

Chapter 3

Initial value problems

3.1 introduction

3.1.1 ordinary differential equations

Ordinary differential equations (ODEs) are equations in which the dependent variable is a function of a single independent variable. Ordinary differential equations may be of any order. Linear ODEs all follow the basic form

$$a_n \frac{d^n y}{dx^n} + a_{n-1} \frac{d^{n-1} y}{dx^{n-1}} + \dots + a_1 \frac{dy}{dx} + a_0 y = f(x) \quad (3.1)$$

The equation is said to be *linear* if the coefficients a are constant or depend only on the independent variable x and *non-linear* if one or more of the a depend on the dependent variable y . The equation is *homogenous* if $f(x) = 0$. Higher-order ODEs may always be reduced to a coupled set of first-order equations by defining suitable auxiliary functions and re-writing the original equations.

The **decay equation**:

$$\frac{dc}{dt} = -\lambda c \quad (3.2)$$

in which c represents the amount of a quantity, t represents time, and λ represents the decay constant for the quantity c , is a common first order ODE. Changing the sign of the right hand side yields an exponential growth equation. The right-hand side of the equation is a *direction field* representing the family of solutions dc/dt (figure 3.1.1). Without a specified initial value, we don't know which solution is the useful one. For a given initial condition, the direction field forms tangents to the exact solution. When an analytical solution to a differential equation is not available, we must use a computational scheme to work our way through the direction field.

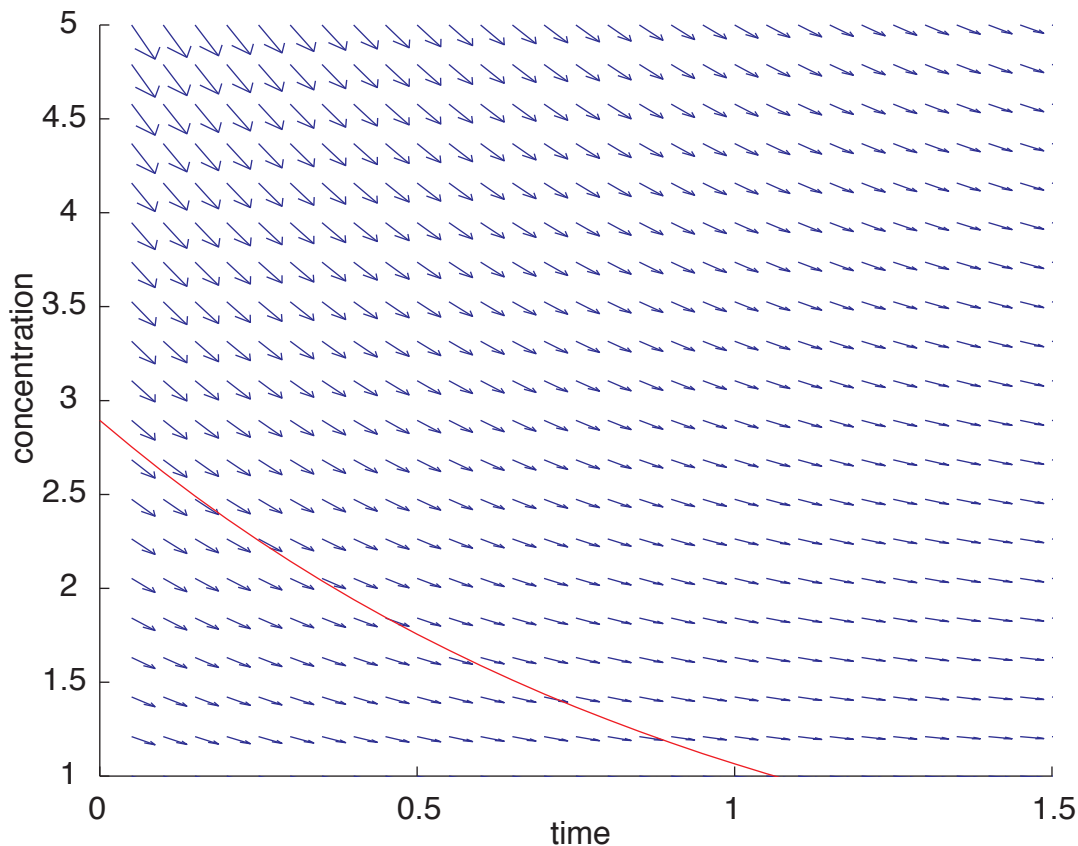


Figure 3.1: direction field of radioactive decay $\lambda = 1$, exact solution for $c_0 = 2.9$.

Using an initial value $c = c_0$ at $t = 0$, the exact solution is

$$c = c_0 e^{(-\lambda t)} \quad (3.3)$$

Population equations describe the dependence of the rate of change in a population on the size of the population. A very simple expression may be formed by taking the rate of change to be the difference between birth rate and death rate. In that case, an equation may be written by including a growth term on the right hand side of the decay equation (3.2) so that

$$\frac{dc}{dt} = \alpha c - \beta c \quad (3.4)$$

The product αc is the growth rate and βc is the death rate in the population c and t represents time. The equation can be integrated using an initial value, $c = c_0$ at $t = t_0$. It may help you do this to recognize $(\alpha - \beta)$ as just another coefficient. The exact expression is:

$$c(t) = c_0 e^{[(\alpha - \beta)t]} \quad (3.5)$$

If $\beta > \alpha$, the exponent is negative and the population declines over time, as in the decay equation (3.2). If we think about the application of this equation to a biological system, we quickly recognize that more is needed. How are the coefficients α and β determined? Do they both depend only on c or on the populations of other organisms as well?

The growth and decay rates may be related to one another through an environment “carrying capacity” for a particular organism. Imagine a population introduced to a new region where food is abundant and there are no predators. As a small starting population grows, its growth rate ought to increase as food is easy to obtain. What happens as the population grows and there is competition for food? Perhaps then the death rate will depend on the population size and the rate of population growth will decline. A **logistic** type function can be used to describe a situation in which the death rate depends on the population size according to some coefficient γ , so that

$$\frac{dc}{dt} = \alpha c - (\beta + \gamma c) c \quad (3.6)$$

or grouping as above,

$$\frac{dc}{dt} = (\alpha - \beta) c - \gamma c^2 \quad (3.7)$$

The carrying capacity of the ecosystem is the steady state solution to equation (3.7), that is, when $dc/dt = 0$. Equation (3.7) has two such solutions, one of which is zero and the other of which is $c = (\alpha - \beta)/\gamma$. The latter seems more interesting than the former. Defining a carrying capacity k to be the non-zero equilibrium solution, the **logistic equation** may be written:

$$\frac{dc}{dt} = (\alpha - \beta) \left(1 - \frac{c}{k}\right) c \quad (3.8)$$

The equation is easy to read. As c approaches k , $\delta c/\delta t$ declines because $(1 - c/k)$ approaches zero. The trajectory by which c approaches that value depends on the coefficients α , β , and γ . Whether or not c it obtains that value at all depends on the initial value $c(t_0)$. The exact analytical solution to equation 3.1.1 is found by separating the variables

$$\frac{dc}{(1 - c/k) c} = (\alpha - \beta) dt$$

and using the method of partial fractions to rearrange the terms on the left hand side

$$\frac{dc}{(k - c)} + \frac{dc}{c} = (\alpha - \beta) dt$$

and then integrating to yield

$$-\ln |k - c| + \ln |c| = (\alpha - \beta)t + C$$

in which the constant C arises from the indefinite integral. Using the properties of natural logs to rearrange,

$$\ln \frac{|c|}{|k - c|} = e^{(\alpha - \beta)t} C$$

A particular solution is found by evaluating the constant C for an initial value $c = c_0$ at $t = 0$

$$c(t) = \frac{c_0 k e^{(\alpha-\beta)t}}{k + c_0 (e^{(\alpha-\beta)t} - 1)}$$

The logistic model has our population c living in isolation, in a petri dish or on a deserted island perhaps. It would be more realistic to introduce some competition with another organism, perhaps a predator. The **Lotka-Volterra** equations offer a simple predator-prey relationship:

$$\frac{dc_1}{dt} = c_1 (\alpha - \beta c_2) \quad (3.9)$$

$$\frac{dc_2}{dt} = -c_2 (\gamma - \delta c_1) \quad (3.10)$$

in which the negative tells you that c_2 is the predator and the parameters α , β , γ , and δ describe interactions between the two organisms. There is no intraspecific competition (the carrying capacity notion) in these equations but it could be added. This is a set of two coupled first-order ordinary differential equations. The set of equations could be expanded to include more species interactions.

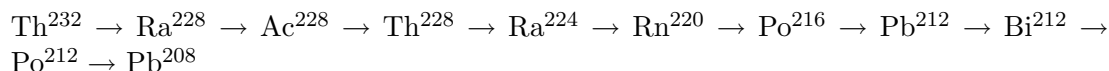
It may not be the case that the birth rate and death rate factors α and β are independent of the population size c . If either coefficient does depend on the dependent variable, then the population equation is nonlinear. Such a situation may arise in populations that depend on a minimum group size for safety (for example, in protecting young from predators). Nonlinearities of this type may be associated with species extinction.

The systems of logistic equations presented thus far are deterministic. Models built with these equations will always yield the same results. A stochastic version of the model may be preferred, in which the probabilities of birth and death are considered.

3.1.2 coupled ODEs

Coupled ordinary differential equations are sets of two or more equations sharing common dependent variables. They may arise from the reduction of a higher-order ODE to a set of first order equations or be mathematical descriptions of different processes that affect a common variable.

A radioactive decay series is a simple example of coupled ODEs. When a parent nuclide decays, its concentration decreases and the concentration of its daughter increases. When the daughter product is also radioactive, the daughter decays as well so an equation describing its concentration over time must include both an accumulation term and a decay term. Many radiogenic elements are themselves radioactive, producing decay chains of various lengths. For example, thorium-232 decays through a series of steps to become the stable lead-208. The complete decay series is:



also



For a decay chain with one radioactive daughter and one stable daughter, we would write:

$$\frac{dc_1}{dt} = -\lambda_1 c_1 \quad (3.11)$$

$$\frac{dc_2}{dt} = \lambda_1 c_1 - \lambda_2 c_2 \quad (3.12)$$

$$\frac{dc_3}{dt} = \lambda_2 c_2 - \lambda_3 c_3 \quad (3.13)$$

where the subscripted numeral indicates a particular isotope. Equations 3.11 and 3.12 are coupled by the variable c_1 , and equations 3.12 and 3.13 are coupled by the variable c_2 . Note the similarity of equations (3.12) and (3.13) to equation (3.4).

3.1.3 initial value problems

Initial value problems are ones in which we know the initial state of a system and wish to compute how the system changes as the value of the independent variable changes. The independent variable might be time, a space dimension, or another quantity. Starting from a known (initial) value of the dependent variable, a numerical method is used to compute successive values of the dependent variable at regular intervals of the independent variable. The job of the numerical method is to approximate the differential equation over each interval.

3.2 numerical solutions to ODEs

3.2.1 general mathematical form of the problem

The general form of the equations we wish to solve is:

$$\frac{dy}{dx} = f(x, y) \quad y = y_0, x = x_0 \quad (3.14)$$

where f is some mathematical function that maps one set of real numbers x to another set of real numbers y and (y_0, x_0) is an initial condition. We can begin by integrating (3.14)

$$\begin{aligned} \int dy &= \int f(x, y) dx & (3.15) \\ y - y_0 &= \int_{x_0}^x f(x, y) dx \\ y &= y_0 + \int_{x_0}^x f(x, y) dx \end{aligned}$$

but now what? We can't evaluate the remaining integral because we don't know yet the values of y after y_0 .

Our next step is to discretize the problem, to define a small interval x_j to x_{j+1} over which the integral will be evaluated:

$$y_{j+1} = y_j + \int_{x_j}^{x_{j+1}} g(x) dx \quad (3.16)$$

in which $g(x)$ represents $f(x, y)$ over the finite interval. A subscript is used (in this case, j) to denote a specific value of the independent variable and the corresponding value of the dependent variable. The goal in developing a numerical method is to make an assumption about the form of the function $g(x)$ that allows the integral to be evaluated.

3.2.2 continuous functions

It is important at this point to be sure that $f(x)$ is a continuous function. In your calculus textbook, continuity was probably defined following Heine (a German mathematician; 1821 to 1881). If the function f is defined in the neighborhood of a , then f is *continuous* at a provided that as $n \rightarrow \infty$, $f(x_n)$ exists and goes to $f(a)$:

$$\lim_{n \rightarrow \infty} f(x_n) = f(a)$$

Another definition of continuity, which provides a handy perspective in our formulation of numerical methods, is due to Cauchy (the French mathematician who formulated and proved many Calculus theorems; 1789 to 1857). The function f is said to be *continuous* at a if for any number $\epsilon > 0$, there exists some number $\delta > 0$ such that for all x in the domain with

$$a - \delta < x < a + \delta$$

the value of x satisfies

$$f(a) - \epsilon < f(x) < f(a) + \epsilon$$

If we have a continuous function, we are assured that $g(x)$ in equation (3.16) exists for any interval of the independent variable x .

In addition to being continuous, it is important that $f(x)$ is a *smooth* function over the interval of interest in our problem. Smoothness guarantees that both the function and its derivatives are continuous (that is, there are no corner points in $f(x)$).

3.2.3 series

The definite integral in equation (3.16) is not known because we don't know the form of the function $g(x)$. At least we have a definite integral, that's something! The goal in developing a numerical model is to approximate the value of the function and evaluate the integral as accurately and as efficiently as possible, given whatever computational constraints we face.

If it happens to be the case that the function is linear, then the product of the first derivative and the interval size, $\frac{df}{dx}(x-a)$, gives us the value of the integral. We can see that the whole solution would be a simple first-order polynomial

$$f(x) = f(a) + \frac{df(a)}{dx}(x-a) \quad (3.17)$$

or using the notation in equation (3.16),

$$y_{j+1} = y_j + \frac{df(x_j)}{dx}(x_{j+1} - x_j) \quad (3.18)$$

The linear approximation (equation 3.17) assumes that higher derivatives are zero. Unfortunately, we don't know *a priori* if the function $g(x)$ is really this simple over the interval from a to x . On some scales in x it almost certainly is this simple, but those may be very small indeed.

We may do a better job of representing the function $g(x)$ by adding the second derivative to the polynomial

$$f(a) + \frac{df(a)}{dx}(x-a) + \frac{d^2f(a)}{dx^2}(x-a)^2$$

This would capture some curvature, if it exists, in $g(x)$. Recall that the first derivative of $f(a)$ is the tangent at a . Each higher-order term we add to our polynomial *series* improves our chance of capturing whatever complicated shape the function might have over the interval but we are still limited to only those terms we choose to add. Clearly, we need a more general expression that allows for any form of $g(x)$, an expression that includes all derivatives of the function.

In mathematics, the term *series* is used to mean a summation of a sequence of terms:

$$\sum_{k=0}^n a_k = a_0 + a_1 + \dots + a_{n-1} + a_n$$

The Greek letter Σ is used to indicate summation and the subscript indicates the starting value of k and the superscript indicates its ending value. The terms are produced according to some rule. If n is finite then the summation is a finite series. Series may also be infinite, in which case we take the limit as n goes to infinity.

As an example, consider the finite *geometric* sequence of terms 2^k where $k = \{0, 1, 2, 3\}$. The finite series is

$$\begin{aligned} \sum_{k=0}^3 2^k &= 2^0 + 2^1 + 2^2 + 2^3 \\ &= 1 + 2 + 4 + 8 \\ &= 15 \end{aligned}$$

The infinite series 2^k is written

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n 2^k$$

Series may either diverge or converge. The geometric series

$$\sum_{k=0}^n a\alpha^k$$

converges when $|\alpha| < 1$. At each successive value of k in the sequence, the product $a\alpha^k$ becomes smaller. When $|\alpha| > 1$, the product $a\alpha^k$ grows larger with increasing k and the series diverges. The sum for finite n may be calculated by applying the geometric series theorem

$$\sum_{k=0}^n a\alpha^k = a \left(\frac{\alpha^{n+1} - 1}{\alpha - 1} \right)$$

You could think of this series as representing a feedback process in Earth's climate system, in which a is a climate parameter and α is a feedback. When $\alpha > 1$, the series diverges, meaning that the feedback amplifies over time and we have the possibility for a rapid change from one state to another.

3.2.4 Taylor series

The numerical methods we develop and use in this class are based on the notion that a function can be represented as a sum of terms involving the derivatives of the function at a point. This approach to computing the values of functions is ancient, going back to Archimedes (in the 3rd Century BCE) and Liu Hui (in the 3rd Century). The Indian mathematician Madhava took it up again in the 14th Century, developing some special cases for trigonometric functions. The general method was published in 1715 (along with other useful things) by Brook Taylor in his book *Methodus in crementorum directa et inversa*.

The Taylor series representation of a function is

$$f(x) = \lim_{n \rightarrow \infty} \left(\sum_{k=0}^n \frac{f^{(k)}(a)}{k!} (x - a)^k \right) \quad (3.19)$$

in which $f^{(k)}$ is the k 'th derivative of the function f .

As a practical matter, we cannot compute an infinite number of terms and so we approximate our function using a finite-length polynomial $P_n(x)$ constructed of the first n terms in the Taylor series. The truncation of the infinite series yields an error $R_n(x)$:

$$f(x) = P_n(x) + R_n(x) \quad (3.20)$$

The *truncation error* in our approximation is of the order $n + 1$. A first step in dealing with the integral of $g(x)$ is determining how many terms of the Taylor series we will use to compute $P_n(x)$.

3.2.5 stability

By their nature, numerical integration schemes introduce error into our calculations. Schemes that do not magnify those errors as the calculation proceeds are called numerically **stable**. Different schemes and different types of equations have different stability characteristics and we will discuss these as they arise in future chapters. A differential equation is classified as *stiff* when very small time steps are required to maintain numerical stability. This happens when derivative terms in the model are large with respect to scales you care about in the problem and when there is a mismatch between relevant scales in a system of coupled equations. *Non-stiff* means that you don't have these problems.

Decay chains offer a good illustration of stiffness. Suppose you are interested in a decay series in which one of the daughter products has a very short half-life compared to its parent and its daughter isotope. The long-lived isotopes allow for relatively long time steps, but if you select a long time step it is possible for the short-lived isotope to decay away entirely within one step. That would not be useful if you need to use the concentration of that isotope to calculate the concentration of its daughter.

3.3 solution methods for ODE initial value problems

3.3.1 the model domain

The “model domain” is the range of the independent variable over which we wish to solve the differential equation. The differential equation we wish to solve must be continuous over the model domain. At least two tasks are required to define the model domain.

- A range of values for the independent variable must be selected (*for example, minimum and maximum depth or start and stop time*).
- That range must be divided into discrete intervals within which the function is estimated (the integral in equation (3.16)).

The interval size, or “step size,” may be specified and then used to discretize the domain or the number of intervals may be specified and used to compute a corresponding step size. The discrete values of the independent variables are often called “nodes” in the model domain. If there are N intervals, then there are $N + 1$ nodes in the domain.

3.3.2 Euler single-step method

The Euler single-step method makes the very simple assumption that $g(x)$ is constant over the interval $\{x_j : x_{j+1}\}$ and equal to $f(x_j, y_j)$. In this case, equation 3.16 becomes:

$$y_{j+1} = y_j + \int_{x_j}^{x_{j+1}} f(x_j, y_j) dx \quad (3.21)$$

$$\begin{aligned}
&= y_j + f(x_j, y_j) \int_{x_j}^{x_{j+1}} dx \\
&= y_{j_j} + h f(x_j, y_j)
\end{aligned}$$

in which h represents the step size,

$$x_{j+1} = x_j + h. \quad (3.22)$$

In the language of numerical modeling, we would call this forward-stepping solution an explicit scheme. It is named for the Swiss mathematician Leonard Euler (1707-1783).

3.3.3 numerical error in the Euler single-step

The error in our numerical scheme has two parts, a local error due to the approximation we make over each interval of the independent variable and a global error that is the integrated effect of the local errors over our model domain. The error is the sum of the infinite number of terms that make up R_n in equation 3.20. We cannot calculate that value but we can make the reasonable assumption that the largest source of error is the first term after the truncation (the first term in R_n).

The Euler single step method uses the first two terms of the Taylor series, $n = 0$ and $n = 1$. The largest source of error is thus the term involving $f^{(2)}$. The error is

$$\frac{(x_{j+1} - x_j)^2}{2!} f^{(2)}(\xi)$$

in which ξ (the Greek letter Xi) is some unknown value of $x_j \leq \xi \leq x_{j+1}$. We can't evaluate this exactly because we don't know the value of ξ but we can say something about the magnitude of the error. The magnitude of this term goes according to the term $(x_{j+1} - x_j)^2$ so the error must be proportional to h^2 .

The global error is the integrated effect of the local error at each node in the model domain. The number of integration steps is inversely proportional to the step size. Thus, for the Euler single step method, the global error is proportional to h^2/h , that is, h .

The Euler method is lovely for its simplicity but its error is of the order of the step size. To meet reasonable accuracy requirements for most problems, the step size must be very small. Happily, alternatives are available.

3.3.4 higher-order single-step methods

One way to improve upon the simple Euler single-step method is to increase the number of estimates of $f(x, y)$ over the interval of the independent variable step. A weighted sum of the m estimates is then used to compute the $j + 1$ value of the dependent variable:

$$y_{j+1} = y_j + \sum_{l=1}^m \gamma_l k_l \quad (3.23)$$

where γ_l are the weights, and k_l are the function evaluations, none of which involve $f(x_{j+1}, y_{j+1})$. The weights must of course sum to 1.

The 4th-order Runge-Kutta method, in which 4 estimates of $f(x, y)$ are made between x_j and x_{j+1} , is a popular application of the weighted sum scheme. The estimates are made successively, each building on the result of the last:

$$\begin{aligned} k_1 &= f \left(x_j, y_j \right) \\ k_2 &= f \left(x_j + \frac{h}{2}, y_j + \frac{h}{2}k_1 \right) \\ k_3 &= f \left(x_j + \frac{h}{2}, y_j + \frac{h}{2}k_2 \right) \\ k_4 &= f \left(x_j + h, y_j + hk_3 \right) \end{aligned} \tag{3.24}$$

The value y_{j+1} is then the sum of y_j and the weighted sum of the four estimates:

$$y_{j+1} = y_j + h \left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \right) \tag{3.25}$$

As you might guess, the Runge-Kutta method was first developed by Carl Runge (1856-1927), a German mathematical physicist who developed numerical methods in order to solve the differential equations that arose in his studies of atomic spectra. He also worked with Leopold Kronecker on problems in algebra. Another German mathematician, Wilhelm Kutta (1867-1944), extended the method.

3.3.5 predictor-corrector methods

Another approach to increasing the accuracy of our approximation is to use an initial *prediction* of the $j + 1$ value of the dependent variable to re-compute $f(x, y)$ over the interval of the independent variable. A combination of the two estimates is used to calculate a *corrected* value of the dependent variable at $j + 1$.

A variety of predictor-corrector methods are possible. Here, the simple midpoint method is used to demonstrate the process. The midpoint predictor-corrector technique uses the mean of $f(x, y)$ at j and $j + 1$. The approximation of y_{j+1} proceeds in a series of steps.

1. Starting from an initial prediction (an Euler single-step)

$$y_{j+1} = y_j + h f(x_j, y_j) \tag{3.26}$$

2. The mean of $f(x, y)$ evaluated at j and $j + 1$

$$\frac{1}{2} \{f(x_j, y_j) + f(x_{j+1}, y_{j+1})\}$$

and is used to correct the estimate of y_{j+1}

$$y_{j+1} = y_j + \frac{h}{2} \{f(x_j, y_j) + f(x_{j+1}, y_{j+1})\} \quad (3.27)$$

The method results in twice as many calculations as the Euler single step but is second order accurate (the Euler single step is first order accurate).

3.4 example: $dy/dx = x + y$ using an Euler single-step solution

3.4.1 an exact solution for comparison

The equation

$$\frac{dy}{dx} = x + y \quad (3.28)$$

can be solved exactly. With the condition $y_0 = 1$ at $x_0 = 0$, the analytical solution is:

$$y = 2e^x - x - 1 \quad (3.29)$$

3.4.2 Euler single-step solution

In the exercises at the end of this chapter, you are asked to use both an Euler single-step solution (a numerical model of equation (3.28)) and the exact solution so that their results may be compared. Before we write a program to implement the models, we should begin with an algorithm to direct ourselves through that process.

1. define model domain: step size range of the independent variable x .
2. define array for dependent variable, with the same size as the independent variable
3. define initial value for dependent variable
4. compute exact solution at the defined x_j
5. for loop to step through numerical solution; N intervals and $N+1$ nodes
6. draw figure to examine result of calculation

In the Matlab MATLAB script below, $x=[x_i:h:x_f]$; is used to establish values of the independent variable at which the dependent value will be calculated. Each x_j is a *node* in the model domain. We must approximate the solution to our differential equation over a number of *intervals* that is one less than the number of nodes in the model domain. Next, the initial condition on the independent variable y is defined, an array is established to hold all values of y calculated as the model progresses: $y=\mathbf{zeros}(1,N+1)$; and the initial value is inserted into the vector y .

In this example, the function $x + y$ is written within the script itself:

```
%* numerical solution
for j=1:N
    y(j+1) = y(j) + h * (x(j) + y(j));
end
```

The program could also be written such that $x + y$ is represented in a MATLAB function. The terms $(x(j) + y(j))$ above would be replaced with a function call to which the values $x(j)$ and $y(j)$ are passed.

```
%* routine to solve dy/dx=x+y using Euler steps

%* step size
h=0.1;

%* independent variable
xi=0;
xf=5;
x=[xi:h:xf];           % this is the model domain
N=length(x)-1;        % number of intervals , nodes=N+1

%* dependent variable
yi=1;                 % initial value
y=zeros(1,N+1);      % vector to store dependent variable
y(1)=yi;              % insert initial value into y

%* numerical solution
for j=1:N
    y(j+1) = y(j) + h * (x(j) + y(j));
end

%* exact solution for xi=0, yi=1

g=2*exp(x) - x - 1;

%* compare results
figure(1)
clf
plot(x, y, 'b-')
hold on
plot(x, g, 'r-')
title('dy/dx = x + y with x0=0, y0=1')
legend('Euler single-step solution ', 'exact solution ')
```

3.5 example: $dc/dt = -\lambda c$ using a Matlab solver

Below are a script and function written specifically for use with one of MATLAB's Runge-Kutta solvers. The script uses **ode45** but the basic setup is the same for any of the solvers.

MATLAB offers several adaptive step-size higher-order methods for solving ODEs. In an adaptive step-size routine, the numerical algorithm optimizes the step size in order to meet a specified level of accuracy with a minimum number of computational steps. For example, **ode45** accomplishes this by comparing the results of 4th and 5th order Runge-Kutta schemes, hence the name, **ode45**.

MATLAB's description of **ode45**:

```
>> help ode45
Solve non-stiff differential equations, medium order
method. [T,Y] = ODE45('F',TSPAN,Y0) with TSPAN = [T0
TFINAL] integrates the system of differential equations y'
= F(t,y) from time T0 to TFINAL with initial conditions Y0.
'F' is a string containing the name of an ODE file.
Function F(T,Y) must return a column vector. Each row in
solution array Y corresponds to a time returned in column
vector T. To obtain solutions at specific times T0, T1,
..., TFINAL (all increasing or all decreasing), use TSPAN =
[T0 T1 ... TFINAL].

[T,Y] = ODE45('F',TSPAN,Y0,OPTIONS) solves as above with
default integration parameters replaced by values in
OPTIONS, an argument created with the ODESET function. See
ODESET for details. Commonly used options are scalar
relative error tolerance 'RelTol' (1e-3 by default) and
vector of absolute error tolerances 'AbsTol' (all
components 1e-6 by default).

[T,Y] = ODE45('F',TSPAN,Y0,OPTIONS,P1,P2,...) passes the
additional parameters P1,P2,... to the ODE file as
F(T,Y,FLAG,P1,P2,...) (see ODEFILE). Use OPTIONS = [] as a
place holder if no options are set.
```

You can use

```
>> help odeset
```

to learn about all the options. A list of all the MATLAB ODE solvers appears at the end of the help for each of them.

3.5.1 script

One of the variables that must be passed to **ode45** is an array of characters containing the name of a user-supplied function that represents the right-hand side of the differential equation:

```
myfun='f_decay_rk';
```

The function definition must also include some information specific to **ode45**. Here is the function declaration (see the next section for the complete function listing):

```
function R=f_decay_rk(t,c,flag,k)
```

and here is the **ode45** function call:

```
[trk crk]=ode45(myfun, [t0 tf], c0,options,lambda);
```

The order of the variables passed to **ode45** is the same as the order of the variables appearing in the user-defined function `f_decay_rk`. The variables passed to **ode45** are the name of the user-defined function, an array containing the range of the independent variable, the initial value of the dependent variable, a list of special options that tell **ode45** how you want the numerical scheme to proceed, and the value of our one constant coefficient. **ode45** returns arrays containing all values of the independent and dependent variables. The solver employs an adaptive step size strategy so the values of the independent variable may not be evenly spaced.

```
%* Matab script to calculate exponential decay using
% adaptive Runge-Kutta method

%
% variables set by user:
% lambda:      decay constant
% c0:         initial concentration
% t0:         initial time
% tf:         end time
% myfun:      name of function containing RHS of ODE
%
% results:
% crk:       vector containing concentration at all steps
% trk:       vector containing time at all time steps
clear

lambda=1;
c0=1;
ti=0;
tf=5;
myfun='f_decay_rk';

%* numerical solution
% set some ode45 options
% epsilon=1e-6;           %0.0001% error, the abs default
% options=odeset('RelTol',epsilon,'AbsTol',epsilon);

options=[];                % or use default values
```

```
[trk crk]=ode45(myfun, [ti tf], c0,options,lambda);

%* analytic solution
g=c0 * exp(-lambda*trk);

%* compare results
figure(1)
clf
axis([t0 tf 0 1]), hold on
plot(trk, g, 'r-')
plot(trk, crk, 'bo')
legend('analytic solution', 'adaptive Runge-Kutta solution')
title('concentration of a radioactive element over time')
xlabel('time'), ylabel('concentration')
```

3.5.2 function

```
function R=f_decay_rk(t,c,flag,k)

%* R=f_decay(t,c,flag,k)
%
% NOTE: for this to work in Matlab, the parameter flag
% must be included before any passed parameters
% input: k: decay constant
% c: concentration at current time
% return: R=decay rate this time step

R=k*c;
```

3.6 coupled ODEs example: decay chain

Radioactive decay chains, as discussed in section 3.1.2, are a simple example of coupled systems of ODEs. As with most problems we address, there is more than one numerical method we could use to solve our system of equations and more than one way to apply that method. Here, a MATLAB function is written to compute the change in concentration for a parent and daughter system with any number of daughters. A script is written to use that function with a short decay series. The Euler single-step method is applied.

3.6.1 function

The function `f_decayseries` uses a **for** loop to step through all the daughter products in the decay series at a given time step. For each daughter, material is added according to the decay of its immediate parent and material is lost according to its own decay. In the case of the stable daughter

product, the decay is always zero. The function is more complicated than the others we have constructed thus far because the input variable is an $N \times M$ array and the return value is also an $N \times M$ array.

```
function dyv=f_decayseries(kv,cv)

%* function f_decayseries(kv,cv)
%
% input variables:
% kv    decay constants for elements in series, vector
% cv    concentrations of elements in series, vector

%
% return value:
% vector containing concentrations of all elements after this
% decay interval

M=length(cv);
dyv=zeros(1,M);

dyv(1)=-kv(1)*cv(1);

if M>1
    for m=2:M
        dyv(m)= -kv(m)*cv(m) + kv(m-1)*cv(m-1);
    end
end
```

3.6.2 script

This script uses `f_decayseries` to model the concentrations of a radioactive parent and two daughters, the second of which is stable. The variable name `lambda` is used to store an array of decay constants, including a value of 0 for the stable daughter. The decay constants are scaled but you could use measured physical values as well (though in this case we would need to be careful about time scales for various isotopes in the chain and may wish to adopt a strategy appropriate for stiff sets of equations). The variable `c0` is now an array used to store initial values for all of the isotope concentrations.

```
%* Euler single-step solution for decay series system of ODEs
% lambda    decay constants, incl. stable daughter
% c         concentration (dependent variable)
% c0        initial values on c
% t         time (independent variable)
% ti        initial time
% tf        ending time for calculation
% h         step size
clear

lambda=[1 1 0];
```

```

c0=[1 0 0];
t0=0;
tf=5;
h=0.05;

t=[t0:h:tf]';
N=length(t)-1;           % number of steps, N+1 = number of nodes

c=zeros(N+1, length(c0));
c(1,:)=c0;               % insert initial values

for j=1:N
    c(j+1,:) = c(j,:) + h * f_decayseries(lambda, c(j,:));
end

%* information for plotting
ymin=min(min(c));
ymax=max(max(c));

figure(1)
clf
axis([min(t) max(t) ymin ymax]), hold on
plot(t, c(:,1), 'b-')
plot(t, c(:,2), 'r-')
plot(t, c(:,3), 'r-')
legend('parent', 'daughter 1, radioactive', 'daughter 2, stable')
ylabel('concentration')
xlabel('time')
title('decay series')

```

3.7 coupled ODEs example: ocean biogeochemical cycles

3.7.1 introduction

The distribution of various chemicals in Earth's oceans is of interest in many fields of study. For example, the supply of nutrients in upwelling deep water is a fundamental component of the ocean food web because it provides necessary resources for primary productivity (photosynthesis) in the shallow ocean. As ocean circulation changes, the rate of nutrient delivery changes and the whole food web may be affected. Photosynthesis in the shallow ocean also influences global climate by consuming CO₂.

A complete description of ocean circulation is both complicated and computationally intensive. Often, significant insights into ocean biogeochemical cycles can be gained using simpler, linear reservoir representations usually called *box models* (for example Lefèvre and Watson, 1999; Toggweiler and others, 2003a; 2003b). In this context, the word *linear* means that a linear proportionality is specified between the dependent variable and coefficients representing physical and biological

	ocean	rivers
C	2,300,000	
N	605,000	28,000
O	162,000	
Na	480,070,000	315,000
Si	84,000	193,000
P	2,000	1,300
Ca	10,600,000	364,000
Fe	0.7	700

Table 3.1: Dissolved content of several chemicals in the modern ocean and rivers in units of $\mu\text{mol}/\text{m}^3$. Based on table 1.1.1 in Sarmiento and Gruber, 2006.

processes. Ocean biogeochemical box models are systems of coupled ODEs that represent transfers of chemicals among major ocean reservoirs due to various processes.

One differential equation is written for the change with time of each chemical of interest for each reservoir in the model according to *inputs* and *outputs*. For the mass of the chemical to be conserved, a requirement of basic physics (see Appendix A), inputs, outputs, and the rate of change in the mass of a given chemical must balance for each reservoir. Individual reservoirs are connected by fluxes that transfer the chemical from one reservoir to another.

The presentation given in this chapter is brief. For a more complete discussion, see the excellent text book by Sarmiento and Gruber, *Ocean Biogeochemical Dynamics* (2006).

3.7.2 controls on ocean chemistry: two hypotheses

One way to state the fundamental question in ocean biogeochemistry is: “*What controls the chemical composition of the ocean?*” Have elements and chemical compounds simply been accumulating via river input since the oceans formed 3.85 billion years ago (the *accumulation hypothesis*)? Does the chemical composition represent a balance between input and removal processes, mediated by biological processes (the *kinetic control hypothesis*)?

Mean global ocean and river concentrations of various elements can be used to evaluate the merits of the two hypotheses (table 3.7.2). If there is a consistent relationship between ocean and river concentrations from element to element, then the *accumulation hypothesis* may be correct. If there is no consistent relationship, then the *accumulation hypothesis* cannot be correct. In this section, we will develop models to explore these hypotheses.

3.7.2.1 a well mixed ocean

A good place to start our investigation of ocean chemical composition is with the assumption that the ocean as a whole is well mixed. This means that chemical concentrations can be treated as uniform throughout. The assumption frees us from working with the full set of hydrodynamic equations. We know the assumption is not true at river inflows, where sea ice is melting, and so

on, so we must be careful to limit ourselves to asking appropriate questions. This means we must confine our interest to processes that transpire on relatively large spatial scales (in fact, the whole ocean). This might seem excessively simple but the model will be easy to construct, quick to run, can lead to some important insights, and is easy to expand.

The well-mixed assumption leads to a single-reservoir (or one-box) model of ocean chemical concentration with a form identical to the population model introduced earlier. The only quantity we need to conserve is mass:

$$\frac{\delta M_{oc}}{\delta t} = input - output \quad (3.30)$$

Equation 3.30 states that a small change in time δt will produce a small change in the mass M_{oc} of an element if the difference between *input* and *output* is not zero. We will use moles as the fundamental dimension (and units) for M . Taking the limit as δ goes to zero yields a differential equation.

We assume that the only *input* is river discharge into the oceans. For our one-reservoir model, this means that the mass input is the product of the total global river discharge rate v_r and the global mean river concentration c_r of a particular element (or chemical compound) in that flow. The two terms have dimensions L^3/T and mol/L^3 , respectively so the product

$$input = v_r c_r$$

has dimension mol/T .

The output is sedimentation to the sea floor. The sedimentation rate must in some way be related to the concentration c_{oc} of a particular chemical in the ocean. We can't describe all the processes involved in sea floor sedimentation so instead we must make an assumption about the nature of the relationship between mass sedimentation rate and c_{oc} . Assuming a linear relationship,

$$output = k V_{oc} c_{oc}$$

in which V_{oc} is the volume of the ocean and k is a rate constant for removal by sedimentation. The physical processes responsible for k are, in the world of this model, a mystery. The dimension of V_{oc} is L^3 so the dimension of k must be T^{-1} .

3.7.2.2 accumulation hypothesis

Design

The *accumulation hypothesis* states that the concentrations of all elements dissolved in ocean water reflect the accumulated river input over time:

$$\frac{dM_{oc}}{dt} = v_r c_r . \quad (3.31)$$

Assuming constant ocean volume, equation 3.31 becomes

$$V_{oc} \frac{dc_{oc}}{dt} = v_r c_r . \quad (3.32)$$

Equation 3.32 is a linear reservoir model we can use to test our hypothesis.

Implementation

How can we use this model to test the *accumulation hypothesis*? One possibility is to write a forward-stepping numerical routine with an initial value $c_{oc} = 0$ at $t = 0$ and compute forward in time to $t = 3.85 \times 10^9$ years. Comparison of $c_{oc}(t = 3.85 \times 10^9)$ with modern observed concentrations would test the hypothesis.

A more computationally efficient option is to find an analytical solution to equation 3.32 that will answer the question of interest to us. Integrating equation (3.32) for the case in which none of the coefficients depend on t (that is, v_r and c_r do not vary over time) yields an exponential relationship,

$$c(t) = c_{oc} \exp\left(\frac{v_r c_r}{V_{oc}} t\right) \quad (3.33)$$

which we know approaches a steady value of the dependent variable at sufficiently large values of the independent variable. If we suppose that 3.85 billion years is sufficient time to reach the very slowly changing tail of the exponential then we can say

$$\frac{dc_{oc}}{dt} \approx \frac{(c_{oc \text{ modern}} - c_{oc \text{ initial}})}{(\tau_a - t_{\text{initial}})} = \frac{c_{oc \text{ modern}}}{\tau_a}$$

in which τ_a is the time required to reach the observed “modern” concentration. The accumulation time is calculated

$$\tau_a = \frac{V_{oc}}{v_r} \frac{c_{oc}}{c_r} . \quad (3.34)$$

Analysis

Using values from table 3.7.2, accumulation times for phosphorous and sodium are 0.44×10^6 and 43×10^6 years, respectively.

The first step in analyzing our model result should be to ask if these time scales are reasonable. There are two reasons to conclude that they are not:

- The τ_a are much smaller than the age of the ocean.
- The range among τ_a is very large. Were we to compute τ_a for additional elements, we would find a range of about 8 orders of magnitude in its value.

The second step in analyzing our result is to evaluate the assumptions we made in constructing the model. Perhaps river flows or concentrations were different in the past. The many orders of

magnitude range in τ_a indicates that this is not the solution to our problem. Thus, we can conclude that the *accumulation hypothesis* is not correct.

It is important to take a moment here to think about what our model can and cannot tell us. If we had found similar time scales for all the elements we tested, that could indicate that the *accumulation hypothesis* is correct but it might also be possible for some other set of assumptions to produce a similar result. It would be important to test other possibilities before concluding that the *accumulation hypothesis* is correct.

3.7.2.3 kinetic control hypothesis in one reservoir

Design

The *kinetic control hypothesis* asserts that the chemical composition of the ocean represents a balance between chemical and biological input and removal processes. For our single reservoir, this hypothesis is stated

$$\frac{dM_{oc}}{dt} = v_r c_r - k V_{oc} c_{oc} \quad (3.35)$$

in which k is a constant of proportionality that represents what may in fact be complicated chemical and biological processes. Assuming the ocean volume to be constant over time,

$$\frac{dc_{oc}}{dt} = \frac{v_r c_r}{V_{oc}} - k c_{oc}. \quad (3.36)$$

Analysis

This model shows us one fundamentally important attribute of our system: the variation we found in τ_a must reflect differences in removal processes among chemicals dissolved in the ocean. Suppose that over reasonable time scales, ocean composition is at steady state. In this case, the one-reservoir kinetic control model is

$$c_{oc} = \frac{1}{k} \frac{v_r c_r}{V_{oc}}. \quad (3.37)$$

Comparison with a rearranged equation 3.34

$$c_{oc} = \tau_a \frac{v_r c_r}{V_{oc}}. \quad (3.38)$$

shows the equivalence between the two coefficients. τ_a varies among elements because k varies among elements. This is an important insight into the diverse behaviors of chemical species in the ocean system. Relatively small τ_a for elements such as phosphorous and iron indicate that their concentrations in the ocean are modulated in important ways by physical and biological processes that are not resolved by our model.

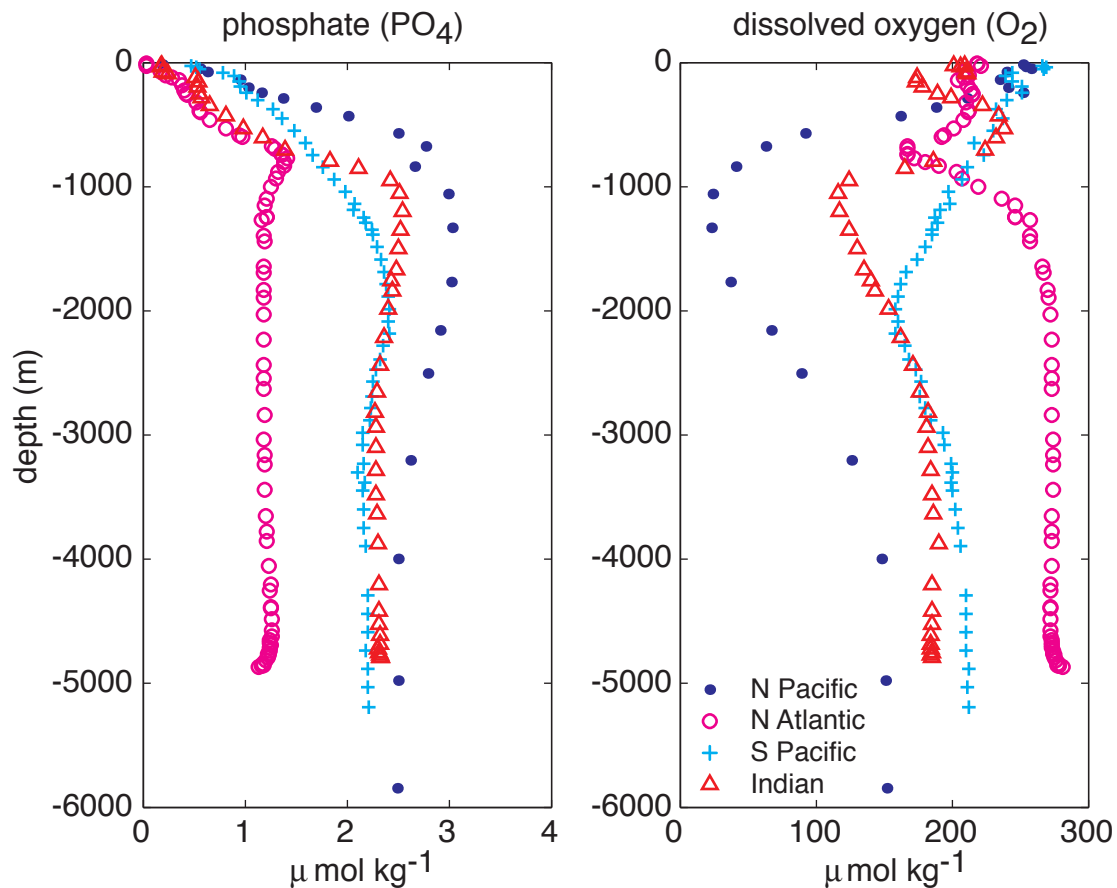


Figure 3.2: Concentrations of phosphate and dissolved oxygen at four locations measured in the 1970's during the Geochemical Ocean Sections Study expeditions. The data are archived at the IRI/LDEO Climate Data Library.

3.7.3 a two reservoir ocean model

In our first ocean model we assumed the whole ocean to be well mixed. A better assumption is suggested by measurements of the depth variation in chemical concentrations within the ocean (figure 3.7.3). Elements such as phosphorous, nitrogen, and iron tend to have low concentrations at shallow depths and higher concentrations in the deep ocean. Oxygen content is high near the surface, declines, and then increases again with depth. These patterns suggest that the concentrations of certain chemicals are determined by different processes in the shallow and deep parts of the ocean. Only chemicals that are associated with biological processes exhibit patterns such as these. They are the signature of life in the ocean.

Design

An easy improvement to our model would be to divide it into two reservoirs:

- a sunlit shallow ocean in which photosynthesis uses CO_2 , nutrients, and energy to produce

organic matter and O_2 and

- a deep ocean in which organic matter is remineralized.

The shallow and deep parts of the ocean also differ in their circulation. Circulation in the shallow ocean is primarily wind-driven while circulation in the deep ocean is driven by density differences (the *thermohaline* circulation). The two-reservoir (or two-box) model requires two components not present in our one-reservoir model, water exchange between the shallow and deep reservoirs, and a term for the transfer associated with sinking organic material.

Water exchange between shallow and deep results from *downwelling* in the high latitude North Atlantic and Southern oceans and *upwelling* elsewhere. We must treat this overturning simply in our model, as a volume exchange rate v_x between the two reservoirs. The dimension of v_x is L^3/T . This overturning carries dissolved materials from the shallow to the deep reservoir and from the deep to the shallow reservoir.

The model must also now include a parameter for the sinking of organic material from the shallow ocean where it is created into the deep ocean where it is remineralized. Much of the organic material produced in the shallow ocean is consumed in the shallow ocean. Export from shallow to deep occurs when photosynthetic production outpaces consumption by higher organisms (zooplankton). Oceanographers who study ocean biochemistry often call this the *organic pump* (or the *carbon pump*). We will define an organic pump term Φ to account for this transfer.

Two reservoirs require a set of two coupled ordinary differential equations, one for the shallow ocean and one for the deep ocean. As with the one-reservoir model, these are mass conservation equations for any chemical compound of interest to our study (as in equation 3.30):

$$V \frac{dc}{dt} = \text{input} - \text{output}$$

where V represents the volume of the reservoir and c represents the concentration of some chemical within it.

Conservation of mass in the shallow ocean is written:

$$V_s \frac{dc_s}{dt} = (v_r c_r + v_x c_d) - (v_x c_s + \Phi) \quad (3.39)$$

where the subscripts s and d indicate the shallow and deep reservoirs, respectively. The term $v_x c_d$ represents transfer of dissolved material from the deep ocean by upwelling while $v_x c_s$ represents transfer to the deep ocean by downwelling. If we suppose that organic productivity is proportional to nutrient supply, Φ may be defined:

$$\Phi = \frac{c_s V_s}{\tau_s} \quad (3.40)$$

where τ_s represents the residence time of a chemical before it is used by organisms in the shallow ocean. This is a fair assumption for the ocean, where organic productivity is nutrient-limited.

For the deep ocean, the conservation equation is:

$$V_d \frac{dc_d}{dt} = (v_x c_s + \Phi) - (v_x c_d + k V_d c_d) \quad (3.41)$$

where the second of the two output terms represents sedimentation to the sea floor. Now we have two equations for two dependent variables and a small collection of coefficients.

The coefficients Φ and k in 3.39 and 3.41 are mysterious. They are constants of proportionality that we have defined in order to write equations. They represent processes associated with biologic productivity and sedimentation but we have not yet considered how to determine their values or how they might be related.

Conclusions we drew from the single reservoir model (section 3.7.2.2) suggest that the combined effects of Φ and k must be reflected in modern concentrations of elements involved in biological processes. One way to make progress would be to use modern observational data for our dependent variables c_s and c_d and an *inverse* technique to solve for the coefficients Φ and k simultaneously. Here we will make some simple assumptions about the processes that govern Φ and k and stick with our forward modeling framework.

The material that accumulates on the sea floor is a combination of remineralized organic matter, un-oxidized organic matter, precipitates from the ocean water, and lithic grains of terrestrial origin. If we assume that in the ocean as a whole, remineralized organic matter dominates (a reasonable assumption), we can define a relationship between Φ and k :

$$k V_d c_d = \alpha \Phi \quad (3.42)$$

where α is a sedimentation efficiency (the proportion of remineralized organic material lost to sedimentation). Again, we have chosen a linear relationship. Larger α produces a larger accumulation rate. Rewriting equation (3.41),

$$V_d \frac{dc_d}{dt} = v_x c_s - v_x c_d + (1 - \alpha) \Phi \quad (3.43)$$

is the deep reservoir ODE.

Implementation

The next step is to choose and implement a numerical technique with which the model's governing equations (3.39) and (3.43) may be solved. This requires definition of initial values on c_s and c_d and of the coefficients v_x , v_r , τ_s , Φ , and α .

MATLAB's built-in ODE solvers are a handy tools with which to solve our system of equations. This begins with a function (or functions) for the right-hand-sides of our governing equations. Both equations in our model contain both of the dependent variables so we write one function that includes both right-hand-sides.

```

function rt=f_dcdt(t, c, flag, fr, vx, V, alpha, taus)

% function rt=f_dcdt(t, c, flag, fr, vx, V, alpha, taus)
%
% input
% t:          time
% c:          concentrations of shallow & deep [1x2]
% flag:       Matlab ODE solver options flag
% fr:         river source, product of flow rate and concentration
% vx:         volume exchange shallow/deep
% V:          volumes of shallow & deep reservoirs [1x2]
% alpha:      sedimentation efficiency
% taus:       shallow ocean residence time for c
%
% return value
% rt:         concentrations of shallow & deep [1x2]

rt=zeros(size(c));
phi=c(1)*V(1)/taus;

rt(1)=(fr + vx*c(2) - vx*c(1) - phi)/V(1);
rt(2)=(vx*c(1) - vx*c(2) + (1-alpha)*phi)/V(2);

```

We also need a script that uses the function to “drive” the model. All initial values and coefficients must be defined and an ODE solver must be selected. Selection of an appropriate solver depends on the time scales of processes in the two reservoirs. Is this a *stiff* set of equations?

The function `f_dcdt` is written to receive two $[1 \times 2]$ arrays, `c` and `V`, and to return one $[1 \times 2]$ array, `rt`. The corresponding variables in the script that drives the model must have the same dimensions. Looking at the function listing, we can see that the first place in `c` and in `V` represents the shallow reservoir and the second place in `c` and in `V` represents the deep reservoir. If this isn’t clear, compare the calculations for `rt` with equations 3.39 and 3.43.

3.7.4 concluding thoughts

Our model could be improved substantially by the inclusion of just one more reservoir. In addition to distinguishing shallow and deep ocean biochemical processes, the model should account for differences between high and low latitude biochemical processes and water transfers. The ocean’s deep water is produced at high latitudes, in the north Atlantic and in the southern ocean’s Ross and Weddell Seas, locations with distinct biochemical signatures.

Linear reservoir models are powerful tools for gaining insight into complicated physical systems. In the field of oceanography, differences between box models and more computationally intensive coupled climate system models have revealed where understanding of the relevant processes is limited or perhaps flawed. The comparison pushes scientists to dig deeply into some very fundamental questions about ocean biogeochemical processes, which in turn have important implications for our understanding of both past and future climate. Papers by Archer and others (2000) and Toggweiler and others (2003a; 2003b) offer a superb overview of this process.

3.8 exercises

1. Solve equation (3.28) over the span $x = \{0 : 2\}$ with an initial value $y = 1$ at $x = 0$ using an Euler single step and a fourth order Runge-Kutta scheme. Write the right-hand side of the equation as a function and then use that function within a script written to answer the following questions. *Please write your own scripts, do not use Matlab solvers.*
 - (a) As always, begin by writing an implementation algorithm.
 - (b) Write a script that solve the equation using the single-step and 4th order Runge-Kutta schemes with step sizes 0.5 and 0.05. Make a figure that can be used to compare the various numerical results with the exact solution to for the specified initial value.
 - (c) Which numerical scheme is more efficient? *In answering this question, think about the total number of calculations made across the model domain.*
 - (d) (extra credit; no chance for rewrite) Show how the particular solution equation 3.29 is obtained.
2. As described in section 3.1.2, radioactive decay series can be modeled using coupled ODEs and the numerical methods we've developed in this chapter, given initial concentrations for the elements in the chain and all the relevant decay constants.
 - (a) Write a system of coupled differential equations for the concentration of a radioactive parent nuclide, three radioactive daughter products and stable final product of the decay chain.
 - (b) Draw a schematic showing how information is passed through the five-component decay series over time. At the initial time, only the parent concentration is non-zero. At the end of the first time step, the parent concentration has changed and the first daughter has a non-zero concentration, and so on. Draw your schematic so that it looks like a matrix, in which each row corresponds to a time and each column corresponds to a different radiogenic element. Please limit the schematic to six rows (five time steps). Draw arrows between the boxes to show how information is passed through the decay series over time.
 - (c) Modify the script in section 3.6 to work for your system of five isotopes. Use decay constants $\{1, 0.2, 1, 0.4, 0\}$ and initial concentrations $\{1, 0, 0, 0, 0\}$ for the parent and four daughters, and a step size $h = 0.05$. Run the model and include a plot of its result in your lab write-up.
 - (d) At what value of t does the third daughter product reach its maximum concentration? MATLAB's **find** function may help you with this answer.
3. In practice, we often use numerical solvers provided to us by software companies instead of building our own (though it is a good idea to understand what goes into them). As introduced in section 3.3.4, MATLAB provides a set of solvers for both non-stiff and stiff ODEs.
 - (a) Describe a physical process that would result in stiff equations.

<i>variable</i>	<i>standard run</i>	<i>units</i>
v_r	45.5×10^{12}	$\text{m}^3 \text{a}^{-1}$
v_x	1.2×10^{15}	$\text{m}^3 \text{a}^{-1}$
V_s	0.03×10^{18}	m^3
V_d	1.26×10^{18}	m^3
c_r P	1300	$\mu \text{mol m}^{-3}$
τ_s P	10	a
α	0.01	

Table 3.2: Variables for exercise 4

- (b) Use MATLAB's **ode45** function to solve the single-isotope radioactive decay problem using the default options, $\lambda = 1$ and an initial value $c = 1$ at $t = 0$. How many steps does the function require for 0.01% absolute accuracy?
- (c) How many Euler steps are required to attain the same quality result? You can answer this by calculating a difference between the two results but for our purposes it would be fine to simply approximate the answer by visual inspection of $c(t)$ produced by both numerical methods.
4. Implement the two-reservoir ocean biogeochemistry model in section 3.7.3 in order to study the effect of the overturning rate on the ocean's phosphorous concentration. A function representing the right hand sides of the shallow and deep reservoir ODEs is listed in section 3.7.3. It is written to be used with one of the MATLAB ODE solvers.

The sequence of steps you will need to follow in writing your script are:

- set up the model coefficients and variables
- run the model for the standard set of coefficients to find a *steady state* solution
- starting from the steady state, change v_x to conduct your perturbation experiment
- plot the results of your model experiments

Use the model to conduct two perturbation experiments, one in which v_x is halved and one in which v_x is doubled.

- (a) Write the script needed to perform the experiment described above. The best way to start is to write the series of steps you need to complete, either on a piece of paper or as comments in your script file. Test the script in parts as you write it. Please include the script in your lab write-up.
- (b) Describe and explain the transient effects on c_s and c_d of doubling and halving the overturning rate
- (c) Compare and explain the post-perturbation steady state values of c_s and c_d in the two experiments.

3.9 References

D. Archer, G. Eshel, A. Winguth, and W. Broecker, 2000, Atmospheric CO₂ sensitivity to the biological pump in the ocean, *Global Biogeochemical Cycles*, 14, 1219-1230.

Lefèvre, N. and A.J. Watson, 1999, Modeling the geochemical cycle of iron in the oceans and its impact of atmospheric CO₂ concentrations, *Global Biogeochemical Cycles*, 13(3), 727-736.

Sarmiento, J.L. and N. Gruber, 2006, *Ocean Biogeochemical Dynamics*, Princeton University Press.

Toggweiler, J.R., R Murnane, S Carson, A Gnanadesikan, and J.L. Sarmiento, 2003a, Representation of the carbon cycle in box models and GCMs 1: solubility pump, *Global Biogeochemical Cycles*, 17(1), doi: 10.1029/2001GB001401.

Toggweiler, J.R., R Murnane, S Carson, A Gnanadesikan, and J.L. Sarmiento, 2003b, Representation of the carbon cycle in box models and GCMs 2: carbon pump, *Global Biogeochemical Cycles*, 17(1), doi: 10.1029/2001GB001841.

