Computational Methods

- We summarize many of the fundamental computational procedures required in reactor analysis and design.
- We illustrate these procedures using the numerical computing language Octave [4, 5] or, equivalently, MATLAB.
- Octave is freely available for a variety of hardware platforms and can be downloaded from `www.octave.org`.
- The MATLAB package is commercially available from The MathWorks, Inc., and is becoming a commonly available tool of industrial engineering practice.

Software

- Many of the figures and tables in this text required numerical solutions of various types of chemical reactor problems.
- All computations required to produce every figure and table in this text were performed with Octave.
- The Octave files required to produce these figures and tables are freely available at these two websites:
  www.nobhillpublishing.com
  www.engineering.ucsb.edu/~jbraw/chemreacfun

- After starting an Octave session, type `help -i introduction` at the Octave prompt for information on how to enter matrices and perform matrix multiplication.
- Consider again the water gas shift reaction chemistry

$$\begin{bmatrix} 0 & 1 & 0 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{H} \\ \mathrm{H_2} \\ \mathrm{OH} \\ \mathrm{H_2O} \\ \mathrm{CO} \\ \mathrm{CO_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Linear Independence

- Say we would like to find the number of linearly independent reactions, and then compute the species production rates from given reaction rates, $[r_1\ r_2\ r_3] = [1\ 2\ 3]$.

Linear Independence

- A typical set of Octave commands to perform these operations would appear as follows

```
octave:1> stoi=[0 1 0 -1 -1 1; -1 1 1 -1 0 0; \
                1 0 -1 0 -1 1];
octave:2> rank(stoi)
ans = 2
octave:3> r=[1;2;3];
octave:4> R=stoi'*r
R =
   1
   3
  -1
  -3
  -4
   4
```

Least Squares Estimation of $\boldsymbol{r}$

- As we explored in Chapters 2 and 9, from measured $\boldsymbol{R}$, we can solve for $\boldsymbol{r}$ as a least-squares problem.
- Given the production rates and reaction rates are related by

$$\boldsymbol{R} = \boldsymbol{\nu}^T \boldsymbol{r} \tag{A.1}$$

- The least-squares solution is given by

$$\boldsymbol{r} = (\boldsymbol{\nu}\, \boldsymbol{\nu}^T)^{-1} \boldsymbol{\nu} \boldsymbol{R}$$

in which the superscript -1 indicates a matrix inverse.

Least Squares

- The command in Octave and MATLAB for the least-squares solution is

```
r = stoi' \ R
```

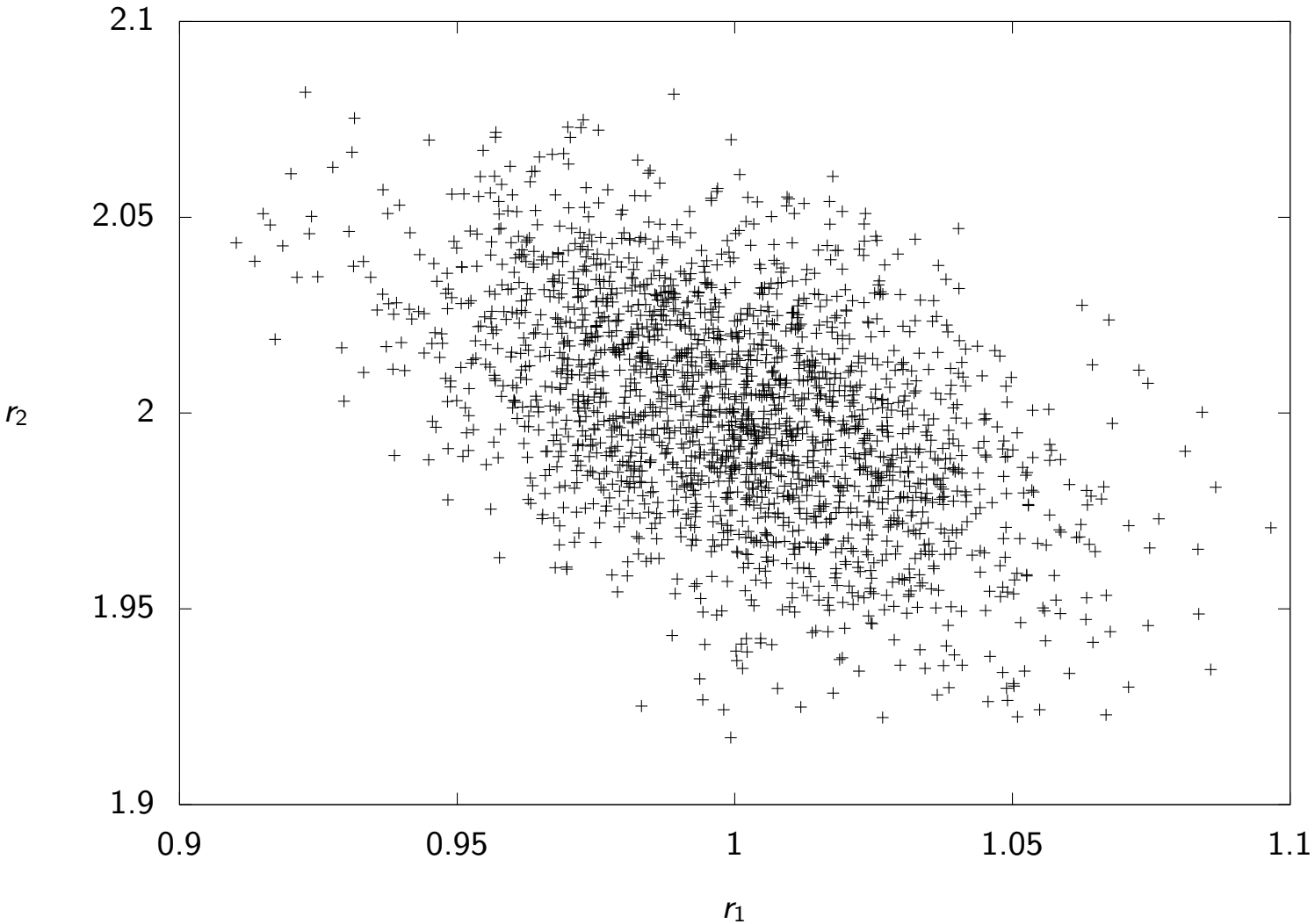
- This formula can be recalled with the mnemonic of a "left division" of Equation A.1 by $\boldsymbol{\nu}^T$.

Example A.1: Estimating reaction rates

- Consider the first and second water gas shift reaction as an independent set and let $[r_1\ r_2] = [1\ 2]$.
- Create 2000 production rate measurements by adding noise to $\boldsymbol{R} = \boldsymbol{\nu}^T \boldsymbol{r}$ and estimate the reaction rates from these data.
- Plot the distribution of estimated reaction rates.

□

The Distribution of Estimated Reaction Rates



The Octave Code

- The Octave commands to generate these results are

```
stoi=[0 1 0 -1 -1 1; -1 1 1 -1 0 0];
r=[1;2];
R=stoi'*r;
npoints = 2000;
for i=1:npoints R_meas(:,i)=0.05*randn(6,1)+R; endfor
r_est=stoi' \ R_meas;
```

- The figure shows the estimates.
- We know from Chapter 9 that the distribution of estimates is a multivariate normal.
- We also know how to calculate the α-level confidence intervals.

Nonlinear Algebraic Equations and Optimization

- Determining equilibria for reacting systems with multiple phases and reactions can require significant computational effort.
- As we saw in Chapter 3, the phase and reaction equilibrium conditions generally lead to mathematical problems of two types: solving nonlinear algebraic equations and minimizing a nonlinear function subject to constraints.
- In this section it is assumed that the required thermochemical data are available, but finding or measuring these data is often another significant challenge in computing equilibria for systems of industrial interest.
- See also Section 3.2.1 for a brief discussion of thermochemical databases.

Defining Functions (`function`)

- Octave and MATLAB provide many convenient built-in functions.
- Some that we have used in this text include the matrix exponential, `expm`; incomplete gamma function, `gammai`; pseudorandom number generators, `rand` and `randm`; and several others.
- Type `help -i` and select the menu item `Function Index` for a complete list of Octave's built-in functions.

Defining Functions (`function`)

- But often we need to define our own functions.
- To solve algebraic equations or optimization problems, for example, the user needs to provide a function that can be called by the solver to evaluate the nonlinear equations or the objective function.
- To define a function in Octave, $f(x) = x^2$ for example, the command structure is

```
function y = f(x)
  y = x*x;
endfunction
```

Defining Functions (`function`)

```
function y = f(x)
  y = x*x;
endfunction
```

- The first line states that the value of the variable `y` is returned by invoking the function named `f` with the input value `x`.
- The body of the function, then, is the calculation of the value of `y` to be returned.

The m-files

- As the computational complexity of the function increases, we would like to store our work so that it does not need to be reentered at the beginning of each subsequent Octave session.
- Octave commands and function definitions can be stored in text files, called m-files
- The naming convention is `filename.m`.
- Editing these files with a text editor during an interactive Octave session provides a convenient means for debugging new calculations.

Solving Nonlinear Algebraic Equations (`fsolve`)

- We consider again the problem in Example 3.5. The equilibrium condition requires the solution of two equations in two unknowns,

$$PK_1 y_I y_B - y_{P_1} = 0$$
$$PK_2 y_I y_B - y_{P_2} = 0$$

- $y_I, y_B, y_{P_1}, y_{P_2}$ are defined in terms of the two reaction extents in Equations 3.66.

Function Definition

- An Octave function defining these two equations is

```
function residual = dgdx(x)
 K1 = 108; K2 = 284; P  = 2.5;
 yI0 = 0.5; yB0 = 0.5; yP10 = 0; yP20 = 0;
 d   = 1 - x(1) - x(2);
 yI  = (yI0 - x(1) - x(2)) / d;
 yB  = (yB0 - x(1) - x(2)) / d;
 yP1 = (yP10 + x(1)) / d;
 yP2 = (yP20 + x(2)) / d;
 residual(1) = P*K1*yI*yB - yP1;
 residual(2) = P*K2*yI*yB - yP2;
endfunction
```

Using the Function

- Notice that `x` is a vector containing the two reaction extents provided to the function named `dgdx` when it is invoked, and `residual` is a vector containing the values of the two equilibrium conditions to be returned by the function.
- We seek values of `x` such that `residual` is zero. With this function defined, we can use the built-in nonlinear equation solver to find its solution, by entering

```
[x,info] = fsolve("dgdx", x0)
```

`fsolve`

- The `fsolve` function requires the name of the function, inside quotes, and an initial guess for the unknown extents, provided in the variable `x0`, and returns the solution in `x` and a flag `info` indicating if the calculation was successful.
- It is highly recommended to examine these information flags after all calculations.
- Reactor equilibrium problems can be numerically challenging, and even the best software can run into problems.

Solving the Problem

- After the function `dgdx` is defined, the following is a typical session to compute the solution given in Example 3.5

```
octave:1> x0=[0.2;0.2];
octave:2> [x,info] = fsolve("dgdx",x0)
x =
  0.13317
  0.35087
info = 1
```

- The value of `info = 1` indicates that the solution has converged.

Solving Nonlinear Optimization Problems (`fmincon`)

- The other main approach to finding the reaction equilibrium is to minimize the appropriate energy function, in this case the Gibbs energy.
- This optimization-based formulation of the problem, as shown in Example 3.3, can be more informative than the algebraic approach discussed above.

Gibbs Energy Function

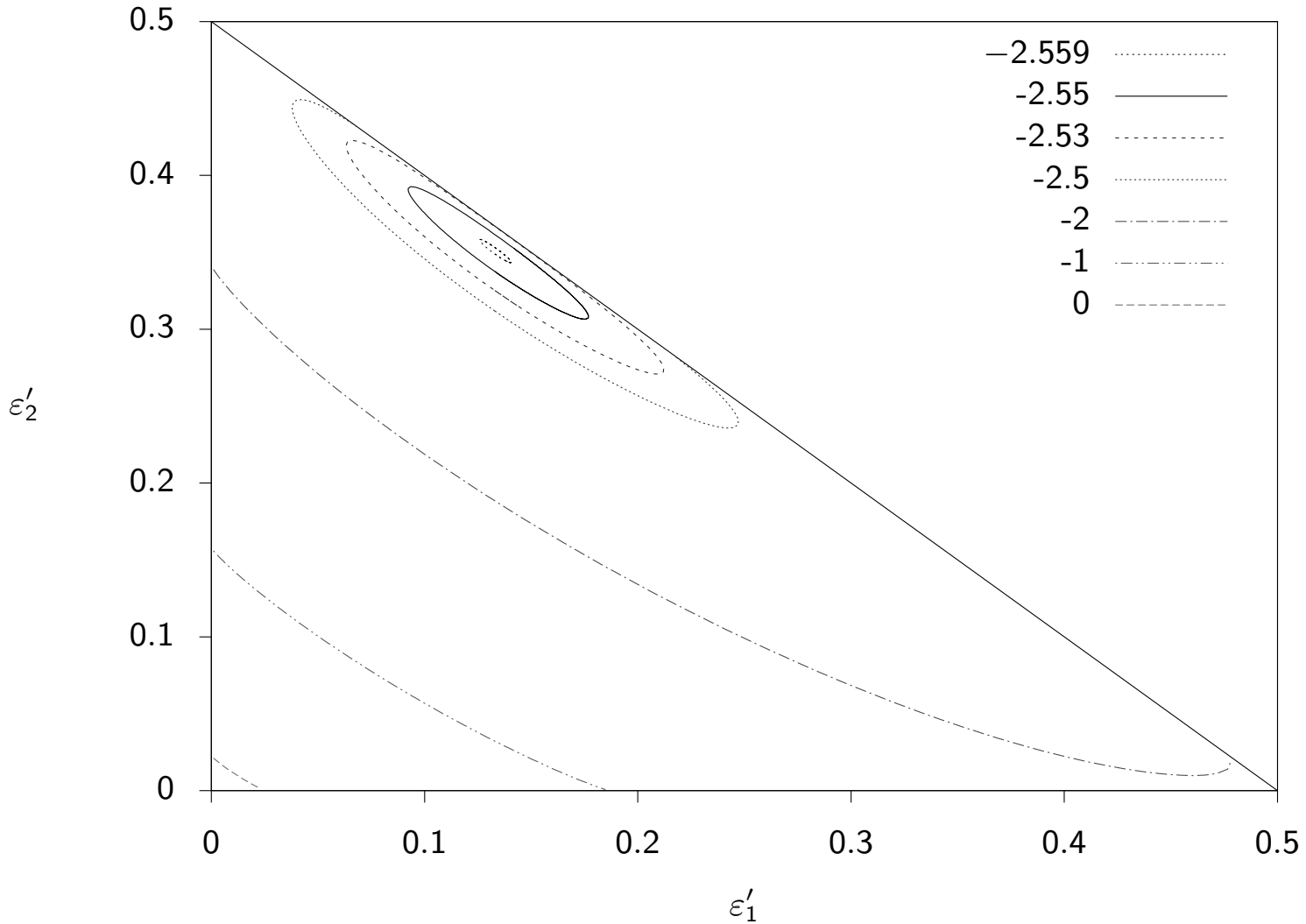
- In Chapter 3, we derived the following Gibbs energy function for an ideal-gas mixture

$$\begin{aligned}\tilde{G} &= -\sum_i \varepsilon_i' \ln K_i + \left(1 + \sum_i \bar{\nu}_i \varepsilon_i'\right) \ln P \\ &\quad + \sum_j \left(y_{j0} + \sum_i \nu_{ij} \varepsilon_i'\right) \ln \left[\frac{y_{j0} + \sum_i \nu_{ij} \varepsilon_i'}{1 + \sum_i \bar{\nu}_i \varepsilon_i'}\right]\end{aligned}$$

- The minimization of this function of ε_i' then determines the two equilibrium extents.

Minimization of $\tilde{G}$

The figure shows the lines of constant Gibbs energy as a function of the two reaction extents.



Minimization of $\tilde{G}$

- We see immediately that the minimum is unique.
- Notice that $\tilde{G}$ is not defined for all reaction extents, and the following constraints are required to ensure nonnegative concentrations

$$0 \leq \varepsilon_1' \qquad 0 \leq \varepsilon_2' \qquad \varepsilon_1' + \varepsilon_2' \leq 0.5$$

Octave Code

- First we define a function that evaluates $\tilde{G}$ given the two reaction extents.

```
function retval=gibbs(x)
 dg1= -3.72e3; dg2= -4.49e3; T=400; R=1.987; P=2.5;
 K1 = exp(-deltag1/(R*T)); K2 = exp(-deltag2/(R*T));
 yI0 = 0.5; yB0 = 0.5; yP10 = 0; yP20 = 0;
 d   = 1 - x(1) - x(2);
 yI  = (yI0 - x(1) - x(2)) / d;
 yB  = (yB0 - x(1) - x(2)) / d;
 yP1 = (yP10 + x(1)) / d;
 yP2 = (yP20 + x(2)) / d;
 retval = - (x(1)*log(K1)+x(2)*log(K2)) + ...
    (1-x(1)-x(2))*log(P) + yI*d*log(yI) + ...
    yB*d*log(yB) + yP1*d*log(yP1) + yP2*d*log(yP2);
endfunction
```

Minimizing

- To solve the optimization we use the Octave or MATLAB function `fmincon`

```
[x,obj,info]=fmincon(@gibbs,x0, A, b, Aeq, beq, lb,ub)
```

- `x0` is the initial guess as before and the new variables are used to define the constraints (if any).
- The variable `lb` provides lower bounds for the extents and `ub` provides upper bounds for the extents. These are required to prevent the optimizer from "guessing" reaction extent values for which the Gibbs energy function is not defined.

Constraints

- For example, consider what the `gibbs` function returns given negative reaction extents.
- Finally, we specify the sum of extents is less than 0.5 with the last three arguments, which define linear inequality constraints other than simple bounds,

$$a_{\mathrm{lb}} \leq A\varepsilon' \leq a_{\mathrm{ub}}$$

$$\varepsilon_1' + \varepsilon_2' \leq 1/2$$

$$A = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad a_{ub} = 1/2$$

- We can set $a_{lb} = 0$ because both extents are already specified to be positive.

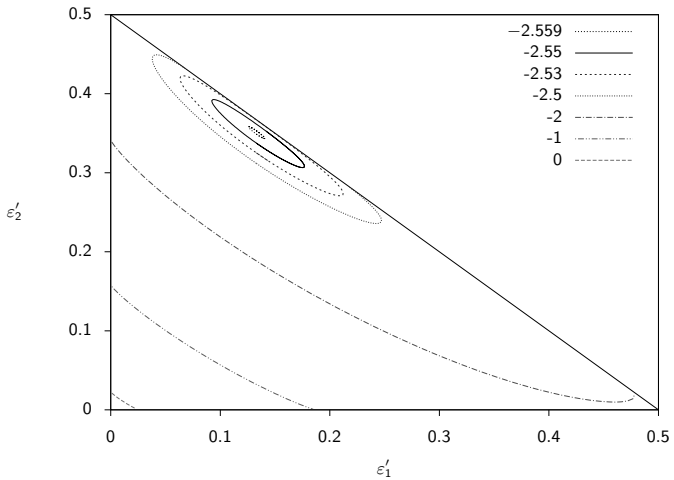
Solving the Problem

- A typical session to solve for equilibrium composition by minimizing Gibbs energy is

```
octave:1> A=[1 1]; b=0.4999
octave:2> lb=[0;0]; ub=[0.5;0.5];
octave:3> x0=[0.2;0.2];
octave:4> [x,obj,info]=fmincon(@gibbs,x0,A,b,[],[],lb,ub)
x =
   0.13317
   0.35087
obj = -2.5593
info =  2
```

- The value of `info = 2` indicates that the solution has converged

Comparing Solutions



- The optimization results are in good agreement with those computed using the algebraic approach, and the Gibbs energy contours

- Optimization is a powerful tool for solving many types of engineering modeling and design problems.
- We also rely heavily on optimization tools in Chapter 9 on parameter estimation.

Differential Equations (`lsode`) I

Type `doc` and select the menu item `Differential Equations` for an overview of Octave's ODE solver, `lsode` [6].
Type `help ode15s` for an overview of MATLAB's ODE solvers. We recommend always using the so-called "stiff" solvers for chemical reactor modeling. The time scales for different reactions may be widely separated, leading to stiff ODEs.
The ODEs required to describe transient-batch, CSTR, and steady-state PFR reactors are of the form

$$\frac{dx}{dt} = f(x)$$
$$x(0) = x_0$$

in which $x(t)$ is a vector of species concentrations to be found, t is time in the batch reactor and CSTR, but volume or length in the steady-state PFR, and f is the function defining the material balances for the species. If the differential equations are nonlinear (f is not a linear function) and one must keep track of several species simultaneously, then analytical solution is not possible and a numerical procedure is required.
The first task is to formulate a function which defines f. If we examine the benzene pyrolysis example, Example 4.5, a suitable function is

Differential Equations (`lsode`) II

```
function xdot = benzene_pyrol_rhs(ext,volume)
 global k1 K1 k2 K2 R T P NBf
 NB = NBf - 2*ext(1) - ext(2);
 ND = ext(1) - ext(2);
 NH = ext(1) + ext(2);
 NT = ext(2);
 Q = NBf*R*T/P;
 cB = NB/Q;  cD = ND/Q;  cT = NT/Q;  cH = NH/Q;
 xdot(1) = k1*(cB*cB - cD*cH/K1);
 xdot(2) = k2*(cB*cD - cT*cH/K2);
endfunction
```

in which the meanings of the symbol names should be apparent from the example.
We then need to specify at which t values, volumes in this case, the solution should be reported, the initial condition (in this case, feed condition), x_0, and call the ODE solver.
A typical Octave session would be

Differential Equations (lsode) III

```
NBf = 60e3; R  = 0.08205; T  = 1033; P  = 1;
k1 = 7e5; k2 = 4e5; K1 = 0.31; K2 = 0.48;
x0=[0;0];
vout = linspace(0,1600,50)';
x = lsode("benzene_pyrol_rhs", x0, vout);
conv = ( 2*x(:,1) + x(:,2) ) /NBf;
yB   = ( NBf - 2*x(:,1) - x(:,2) ) /NBf;
yD   = ( x(:,1) - x(:,2) ) /NBf;
yT   = ( x(:,2) ) /NBf;
yH   = ( x(:,1) + x(:,2) ) /NBf;
```

in which the function `linspace` computes 50 linearly spaced points between 0 and 1600 L. Finally, any quantities such as conversions or yields, which are computable from x, are calculated for plotting. The results of this calculation are shown in Figures 4.20 and 4.21.

Differential-Algebraic Equations (daspk)

Some reactor models require a more general structure than the ODE, $d\boldsymbol{x}/dt = \boldsymbol{f}(\boldsymbol{x}, t)$. The nonconstant density, nonconstant volume semi-batch and CSTR reactors in Chapter 4 are more conveniently expressed as differential-algebraic equations (DAEs). To address these models, consider the more general form of implicit ODEs

$$\boldsymbol{0} = \boldsymbol{f}(d\boldsymbol{x}/dt, \boldsymbol{x}, t) \tag{A.3}$$

Both DAEs and ODEs can be considered special cases of this structure. Brenan et al. [1] provide further reading on existence and uniqueness of solutions to these models, which are considerably more complex issues than in the case of simple ODEs. Initial conditions are required for $d\boldsymbol{x}/dt$ as well as $\boldsymbol{x}$ in this model,

$$\frac{d\boldsymbol{x}}{dt}(t) = \dot{\boldsymbol{x}}_0 \qquad \boldsymbol{x}(t) = \boldsymbol{x}_0, \qquad \text{at } t = 0$$

Brown, Hindmarsh and Petzold [2, 3] have provided a numerical package, `daspk`, to compute solutions to implicit differential, and differential-algebraic equations. The main difference between using `daspk` and `lsode` is the form of the user-supplied function defining the model. A second difference is that the user must supply $\dot{\boldsymbol{x}}_0$ as well as $\boldsymbol{x}_0$.

Example Using daspk

As an example, we can solve the previous ODEs using daspk. First we modify the right-hand side function benz_pyrol_rhs of the ODEs to return the *residual* given in Equation A.3. We call the new function benz_pyrol

```
function res = benz_pyrol(ext, extdot, volume)
 global k1 K1 k2 K2 R T P NBf
 NB = NBf - 2*ext(1) - ext(2);
 ND = ext(1) - ext(2);
 NH = ext(1) + ext(2);
 NT = ext(2);
 Q = NBf*R*T/P;
 cB = NB/Q;  cD = ND/Q;  cT = NT/Q;  cH = NH/Q;
 res(1) = extdot(1) - k1*(cB*cB - cD*cH/K1);
 res(2) = extdot(2) - k2*(cB*cD - cT*cH/K2);
endfunction
```

Notice that when this function returns zero for res, the variables extdot and ext satisfy the differential equation. With this function defined, we then solve the model by calling daspk

```
x0=[0;0]; xdot0 = [0;0];
vout = linspace(0,1600,50)';
[x, xdot] = daspk(@benz_pyrol, x0, xdot0, vout);
```

Notice that both the state x and state time (volume) derivative xdot are returned at the specified times (volumes) vout.

Automatic Stopping Times (dasrt)

We often need to stop ODE solvers when certain conditions are met. Two examples are when we have reached a certain conversion, and when we have created a new phase and need to change the ODEs governing the system. The program dasrt enables us to stop when specified conditions are met exactly. The following code illustrates how to find the PFR volume at which 50% conversion of benzene is achieved. The user provides one new function, stop, which the ODE solver checks while marching forward until the function reaches the value zero. The PFR volume is found to be $V_R = 403.25$ L, in good agreement with Figure 4.20.

```
function val = stop(ext, volume)
  global NBf xBstop
  NB = NBf - 2*ext(1) - ext(2);
  xB = 1 - NB/NBf;
  val = xBstop - xB;
endfunction
```

```
octave:1> x0 = [0;0]; xdot0 = [0;0];
octave:2> vout = linspace(0,1600,50)'; xBstop = 0.5;
octave:3> [x, xdot, vstop] = dasrt("benzene_pyrol_rhs", \
                                   "stop", x0, xdot0, vout);
octave:4> vstop(length(vstop))
ans = 403.25
```

Parametric Sensitivities of Differential Equations (cvode_sens) I

As discussed in Chapter 9, it is often valuable to know not just the solution to a set of ODEs, but how the solution changes with the model parameters. Consider the following vector of ODEs with model parameters $\boldsymbol{\theta}$.

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta})$$
$$\boldsymbol{x}(0) = \boldsymbol{g}(\boldsymbol{x}_0;\boldsymbol{\theta})$$

The sensitivities, $\boldsymbol{S}$, are defined as the change in the solution with respect to the model parameters

$$S_{ij} = \left(\frac{\partial x_i}{\partial \theta_j}\right)_{\theta_{k\neq j}}$$
$$\boldsymbol{S} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}^T}$$

We can differentiate the S_{ij} to obtain

$$\frac{dS_{ij}}{dt} = \frac{d}{dt}\left[\frac{\partial x_i}{\partial \theta_j}\right] = \frac{\partial}{\partial \theta_j}\frac{dx_i}{dt} = \frac{\partial}{\partial \theta_j} f_i(\boldsymbol{x};\boldsymbol{\theta})$$

Parametric Sensitivities of Differential Equations (cvode_sens) II

Using the chain rule to perform the final differentiation gives

$$\left(\frac{\partial f_i}{\partial \theta_j}\right)_{\theta_{k\neq j}} = \sum_k \left(\frac{\partial f_i}{\partial x_k}\right)_{x_{l\neq k},\theta_j} \left(\frac{\partial x_k}{\partial \theta_j}\right)_{\theta_{k\neq j}} + \left(\frac{\partial f_i}{\partial \theta_j}\right)_{x_l,\theta_{k\neq j}} \tag{A.4}$$

Substituting this back into Equation A.4 and writing the sum as a matrix multiply yields

$$\frac{d\boldsymbol{S}}{dt} = \boldsymbol{f}_x \boldsymbol{S} + \boldsymbol{f}_\theta$$

in which the subscript denotes differentiation with respect to that variable. Notice the sensitivity differential equation is linear in the unknown $\boldsymbol{S}$. The initial conditions for this matrix differential equation is easily derived from the definition of $\boldsymbol{S}$

$$S_{ij}(0) = \frac{\partial x_i(0)}{\partial \theta_j} = \frac{\partial g_i}{\partial \theta_j}$$

$$\frac{d\boldsymbol{S}}{dt} = \boldsymbol{f}_x \boldsymbol{S} + \boldsymbol{f}_\theta$$
$$\boldsymbol{S}(0) = \boldsymbol{g}_\theta$$

Notice that if none of the initial conditions are parameters for which one is computing sensitivities, then $\boldsymbol{S}(0) = \boldsymbol{0}$.

Two simple sensitivity calculations

Example A.2: Two simple sensitivity calculations

Consider the differential equation describing a first-order, isothermal, irreversible reaction in a constant-volume batch reactor

$$\begin{aligned} \frac{dc_A}{dt} &= -kc_A \\ c_A(0) &= c_{A0} \end{aligned} \tag{A.5}$$

1. First write out the solution to the differential equation.
2. Next consider the rate constant to be the parameter in the problem, $\theta = k$. Take the partial derivative of the solution directly to produce the sensitivity to this parameter, $S = \partial c_A / \partial k$.
3. Next write the differential equation describing dS/dt. What is $S(0)$? Solve this differential equation and compare to your previous result. Plot $c_A(t), S(t)$ versus t.
4. Repeat steps 2 and 3 using the initial concentration as the parameter in the problem, $\theta = c_{A0}$.

□

Solution I

1. Equation A.5 is our standard, first-order linear equation, whose solution is

$$c_A = c_{A0} e^{-kt}$$

2. Taking the partial derivative of $c_A(t)$ with respect to $\theta = k$ produces directly

$$S_1 = -t c_{A0} e^{-kt} \tag{A.6}$$

Evaluating the various partial derivatives gives

$$f_x = -k \qquad f_\theta = -c_A \qquad g_\theta = 0$$

so the sensitivity differential equation is

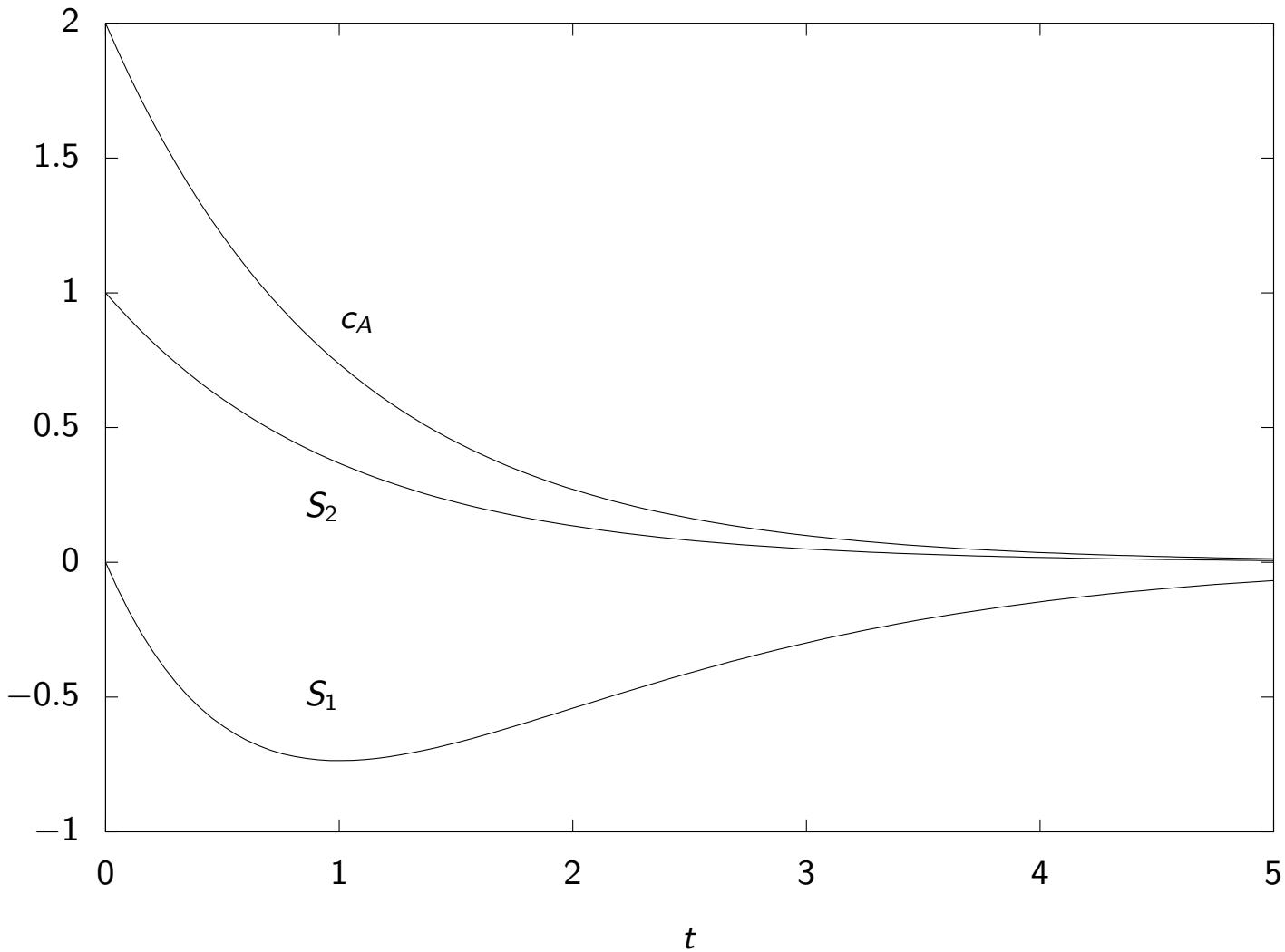
$$\frac{dS_1}{dt} = -kS_1 - c_A$$
$$S_1(0) = 0$$

This equation is linear and can be solved with the method used in the series reactions in Chapter 4, or with Laplace transforms as in Exercise 4.6. The result is

$$S_1(t) = -tkc_{A0} e^{-kt}$$

which agrees with Equation A.6. The functions $c_A(t), S_1(t)$ are shown in Figure A.3.

Solution II



Solution III

Figure A.3: Solution to first-order differential equation $dc_A/dt = -kc_A$, and sensitivities $S_1 = \partial c_A/\partial k$ and $S_2 = \partial c_A/\partial c_{A0}$.

3. Taking the partial derivative of $c_A(t)$ with respect to $\theta = c_{A0}$ produces directly

$$S_2(t) = e^{-kt} \tag{A.7}$$

Evaluating the various partial derivatives for this parameter choice gives

$$f_x = -k \qquad f_\theta = 0 \qquad g_\theta = 1$$

so the sensitivity differential equation is

$$\frac{dS_2}{dt} = -kS_2$$
$$S_2(0) = 1$$

This equation is our standard, first-order linear equation whose solution is

$$S_2(t) = e^{-kt}$$

which agrees with Equation A.7. This sensitivity also is plotted in Figure A.3. Notice that only when the initial condition is the parameter is the sensitivity initially nonzero.

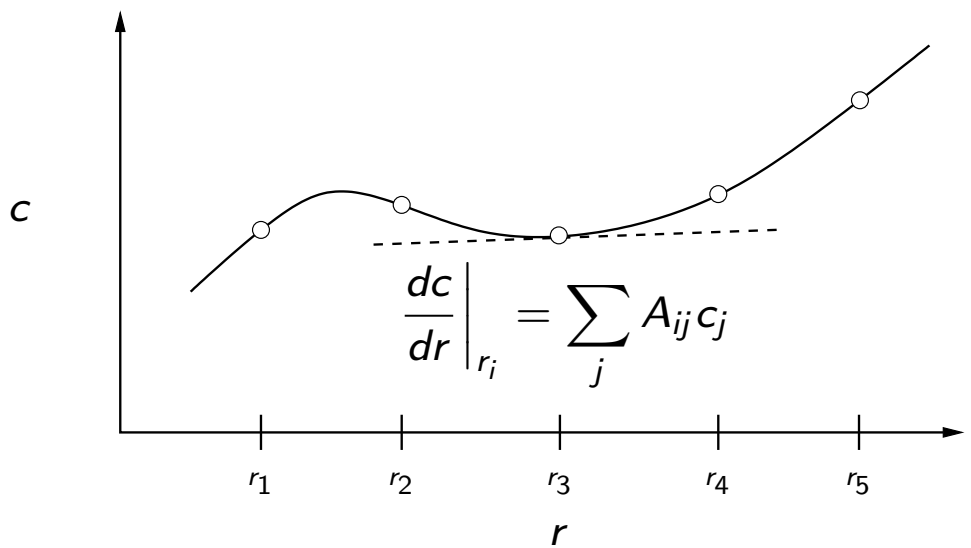
Computing sensitivities

The following Octave commands show how to use paresto to solve Example A.2.

```
ca0 = 2; k   = 1; tfinal = 5;
nts    = 100; tout = linspace(0,tfinal,nts)';

model = struct;
model.x = {'ca'};
model.p = {'k'};
model.d = {'dummy'};
model.tout = tout;
model.ode = @(t, y, p) {-p.k*y.ca};
% dummy objective function; just finding sensitivities
model.lsq = @(t, y, p) {y.ca};
pe = paresto(model);

%% initialize state and parameters
x0 = ca0; thetaac = [k; x0];

theta0 = thetaac;
lbtheta = theta0;
ubtheta = theta0;

[est, y, p] = pe.optimize(zeros(1,nts), ...
                theta0, lbtheta, ubtheta);

plot (tout, [y.ca', est.dca_dk, est.dca_dca0]);
```

Boundary-Value Problems and Collocation (colloc) I

In the collocation method, we approximate a function by passing a polynomial through values of the function at selected points. The selected points are known as collocation points. The locations of the collocation points have a large impact on how well the method works. Evenly spaced points, which seems a natural first choice, turns out to have mediocre properties. Choosing the points as zeros of a member of a family of orthogonal polynomials turns out to have much better properties. This choice is referred to as orthogonal collocation.



Boundary-Value Problems and Collocation (`colloc`) II

Figure A.4: Function $c(r)$ and its values at five collocation points. Derivatives and integrals of the polynomial interpolant are linear combinations of the function values at the points.

Figure A.4 shows values of a function $c(r)$ at five collocation points. The function is approximated by passing a polynomial through these points. To solve differential equations and boundary-value problems (BVP), we first compute the required derivatives of the polynomial approximation. Then we then find the values of the function such that the differential equation is satisfied at the collocation points. If we increase the number of collocation points, n_c, we require the differential equation to be satisfied at more locations, and we obtain a more accurate solution.

Boundary-Value Problems and Collocation (colloc) III

The derivatives and integrals of the polynomial interpolant can be computed as linear combinations of the values at the collocation points

$$\left.\frac{dc}{dr}\right|_{r_i} = \sum_{j=1}^{n_c} A_{ij} c_j$$

$$\left.\frac{d^2c}{dr^2}\right|_{r_i} = \sum_{j=1}^{n_c} B_{ij} c_j$$

$$\int_0^1 f(c)dc = \sum_{j=1}^{n_c} Q_j f(c_j)$$

To obtain the locations of the collocation points and derivatives and integral weighting matrices and vectors, we use the function `colloc`, based on the methods described by Villadsen and Michelsen [7].

```
[R A B Q] = colloc(npts-2, "left", "right");
```

The strings `"left"` and `"right"` specify that we would like to have collocation points at the endpoints of the interval in addition to the zeros of the orthogonal polynomial. We solve for the concentration profile for the reaction-diffusion problem in a catalyst pellet to illustrate the collocation method.

Example A.3: Single-pellet profile

Consider the isothermal, first-order reaction-diffusion problem in a spherical pellet

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dc}{dr}\right) - \Phi^2 c = 0 \tag{A.8}$$

$$c = 1, \qquad r = 3$$
$$\frac{dc}{dr} = 0, \qquad r = 0$$

The effectiveness factor is given by

$$\eta = \left.\frac{1}{\Phi^2}\frac{dc}{dr}\right|_{r=3}$$

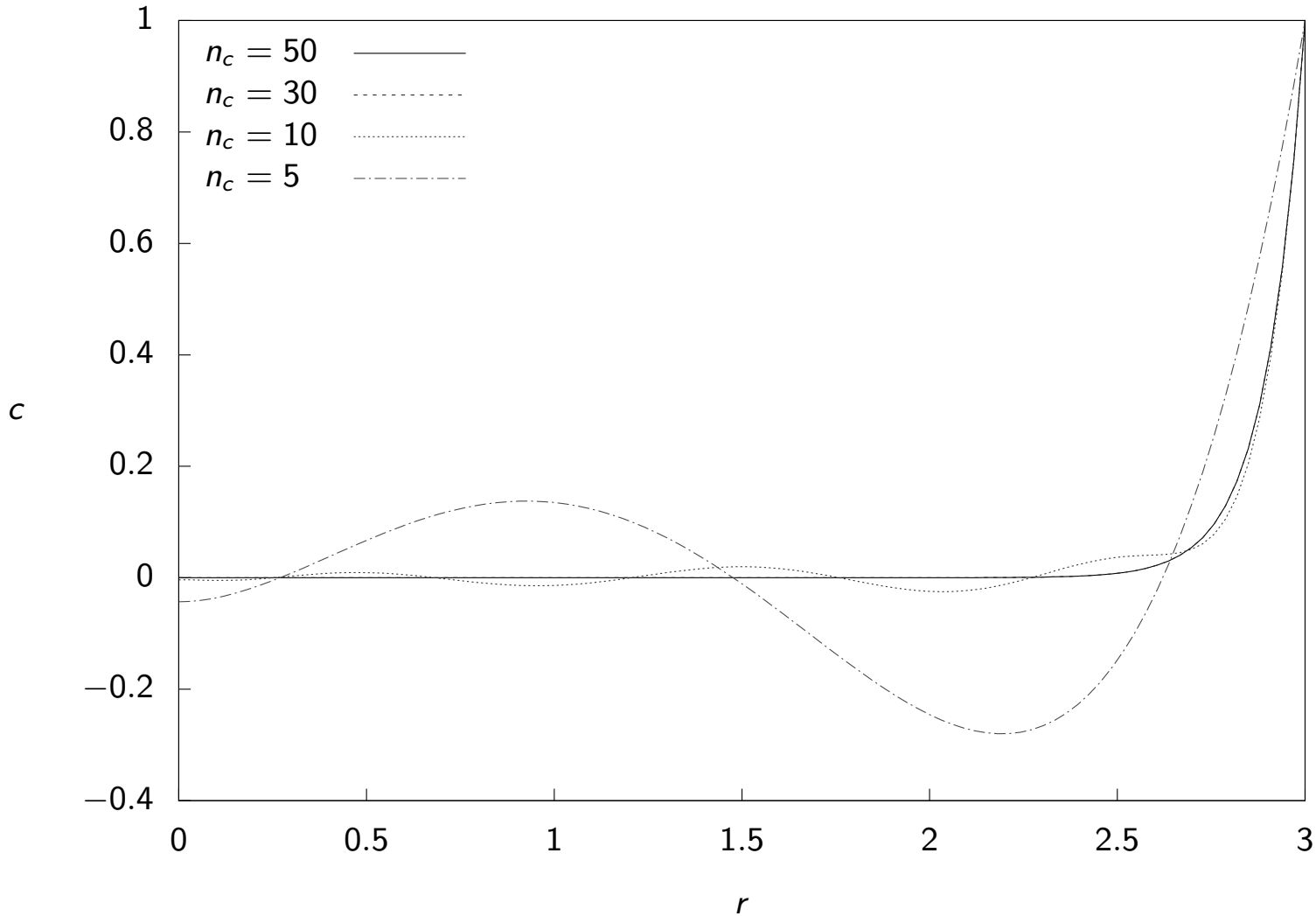
1. Compute the concentration profile for a first-order reaction in a spherical pellet. Solve the problem for $\phi = 10$.
2. Plot the concentration profile for $n_c = 5$, 10, 30 and 50. How many collocation points are required to reach accuracy in the concentration profile for this value of Φ.

Boundary-Value Problems and Collocation (`colloc`) V

3. How many collocation points are required to achieve a relative error in the effectiveness factor of less than 10^{-5}?

□

Solution I



Solution II

Figure A.5: Dimensionless concentration versus dimensionless radial position for different numbers of collocation points.

First we perform the differentiation in Equation A.8 to obtain

$$\frac{d^2c}{dr^2} + \frac{2}{r}\frac{dc}{dr} - \Phi^2 c = 0$$

We define the following Octave function to evaluate this equation at the interior collocation points. At the two collocation endpoints, we satisfy the boundary conditions $dc/dr = 0$ at $r = 0$ and $c = 1$ at $r = 3$. The function is therefore

```
function retval = pellet(c)
  global Phi A Aint Bint Rint n
  npts = length(c);
  cint = c(2:npts-1);
  retval(1)     = A(1,:)*c;
  retval(2:npts-1) = Bint*c .+ 2*Aint*c./Rint .- \
      Phi^2*cint;
  retval(npts) = 1 - c(npts);
endfunction
```

Figure A.5 shows the concentration profiles for different numbers of collocation points.

Solution III

We require about $n_c = 30$ to obtain a converged concentration profile. Figure A.6 shows the relative error in the effectiveness factor versus number of collocation points. If one is only interested in the pellet reaction rate, about 16 collocation points are required to achieve a relative error of less than 10^{-6}.

Solution IV

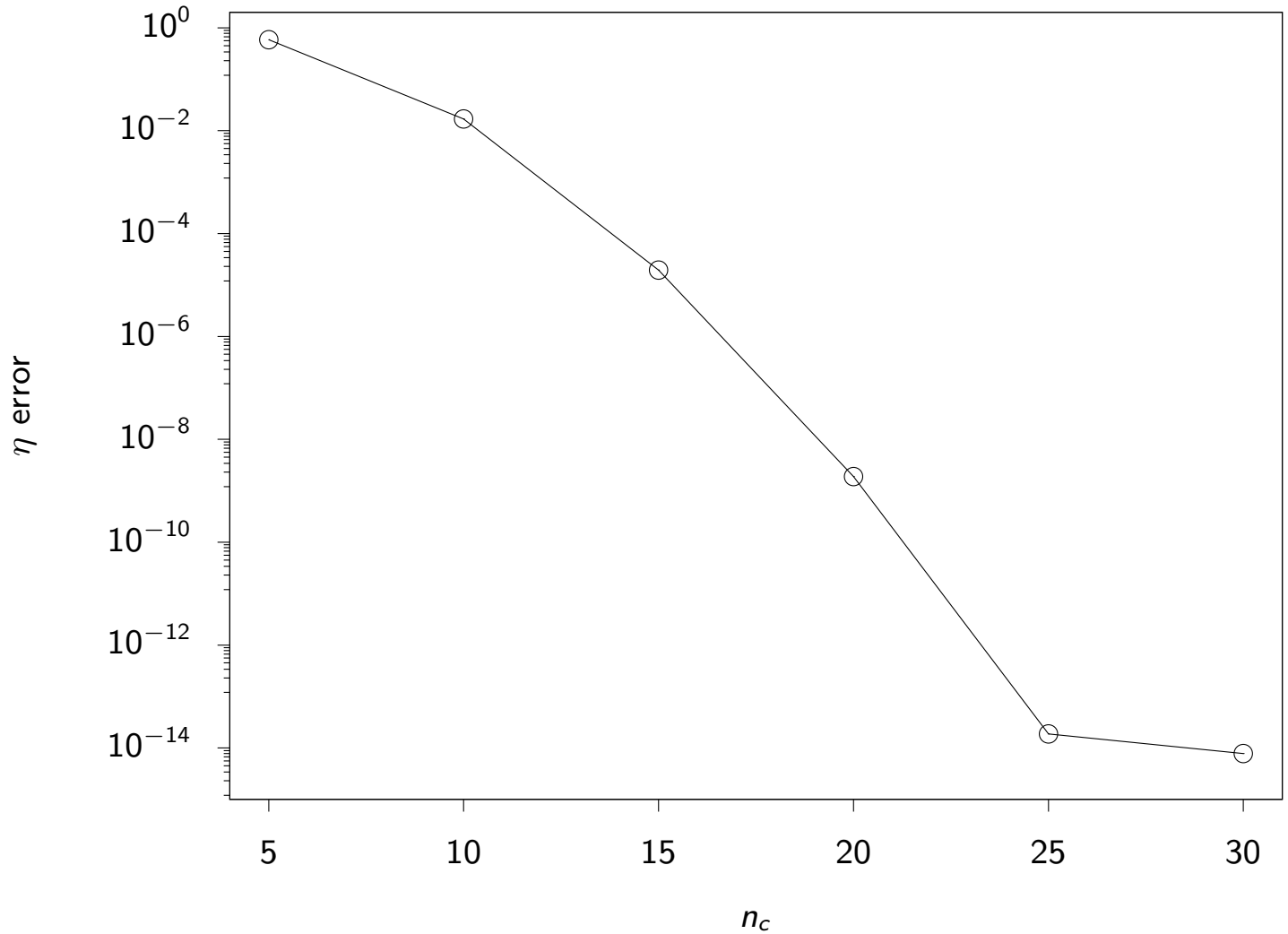


Figure A.6: Relative error in the effectiveness factor versus number of collocation points.

Example A.4: A simple fixed-bed reactor problem

Here we set up the collocation approach of this section for the single, second-order reaction

$$\mathrm{A} \longrightarrow \mathrm{B} \qquad r = kc_A^2$$

Solve the model exactly and compare the numerical solution to the solution obtained with the *approximate* Thiele modulus and effectiveness factor approaches shown in Figure 7.26.

□

Solution I

The model for the fluid and particle phases can be written from Equations 7.67–7.77

$$\frac{dN_A}{dV} = R_A = -(1-\epsilon_B)\frac{S_p}{V_p} D_A \left.\frac{d\widetilde{c}_A}{dr}\right|_{r=R}$$

$$\frac{d^2\widetilde{c}_A}{dr^2} + \frac{2}{r}\frac{d\widetilde{c}_A}{dr} + \frac{\widetilde{R}_A}{D_A} = 0$$

$$\widetilde{c}_A = c_A \qquad r = R$$

$$\frac{d\widetilde{c}_A}{dr} = 0 \qquad r = 0$$

The collocation method produces algebraic equations for the pellet profile as for the previous example. We use a DAE solver to integrate the differential equation for N_A and these coupled algebraic equations.

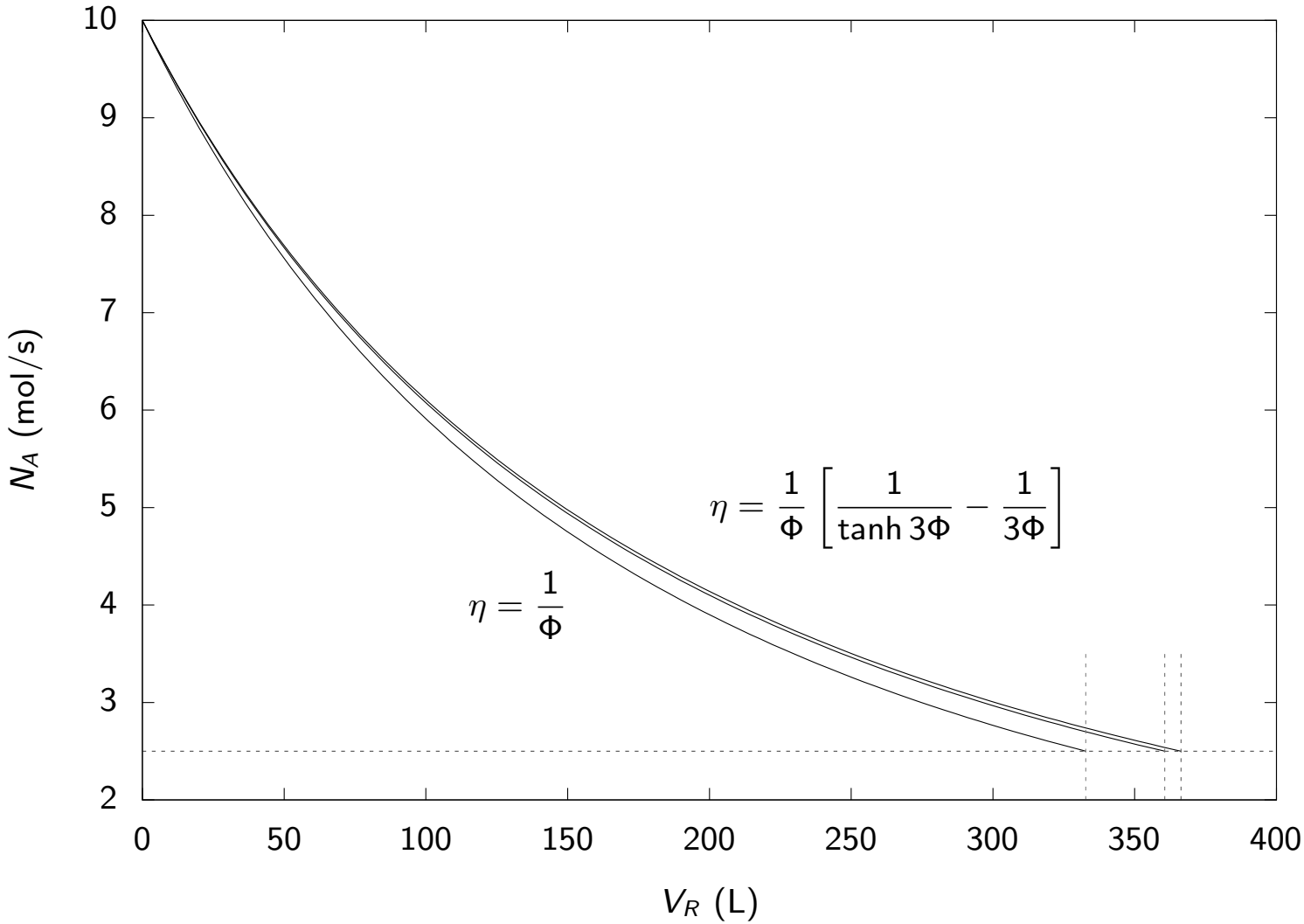


Figure A.7: Molar flow of A versus reactor volume for second-order, isothermal reaction in a fixed-bed reactor; two approximations and exact solution.

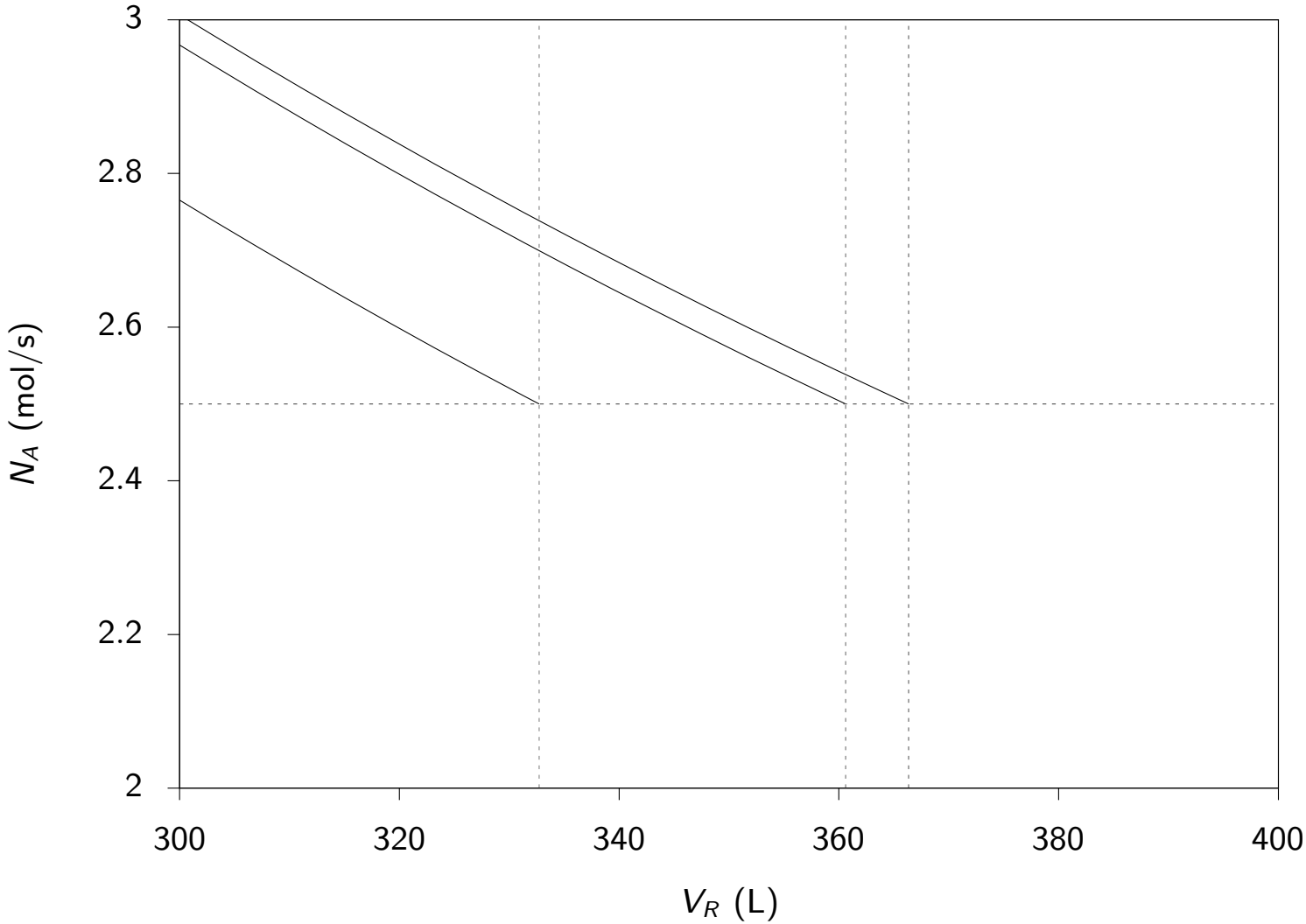


Figure A.8: Magnified view of Figure A.7.

The results for N_A versus reactor length using 25 collocation points for the pellet are shown in Figure A.7. Also shown are the simplified effectiveness factor calculations for this problem from Example 7.5. A magnified view is shown in Figure A.8. Notice the effectiveness factor approach gives a good approximation for the bed performance. It is not exact because the reaction is second order.

Example A.5: Estimating two rate constants in reaction A→ B→ C

Consider the irreversible series reactions

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C}$$

Estimate the two rate constants k_1 and k_2 from the measurements shown in Figure A.9. We would also like to know how much confidence to place in these parameter estimates.

□

Solution I

For this problem we require only the following simple `paresto` options.

```
model.x = {'ca', 'cb', 'cc'};
model.p = {'k1', 'k2'};
model.d = {'m_ca', 'm_cb', 'm_cc'};

model.ode = @massbal_ode;
model.lsq = @(t, y, p) {y.ca-y.m_ca, y.cb-y.m_cb, y.cc-y.m_cc};

tfinal = 6;
nplot = 75;
tplot = linspace(0, tfinal, nplot)';

k1 = 2; k2 = 1; p_ac = [k1; k2];

ca0 = 1; cb0 = 0; cc0 = 0; x_ac = [ca0; cb0; cc0];

thetaic   = [0.5; 3; x_ac];
thetalb   = [1e-4; 1e-4; 0.9*x_ac];
thetaub   = [10; 10; 1.1*x_ac];
est_ind = 1:2;

%% load measurements from a file
tabledat = load ('ABC_data.dat');
tmeas = tabledat(:,1);
ymeas = tabledat(:,2:4);
```

Solution I

```
[tout,~,ic] = unique([tmeas; tplot]);
% index of times at which measurement is made
meas_ind = ic(1:numel(tmeas));
model.tout = tout;
% interploate measurement onto new grid
y_noisy = interp1(tmeas, ymeas, tout, 'previous');
y_noisy(isnan(y_noisy)) = 0.;
% use index of measurement times in objective
model.lsq_ind = meas_ind';

% Create a paresto instance
pe = paresto(model);

%% estimate the parameters
est = pe.optimize(y_noisy', thetaic, thetalb, thetaub);

% Also calculate confidence intervals with 95 % confidence
theta_conf = pe.confidence(est, est_ind, 0.95);

disp('Estimated Parameters and Bounding Box')
[est.theta(est_ind)  theta_conf]

%plot the model fit to the noisy measurements
plot(model.tout, est.x, tmeas, ymeas', 'o');
```

We also require the function `massbal_ode` to define the right-hand side of the differential equations

```
function xdot = massbal_ode(t, x, p)
  r1 = p.k1*x.ca;
  r2 = p.k2*x.cb;
  xdot = {-r1, r1-r2, r2};
endfunction
```

Solution II

Running the `ABC.m` file produces the following parameter estimates and confidence intervals.

$$\widehat{\boldsymbol{\theta}} = \begin{bmatrix} 2.02 \\ 0.998 \end{bmatrix} \pm \begin{bmatrix} 0.0977 \\ 0.0357 \end{bmatrix} \qquad \boldsymbol{\theta}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

The estimates are close to the true values $\boldsymbol{\theta}_0$ used to generate the data. The confidence intervals are reasonably tight given the three species measurements with the noise level indicated in Figure A.9. The fit of the model using the estimated parameters is shown in Figure A.10.

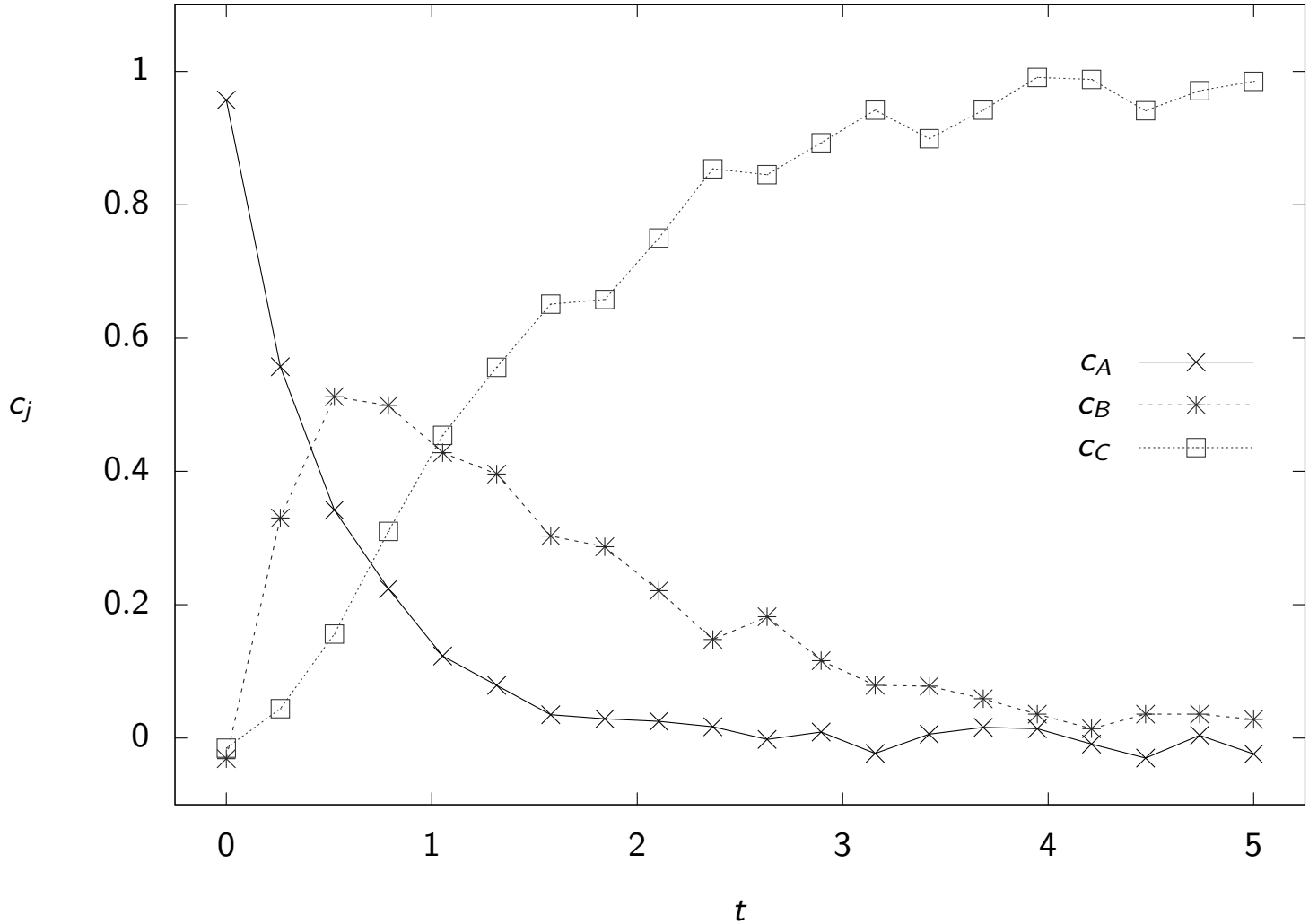


Figure A.9: Measurements of species concentrations in Reactions A.9 versus time.

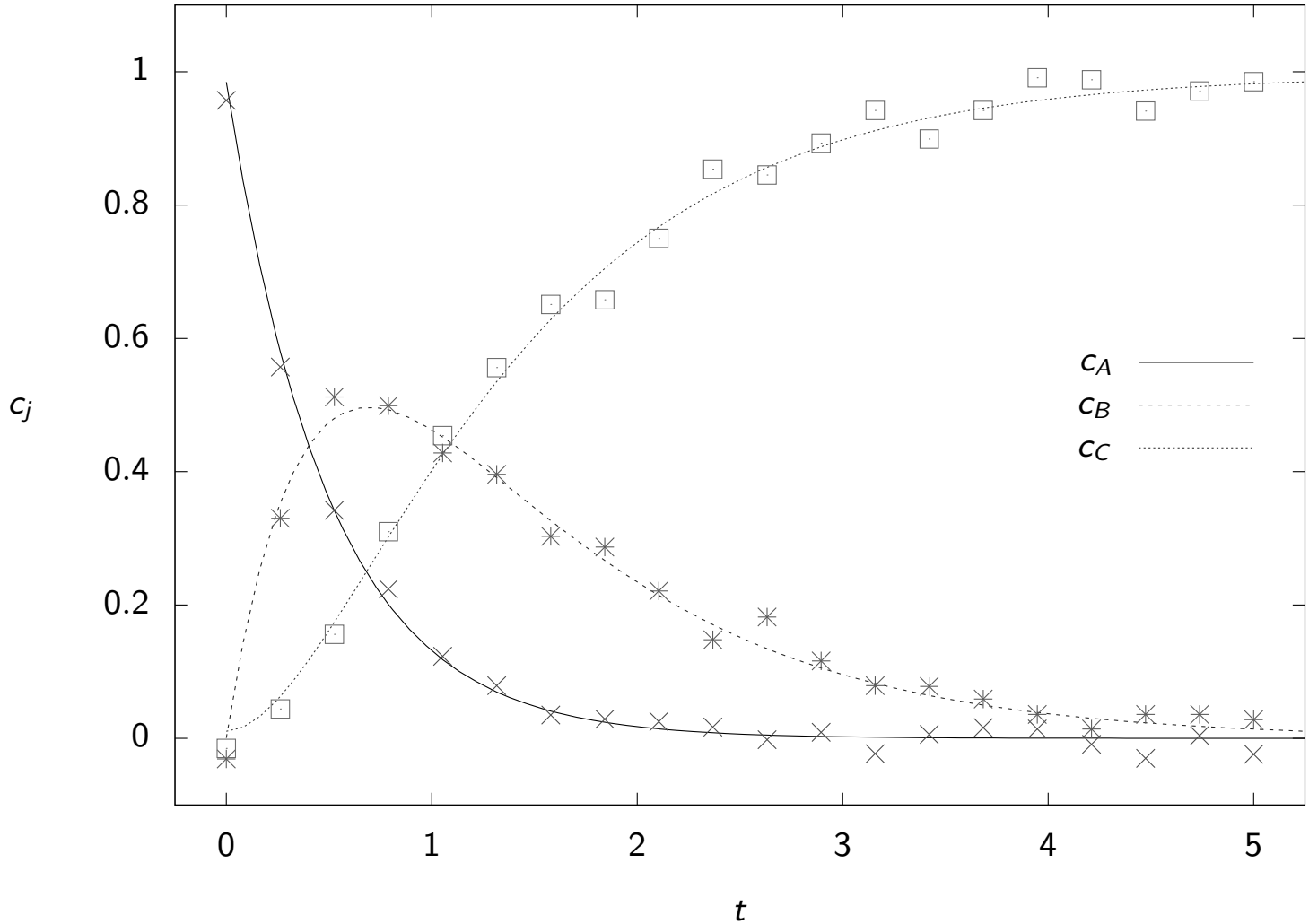


Figure A.10: Fit of model to measurements using estimated parameters.

References I

K. E. Brenan, S. L. Campbell, and L. R. Petzold.

Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.
Elsevier Science Publishers, New York, 1989.

P. Brown, A. Hindmarsh, and L. Petzold.

Consistent initial condition calculation for differential-algebraic systems.
SIAM J. Sci. Comp., 19(5):1495–1512, 1998.

P. N. Brown, A. C. Hindmarsh, and L. R. Petzold.

Using Krylov methods in the solution of large-scale differential-algebraic systems.
SIAM J. Sci. Comp., 15(6):1467–1488, November 1994.

J. W. Eaton.

Octave: Past, present and future.
In K. Hornik and F. Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15-17, 2001, Technische Universität Wien, Vienna, Austria*, 2001.
http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/, ISSN 1609-395X.

J. W. Eaton and J. B. Rawlings.

Octave—a high level interactive language for numerical computations.
CACHE News, (40):11–18, Spring 1995.

A. C. Hindmarsh.

ODEPACK, a systematized collection of ODE solvers.
In R. S. Stepleman, editor, *Scientific Computing*, pages 55–64, Amsterdam, 1983. North-Holland.

J. Villadsen and M. L. Michelsen.

Solution of Differential Equation Models by Polynomial Approximation.
Prentice-Hall, Englewood Cliffs, New Jersey, 1978.