```
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

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

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.

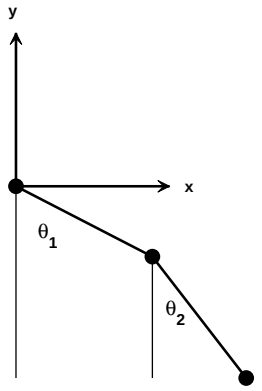

Figure 7.11. *Double pendulum.*

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

rods. The positions of the bobs are

$$x_1 = \ell_1 \sin\theta_1, \quad y_1 = -\ell_1 \cos\theta_1,$$
$$x_2 = \ell_1 \sin\theta_1 + \ell_2 \sin\theta_2, \quad y_2 = -\ell_1 \cos\theta_1 - \ell_2 \cos\theta_2.$$

The only external force is gravity, denoted by g. Analysis based on the Lagrangian formulation of classical mechanics leads to a pair of coupled, second-order, nonlinear ordinary differential equations for the two angles $\theta_1(t)$ and $\theta_2(t)$:

$$\begin{aligned}(m_1+m_2)\ell_1\ddot{\theta}_1 + m_2\ell_2\ddot{\theta}_2\cos(\theta_1-\theta_2) &= -g(m_1+m_2)\sin\theta_1 \\ &\quad -m_2\ell_2\dot{\theta}_2^2\sin(\theta_1-\theta_2), \\ m_2\ell_1\ddot{\theta}_1\cos(\theta_1-\theta_2) + m_2\ell_2\ddot{\theta}_2 &= -gm_2\sin\theta_2 + m_2\ell_1\dot{\theta}_1^2\sin(\theta_1-\theta_2).\end{aligned}$$

To rewrite these equations as a first-order system, introduce the 4-by-1 column vector $u(t)$:

$$u = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T.$$

With $m_1 = m_2 = \ell_1 = \ell_2 = 1$, $c = \cos(u_1 - u_2)$, and $s = \sin(u_1 - u_2)$, the equations become

$$\begin{aligned}\dot{u}_1 &= u_3, \\ \dot{u}_2 &= u_4, \\ 2\dot{u}_3 + c\dot{u}_4 &= -g\sin u_1 - su_4^2, \\ c\dot{u}_3 + \dot{u}_4 &= -g\sin u_2 + su_3^2.\end{aligned}$$

Let $M = M(u)$ denote the 4-by-4 *mass matrix*

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & c \\ 0 & 0 & c & 1 \end{pmatrix}$$

and let $f = f(u)$ denote the 4-by-1 nonlinear *force function*

$$f = \begin{pmatrix} u_3 \\ u_4 \\ -g\sin u_1 - su_4^2 \\ -g\sin u_2 + su_3^2 \end{pmatrix}.$$

In matrix-vector notation, the equations are simply

$$M\dot{u} = f.$$

This is an *implicit* system of differential equations involving a nonconstant, nonlinear mass matrix. The double pendulum problem is usually formulated without the mass matrix, but larger problems, with more degrees of freedom,

are frequently in implicit form. In some situations, the mass matrix is singular and it is not possible to write the equations in explicit form.
The NCM M-file `swinger` provides an interactive graphical implementation of these equations. The initial position is determined by specifying the starting coordinates of the second bob, (x_2, y_2), either as arguments to `swinger` or by using the mouse. In most situations, this does not uniquely determine the starting position of the first bob, but there are only two possibilities and one of them is chosen arbitrarily. The initial velocities, $\dot{\theta}_1$ and $\dot{\theta}_2$, are zero.
The numerical solution is carried out by `ode23` because our textbook code, `ode23tx`, cannot handle implicit equations. The call to `ode23` involves using `odeset` to specify the functions that generate the mass matrix and do the plotting

```
opts = odeset('mass',@swingmass, ...
       'outputfcn',@swingplot);
ode23(@swingrhs,tspan,u0,opts);
```

The mass matrix function is

```
function M = swingmass(t,u)
c = cos(u(1)-u(2));
M = [1 0 0 0; 0 1 0 0; 0 0 2 c; 0 0 c 1];
```

The driving force function is

```
function f = swingrhs(t,u)
g = 1;
s = sin(u(1)-u(2));
f = [u(3); u(4); -2*g*sin(u(1))-s*u(4)^2;
     -g*sin(u(2))+s*u(3)^2];
```

It would be possible to have just one ordinary differential equation function that returns `M\f`, but we want to emphasize the implicit facility.
An internal function `swinginit` converts a specified starting point (x, y) to a pair of angles (θ_1, θ_2). If (x, y) is outside the circle

$$\sqrt{x^2 + y^2} > \ell_1 + \ell_2,$$

then the pendulum cannot reach the specified point. In this case, we straighten out the pendulum with $\theta_1 = \theta_2$ and point it in the given direction. If (x, y) is inside the circle of radius two, we return one of the two possible configurations that reach to that point.
Here are some questions to guide your investigation of `swinger`.
(a) When the initial point is outside the circle of radius two, the two rods start out as one. If the initial angle is not too large, the double pendulum continues to act pretty much like a single pendulum. But if the initial angles are large enough, chaotic motion ensues. Roughly what initial angles lead to chaotic motion?
(b) The default initial condition is

```
swinger(0.862,-0.994)
```

Why is this orbit interesting? Can you find any similar orbits?
(c) Run `swinger` for a while, then click on its `stop` button. Go to the MATLAB command line and type `get(gcf,'userdata')`. What is returned?
(d) Modify `swinginit` so that, when the initial point is inside the circle of radius two, the other possible initial configuration is chosen.
(e) Modify `swinger` so that masses other than $m_1 = m_2 = 1$ are possible.
(f) Modify `swinger` so that lengths other than $\ell_1 = \ell_2 = 1$ are possible. This is trickier than changing the masses because the initial geometry is involved.
(g) What role does gravity play? How would the behavior of a double pendulum change if you could take it to the moon? How does changing the value of `g` in `swingrhs` affect the speed of the graphics display, the step sizes chosen by the ordinary differential equation solver, and the computed values of `t`?
(h) Combine `swingmass` and `swingrhs` into one function, `swingode`. Eliminate the `mass` option and use `ode23tx` instead of `ode23`.
(i) Are these equations stiff?
(j) This is a difficult question. The statement `swinger(0,2)` tries to delicately balance the pendulum above its pivot point. The pendulum does stay there for a while, but then loses its balance. Observe the value of t displayed in the title for `swinger(0,2)`. What force knocks the pendulum away from the vertical position? At what value of t does this force become noticeable?