i+1}	$\varepsilon_i = \lvert v_{i+1}-v_i \rvert$	q
1	10	545193.3	0.003892	11.58185	1.581854	1559.383
2	11.58185	631434.9	0.003849	11.64971	0.067853	1806.055
3	11.64971	635134.2	0.003847	11.65229	0.002578	1816.636
4	11.65229	635274.8	0.003847	11.65238	9.75E-05	1817.038
5	11.65238	635280.1	0.003847	11.65239	3.69E-06	1817.053

After the correct solution has been obtained for one set of pipe length and diameter values the "Two Input Table" option of Excel can be conveniently used for carrying out all the calculations that required in part (b) of the problem statement in Appendix A. The plot of flow velocity for all combinations of pipe length and diameter is shown in Figure 4.

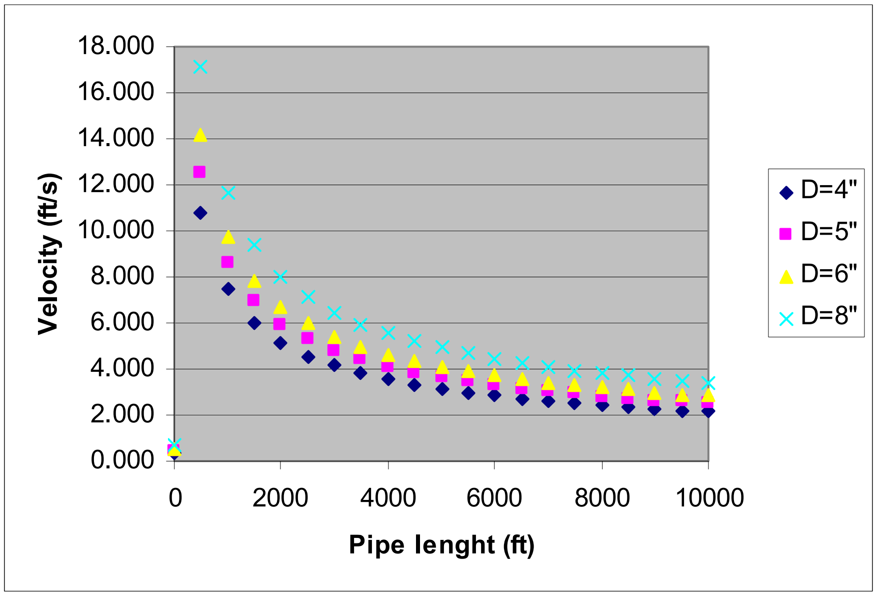


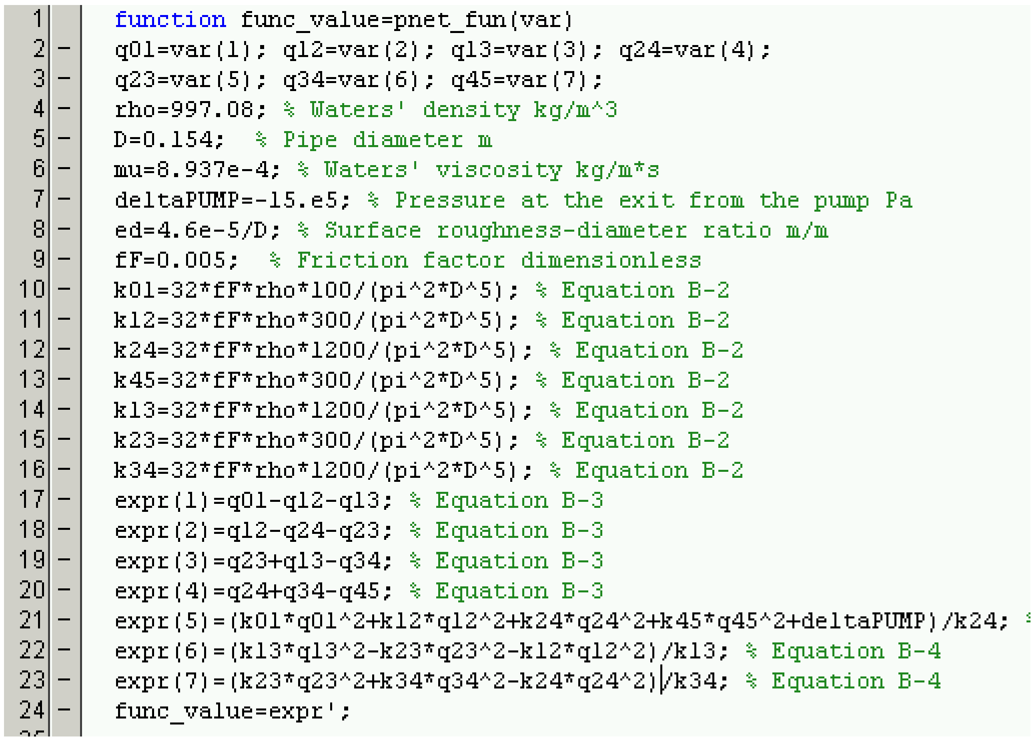
Figure 4 - Flow Velocity versus Pipe Length and Diameter for Example 1

Example 2 - Flow Distribution in a Pipeline Network Solved by Broyden's Method

This example is based on Problem 5.11 presented by Cutlip and Shacham[1]. The detailed problem statement is shown in Appendix B. The problem involves calculation of flow rates and pressure drops in a pipeline network that includes seven interconnected pipe segments. The problem can be brought into a form of seven algebraic equations (where three of them are nonlinear) with seven unknown pipe-segment flow rates. The use of Broyden's Quasi-Newton method for this type of problems is attractive because it has super-linear convergence and it does not require calculations of the matrix of partial derivatives. The various matrix-matrix and matrix-vector multiplications required by this method can be easily carried out when MATLAB is used for implementation. Students may find this method difficult to program if another programming package is used that does not support algebraic expressions that contain matrices and matrix operations.

In this example it is very important to verify that the mathematical model is correct and has a solution before attaching to it the numerical solution technique, because the Broyden's' method may not converge to the solution even if the problem does have a solution. Some potential causes for no convergence will be demonstrated in this example.

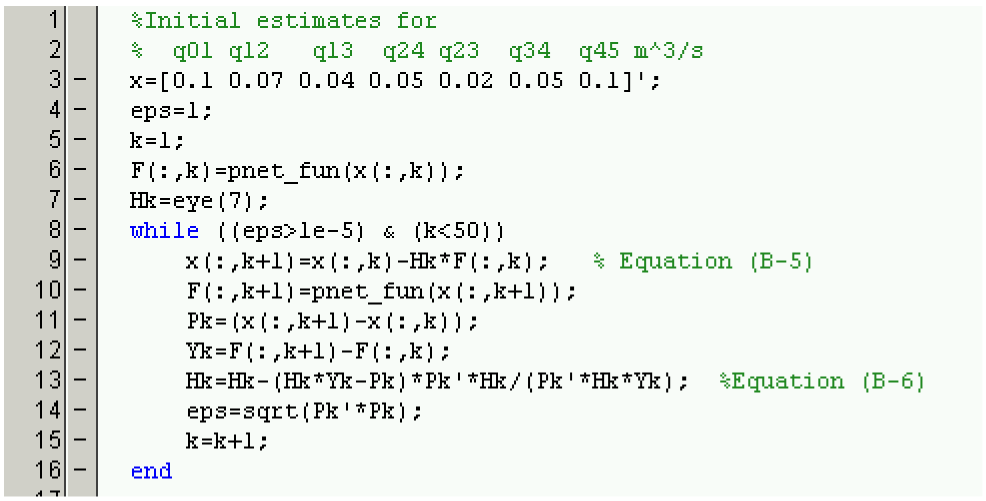
The model equations, as entered into the POLYMATH nonlinear algebraic equation solver program are shown in p. 188 of Cutlip and Shacham[1]. The problem can be solved with POLYMATH when using the initial estimates shown in the problem statement without any difficulties. The POLYMATH equations can be easily converted to a MATLAB function. We have used an automatic conversion routine to do that but the conversion can be done easily by hand also. The MATLAB function that was obtained by the conversion routine and edited in order to bring it to a more compact form is shown in Figure 5.



```
function func_value=pnet_fun(var)
q01=var(1); q12=var(2); q13=var(3); q24=var(4);
q23=var(5); q34=var(6); q45=var(7);
rho=997.08; % Waters' density kg/m^3
D=0.154;  % Pipe diameter m
mu=8.937e-4; % Waters' viscosity kg/m*s
deltaPUMP=-15.e5; % Pressure at the exit from the pump Pa
ed=4.6e-5/D; % Surface roughness-diameter ratio m/m
fF=0.005;  % Friction factor dimensionless
k01=32*fF*rho*100/(pi^2*D^5); % Equation B-2
k12=32*fF*rho*300/(pi^2*D^5); % Equation B-2
k24=32*fF*rho*1200/(pi^2*D^5); % Equation B-2
k45=32*fF*rho*300/(pi^2*D^5); % Equation B-2
k13=32*fF*rho*1200/(pi^2*D^5); % Equation B-2
k23=32*fF*rho*300/(pi^2*D^5); % Equation B-2
k34=32*fF*rho*1200/(pi^2*D^5); % Equation B-2
expr(1)=q01-q12-q13; % Equation B-3
expr(2)=q12-q24-q23; % Equation B-3
expr(3)=q23+q13-q34; % Equation B-3
expr(4)=q24+q34-q45; % Equation B-3
expr(5)=(k01*q01^2+k12*q12^2+k24*q24^2+k45*q45^2+deltaPUMP)/k24; %
expr(6)=(k13*q13^2-k23*q23^2-k12*q12^2)/k13; % Equation B-4
expr(7)=(k23*q23^2+k34*q34^2-k24*q24^2)/k34; % Equation B-4
func_value=expr';
```

Figure 5 - MATLAB Function Representing the Pipeline Network of Example 2

The main program for solving the pipeline network problems using Broyden's method is shown in Figure 6. Note that only the computational commands are shown, the input output commands were removed for brevity.



```
%Initial estimates for
%  q01 q12   q13  q24 q23  q34  q45 m^3/s
x=[0.1 0.07 0.04 0.05 0.02 0.05 0.1]';
eps=1;
k=1;
F(:,k)=pnet_fun(x(:,k));
Hk=eye(7);
while ((eps>1e-5) & (k<50))
    x(:,k+1)=x(:,k)-Hk*F(:,k);   % Equation (B-5)
    F(:,k+1)=pnet_fun(x(:,k+1));
    Pk=(x(:,k+1)-x(:,k));
    Yk=F(:,k+1)-F(:,k);
    Hk=Hk-(Hk*Yk-Pk)*Pk'*Hk/(Pk'*Hk*Yk);  %Equation (B-6)
    eps=sqrt(Pk'*Pk);
    k=k+1;
end
```

Figure 6 - MATLAB Main Program for Solving Example 2 Using Broyden's Method

Note that Broyden's method, as implemented in Figure 6 diverges from the initial estimates specified in Appendix B. One reason for that is the order of magnitude differences between the functions associated with Equation (B-3) and those associated with Equation (B-4). The Equations (B-3) are comprised of terms of the order of 0.1 while the Equations (B-4) are comprise of terms of the order of 10^6.

The equations can be rescaled by dividing the Equations (B-3) by the largest k_{ij} associated with the particular equation. This rescaling was already carried out in the function shown in Figure 5. However, Broyden's method diverges even after rescaling the equations. To validate the solution technique an initial estimate, closer to the solution is selected, based on the solution obtained by POLYMATH. From this initial estimate Broyden's method converges to the correct solution shown in Table 2. But even when starting from an initial estimate close to the solution the convergence of Broyden's method is non-monotonic as can be seen in Figure 7. This Figure shows the logarithm of the error norm (defined in Appendix B) as function of the iteration number. The error norm gets reduced for several iterations and increases once again until, finally, gets close enough to the solution to achieve monotonic convergence. Sophisticated nonlinear equation solver packages use a one-dimensional search in order to prevent the increase of the error norm and this is a good example to demonstrate the need for such modification of the solution algorithm.

Table 2 - Initial Estimates and Solutions for Pipeline Network Flow-rates

	Initial Estimate	Solution
q_{01} (m^3/s)	0.1	0.098134
q_{12} (m^3/s)	0.07	0.06482
q_{13} (m^3/s)	0.04	0.033314
q_{24} (m^3/s)	0.05	0.049372
q_{23} (m^3/s)	0.02	0.015449
q_{34} (m^3/s)	0.05	0.048763
q_{45}(m^3/s)	0.1	0.098135

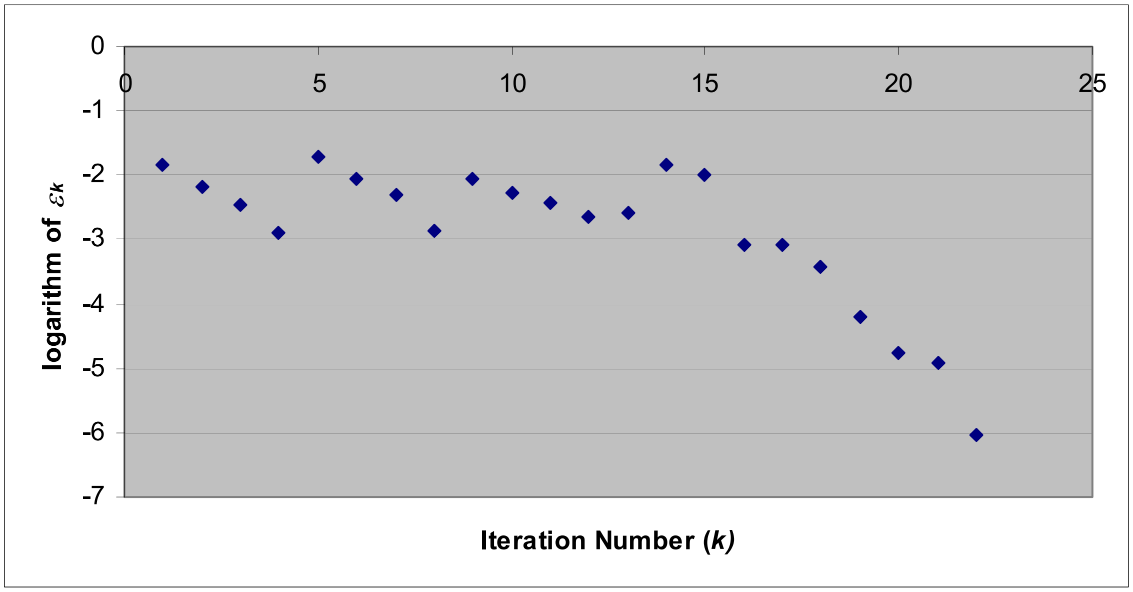


Figure 7 - Convergence Pattern of the Broyden's Method for Example 2

Conclusions

We have demonstrated, using two practical examples from the fluid mechanics field several benefits of the use of a number of software packages in engineering problem solving. When teaching numerical methods the preparation, coding and debugging of the mathematical model can be separated from the coding and debugging of the solution algorithm by using a software package that takes care of the technical details of the numerical solution. This enables solving complex, realistic problems increasing the students' motivation to learn CPBS techniques.

Once a correct, verified solution has been obtained for a particular model a different software package, more appropriate for coding the numerical solution algorithm can be used. The existence of a solution for the problem can alleviate considerably the solution algorithm and can help to identify weak points of the basic algorithm such as the need to rescale the functions and variables, start the iterations from points close to the solution or carry out one dimensional search for minimum.

Example 1, in slightly modified form was presented as a final exam question to the students of a "Process Modeling and Numerical Methods" course in the Chemical Engineering Department of the Ben-Gurion University. Using two software packages (POLYMATH and MATLAB) for solution most of the students managed to solve correctly the problem in two hours. This clearly demonstrates the benefits of the use of several software packages simultaneously to enable solving realistic problems by computer in reasonable timeframes.

Bibliography

(1) Cutlip, M. B. and Shacham, M., Problem *Solving In Chemical Engineering with Numerical Methods*, Prentice-Hall, Upper Saddle River, New-Jersey, 1999.

(2) Jones J. B., "The Non-Use of Computers in Undergraduate Engineering Science Courses", *Journal of Engineering Education*, Vol. 88, 1998, pp. 11-14.

(3) Kantor, J. C. and T. Edgar, "Computing Skills in the Chemical Engineering Curriculum", pp. 9-20 in Carnahan, B. (Ed), *Computers in Chemical Engineering Education*, CACHE, Austin, Texas, 1996.

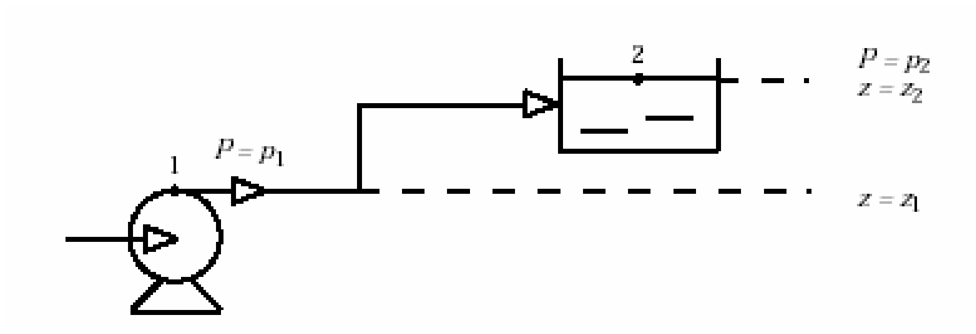
(4) Shacham, M. and M.B. Cutlip, “A Comparison of Six Numerical Software Packages for Educational Use in the Chemical Engineering Curriculum”, *Computers in Education Journal*, Vol. IX, No. 3, 1999, pp. 9-15

Appendix A

Problem Statement for Example 1

Calculation of the Flow Rate in a Pipeline

The Figure below shows a pipeline which delivers water at constant temperature $T = 60$ °F from point 1 where the pressure is $p_1 = 150$ psig and the elevation is $z_1 = 0$ ft to point 2 where the pressure is atmospheric and the elevation is $z_2 = 300$ ft.



(a) Calculate the flow rate q (in gal/min) for a pipeline with effective length of L = 1000 ft and made of nominal 8-inch diameter schedule 40 commercial steel pipe. (Solution: v = 11.65 ft/s, q = 1817 gpm)
(b) Calculate the flow velocity and flow rate for pipelines with effective length of L = 500, 1000, … 10000 ft and made of nominal 4,5,6 and 8-inch schedule 40 commercial steel pipe. Use the successive substitution method for solving the equations for the various cases and present the results in tabular form. Prepare plots of flow velocity versus D and L and flow rate versus D and L.

Equations and numerical data

The general mechanical energy balance on an incompressible liquid that applicable to this case is the following

$$-\frac{1}{2}v^2 + g\Delta z + \frac{g_c \Delta P}{\rho} + 2\frac{f_F L v^2}{D} = 0 \qquad \text{(A-1)}$$

where v is the flow velocity in ft/s, g is the acceleration of gravity given by $g = 32.174$ ft/s^2, $\Delta z = z_2 - z_1$ is the difference in elevation (ft), g_c is a conversion factor (in English units $g_c = 32.174$ ft·lbm/lbf·s^2), $\Delta P = P_2 - P_1$ is the difference in pressure lb$_m$/ft^2), ρ is the fluid density (for water at $T = 60$ °F, $\rho = 62.3$ lb$_m$/ft^3), f_F is the Fanning friction factor, L is the length of the pipe (ft) and D is the inside diameter of the pipe (ft). To use the Successive Substitution method equation (A-1) should be rewritten:

$$v = \sqrt{\frac{g\Delta z + \dfrac{g_c \Delta P}{\rho}}{0.5 - 2\dfrac{f_F L}{D}}} \tag{A-2}$$

The equation used for calculating the Fanning friction factor depends on the flow regime. The flow regime is characterized by the Reynold's number Re. The Reynold's number is a dimensionless number $Re = v\rho D/\mu$ where μ is the viscosity (for water at $T = 60$ °F, $\mu = 0.76\times10^{-3}$ lb$_m$/ft·s). For laminar flow (Re < 2100) the Fanning friction factor can be calculated from the equation

$$f_F = \frac{16}{Re} \tag{A-3}$$

For turbulent flow (Re > 2100) the Shacham[D] equation can be used.

$$f_F = \frac{1}{16\left\{\log\left[\dfrac{\varepsilon/D}{3.7} - \dfrac{5.02}{Re}\log\left(\dfrac{\varepsilon/D}{3.7} + \dfrac{14.5}{Re}\right)\right]\right\}^2} \tag{A-4}$$

where ε/D is the surface roughness of the pipe ($\varepsilon = 0.00015$ ft for commercial steel pipes).

The flow velocity in the pipeline can be converted to flow rate by multiplying it by the cross section of the pipe, thus $q = v\pi D^2/4$ (ft^3/s). The inside diameter (D) of commercial steel pipes can be found, for example, in Table D-5 of the book by Cutlip and Shacham (1999). The iteration function of the successive substitution method for calculation of the flow velocity is

$$v_{i+1} = F(v_i) \quad i = 0, 1, \ldots \tag{A-5}$$

where i is the iteration number, F is the function in the right side of Equation (A-2) and v_0 is an initial estimate for the flow velocity (use $v_0 = 10$ ft/s). Equation (A-6) provides an error estimate at iteration i:

$$\hat{\varepsilon}_i = | v_i - v_{i+1} | \tag{A-6}$$

The iterations can be stopped when $\hat{\varepsilon}_i < 10^{-5}$.

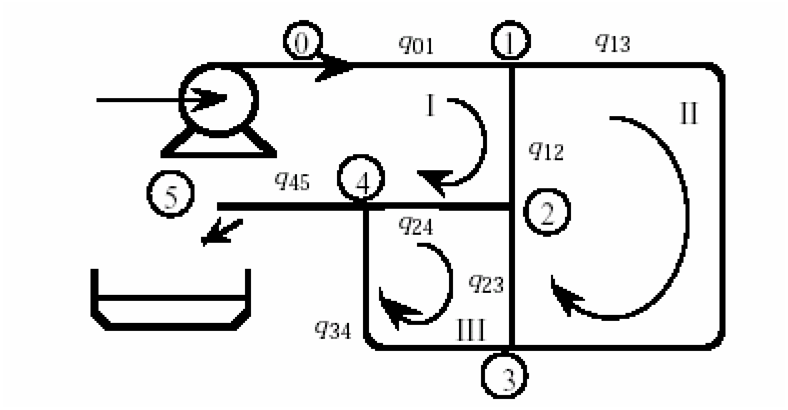
[D] Shacham, M. *Ind. Eng. Chem. Fund., 19*, 228-229(1980)

Appendix B

Problem Statement for Example 2

Flow Distribution in a Pipeline Network

Water at 25 °C is flowing in the pipeline network given in Figure B-1. The pressure at the exit of the pump is 15 bar (15×10^5 Pa) above atmospheric, and the water is discharged at atmospheric pressure at the end of the pipeline. All the pipes are 6-inch schedule 40 steel with an inside diameter of 0.154 m. The equivalent lengths of the pipes connecting different nodes are the following: $L_{01} = 100$ m, $L_{12} = L_{23} = L_{45} = 300$ m, and $L_{13} = L_{24} = L_{34} = 1200$ m.



a) Use POLYMATH to calculate all the flow rates and pressures at nodes 1, 2, 3, and 4 for the pipeline network shown in the Figure. The Fanning friction factor can be assumed to be constant at $f_F = 0.005$ for all pipelines. The initial estimates for all the volumetric flow rates can be set at 0.1 m^3/s.

b) Use Broyden's Quasi-Newton method (programmed with MATLAB) to solve the system of equations obtained in part (a).

Equations and data

For the solution of this problem it is convenient to express the pressure drop from node *i* to node *j* as

$$\Delta P_{ij} = k_{ij}(q_{ij})^2 \qquad \text{(B-1)}$$

where ΔP_{ij} is the pressure drop and q_{ij} is the volumetric flow rate between nodes *i* and *j*. The k_{ij} terms in Equation (B-1) are related to the Fanning friction factors and average fluid velocities.

$$k_{ij} = \frac{32 f_F \rho L_{ij}}{\pi^2 D^5} \qquad \text{(B-2)}$$

There are two relationships that govern the steady-state flow rate in pipeline networks. *First, the algebraic sum of the flow rates at each node must be zero. Second, the algebraic sum of all pressure drops in a closed loop must be zero.*

The flow rate summation equations for nodes 1, 2, 3 and 4 are the following:

$$\begin{aligned} q_{01} - q_{12} - q_{13} &= 0 \\ q_{12} - q_{24} - q_{23} &= 0 \\ q_{23} + q_{13} - q_{34} &= 0 \\ q_{24} + q_{34} - q_{45} &= 0 \end{aligned} \tag{B-3}$$

The pressure drop summation equations on Loops I, II and III (see Figure B-1) are respectively

$$\begin{aligned} \Delta P_{01} + \Delta P_{12} + \Delta P_{24} + \Delta P_{45} + \Delta P_{PUMP} &= 0 \\ \Delta P_{13} - \Delta P_{23} - \Delta P_{12} &= 0 \\ \Delta P_{23} + \Delta P_{34} - \Delta P_{24} &= 0 \end{aligned} \tag{B-4}$$

The pressure drops can be expressed as functions of q_{ij} using Equation (B-1). This substitution leads to seven equations with seven unknown flow rates.

The iteration function of Broyden's quasi-Newton method[E] for solving systems of nonlinear algebraic equations is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_k \mathbf{f}(\mathbf{x}_k) \quad k = 0, 1, \ldots \tag{B-5}$$

where **x** is an *n* vector of variables, **f** is an *n* vectors of functions and $\mathbf{H}_k$ is the k^{th} estimate for the inversed Jacobian matrix (matrix of partial derivatives). The identity matrix **I** is often used as initial estimate for the inversed Jacobian matrix $\mathbf{H}_0$. This matrix is updated in iteration *k* using the equation

$$.\mathbf{H}_{k+1} = \mathbf{H}_k - \frac{(\mathbf{H}_k \mathbf{y}_k - \mathbf{p}_k)\mathbf{p}_k^T.\mathbf{H}_k}{\mathbf{p}_k^T \mathbf{H}_k \mathbf{y}_k} \tag{B-6}$$

where $\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \mathbf{f}(\mathbf{x}_{k+1}) - \mathbf{f}(\mathbf{x}_k)$.

The Euclidean norm of $\mathbf{p}_k$ can be used as error estimate. Thus $\tilde{\varepsilon}_k = \| \mathbf{p}_k \|$. The iterations can be stopped when $\hat{\varepsilon}_k < 10^{-5}$.

[E] Broyden, C. G. Mathematics of Computation, 19, 577-593(1965)