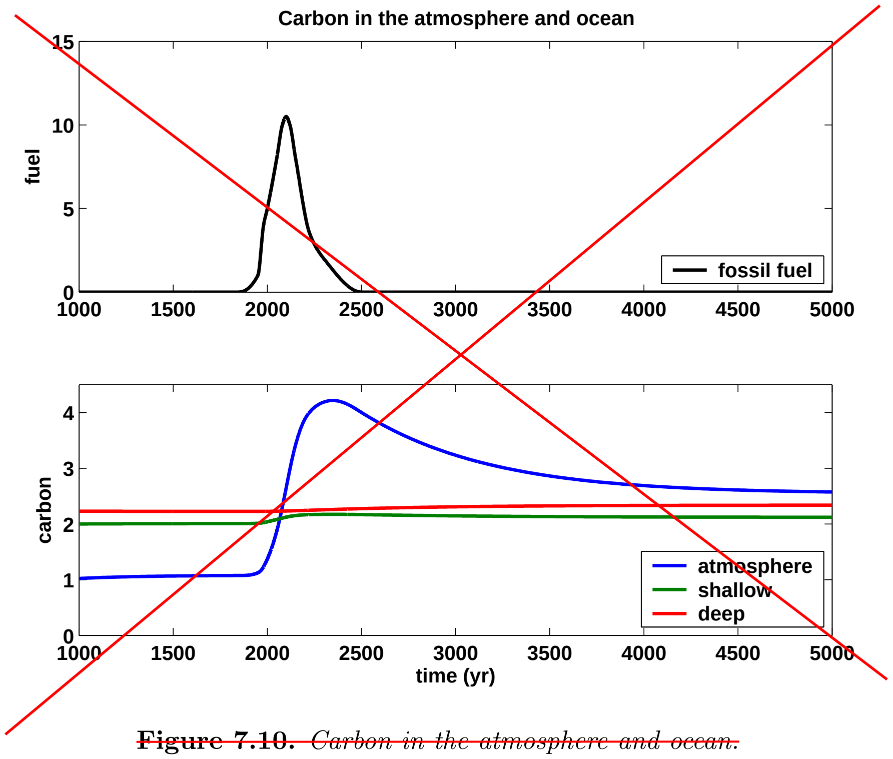
~~(a) Reproduce Figure 7.10. Use `pchiptx` to interpolate the fuel table and `ode23tx` with the default tolerances to solve the differential equations.~~
~~(b) How do the amounts of carbon in the three regimes at year 5000 compare with the amounts at year 1000?~~
~~(c) When does the atmospheric carbon dioxide reach its maximum?~~
~~(d) These equations are mildly stiff, because the various chemical reactions take place on very different time scales. If you zoom in on some portions of the graphs, you should see a characteristic sawtooth behavior caused by the small time steps required by `ode23tx`. Find such a region.~~
~~(e) Experiment with other MATLAB ordinary differential equation solvers, including `ode23`, `ode45`, `ode113`, `ode23s`, and `ode15s`. Try various tolerances and report computational costs by using something like~~

```
~~odeset('RelTol',1.e-6,'AbsTol',1.e-6,'stats','on');~~
```

Which method is preferable for this problem?

7.22. This problem makes use of quadrature, ordinary differential equations, and zero finding to study a nonlinear boundary value problem. The function $y(x)$ is defined on the interval $0 \le x \le 1$ by

$$y'' = y^2 - 1,$$



~~**Figure 7.10.** *Carbon in the atmosphere and ocean.*~~

$$\begin{aligned} y(0) &= 0, \\ y(1) &= 1. \end{aligned}$$

This problem can be solved four different ways. Plot the four solutions obtained on a single figure, using `subplot(2,2,1),..., subplot(2,2,4)`.
(a) Shooting method. Suppose we know the value of $\eta = y'(0)$. Then we could use an ordinary differential equation solver like `ode23tx` or `ode45` to solve the initial value problem

$$\begin{aligned} y'' &= y^2 - 1, \\ y(0) &= 0, \\ y'(0) &= \eta. \end{aligned}$$

on the interval $0 \le x \le 1$. Each value of η determines a different solution $y(x;\eta)$ and corresponding value for $y(1;\eta)$. The desired boundary condition $y(1) = 1$ leads to the definition of a function of η:

$$f(\eta) = y(1;\eta) - 1.$$

Write a MATLAB function whose argument is η. This function should solve the ordinary differential equation initial problem and return $f(\eta)$. Then use `fzero` or `fzerotx` to find a value η_* so that $f(\eta_*) = 0$. Finally, use this η_* in the initial value problem to get the desired $y(x)$. Report the value of η_* you obtain.
(b) Quadrature. Observe that $y'' = y^2 - 1$ can be written

$$\frac{d}{dx}\left(\frac{(y')^2}{2} - \frac{y^3}{3} + y\right) = 0.$$

This means that the expression

$$\kappa = \frac{(y')^2}{2} - \frac{y^3}{3} + y$$

is actually constant. Because $y(0) = 0$, we have $y'(0) = \sqrt{2\kappa}$. So, if we could find the constant κ, the boundary value problem would be converted into an initial value problem. Integrating the equation

$$\frac{dx}{dy} = \frac{1}{\sqrt{2(\kappa + y^3/3 - y)}}$$

gives

$$x = \int_0^y h(y,\kappa)\, dy,$$

where

$$h(y,\kappa) = \frac{1}{\sqrt{2(\kappa + y^3/3 - y)}}.$$

This, together with the boundary condition $y(1) = 1$, leads to the definition of a function $g(\kappa)$:

$$g(\kappa) = \int_0^1 h(y, \kappa)\ dy \ - \ 1.$$

You need two MATLAB functions, one that computes $h(y, \kappa)$ and one that computes $g(\kappa)$. They can be two separate M-files, but a better idea is to make $h(y, \kappa)$ an inline function within $g(\kappa)$. The function $g(\kappa)$ should use `quadtx` to evaluate the integral of $h(y, \kappa)$. The parameter κ is passed as an extra argument from g, through `quadtx`, to h. Then `fzerotx` can be used to find a value κ_* so that $g(\kappa_*) = 0$. Finally, this κ_* provides the second initial value necessary for an ordinary differential equation solver to compute $y(x)$. Report the value of κ_* you obtain.
(c and d) Nonlinear finite differences. Partition the interval into $n + 1$ equal subintervals with spacing $h = 1/(n + 1)$:

$$x_i = ih, \ i = 0, \ldots, n + 1.$$

Replace the differential equation with a nonlinear system of difference equations involving n unknowns, $y_1, y_2, \ldots, y_n$:

$$y_{i+1} - 2y_i + y_{i-1} = h^2(y_i^2 - 1), \ i = 1, \ldots, n.$$

The boundary conditions are $y_0 = 0$ and $y_{n+1} = 1$.
A convenient way to compute the vector of second differences involves the n-by-n tridiagonal matrix A with -2's on the diagonal, 1's on the super- and subdiagonals, and 0's elsewhere. You can generate a sparse form of this matrix with

```
e = ones(n,1);
A = spdiags([e -2*e e],[-1 0 1],n,n);
```

The boundary conditions $y_0 = 0$ and $y_{n+1} = 1$ can be represented by the n-vector b, with $b_i = 0, i = 1, \ldots, n - 1$, and $b_n = 1$. The vector formulation of the nonlinear difference equation is

$$Ay + b = h^2(y^2 - 1),$$

where y^2 is the vector containing the squares of the elements of y, that is, the MATLAB element-by-element power `y.^2`. There are at least two ways to solve this system.
(c) Linear iteration. This is based on writing the difference equation in the form

$$Ay = h^2(y^2 - 1) - b.$$

Start with an initial guess for the solution vector y. The iteration consists of plugging the current y into the right-hand side of this equation and then solving the resulting linear system for a new y. This makes repeated use of the sparse backslash operator with the iterated assignment statement

```
y = A\(h^2*(y.^2 - 1) - b)
```

It turns out that this iteration converges linearly and provides a robust method for solving the nonlinear difference equations. Report the value of n you use and the number of iterations required.
(d) Newton's method. This is based on writing the difference equation in the form

$$F(y) = Ay + b - h^2(y^2 - 1) = 0.$$

Newton's method for solving $F(y) = 0$ requires a many-variable analogue of the derivative $F'(y)$. The analogue is the Jacobian, the matrix of partial derivatives

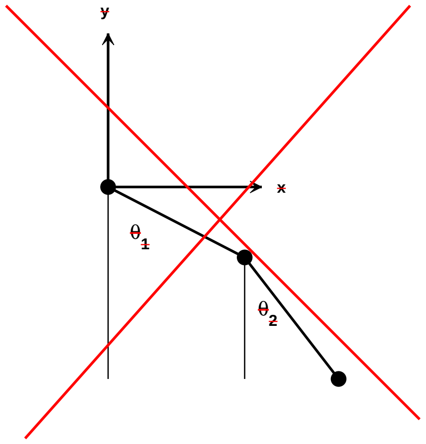
$$J = \frac{\partial F_i}{\partial y_j} = A - h^2 \mathrm{diag}(2y).$$

In MATLAB, one step of Newton's method would be

```
F = A*y + b - h^2*(y.^2 - 1);
J = A - h^2*spdiags(2*y,0,n,n);
y = y - J\F;
```

With a good starting guess, Newton's method converges in a handful of iterations. Report the value of n you use and the number of iterations required.

~~7.23. The double pendulum is a classical physics model system that exhibits chaotic motion if the initial angles are large enough. The model, shown in Figure 7.11, involves two weights, or *bobs*, attached by weightless, rigid rods to each other and to a fixed pivot. There is no friction, so once initiated, the motion continues forever. The motion is fully described by the two angles θ_1 and θ_2 that the rods make with the negative y-axis.~~



~~**Figure 7.11.** *Double pendulum.*~~

~~Let m_1 and m_2 be the masses of the bobs and ℓ_1 and ℓ_2 be the lengths of the~~