ChE class and home problems

The object of this column is to enhance our readers' collections of interesting and novel problems in chemical engineering. We request problems that can be used to motivate student learning by presenting a particular principle in a new light, can be assigned as novel home problems, are suited for a collaborative learning environment, or demonstrate a cutting-edge application or principle. Manuscripts should not exceed 14 double-spaced pages and should be accompanied by the originals of any figures or photographs. Please submit them to Dr. Daina Briedis (e-mail: briedis@egr.msu.edu), Department of Chemical Engineering and Materials Science, Michigan State University, East Lansing, MI 48824-1226.

CONTINUOUS FEED AND BLEED ULTRAFILTRATION

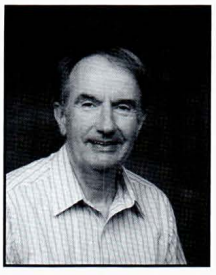
A Demonstration of the Advantages of the Modular Approach for Modeling Multi-Stage Processes

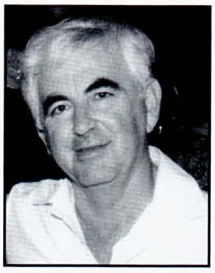
MICHAEL B. CUTLIP[1] AND MORDECHAI SHACHAM[2]
1 University of Connecticut • Storrs, CT 06269
2 Ben-Gurion University of the Negev • Beer-Sheva 84105, Israel

Multiple stage operations are widely used in separation processes. There are two commonly used approaches for modeling of multi-stage processes. The "simultaneous" approach involves modeling of the complete process (which includes all the stages) as one large set of equations. This approach is widely used for modeling of distillation columns and its main advantage is the high computational efficiency. In the "sequential modular" approach a model of a single stage (module) is prepared and tested separately. The complete multi-stage process is then constructed by tying together several modules by means of the material and energy flows between them. The advantages of this approach are that the model building is more straightforward; that the models are easier to construct, to follow, and to debug; and that the computer code can actually serve as problem documentation. Thus this approach is more adequate for educational use than the simultaneous approach.

An example presented by Foley[1] that concerns the design of a multi-stage ultrafiltration unit operated in a feed and bleed mode will be used to demonstrate the advantages of the "sequential modular" approach in the solution of various types of problems. This example is suitable for courses in separation processes, introduction to modeling and computation, and process and product design.

Michael B. Cutlip *is professor emeritus of the Chemical, Materials, and Biomolecular Engineering Department at the University of Connecticut and has served as department head and director of the university's Honors Program. He has B.Ch.E. and M.S. degrees from Ohio State and a Ph.D. from the University of Colorado. His current interests include the development of general software for numerical problem solving and application to chemical and biochemical engineering.*





Mordechai Shacham *is professor emeritus of the Department of Chemical Engineering at the Ben-Gurion University of the Negev in Israel. He received his B.Sc. and D.Sc. degrees from the Technion, Israel, Institute of Technology. His research interests include analysis, modeling and regression of data, applied numerical methods, and prediction and consistency analysis of physical properties.*

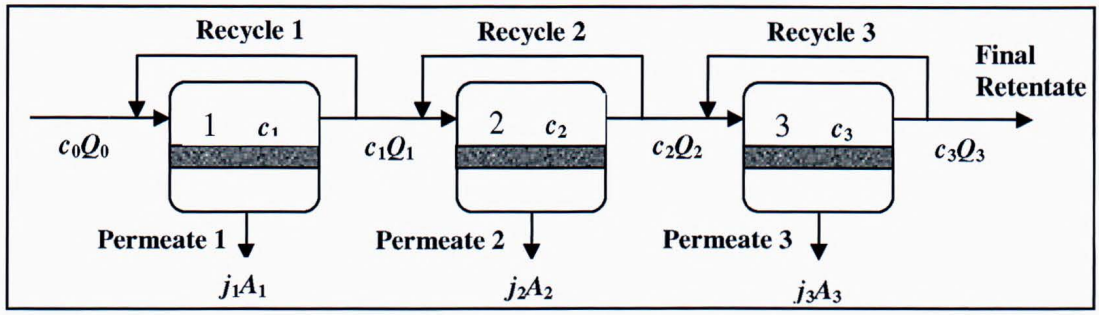


Figure 1. *Schematic plot of the Multi-stage Ultrafiltration System.*

PROBLEM BACKGROUND

Continuous concentration of a protein solution by multi-stage feed and bleed ultrafiltration

A typical 3-stage feed-and-bleed ultrafiltration process is shown in Figure 1. Fresh feed with a volumetric flow rate of Q_0 (m^3/s) and solute concentration of c_0 (g/L) is mixed with retentate (product stream) recycle from the first stage and enters the first membrane module. Some of the solvent (and possibly some of the solute) passes the membrane and exits the unit as permeate (filtrate). The concentrated product stream (with solid concentration of c_1 g/L) is partially recycled in order to increase the flow rate into the unit to ensure well-mixed conditions. The product (bleed) stream from the first unit is fed into the second stage with a volumetric flow rate of Q_1 (m^3/s).

Detailed discussion of the multi-stage ultrafiltration process, including the associated design equations, is provided, for example, by Seader and Henley.[2] Here the assumptions suggested by Foley[1] are used for simplification of the model. These assumptions are that 1) no solute passes the membrane (complete rejection), and 2) the gel polarization model applies. Thus the membrane is operating at the limiting flux (j, m/s) given by

$$j = k \ln \frac{c_g}{c_i} \tag{1}$$

where k is the mass transfer coefficient (m/s), c_g is the limiting or "gel" concentration (g/L), and c_i is the solute concentration in stage i. The solute balance on stage i (assuming complete rejection) yields

$$Q_i c_i = Q_{i-1} c_{i-1} \tag{2}$$

The total balance on the feed, retentate, and permeate streams of stage i can be written

$$f_i = Q_{i-1} - Q_i - jA_i \tag{3}$$

where A_i is the membrane area in stage i.

Eqs. (1), (2), and (3) represent a complete model of stage i. At the solution f_i must vanish ($f_i = 0$). One of the variables associated with stage i (say c_i, A_i, etc.) can be selected as unknown and values for the rest must be specified.

The sequential modular approach requires building the model of a single unit so that it can calculate the "output" variables (Q_i, c_i, and the permeate flow rate) if the input variables (Q_{i-1} and c_{i-1}) and the design parameters (k, c_g, and A_i) are specified. One possibility to carry out this calculation is to solve Eq. (2) for Q_i yielding

$$Q_i = Q_{i-1} c_{i-1} / c_i \tag{2A}$$

Introducing Eqs. (1) and (2A) into Eq. 3, and replacing the input variables and design parameters by their numerical values, yields a single nonlinear algebraic equation that can be solved for the unknown output variable c_i. The other output variable Q_i can be consequently calculated directly by solving Eq. (2A).

If all the input variables and design parameters are specified and the output variables need to be calculated, the problem is categorized as a "simulation problem." In a "design problem" some of the output variables are specified and the same number of input variables and/or design parameters need to be determined, so that the specification regarding the output variables is met. Adding an objective function that contains input variables and/or design parameters that need to be minimized to meet an economic objective, subject to constraints related to the output variables, constitutes an "optimization problem."

The different parts of the following problem statement are related to these three types of problems.

PROBLEM STATEMENT

The assignments and numerical data presented by Foley[1] are considered. A multistage feed-and-bleed ultrafiltration unit is used for concentrating a protein solution. Fresh feed enters the first stage at the rate of Q_0 = 1 L/min with solute (protein) concentration of c_0 = 10 g/L. Complete rejection can be assumed for the protein. The membrane is operating at the limiting flux with the mass transfer coefficient: $k = 3.5 \cdot 10^{-6}$ m/s and the gel concentration c_g = 300 g/L.

a) Given that the membrane area in the first stage is A_1 = 2.7 m^2, calculate the product's outlet concentration: c_1 and flow rate: Q_1 from this stage.

b) A 3-stage system with equal membrane areas of A_i =0.9 m^2 is used to separate the protein solution. Calculate the

product's outlet concentrations and flow rates for the three stages (a simulation problem).

c) *Find the total membrane area of a three-stage ultrafiltration system that yields retentate concentration leaving the third stage of 100 g/L assuming equal membrane areas for the three stages (a design problem).*

d) *Repeat question (c) but this time allow for different membrane areas for the different stages so as to obtain the same final concentration but minimizing the total membrane area (an optimization problem).*

PROBLEM SOLUTION

Modeling a single stage using the sequential modular approach

Modeling a single stage can proceed following the algorithm outlined in the problem background section. Several available software packages can be used for solving the nonlinear algebraic equation of the single-stage model. The POLYMATH[3] software package was used for this purpose.

The input of part (a) of the problem into the POLYMATH software package is shown in Table 1. The POLYMATH program includes the code and comments (text that starts with the "#" sign and ends with the end of the line). The row numbers shown in Table 1 are not part of the program; they were added as references for the explanations that follow.

In the POLYMATH program, the equations and data are grouped into "model equations" [lines 1-4, Eqs. (1), (3), and (2A)], problem-specific data, including units (lines 6-11, including the input variable and design parameter values), and initial estimates for the unknown exit concentration. The POLYMATH software automatically orders the equations prior to solution. The solution with POLYMATH using this equation set was $c_1 = 66.93$ g/L and $Q_1 = 0.15$ L/min.

TABLE 1
POLYMATH program for the solution of Part (a) of the assignment

No.	Equation/ # Comment
1	#Model equations
2	j=k*ln(cg/c1) #Membrane Flux
3	Q1=c0*Q0/c1 #Complete Rejection of Protein
4	f(c1) = Q0-Q1-j*A #Overall Material Balance
5	
6	# Problem specific data
7	Q0=1/(60*1000) #m^3/s
8	k=3.5e-6 #m/s
9	c0=10 # g/L
10	A=2.7 # m^2
11	cg =300 # g/L
12	
13	# Initial estimates
14	c1(min)=20
15	c1(max)=100

The program shown in Table 1 provides a clear, precise, and complete documentation of the problem and its mathematical model. Observe that there is actually no need to combine the three basic equations into one equation and this provides an easier-to-follow problem documentation. Combining the equations can also be a source of errors.

Modeling the three-stage system using the sequential modular approach

The model used for the first stage can be used for the subsequent stages except that the outlet variables of the earlier stage become the inlet variables of the subsequent one and the design parameters need to be updated, if necessary. Thus, modeling the operation of three consecutive stages of the ultrafiltration system [Part (b) of the problem statement] involves writing the model equations that were used for the first stage,

TABLE 2
POLYMATH program for the solution of Part (b) of the assignment

No.	Equation/ # Comment
1	#Model equations
2	#First Stage
3	j1=k*ln(cg/c1) #Membrane Flux
4	Q1=c0*Q0/c1 #Complete Rejection of Protein
5	f(c1) = Q0-Q1-j1*A #Overall Material Balance
6	#Second Stage
7	j2=k*ln(cg/c2) #Membrane Flux
8	Q2=c1*Q1/c2 #Complete Rejection of Protein
9	f(c2)=Q1-Q2-j2*A #Overall Material Balance
10	#Third Stage
11	j3=k*ln(cg/c3) #Membrane Flux
12	Q3=c2*Q2/c3 #Complete Rejection of Protein
13	f(c3)=Q2-Q3-j3*A #Overall Material Balance
14	
15	#Problem specific data
16	Q0=1/(60*1000) #m^3/s
17	k=3.5e-6 #m/s
18	A=0.9 # m^2
19	c0=10 # g/L
20	cg =300 # g/L
21	
22	# Initial estimates
23	c1(0)=20
24	c2(0)=50
25	c3(0)=170

TABLE 3
Results summary for Parts (b), (c), and (d) of the multi-stage ultrafiltration problem

	Part (b)		Part (c)			Part (d)		
Stage No.	c_i (g/L)	Q_i (m^3/s)	A_i (m^2)	c_i (g/L)	Q_i (m^3/s)	A_i (m^2)	c_i (g/L)	Q_i (m^3/s)
1	20.34	0.492	0.712	17.4	0.575	0.947	22.14	0.452
2	56.07	0.178	0.712	37.7	0.265	0.505	49.73	0.201
3	163.45	0.061	0.712	100	0.100	0.567	100	0.100
Total			2.136			2.019		

	A	B	C	D	E	F
1	**POLYMATH NLE Migration Document**					
2		**Variable**	**Value**		***Polymath Equation***	***Comments***
3	Explicit Eqs	j1	9.1218E-06		*j1=k*ln(cg/c1)*	*Membrane Flux*
4		Q1	7.5266E-06		*Q1=c0*Q0/c1*	*Complete Rejection of Protein*
5		j2	6.2903E-06		*j2=k*ln(cg/c2)*	*Membrane Flux*
6		Q2	3.3516E-06		*Q2=c1*Q1/c2*	*Complete Rejection of Protein*
7		j3	3.8451E-06		*j3=k*ln(cg/c3)*	*Membrane Flux*
8		Q3	1.6667E-06		*Q3=c2*Q2/c3*	*Complete Rejection of Protein*
9		**At**	**2.01539045**		***At=A1+A2+A3***	***Objective function***
10		Q0	1.6667E-05		*Q0=1/(60*1000)*	*m^3/s*
11		k	0.0000035		*k=3.5e-6*	*m/s*
12		c0	10		*c0=10*	*g/L*
13		c3	100		*c3=100*	*g/L*
14		cg	300		*cg=300*	*g/L*
15		**A1**	**0.94676198**		*A1=0.7*	
16		**A2**	**0.50474175**		*A2=0.7*	
17	Implicit Vars	c1	22.1436941		*c1(0)=20*	*Overall Material Balance*
18		c2	49.7272272		*c2(0)=50*	*Overall Material Balance*
19		**A3**	**0.56388672**		*A3(0)=1*	*Overall Material Balance*
20	Implicit Eqs	f(c1)	5.0389E-07		*f(c1)=Q0-Q1-j1*A1*	
21		f(c2)	1E-06		*f(c2)=Q1-Q2-j2*A2*	
22		f(A3)	-4.833E-07		*f(A3)=Q2-Q3-j3*A3*	
23	**Sum of Squares:**		**1.4875E-12**		*F = f(c1)^2+f(c2)^2+f(A3)^2*	

Figure 2. *Excel Worksheet for the solution of Part (d) of the multi-stage ultrafiltration problem.*

for the second and third stage while changing the indices of the inlet and outlet streams and introducing equal $A_i = 0.9$ m^2 membrane areas as design parameters. The POLYMATH program prepared according to these principles is shown in Table 2 and the results are presented in Table 3. Observe that (as was pointed out by Foley[1]) the final concentrate in this case is $c_3 = 163.45$ g/L, which is much higher than the concentration reached by a single stage unit of the same total membrane area [in Part (a), $c_1 = 66.93$ g/L].

The model shown in Table 2 can be easily modified to solve Part (c) of the assignment. In this part the exit concentration of the protein from the third stage (c_3) is specified and it is required to calculate the membrane areas in the three stages, assuming equal areas (A_i). There is no need to modify the model equations in Table 2 except to change the status of c_3 to a specified design parameter (instead of unknown) and to include the unique A value as one of the unknowns. The results of the computation for Part (c) are shown also in Table 3. Observe that the total area required to reach $c_3 = 100$ g/L is $A_t = 2.136$ m^2.

These examples demonstrate the flexibility provided by the sequential-modular approach to investigate various design alternatives.

Minimizing the total membrane area of the ultrafiltration system

In part (d), it is requested to minimize the total area of the multi-stage system with a pre-specified c_3 value while allowing different areas A_1, A_2, and A_3 for the three stages. This problem can be formulated as a constrained minimization problem to minimize $A_t = A_1 + A_2 + A_3$ subject to the constraints $f(c_1) = 0$, $f(c_2) = 0$,and $f(A_3) = 0$. The variables are A_1, A_2, A_3, c_1, and c_2.

Only minor changes need to be introduced in the program shown in Table 2 to accommodate this new problem formulation. As the POLYMATH package does not solve constrained nonlinear optimization problems, the program should be solved with a software package that includes tools for solving such problems. POLYMATH can automatically export programs to MATLAB[4] or Excel[5] that can solve a constrained minimization problem. The Excel solution will be shown here.

In Figure 2 the Excel worksheet—as obtained by exporting the POLYMATH program to Excel—is shown. Note that the Excel formulas of the various equations are in column C. The additional information, generated by the POLYMATH export routine, is textual and serves as documentation. The names of the variables, as defined in the POLYMATH program, are shown in column B. The original POLYMATH equations are displayed in column E and the associated comments are given in column F. The Excel "Solver" Add-In is used for finding the minimal area. The following information is used to specify the "Solver" parameters; the minimum is sought for "Target cell" C9 (A_t) by changing cells C15 through C19 (A_1, A_2, A_3, c_1, and c_2), subject to the constraints C20 = 0, C21 = 0, and C22 = 0

[which forces the residuals of the nonlinear equations given by $f(c_1)$, $f(c_2)$, and $f(A_3)$ to be zero]. The optimal solution found is shown in Table 3 and Figure 2. At the minimum, $A_t = 2.015$ m_2, which is slightly lower than the total area required (A_t = 2.136 m^2) when stages with equal areas are used to achieve the same final concentration.

Solution approach presented by Foley

To highlight the educational advantages of the "sequential modular" approach for solving problems that involve staged processes, the solutions provided here can be compared with the solution techniques used by Foley.[1] According to his approach, the equations representing a single stage [Eqs. (1), (2), and (3)] are brought into the form of a single nonlinear algebraic equation containing two unknowns, x_{i-1} and x_i, where $x_i = c_0/c_i$.

$$\frac{Q_0}{kA}(x_{i-1} - x_i) - \ln\frac{C_g}{C_0} - \ln x_i = 0 \qquad (4)$$

This model is inconsistent with the "sequential modular" approach as it includes the input variables of the first stage (c_0 and Q_0) in the models of all the subsequent stages. While the definition of the unknown x_i used in this equation was essential for graphical solution of the ultrafiltration problems, it is absolutely unnecessary for numerical solution. Keeping the original variables associated with a particular stage makes it much easier to understand, to interpret, and to modify the model in order to fit the various problem types (design, optimization, etc.). Furthermore, the model equations cannot provide full documentation of the problem as some of the original variables (Q_i and c_i) do not appear in them.

USING THE EXAMPLE TO DEMONSTRATE GOOD PROGRAMMING PRACTICES

Software packages such as POLYMATH and Excel are very convenient tools for problem solving, but there are more complex tasks that may require programming. One such complex assignment can be optimization of the membrane areas of the multi-stage system for different final concentration: c_3 values (parametric runs). In this section the membrane area optimization assignment (d) is carried out for various c_3 values: c_3 = 50, 60 ... 150 g/L and the resultant optimal areas are plotted vs. c_3. Such parametric optimization runs can be carried out with Excel by manually changing the parameter values. This approach is inefficient and cumbersome, however, particularly for problems where there are many parameters and a wide range of parameter values to be considered. In such cases, programming is required for repetitive solution of the problem with the various parameter values. One option is to carry out the parametric runs efficiently using MATLAB. The development of the MATLAB program can serve for demonstration of good programming practices.

The MATLAB function representing the operation of the multistage ultrafiltration unit can be automatically and efficiently generated by POLYMATH (Table 4). The function is named *MNLEfun*. It accepts c_1, c_2, and A_3 as input parameters and returns the values of $f(c_1)$, $f(c_2)$, and $f(A_3)$ to the calling program. Note that MATLAB requires input of the variable values into the function in a single array (x, in this case), and return of the function values in a single array (fx, lines 18-20 in Table 4). The variable values are put back into variables with the same names as used in the POLYMATH model (lines 2-5) to make the MATLAB code more meaningful. POLYMATH orders the equations sequentially as required by MATLAB and converts any needed intrinsic functions and logical expressions.

The function in Table 4 contains several variables (c_g, c_0, c_3, Q_0, A_1, and A_2) to which constant numerical values are assigned. Assigning numerical values to variables inside functions is considered poor programming practice as it limits the use of the function to one particular problem with just one set of parameter values. To enable general use of the function the numerical values of these variables must be passed to the function by the program that calls this function. One possibility to pass variable values to a

TABLE 4
MATLAB function (model) for the solution of part (d) of the ultrafiltration problem

No.	Command % Comment
1	function fx = MNLEfun(x);
2	c1 = x(1);
3	c2 = x(2);
4	A3 = x(3);
5	cg = 300; %g/L
6	c0 = 10; %g/L
7	k = .0000035; %m/s
8	Q0 = 1 / (60 * 1000); %m^3/s
9	j1 = k * log(cg / c1); %Membrane Flux
10	c3 = 100; %g/L
11	A2 = .7 %m^2
12	Q1 = c0 * Q0 / c1; %m^3/s, Complete Rejection of Protein
13	j2 = k * log(cg / c2); %Membrane Flux
14	Q2 = c1 * Q1 / c2; %m^3/s, Complete Rejection of Protein
15	Q3 = c2 * Q2 / c3; %m^3/s, Complete Rejection of Protein
16	j3 = k * log(cg / c3); %Membrane Flux
17	A1 = .7 %m^2
18	fx(1,1) = Q0 - Q1 - (j1 * A1); %Overall Material Balance
19	fx(2,1) = Q1 - Q2 - (j2 * A2); %Overall Material Balance
20	fx(3,1) = Q2 - Q3 - (j3 * A3); %Overall Material Balance

function is by defining these variables as "global" variables. Unlike "local" variables, which are separate for the different functions and the main program, a single copy of a "global" variable is shared by all of them. The use of "global" variables is not considered good programming practice, however, as it overrides the hierarchical structure of the program. This means that change of a global variable in a lower-hierarchy-level function may cause unforeseen changes in higher-level functions or in the main program. Good programming practice requires the passing of the numerical values of constants as input parameters to the function.

In Table 5 the revised form of the *MNLEfun* is shown. Observe that in this version all the numerical values of the variables are passed through one array, named *parm*. Good programming practice requires introducing the numerical values into the original variables (see lines 5 to 11 in Table 5) so that the original forms of the equations (lines 12 through 20) can serve as clear documentation of the ultafiltration system model.

The MATLAB multiple variable minimization function: *fminsearch* combined with the nonlinear equation solver function: *fsolve* can be used to find the minimal membrane area configuration for various c_3 values. The *fminsearch* function is called with the following parameters:

[Aopt,TArea,exitflag] = fminsearch(@minA,A1A2,[],parm);

The input parameters are: *minA* is the name of the function that calculates the objective function At = A1+ A2 + A3 value, *A1A2* is an array containing the current values of A_1 and A_2, and *parm* is the same array of the parameter values that is used in the *MNLEfun* function (Table 5). The output parameters are: *Aopt* is an array that contains the optimal values of A_1 and A_2, *TArea* contains the optimal value of the total membrane area, and *exitflag* indicates whether a minimum has been found (*exitflag* = 1) or the search has been terminated for other reasons (*exitflag* ≠ 1).

The function *minA* is shown in Table 6. This function obtains the values of *A1A2* and *parm* from the *fminsearch* function and returns *Asum* (A_t). The *minA* function passes the current A1 and A2 values to the *MNLEfun* function (through the *parm* array) and uses the nonlinear equation solver function *fsolve* to solve the system of nonlinear equations: $f(c_1) = 0$, $f(c_2) = 0$, and $f(A_3) = 0$. This is necessary in order to find the A3 value that satisfies the constraints with the current set of A1 and A2 values.

The complete MATLAB program can be downloaded from the ftp site: <ftp://ftp.bgu.ac.il/shacham/Ultrafiltration/>. The calculated optimal areas are plotted vs. the exit concentration c_3 in Figure 3 (next page). As expected, higher outlet concentrations require larger membrane areas.

A similar example involving development of a MATLAB program for modeling of imperfect mixing in a Chemostat that involves the use of minimization and nonlinear equation solver functions can be found in Cutlip, et. al.[6] The example presented there can also be used for demonstration of good programming practices.

TABLE 5
Generalized form of the MATLAB function (model) of the ultrafiltration system

No.	Command % Comment
1	function fx = MNLEfun(x,parm);
2	c1 = x(1);
3	c2 = x(2);
4	A3 = x(3);
5	cg = parm(1); %g/L
6	c0 = parm(2); %g/L
7	k = parm(3); % m/s
8	Q0 = parm(4); %m^3/s
9	c3 = parm(5); % m/s
10	A1 = parm(9);% m^3
11	A2 = parm(10);% m^3
12	j1 = k * log(cg / c1); %Membrane Flux
13	Q1 = c0 * Q0 / c1; %m^3/s, Complete Rejection of Protein
14	j2 = k * log(cg / c2); %Membrane Flux
15	Q2 = c1 * Q1 / c2;%m^3/s, Complete Rejection of Protein
16	Q3 = c2 * Q2 / c3; %m^3/s, Complete Rejection of Protein
17	j3 = k * log(cg / c3); %Membrane Flux
18	fx(1,1) = (Q0 - Q1 - (j1 * A1)); %Overall Material Balance
19	fx(2,1) = Q1 - Q2 - (j2 * A2); %Overall Material Balance
20	fx(3,1) = Q2 - Q3 - (j3 * A3); %Overall Material Balance

TABLE 6
MATLAB Function for calculating the total membrane area

No.	Command % Comment
1	function Asum=minA(A1A2,parm)
2	c1=parm(6); c2=parm(7); A3= parm(8);
3	xguess = [c1 c2 A3];
4	A1= A1A2(1); A2 = A1A2(2);
5	parm(9)=A1; parm(10)=A2;
6	options = optimset('Display',['off'],'TolFun',[1e-14],'TolX',[1e-14]);
7	xsolv=fsolve(@MNLEfun,xguess,options,parm);
8	A3=xsolv(3);
9	Asum=A1+A2+A3;

CONCLUSIONS

The example presented here provides an opportunity to practice several aspects of modeling and design of multi-stage processes

- *Using a consistent "sequential modular" approach for modeling the single units.*
- *Solving problems of increasing levels of difficulty—simulation, design, and optimization—while selecting the most effective software tool for numerical solution of the problem at hand.*
- *Building the model and the computer input so that they can serve as clear and complete documentation of the problem and its solution.*
- *Using advanced tools available for solving nonlinear algebraic equations and optimization problems.*

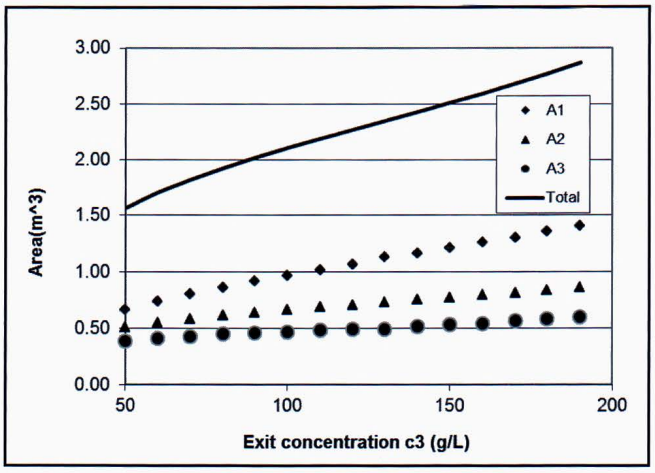


Figure 3. *Plot of membrane areas vs. required exit concentration.*

REFERENCES

1. Foley, G., "Solution of Nonlinear Algebraic Equations in the Analysis, Design, and Optimization of Continuous Ultrafiltration," *Chem. Eng. Ed.*, **45**(1), 59 (2011)
2. Seader, J.D., and E.J. Henley, *Separation Process Principles*, 2nd Ed, New York, Wiley (2006)
3. POLYMATH is a product of Polymath Software, <http://www.polymath-software.com>
4. MATLAB is a product of MathWorks, Inc., <http://www.mathworks.com>
5. Excel is a registered trademark of the Microsoft Corporation, <http://www.microsoft.com>
6. Cutlip, M.B., N. Brauner, and M. Shacham, "Biokinetic Modeling of Imperfect Mixing in a Chemostat—an Example of Multiscale Modeling," *Chem. Eng. Ed.*, **43**(3), 243 (2009) ❐