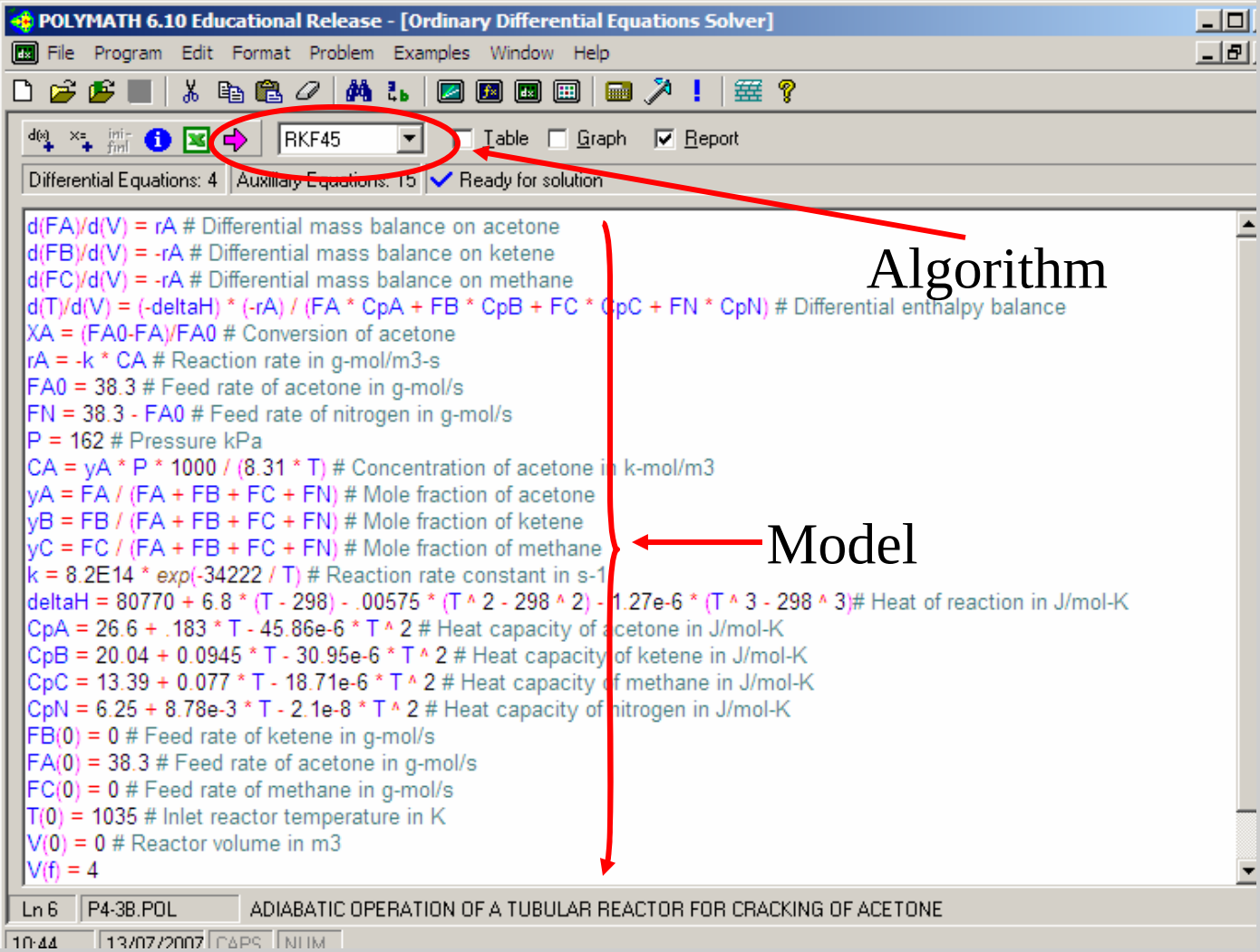
Efficient Solution of Multiple-Model, Multiple-Algorithm Problems in Undergraduate and Graduate Education

Mordechai Shacham
Ben Gurion University of the Negev
Beer-Sheva, Israel

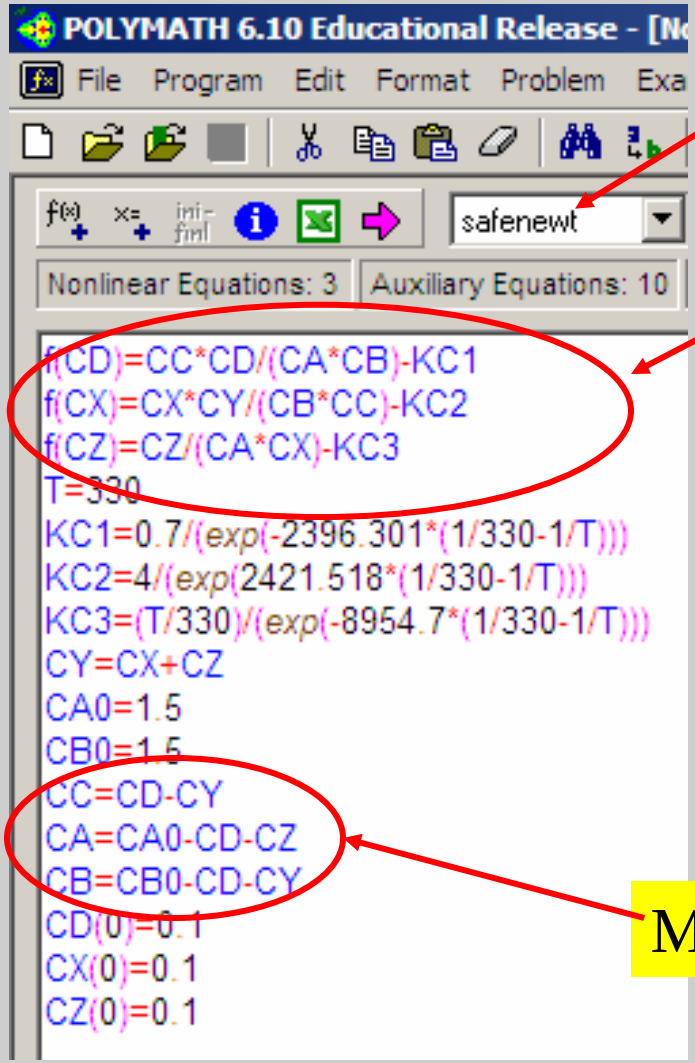
Michael Cutlip
University of Connecticut
Storrs, CT

Michael Elly
Intel Corp.,
Qiryat Gat, Israel

A Single-Model, Single-Algorithm Problem (ODE) - Adiabatic Operation of a Tubular Reactor for Cracking of Acetone



A Single-Model, Single-Algorithm Problem (NLE)
Complex Chemical Equilibrium

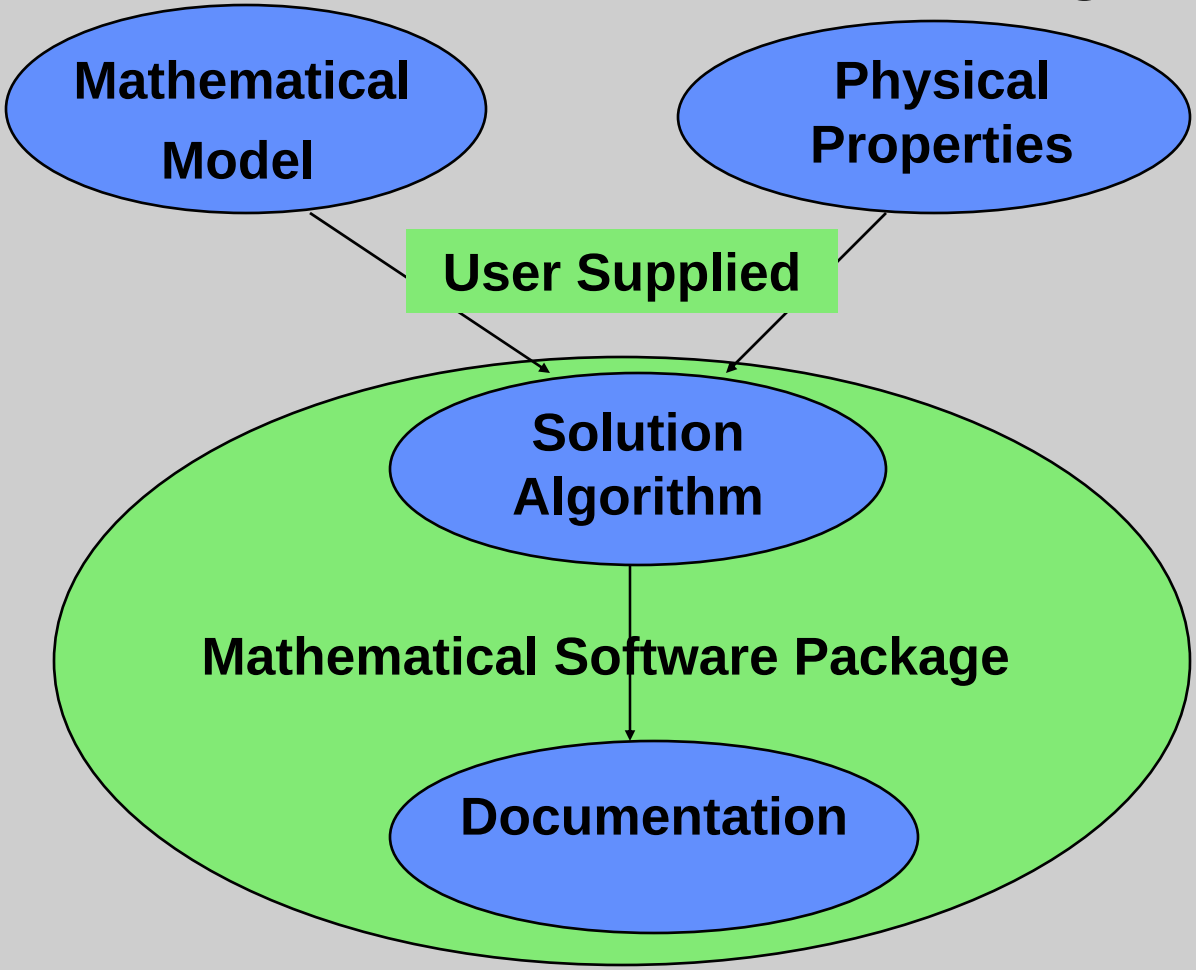


Solution algorithm

Chemical Equilibrium equations

Mole Balance Equations

Single Model – Single Algorithm Problem Solution with Software Packages



Using this approach, the USER supplies the mathematical model and the physical properties and the package provides the numerical solution.

A Multiple-Model, Single-Algorithm (MMSA) Problem
Three Modes in the Operation of the Semi-batch Bioreactor

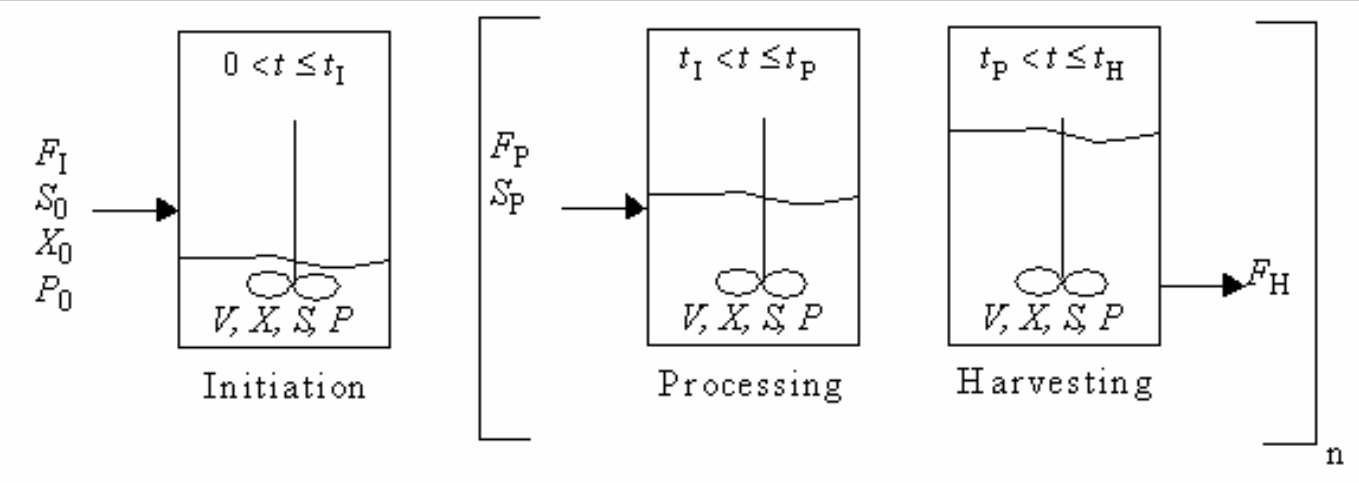


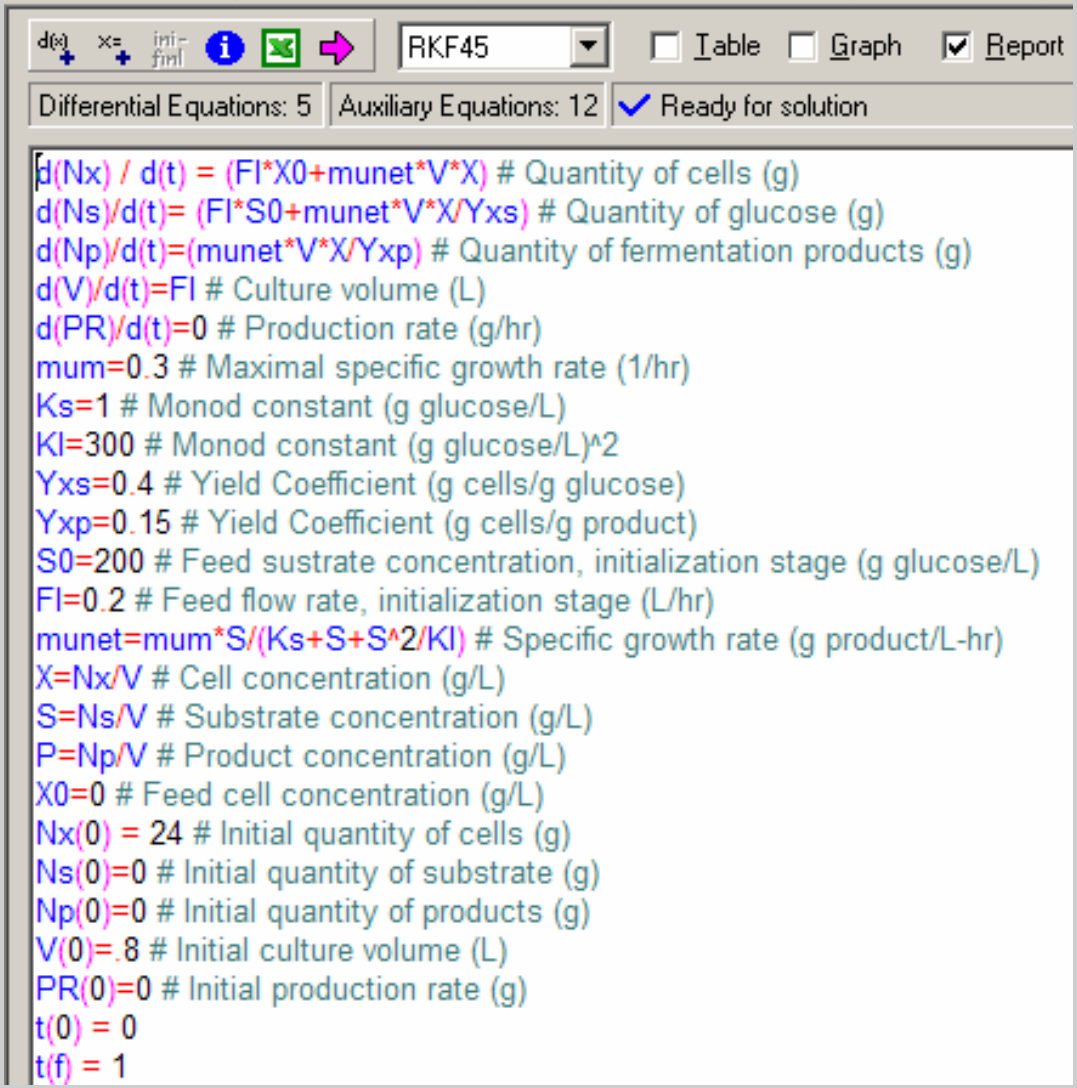
Table 14–7 Differential Equations for Fed Batch and Cyclic Fed Batch Bioreactors

Differential Equations	Initiation	Processing	Harvesting
$\frac{dN_X}{dt} =$	$F_I X_0 + \mu_{net} XV$	$\mu_{net} XV$	$- F_H X + \mu_{net} XV$
$\frac{dN_S}{dt} =$	$F_I S_0 - \frac{\mu_{net} XV}{Y_{X/S}}$	$F_P S_P - \frac{\mu_{net} XV}{Y_{X/S}}$	$- F_H S - \frac{\mu_{net} XV}{Y_{X/S}}$
$\frac{dN_P}{dt} =$	$\frac{\mu_{net} XV}{Y_{X/P}}$	$\frac{\mu_{net} XV}{Y_{X/P}}$	$- F_H P + \frac{\mu_{net} XV}{Y_{X/P}}$
$\frac{dV}{dt} =$	F_I	F_P	$- F_H$

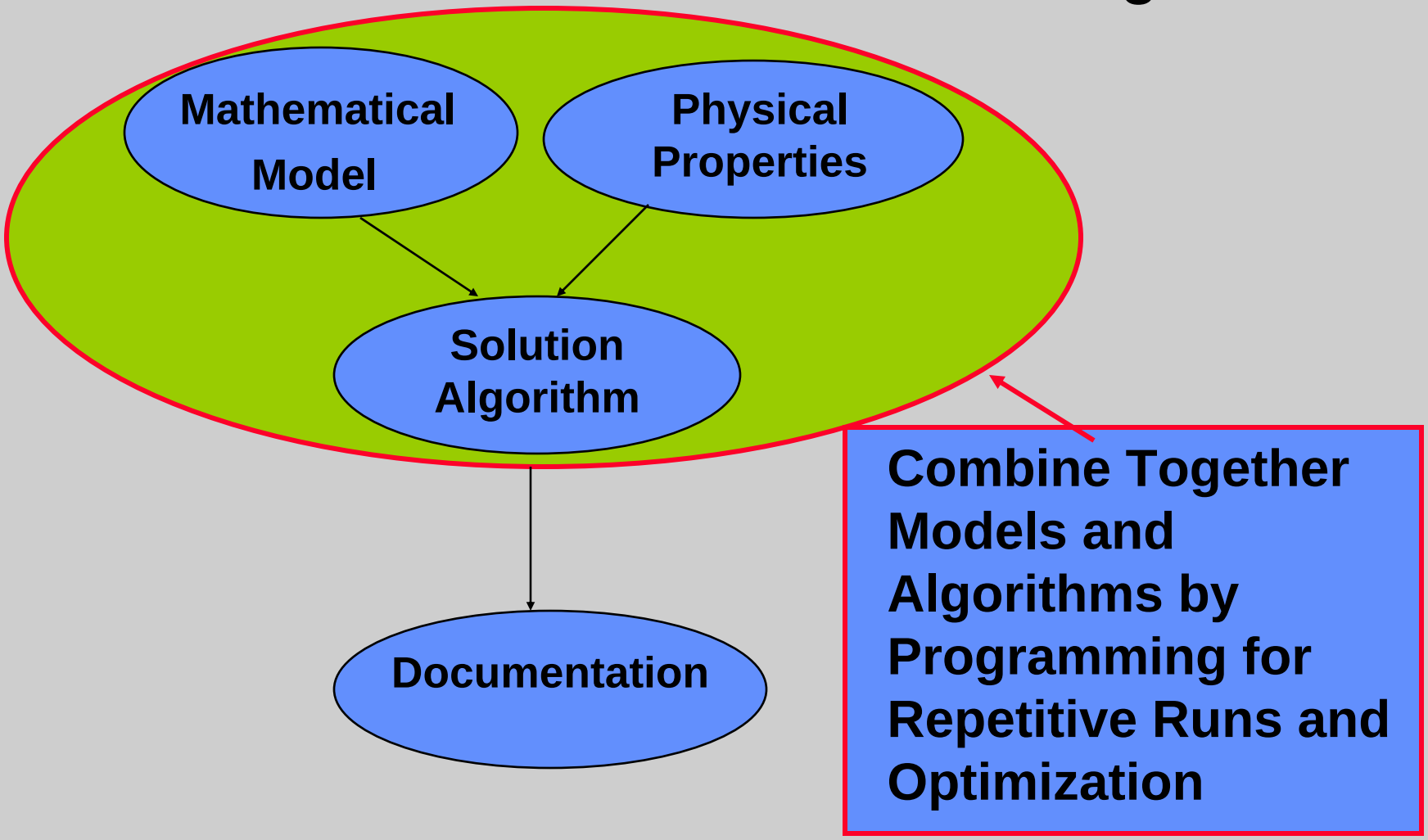
$$\mu_{net} = \mu_g = \frac{\mu_m S}{K_S + S + S^2/K_I}$$

S (substrate reactant) + X (cell) = P (product) + nX

A Multiple-Model, Single Algorithm (MMSA) Problem
Polymath Model of the Initiation Mode of Operation



Multiple Model – Multiple Algorithm Problem Solution with Software Packages



A Single-Model, Multiple Algorithm Problem
Simultaneous Multicomponent Diffusion of Gases*

Gases A and B are diffusing through stagnant gas C between two points 1 and 2 where the compositions and distance apart are known. Calculate and plot the concentration profiles and determine the molar fluxes.

Component	Point 1 Concentration $kg\text{-}mol/m^3$	Point 2 Concentration $kg\text{-}mol/m^3$	Diffusivities at 0.2 atm m^2/s
A	2.229×10^{-4}	0	$D_{AC} = 1.075\times10^{-4}$
B	0	2.701×10^{-3}	$D_{BC} = 1.245\times10^{-4}$
C	7.208×10^{-3}	4.730×10^{-3}	$D_{AB} = 1.47\times10^{-4}$

*p. 10.8 in Cutlip and Shacham, *Problem Solving In Chemical and Biochemical Engineering with Polymath, Excel and MATLAB*. Prentice-Hall, 2008.

Simultaneous Multicomponent Diffusion of Gases

The Stefan-Maxwell equations describe this multi-component diffusion process

$$\frac{dC_A}{dz} = \frac{(x_A N_B - x_B N_A)}{D_{AB}} + \frac{(x_A N_C - x_C N_A)}{D_{AC}}$$

$$\frac{dC_B}{dz} = \frac{(x_B N_A - x_A N_B)}{D_{AB}} + \frac{(x_B N_C - x_C N_B)}{D_{BC}}$$

$$\frac{dC_C}{dz} = \frac{(x_C N_A - x_A N_C)}{D_{AC}} + \frac{(x_C N_B - x_B N_C)}{D_{BC}}$$

where

$$D_{BA} = D_{AB},\ D_{CA} = D_{AC},\ \text{and}\ D_{CB} = D_{BC}$$

Simultaneous Multicomponent Diffusion of Gases

$$\frac{dC_A}{dz} = \frac{(x_A N_B - x_B N_A)}{D_{AB}} + \frac{(x_A N_C - x_C N_A)}{D_{AC}}$$

The parameters N_A and N_B (the molar fluxes of components A and B respectively) are unknown. They can be calculated using the **boundary conditions: at point 2 ($z = 0.001$m) $CA = 0$ and $CB = 2.701$.**

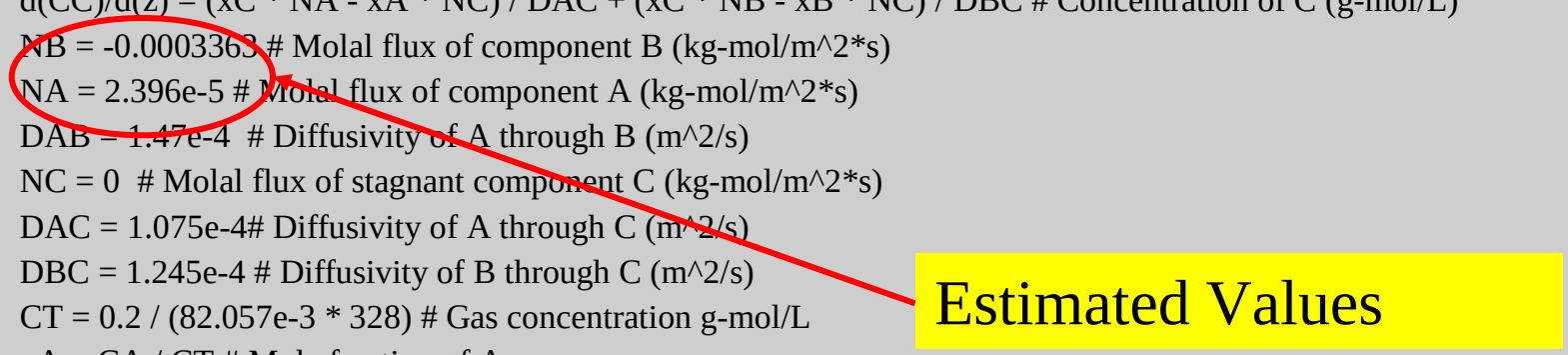
Estimates of *NA* and *NB* can be obtained from application of the Fick's law assuming simple binary diffusion. Estimates for N_A and N_B can be obtained from:

$$N_A = -D_{AC}\frac{(C_A|_2 - C_A|_1)}{(z|_2 - z|_1)} = -1.075 \times 10^{-4}\frac{(0 - 2.229 \times 10^{-4})}{(0.001 - 0)} = 2.396 \times 10^{-5}$$

$$N_B = -D_{BC}\frac{(C_B|_2 - C_B|_1)}{(z|_2 - z|_1)} = -1.245 \times 10^{-4}\frac{(2.701 \times 10^{-3} - 0)}{(0.001 - 0)} = -3.363 \times 10^{-4}$$

Simultaneous Multi-Component Diffusion of Gases – POLYMATH Code

No.	Equation # Comment
1	d(CA)/d(z) = (xA * NB - xB * NA) / DAB + (xA * NC - xC * NA) / DAC # Concentration of A (g-mol/L)
2	d(CB)/d(z) = (xB * NA - xA * NB) / DAB + (xB * NC - xC * NB) / DBC # Concentration of B (g-mol/L)
3	d(CC)/d(z) = (xC * NA - xA * NC) / DAC + (xC * NB - xB * NC) / DBC # Concentration of C (g-mol/L)
4	NB = -0.0003363 # Molal flux of component B (kg-mol/m^2*s)
5	NA = 2.396e-5 # Molal flux of component A (kg-mol/m^2*s)
6	DAB = 1.47e-4 # Diffusivity of A through B (m^2/s)
7	NC = 0 # Molal flux of stagnant component C (kg-mol/m^2*s)
8	DAC = 1.075e-4# Diffusivity of A through C (m^2/s)
9	DBC = 1.245e-4 # Diffusivity of B through C (m^2/s)
10	CT = 0.2 / (82.057e-3 * 328) # Gas concentration g-mol/L
11	xA = CA / CT # Mole fraction of A
12	xB = CB / CT # Mole fraction of B
13	xC = CC / CT # Mole fraction of C
14	z(0) = 0 # Length coordinate at point 1
15	CB(0) = 0 # Concentration of B at point 1
16	CA(0) = 0.0002229 # Concentration of A at point 1
17	CC(0) = 0.007208 # Concentration of C at point 1
18	z(f) = 0.001 # Length coordinate at point 2



Iterations on the NA and NB values have to be carried out to reach the specified final values of CA, CB and CC

Simultaneous Multi-Component Diffusion of Gases – POLYMATH Solution for Estimated N_A and N_B values

Calculated values of DEQ variables

	Variable	Initial value	Minimal value	Maximal value	Final value
1	CA	0.0002229	-1.692E-05	0.0002229	-1.692E-05
2	CB	0	0	0.002284	0.002284
3	CC	0.007208	0.0051638	0.007208	0.0051638
15	xC	0.9700056	0.6949123	0.9700056	0.6949123
16	z	0	0	0.001	0.001

No match between the specified and calculated final values

Component	Point 1 Concentration kg-mol/m^3	Point 2 Concentration kg-mol/m^3
A	2.229×10^{-4}	0
B	0	2.701×10^{-3}
C	7.208×10^{-3}	4.730×10^{-3}

Application of the Newton-Raphson Method for the Solution of Two Point Boundary Value Problems

Let us define **x** as the vector of unknown parameters (in this particular case $\mathbf{x} = (N_A\ N_B)^T$) and **f** as a vector of functions representing the difference between the desired and calculated concentration values as point 2 , thus

$$\mathbf{f} = \begin{bmatrix} C_A\,|_2 - 0 \\ C_B\,|_2 - 2.701\times10^{-3} \end{bmatrix}$$

The Newton-Raphson Method using Forward Difference to Calculate the Derivatives

The Newton-Raphson (NR) method can be written

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1} \mathbf{f}(\mathbf{x}_k) \quad k = 0, 1, 2\ldots$$

where k is the iteration number, $\mathbf{x}_0$ is the initial estimate and $\partial\mathbf{f}/\partial\mathbf{x}$ is the matrix of partial derivatives at $\mathbf{x} = \mathbf{x}_k$. The matrix of partial derivatives can be calculated using forward differences. thus

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i(\mathbf{x}_k + \boldsymbol{\delta}_j) - f_i(\mathbf{x}_k)}{\delta_j} \quad i = 1, 2; \quad j = 1, 2$$

where $\boldsymbol{\delta}_j$ is a vector containing the value of δ_j at the j^{th} position and zeroes elsewhere..

Simultaneous Multi-Component Diffusion of Gases – A MATLAB Function Generated by POLYMATH

Editor - C:\ASEE_SS\Example-6\MultDiffusA.m

File Edit Text Cell Tools Debug Desktop Window Help

```
21      function dYfuncvecdz = ODEfun(z,Yfuncvec);
22 -    CA = Yfuncvec(1)
23 -    CB = Yfuncvec(2)
24 -    CC = Yfuncvec(3)
25 -    NB = -.0003363; % Molal flux of component B (kg-mol/m^2*s)
26 -    NA = .00002396; % Molal flux of component A (kg-mol/m^2*s)
27 -    DAB = .000147; % Diffusivity of A through B (m^2/s)
28 -    NC = 0; % Molal flux of stagnant component A (kg-mol/m^2*s)
29 -    DAC = .0001075; % Diffusivity of A through C (m^2/s)
30 -    DBC = .0001245; % Diffusivity of B through C (m^2/s)
31 -    CT = .2 / (.082057 * 328); % Gas concentration g-mol/L
32 -    xA = CA / CT; % Mole fraction of A
33 -    xB = CB / CT; % Mole fraction of B
34 -    xC = CC / CT; % Mole fraction of C
35 -    dCAdz = (xA * NB - (xB * NA)) / DAB + (xA * NC - (xC * NA)) / DAC; % Concentration of A (g-mol/L)
36 -    dCBdz = (xB * NA - (xA * NB)) / DAB + (xB * NC - (xC * NB)) / DBC; % Concentration of B (g-mol/L)
37 -    dCCdz = (xC * NA - (xA * NC)) / DAC + (xC * NB - (xB * NC)) / DBC; % Concentration of C (g-mol/L)
38 -    dYfuncvecdz = [dCAdz; dCBdz; dCCdz];
```

Input parameters are transferred to the function in an array

Output parameters should be placed into a column vector

Template for Solving an ODE System*

Editor - C:\ASEE_SS\Example-6\MultDiffusA.m*

File Edit Text Cell Tools Debug Desktop Window Help

```
1   function MultDiffusA
2 - clear, clc, format short g, format compact
3 - tspan = [0 0.001]; % Range for the independent
4 - y0 = [0.0002229; 0; 0.007208]; % Initial values for the dependent variables function
5 - disp(' Variable values at the initial point ');
6 - disp([' t  = ' num2str(tspan(1))]);
7 - disp('      y          dy/dt     ');
8 - disp([y0 ODEfun(tspan(1),y0)]);
9 - [t,y]=ode45(@ODEfun,tspan,y0);
10 - for i=1:size(y,2)
11 -   disp([' Solution for dependent variable y' int2str(i)]);
12 -   disp(['        t          y' int2str(i)]);
13 -   disp([t y(:,i)]);
14 -   plot(t,y(:,i));
15 -   title([' Plot of dependent variable y' int2str(i)]);
16 -   xlabel(' Independent variable (t)');
17 -   ylabel([' Dependent variable y' int2str(i)]);
18 -   pause
19 - end
```

Data Generated by POLYMATH

The MATLAB library function *ode45* is used to solve the ODE system

*Available in the HELP section of POLYMATH

Simultaneous Multi-Component Diffusion of Gases – Newton-Raphson Iterations for Identifying the Parameters

Editor - C:\ASEE_SS\Example-6\MultDiffusB.m*

File Edit Text Cell Tools Debug Desktop Window Help

```
1   function MultDiffusB
2   clear, clc, format short g, format compact
3   tspan = [0 0.001]; % Range for the independent variable
4   y0 = [0.0002229; 0; 0.007208]; % Initial values for the dependent variables function
5   NAB(:,1)=[2.396e-5; -3.363e-4];
6   err=1;
7   it=0;
8   while (err>1e-10) & (it<20)
9     it=it+1;
10    itno(it)=it;
11    [t,y]=ode45(@ODEfun,tspan,y0,[],NAB(1,it),NAB(2,it));
12    f(:,it)=[y(end,1); y(end,2)-2.701e-3];
13    err=sqrt(f(:,it)'*f(:,it));
14    for j=1:2
15      delj=abs(NAB(j,it))*0.01;
16      NAB(j,it)=NAB(j,it)+delj;
17      [t,yp]=ode45(@ODEfun,tspan,y0,[],NAB(1,it),NAB(2,it));
18      fp=[yp(end,1); yp(end,2)-2.701e-3];
19      for k=1:2
20        DF(k,j)=(fp(k)-f(k,it))/delj;
21      end
22      NAB(j,it)=NAB(j,it)-delj;
23    end
24    NAB(:,it+1)=NAB(:,it)-inv(DF)*f(:,it);
25  end
```

Initial estimates for N_A and N_B

Input N_A and N_B as a parameters into the function

Derivative calculation loop

Newton-Raphson iterations loop

Multi-Component Diffusion – Results of Parameter Values

Note that five NR iterations, as shown below, are required for convergence with error tolerance of $\varepsilon_d = 10^{-10}$. The iterations of the NR method are stopped when

$$\left\|\mathbf{f}(\mathbf{x}_k)\right\| \leq \varepsilon_d$$

and where ε_d is the desired error tolerance set at $1\text{x}10^{-10}$.

The converged solution values are N_A = 2.1149e-5 and N_B = -4.1425e-4. Using these solution values, the difference between the calculated and desired values of C_A and C_B at point 2 are $<10^{-10}$.

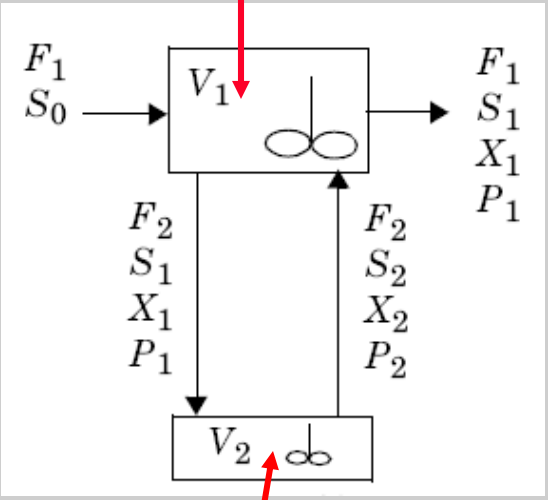
Iteration No.	N_A	N_B	f_1	f_2
0	2.3960E-05	-3.3630E-04	3.35E-05	-5.99E-03
1	2.2076E-05	-1.7614E-04	7.53E-06	-1.40E-03
2	2.1252E-05	-3.8575E-04	8.52E-07	-1.49E-04
3	2.1150E-05	-4.1375E-04	1.77E-08	-2.56E-06
4	2.1149E-05	-4.1424E-04	7.05E-11	-6.41E-09
5	2.1149E-05	-4.1425E-04	1.67E-13	-1.43E-11

Modeling and Optimization of a Chemostat with Imperfect Mixing*

No.	Equation # Comment
1	f(S1) = F1*S0+F2*S2-(1/Yxs)*(mum*S1/(Ks+S1))*X1*V1-F1*S1-F2*S1
	# Substrate balance on the well mixed volume
2	f(S2) = F2*S1-(1/Yxs)*(mum*S2/(Ks+S2))*X2*V2-F2*S2
	# Substrate balance on the stagnant volume
3	f(X1)=F2*X2+(mum*S1/(Ks+S1)-kd)*X1*V1-F1*X1-F2*X2
	# Cell balance on the well mixed volume
4	f(X2)=F2*X1+(mum*S2/(Ks+S2)-kd)*X2*V2-F2*X2
	# Cell balance on the stagnant volume
5	P1=Yps*(S0-S1) # Production (g/dm^3)
6	D=F1/(V1+V2) # Dilution rate (1/hr)
7	S0 = 0.6 # Feed substrate concentration (g/dm^3)
8	kd = 0.002
9	Yxs=0.4 # Yield coefficient (g cells/g substrate)
10	Yps=.2 # Yield coefficeint (g product/g substrate)
11	Ks = 0.2 # Monod constant (g substrate/L)
12	mum = 0.2 #Maximal specific growth rate (1/hr)
13	V1=1.7 # Well mixed voume (dm^3)
14	V2=0.3 # Stagnant volume (dm^3)
15	F1=.17 # Feed flow rate to the well mixed volume (dm^3/hr)
16	F2=0.2*F1 # Feed flow rate to the stagnant volume (dm^3/hr)
17	PR_DX1=D*X1 # Cell production rate (g/hr)
18	PR_DP1=D*P1 # Product production rate (g/hr)
19	S1(0) = 0
20	S2(0)=0
21	X1(0)=0.2
22	X2(0)=0.4

Maximize Cell production rate or Product production rate as function of Dilution rate

Well mixed volume



Stagnant volume

*p. 14.11 in Cutlip and Shacham, *Problem Solving In Chemical and Biochemical Engineering with Polymath, Excel and MATLAB*. Prentice-Hall, 2008.

Modeling and Optimization of a Chemostat with Imperfect Mixing – Results for D = 0.06 (1/hr) and D = 0.085 (1/hr)

POLYMATH Report
Nonlinear Equations

Calculated values of NLE variables

	Variable	Value	f(x)	Initial Guess
1	S1	0.1139428	9.403E-13	0
2	S2	0.0157426	2.343E-14	0
3	X1	0.1814265	-3.762E-13	0.2
4	X2	0.2153234	-9.426E-15	0.4

	Variable	Value
1	D	0.06
2	F1	0.12
3	F2	0.024
4	kd	0.002
5	Ks	0.2
6	mum	0.2
7	P1	0.0972114
8	PR_DP1	0.0058327
9	PR_DX1	0.0108856
10	S0	0.6
11	V1	1.7
12	V2	0.3
13	Yps	0.2
14	Yxs	0.4

POLYMATH Report
Nonlinear Equations

Calculated values of NLE variables

	Variable	Value	f(x)	Initial Guess
1	S1	0.2081633	4.324E-13	0
2	S2	0.0447391	6.833E-14	0
3	X1	0.1408439	-1.73E-13	0.2
4	X2	0.2026376	-2.737E-14	0.4

	Variable	Value
1	D	0.085
2	F1	0.17
3	F2	0.034
4	kd	0.002
5	Ks	0.2
6	mum	0.2
7	P1	0.0783673
8	PR_DP1	0.0066612
9	PR_DX1	0.0119717
10	S0	0.6
11	V1	1.7
12	V2	0.3
13	Yps	0.2
14	Yxs	0.4

Point by point calculation of the objective function values may not be the most efficient way for finding the optimum

Modeling and Optimization of a Chemostat – A MATLAB Function Generated by POLYMATH

No.	Equation % Comment
1	function fx = MNLEfun(x);
2	S1 = x(1); S2 = x(2);
3	X1 = x(3); X2 = x(4);
4	Yps = .2; %Yield coefficeint (g product/g substrate)
5	F1 = .17; %Feed flow rate to the well mixed volume (dm^3/hr)
6	S0 = .6; %Feed substrate concentration (g/dm^3)
7	kd = .002; % Cell death rate (1/hr)
8	Yxs = .4; %Yield coefficient (g cells/g substrate)
9	P1 = Yps * (S0 - S1); %Production (g/dm^3)
10	Ks = .2; %Monod constant (g substrate/L)
11	mum = .2; %Maximal specific growth rate (1/hr)
12	V1 = 1.7; %Well mixed volume (dm^3)
13	V2 = .3; %Stagnant volume (dm^3)
14	D = F1 / (V1 + V2); %Dilution rate (1/hr)
15	F2 = .2 * F1; %Feed flow rate to the stagnant volume (dm^3/hr)
16	PR_DX1 = D * X1; %Cell production rate (g/hr)
17	PR_DP1 = D * P1; %Product production rate (g/hr)
18	fx(1,1) = F1 * S0 + F2 * S2 - (1 / Yxs * mum * S1 / (Ks + S1) * X1 * V1) - (F1 * S1) - (F2 * S1);
19	fx(2,1) = F2 * S1 - (1 / Yxs * mum * S2 / (Ks + S2) * X2 * V2) - (F2 * S2);
20	fx(3,1) = F2 * X2 + (mum * S1 / (Ks + S1) - kd) * X1 * V1 - (F1 * X1) - (F2 * X2);
21	fx(4,1) = F2 * X1 + (mum * S2 / (Ks + S2) - kd) * X2 * V2 - (F2 * X2);

Input parameters are transferred to the function in an array

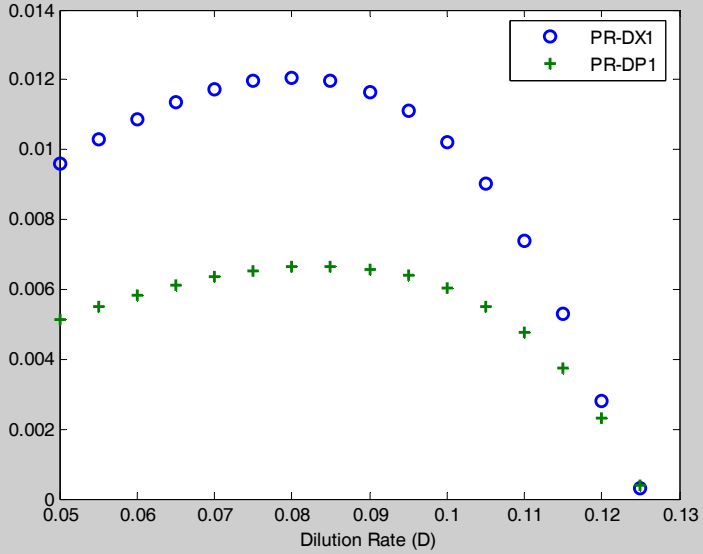
Output parameters should be placed into a column vector

Modeling and Optimization of a Chemostat – MATLAB Main Program and Results of Parametric Runs

```
No. Equation
 1  options = optimset('Diagnostics',['off'],'TolFun',[1e-9],'TolX',[1e-9]);
 2  Yps = 0.2;  S0 = 0.6;  kd = 0.002;  Yxs = 0.4;  Ks = 0.2;
 3  mum = 0.2;  V1 = 1.7;  V2 = 0.3;
 4  F1=0.1;
 5  xguess = [0 0 0.2 0.4]; % initial guess vector
 6  for k=1:16
 7     xsolv=fsolve(@MNLEfun,xguess,options,F1);
 8     S1(k)=xsolv(1); S2(k)=xsolv(2); X1(k)=xsolv(3); X2(k)=xsolv(4);
 9     F1list(k)=F1; D(k) = F1 / (V1 + V2); P1(k)= Yps * (S0 - S1(k));
10     PR_DX1(k) = D(k) * X1(k);  PR_DP1(k) = D(k) * P1(k);
11     F1=F1+0.01;
12  end
```

The MATLAB library function fsolv is used to solve the system of equations.

Calculation of the production rates for parametric runs



Modeling and Optimization of a Chemostat

```
No.  Command
 23  Lb=0.1;
 24  Ub=0.25;
 25   [maxF1, PR_DX] = fminbnd(@ProdRateCell,Lb,Ub);
 26  disp([' Highest Production Rate for Cells is ' num2str(-PR_DX) ' at Dilution Rate of ' num2str(maxF1/(V1+V2))])
 27   [maxF1, PR_DP] = fminbnd(@ProdRateProd,Lb,Ub);
 28  disp([' Highest Production Rate for Product is ' num2str(-PR_DP) ' at Dilution Rate of ' num2str(maxF1/(V1+V2

 29  function PR_DX=ProdRateCell(F1)
 30  V1 = 1.7;
 31  V2 = 0.3;
 32  xguess = [0 0 0.2 0.4]; % initial guess vector
 33  options = optimset('Diagnostics',['off'],'TolFun',[1e-9],'TolX',[1e-9]);
 34   xsolv=fsolve(@MNLEfun,xguess,options,F1);
 35   X1=xsolv(3);
 36   D = F1 / (V1 + V2);
 37   PR_DX = -D* X1;
```

The MATLAB library function ***fminbnd*** is used to find the minimum

The MATLAB library function ***fsolv*** is used to solve the system of equations

Objective function to be minimized

Highest Production Rate for Cells is 0.01207 at Dilution Rate of 0.079932

Highest Production Rate for Product is 0.0066643 at Dilution Rate of 0.083729

CONCLUSIONS

- **For *Single-Model, Single-Algorithm* problems, a *software package* and the user supplied mathematical model and property data are *sufficient for achieving the solution*.**

- **For *Multiple-Model and/or Multiple-Algorithm* problems, the number of possible combinations is so large that it is impossible to provide pre-tailored solution algorithms and *programming is essential* .**

- **For the later a *combinations of software packages* (such as POLYMATH and MATLAB) provides the most *efficient means for solution*.**