Stability Analysis for Systems of Differential Equations

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/

Created: February 8, 2003
Last Modified: March 2, 2008

Contents

1 Introduction

In setting up a physical simulation involving objects, a primary step is to establish the equations of motion for the objects. These equations are formulated as a system of second-order ordinary differential equations that may be converted to a system of first-order equations whose dependent variables are the positions and velocities of the objects. Such a system is of the generic form

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}), \quad t \geq 0, \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{1}$$

where $\mathbf{x}_0$ is a specified initial condition for the system. The components of $\mathbf{x}$ are the positions and velocities of the objects. The function $\mathbf{f}(t, \mathbf{x})$ includes the external forces and torques of the system. A computer implementation of the physical simulation amounts to selecting a numerical method to approximate the solution to the system of differential equations. In many cases a general-purpose solver may be used with little thought about the step size of the solver. If the simulation appears to work properly, then so be it. However, in other cases the simulation might not behave as expected. A numerical analysis of the method is in order to determine if the numerical method is stable, and if so, to select an appropriate step size for the solver.

2 Physical Stability

A solution $\boldsymbol{\phi}(t)$ to the system (1) is said to be *stable* if every solution $\boldsymbol{\psi}(t)$ of the system close to $\boldsymbol{\phi}(t)$ at initial time $t = 0$ remains close for all future time. In mathematical terms, reminiscient of the definition for a limit: For each choice of $\varepsilon > 0$ there is a $\delta > 0$ such that $|\boldsymbol{\psi}(t) - \boldsymbol{\phi}(t)| < \varepsilon$ whenever $|\boldsymbol{\psi}(0) - \boldsymbol{\phi}(0)| < \delta$. If at least one solution $\boldsymbol{\psi}(t)$ does not remain close, then $\boldsymbol{\phi}(t)$ is said to be *unstable*. For a stable solution the ε-δ definition says that you select the maximum amount of error ε you can tolerate between $\boldsymbol{\psi}(t)$ and $\boldsymbol{\phi}(t)$. The value δ, which depends on your choice of ε, tells you how close to $\boldsymbol{\phi}(0)$ you have to start in order to stay within that error. I refer to the stability of the system of differential equations as the *physical stability* of the system, emphasizing that the system of equations is a model of the physical behavior of the objects of the simulation.

In general the stability analysis depends greatly on the form of the function $\mathbf{f}(t, \mathbf{x})$ and may be intractable. In the case of an *autonomous system* where the function does not depend explicitly on t,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad t \geq 0, \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2}$$

the analysis is tractable. An *equilibrium solution* of this system is a constant vector $\mathbf{c}$ for which $\mathbf{f}(\mathbf{c}) = \mathbf{0}$. That is, the constant function $\mathbf{x}(t) \equiv \mathbf{c}$ is a solution to the differential equation with initial condition $\mathbf{x}(0) = \mathbf{c}$. The system may be *linearized* about that constant by using Taylor's Theorem,

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{c}) + D\mathbf{f}(\mathbf{c})(\mathbf{x} - \mathbf{c}) + \mathbf{R}(\bar{\mathbf{x}}) = D\mathbf{f}(\mathbf{c})(\mathbf{x} - \mathbf{c}) + \mathbf{R}(\bar{\mathbf{x}})$$

where $D\mathbf{f}(\mathbf{c})$ is the matrix of first-order partial derivatives of $\mathbf{f}(\mathbf{x})$ evaluated at $\mathbf{c}$. If $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the first-derivative matrix in general is

$$D\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

The remainder is $\mathbf{R}(\bar{\mathbf{x}})$ where $\bar{\mathbf{x}}$ is some value dependent on $\mathbf{x}$ and $\mathbf{c}$ and includes the second- and higher-order terms of the original function. The last equality occurs because $\mathbf{f}(\mathbf{c}) = 0$ by definition of equilibrium solution. The *linearized system* is $\dot{\mathbf{x}} = D\mathbf{f}(\mathbf{c})(\mathbf{x} - \mathbf{c})$, but we can make the change of variables $\mathbf{z} = \mathbf{x} - \mathbf{c}$ and consider instead

$$\dot{\mathbf{z}} = A\mathbf{z}, \ \ t \geq 0, \ \ \mathbf{z}(0) = \mathbf{0} \tag{3}$$

where $A = D\mathbf{f}(\mathbf{c})$.

The physical stability of the linear system (3) is determined completely by the *eigenvalues* of the matrix A which are the roots to the polynomial $p(\lambda) = \det(A - \lambda I) = 0$ where I is the identity matrix. An *eigenvector* $\mathbf{v}$ corresponding to an eigenvalue λ is a nonzero vector for which $A\mathbf{v} = \lambda\mathbf{v}$. The eigenvalues can be real- or complex-valued. If $\lambda = \alpha + \imath\beta$ is an eigenvalue, written as a complex number, the *real part* of λ is the real number α.

Stability of the Linear System

1. Every solution is stable if all the eigenvalues of A have negative real part.
2. Every solution is unstable if at least one eigenvalue of A has positive real part.
3. Suppose that the eigenvalues of A all have real parts that are zero or negative. List those eigenvalues with zero real part as $\lambda_j = \imath\beta_j$ for $1 \leq j \leq \ell$. Let the multiplicity of λ_j be m_j; that is, $p(\lambda) = (\lambda - \lambda_j)^{m_j} q(\lambda)$ where $q(\lambda_j) \neq 0$. Every solution is stable if A has m_j linearly independent eigenvectors for each λ_j. Otherwise, every solution is unstable.

The results have to do with what types of functional terms appear in the solution to the linear system. If $\lambda = \alpha$ (real-valued), then the solution includes terms of the form $e^{\alpha t}$ and possibly $t^p e^{\alpha t}$ for various positive powers p. If $\lambda = \alpha + \imath\beta$ where $\beta \neq 0$ (complex-valued), then the solution includes terms of the form $e^{\alpha t}\sin(\beta t)$, $e^{\alpha t}\cos(\beta t)$, and possibly $t^p e^{\alpha t}\sin(\beta t)$ and $t^p e^{\alpha t}\cos(\beta t)$ for various positive powers p. If $\alpha < 0$ for all eigenvalues, then all the functional terms approach zero in the limit as t approaches infinity. If $\alpha > 0$ for at least one of the eigenvalues, the corresponding functional terms are unbounded as t approaches infinity and consequently cannot stay close to the equilibrium solution zero. If $\lambda = \imath\beta$ for an eigenvalue with all other eigenvalues satisfying $\alpha \leq 0$, the condition that the number of linearly independent eigenvectors of λ is equal to the multiplicity of λ implies that the functional terms only include $\cos(\beta t)$ and $\sin(\beta t)$, both bounded for increasing time and the solutions are stable. If the number of linearly independent eigenvectors is less than the multiplicity of λ, the functional terms include a $t^p\cos(\beta t)$ and a $t^p\sin(\beta)$ for some $p \geq 1$, in which case the solutions become unbounded as t approaches infinity, so the solutions are unstable.

The physical stability of the equilibrium solution $\mathbf{c}$ of the autonomous system (2) is related to that of its linearized system.

Stability of the Autonomous System

1. Every solution is stable if all the eigenvalues of $D\mathbf{f}(\mathbf{c})$ have negative real part.
2. Every solution is unstable if at least one eigenvalue of $D\mathbf{f}(\mathbf{c})$ has positive real part.
3. Suppose that the eigenvalues of A all have real parts that are zero or negative with at least one eigenvalue having zero real part. Not enough information is available to conclude whether or not the equilibrium solution is stable. (More analysis must be done.)

3 Numerical Stability

Physical stability of an equilibrium solution to a system of differential equations addresses the behavior of solutions that start nearby the equilibrium solution. However, we will solve $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ using some numerical method. It is important that the approximations generated by the method are themselves close to the true solution. The concept of closeness in this context is referred to as *numerical stability.* In general to obtain numerical stability, you will need to carefully choose your step size h in the numerical solvers. The end result of our discussion will be that you can only safely do this by understanding the relationship between numerical stability and physical stability.

3.1 Stability for Single-Step Methods

Three concepts are of interest: *consistency*, *convergence*, and *stability. Local truncation error* refers to the terms we discard when generating a numerical method from something such as a Taylor expansion. For example, Euler's method arises from Taylor's Theorem in representing a solution to $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ as

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + h\dot{\mathbf{x}}(t_i) + \frac{h^2}{2}\ddot{(\mathbf{x})}(\bar{t}_i) = \mathbf{x}(t_i) + h\mathbf{f}(t_i, \mathbf{x}(t_i)) + \frac{h^2}{2}\ddot{(\mathbf{x})}(\bar{t}_i)$$

where $\bar{t}_i \in [t_i, t_{i+1}]$, but whose value is generally unknown to us. We discard the second-order term to obtain the numerical method

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}(t_i, \mathbf{y}_i)$$

where $\mathbf{y}_i$ is the approximation to $\mathbf{x}(t_i)$. The discarded term is the local truncation error for Euler's method and is of order $O(h^2)$. As we make h small, the local truncation error for a single iteration is small. If we can make the local truncation errors become small for k iterations, that is a good thing for our numerical method. That leads us to the definition shown below.

Definition. Let τ_i denote the local truncation error at the i-th step of the numerical method. The method is said to be *consistent* with the differential equation it approximates if

$$\lim_{h\to 0} \max_{1\le i\le k} |\tau_i| = 0$$

Intuitively this says that for very small step sizes, the local truncation error made at any time t is very small.

According to this definition, Euler's method is consistent.

In general having very small local truncation errors at any time is not enough to guarantee that $\mathbf{y}_i$ is a good approximation to $\mathbf{x}(t_i)$. We need a definition about closeness.

Definition. A numerical method is said to be *convergent* with respect to the differential equation if

$$\lim_{h\to 0} \max_{1\le i\le k} |\mathbf{x}(t_i) - \mathbf{y}_i| = 0$$

Intuitively this says that for very small step sizes, the maximum error at any time t between the approximation and the true solution is very small.

A careful analysis of Euler's method will show that it is convergent.

Our last definition is about stability itself.

> **Definition**. A numerical method is said to be *stable* if small changes in the initial data for the differential equation produce correspondingly small changes in the subsequent approximations. In formal terms, let $\mathbf{x}_0$ and $\mathbf{x}_1$ be two initial values for the differential equation. Let $\mathbf{y}_i$ be the approximation to the solution $\mathbf{x}(t_i; \mathbf{x}_0)$ where the last component indicates the initial data that generated the solution. Let $\bar{\mathbf{y}}_i$ be the approximation to $\mathbf{x}(t_i; \mathbf{x}_1)$. For each $\varepsilon > 0$ there is a $\delta > 0$ sufficiently small so that $|\mathbf{y}_i - \bar{\mathbf{y}}_i| < \varepsilon$ whenever $|\mathbf{x}_1 - \mathbf{x}_0| < \delta$.

This definition is a statement about continuous dependence of solutions on the initial data.

The relationship between consistency, convergence, and stability for a single-step numerical method is summarized by the following result.

> **Theorem**. Consider the initial value problem $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ for $t \in [t_0, t_0 + \alpha]$ with initial data $\mathbf{x}(t_0) = \mathbf{x}_0$. Let a numerical method for the equation be of the form $\mathbf{y}_0 = \mathbf{x}_0$ and $\mathbf{y}_{i+1} = \mathbf{y}_i + h\boldsymbol{\phi}(t_i, \mathbf{y}_i, h)$ for $i \geq 0$. If there is a value $h_0 > 0$ such that $\boldsymbol{\phi}(t, \mathbf{y}, h)$ is continuous on the domain
> $$D = \{(t, \mathbf{y}, h) : t \in [t_0, t_0 + \alpha],\ \mathbf{y} \in \mathbb{R}^n,\ h \in [0, h_0]\}$$
> and if there exists a constant $L > 0$ such that
> $$|\phi(t, \mathbf{y}, h) - \phi(t, \bar{\mathbf{y}}, h)| \leq L|\mathbf{y} - \bar{\mathbf{y}}|$$
> for all $(t, \mathbf{y}, h), (t, \bar{\mathbf{y}}, h) \in D$, called a *Lipschitz condition.*
>
> 1. The numerical method is stable.
> 2. The method is convergent if and only if it is consistent; that is, if and only if $\boldsymbol{\phi}(t, \mathbf{x}, 0) = \mathbf{f}(t, \mathbf{x}, 0)$ for all $t \in [t_0, t_0 + \alpha]$.
> 3. If the local truncation errors are bounded by $|\tau_i| \leq T(h)$ for some function $B(h)$ independent of i and for $h \in [0, h_0]$, then $|\mathbf{x}(t_i) - \mathbf{y}_i| \leq B(h) \exp(L(t_i - t_0))/L$.

The condition in item 2 of the conclusion is easily verified for all the single-step numerical methods you will encounter. Notice that for Euler's method, $\boldsymbol{\phi}(t, \mathbf{y}, 0) = \mathbf{f}(t, \mathbf{y})$. The same is true for the Runge-Kutta methods. Item 3 of the conclusions gives us an upper bound on the error of the approximation.

3.2 Stability for Multistep Methods

A stability result for multistep methods is only slightly more complicated than the one for single-step methods. The general multistep method can be formulated as

$$\mathbf{y}_{i+1} = \sum_{j=0}^{m-1} a_j \mathbf{y}_{i-j} + h\mathbf{F}(t_i, h, \mathbf{y}_{i+1}, \mathbf{y}_i, \ldots, \mathbf{y}_{i+1-m}), \quad i \geq 0$$

with start-up conditions $\mathbf{y}_i = \mathbf{b}_i$ for selected constants $\mathbf{b}_i$ with $0 \leq i \leq m-1$. Since we started with a differential equation with initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$, we choose $\mathbf{y}_0 = \mathbf{x}_0$. The other start-up values are generated by some single-step method. The prototypical explicit method is Adams-Bashforth and the prototypical implicit method is Adams-Moulton.

The concepts of convergence and stability are the same as for single-step methods, but consistency requires a slight bit more to it. The local truncation errors for a multistep method are of the form

$$\tau_{i+1} = \frac{\mathbf{x}(t_{i+1}) - \sum_{j=0}^{m-1} \mathbf{x}(t_{i-j})}{h} - \mathbf{F}(t_i, h, \mathbf{x}(t_{i+1}), \ldots, \mathbf{x}(t_{i+1-m}))$$

The analysis of the algorithm for the m-step Adams-Bashforth method will show that the local truncation error is $\tau_{i+1} = O(h^m)$. The m-step Adams-Moulton method has local truncation error of $\tau_{i+1} = O(h^{m+1})$. The definition for consistency of a multistep method includes the same condition we used for single-step methods,

$$\lim_{h\to 0} \max_{1\le i\le k} |\tau_i| = 0$$

so the local truncation errors go to zero as the step size becomes small, but we additionally need to make sure that the local truncation errors for the start-up conditions become small, also.

$$\lim_{h\to 0} \max_{1\le i\le m-1} |\mathbf{x}(t_i) - \mathbf{b}_i| = 0$$

The result for multistep methods that is the analogy of the one I stated for single-step methods is the following.

Theorem. Consider the initial value problem $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ for $t \in [t_0, t_0 + \alpha]$ with initial data $\mathbf{x}(t_0) = \mathbf{x}_0$. Consider a multistep method of the form $\mathbf{y}_{i+1} = \sum_{j=0}^{m-1} a_j \mathbf{y}_{i-j} + h\mathbf{F}(t_i, h, \mathbf{y}_{i+1}, \mathbf{y}_i, \ldots, \mathbf{y}_{i+1-m})$ with start-up conditions $\mathbf{y}_0 = \mathbf{x}_0$ and $\mathbf{y}_i = \mathbf{b}_i$ for specified constants $\mathbf{b}_i$ with $1 \le i \le m-1$. Suppose that $\mathbf{F} \equiv \mathbf{0}$ whenever $\mathbf{f} \equiv \mathbf{0}$. Suppose that $\mathbf{F}$ satisfies a Lipschitz condition

$$|\mathbf{F}(t_i, h, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_{i+1-m}) - \mathbf{F}(t_i, h, \bar{\mathbf{y}}_{i+1}, \ldots, \bar{\mathbf{y}}_{i+1-m}| \le L \sum_{j=0}^{m} |\mathbf{y}_{i+1-j} - \bar{\mathbf{y}}_{i+1-j}|$$

for each i with $m-1 \le i \le n$. Define the polynomial $p(\lambda) = \lambda^m - \sum_{j=0}^{m-1} a_j \lambda^{m-1-j}$.

1. The numerical method is stable if and only if all roots of $p(\lambda) = 0$ satisfy $|\lambda| \le 1$ and any root such that $|\lambda| = 1$ must be a simple root (multiplicity is 1).
2. If the numerical method is consistent with the differential equation, the method is stable if and only if it is convergent.

The important aspect of this theorem for practical purposes is the analysis of the roots of $p(\lambda)$. Note that the roots can be nonreal. Also note that $p(1) = 0$ because of the way the a_j were defined for multistep methods. Thus, $\lambda = 1$ is always a root and has magnitude $|\lambda| = 1$. If $\lambda = 1$ is the only root of magnitude 1, all other roots satisfying $|\lambda| < 1$, then the numerical method is said to be *strongly stable.* If more than one root has magnitude one, the others satisfying $|\lambda| < 1$, the numerical method is said to be *weakly stable.* If any root has magnitude $|\lambda| > 1$, the numerical method is said to be *unstable.*

3.3 Choosing a Stable Step Size

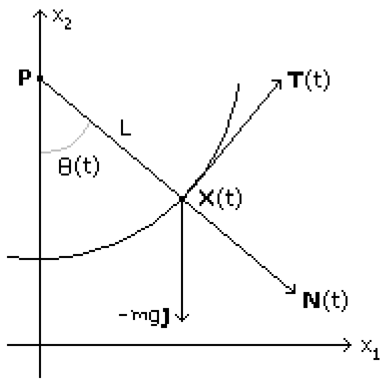
In fact we can go one step further. The analysis of the roots of $p(\lambda)$ when using the linearized system instead of the original autonomous system allows us to decide what step sizes h lead to stability, an important issue for solving the equations on a computer.

Assuming an equilibrium solution of $\mathbf{x}(t) \equiv \mathbf{0}$, the linearized equation is $\dot{\mathbf{x}} = A\mathbf{x}$ where A is an $n \times n$ matrix of constants occuring in the expansion $\mathbf{f}(\mathbf{x}) = A\mathbf{x} + \mathbf{R}$. Let us assume that the eigenvalues of A all have real parts negative so that the equilibrium solution is physically stable. For each eigenvalue λ of A, consider what is called the *modal equation*, $\dot{\mathbf{x}} = \lambda\mathbf{x}$. An m-step method ($m \geq 1$) is applied to the modal equation. The resulting difference equation is linear and has a characteristic polynomial of the type $p(z)$ whose coefficients involve the eigenvalue λ and the step size h. The polynomial includes the linear contribution from the function $\mathbf{F}$ that appears in the multistep method. The condition of $|z| \leq 1$ for all roots and unit-magnitude roots being simple are required for a stable method. This imposes a constraint on how we can choose h. The best way to illustrate this is with an example.

4 The Simple Pendulum

A *simple pendulum* consists of a particle of mass m attached to one end of a wire of length L of negligible mass. The other end of the wire is attached to a joint at position $\mathbf{P}$. The joint is assumed to be frictionless. The only force acting on the particle is gravity. The magnitude of that force is assumed to be a constant. Figure 4.1 illustrates the problem.

Figure 4.1 A simple pendulum.



The pendulum moves only within the plane. The angle formed by the wire and the vertical is $\theta(t)$. The angle has a positive value when the wire is to the right of vertical. The position of the particle is $\mathbf{x}(t) = \mathbf{P} + L\mathbf{N}(t)$ where $\mathbf{N}(t) = (\sin(\theta(t)), -\cos(\theta(t)))$. The acceleration due to gravity points downwards and is labeled as $-g\jmath$ in the figure. The value $g > 0$ is assumed to be constant.

The particle is constrained to move along a circular path. The tangent vector $\mathbf{T}(t) = (\cos(\theta(t)), \sin(\theta(t)))$ and normal vector $\mathbf{N}(t)$ at $\mathbf{x}(t)$, as shown in the figure, form a pair of coordinate axes at the point. The particle cannot move in the normal direction, so the equation of motion involves forces only in the tangential direction. The decomposition of the gravitational acceleration in terms of the tangent and normal is

$$-g\jmath = -g\sin(\theta)\mathbf{T} + g\cos(\theta)\mathbf{N}.$$

The velocity and acceleration of the particle are

$$\dot{\mathbf{x}} = L\frac{d\mathbf{N}}{dt} = L\dot{\theta}\mathbf{T} \text{ and } \ddot{\mathbf{x}} = L(\ddot{\theta}\mathbf{T} - \dot{\theta}\mathbf{N}).$$

Equating the tangential forces, Newton's law states

$$m\ddot{\mathbf{x}} \cdot \mathbf{T} = -mg\jmath \cdot \mathbf{T}$$

which simplifies to

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta). \tag{4}$$

This is a second-order nonlinear ordinary differential equation. The initial value problem requires choosing $\theta(0)$ and $\theta'(0)$. It is not possible to construct a solution in closed form, so numerical methods must be applied to solve this.

If the initial angle is close to zero, then $\theta(t)$ remains close to zero. Using the approximation $\sin\theta \doteq \theta$ for small angles, equation (4) is approximated by

$$\ddot{\theta} = -\frac{g}{L}\theta.$$

This equation is linear and does have a closed-form solution,

$$\theta(t) = \theta(0)\cos\left(\sqrt{\frac{g}{L}}\,t\right) + \sqrt{\frac{L}{g}}\,\theta'(0)\sin\left(\sqrt{\frac{g}{L}}\,t\right).$$

The motion is sinusoidal and is referred to as *simple harmonic motion*. The solution to equation (4) must be periodic, but it is not simple harmonic motion.

4.1 Numerical Solution of the ODE

Equation (4) can be converted to a first-order nonlinear system by setting $u(t) = \theta'(t)$. The system is

$$\theta'(t) = u(t), \quad u'(t) = -c\,\sin(\theta(t)), \quad t \geq 0$$

where $c = \sqrt{g/L}$ and where $\theta(0)$ and $u(0)$ are user-specified inputs.

Without any analysis, let us just try Euler's method to solve the equation. Euler's method uses a forward difference to approximate a derivative:

$$\frac{d\mathbf{x}(t)}{dt} \doteq \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h}.$$

The differential equation $d\mathbf{x}/dt = \mathbf{f}(t, \mathbf{x})$ is approximated as $\mathbf{x}(t+h) \doteq \mathbf{x}(t) + h\mathbf{f}(t, \mathbf{x}(t))$. If $\mathbf{y}_i$ is the approximation to $\mathbf{x}(t_i)$ where t_0 is the initial time, $\mathbf{y}_0 = \mathbf{x}_0$ is the initial condition, h is the step size, and $t_i = t_0 + ih$ for $i \geq 0$, then Euler's method is

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}(t_i, \mathbf{y}_i), \quad i \geq 0.$$

For the problem at hand, $\mathbf{y}_i = (\theta_i, u_i)$ and the iteration equations are

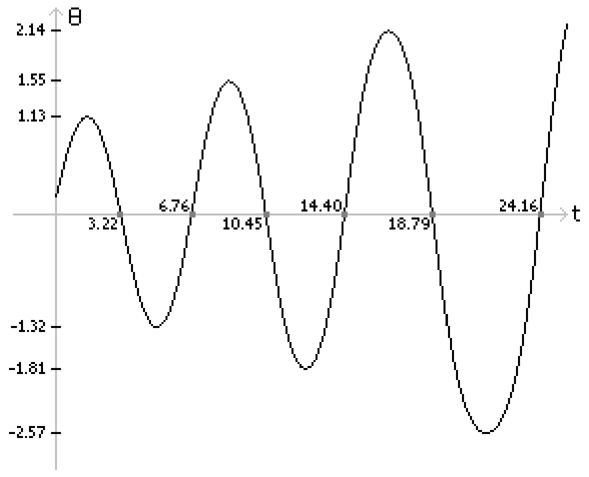
$$\theta_{i+1} = \theta_i + hu_i, \quad u_{i+1} = u_i - hc \, \sin\theta_i, \quad i \geq 0$$

with θ_0 and u_0 known initial values. The value $h > 0$ is a small step in time t. The following code implements Euler's method and generates the sequence of n samples θ_i for $0 \leq i < n$ for specified initial conditions and step size.

```
float* ExplicitEuler (float c, float theta0, float u0, float h, int n)
{
    float* theta = new float[n];
    theta[0] = theta0;
    for (int i = 1; i < n; i++)
    {
        float theta1 = theta0 + h*u0;
        float u1 = u0 - h*c*sin(theta0);
        theta0 = theta1;
        u0 = u1;
        theta[i] = theta0;
    }
    return theta;
}

void Test ()
{
    int n = 256;
    float c = 1.0f, theta0 = 0.1f, u0 = 1.0f, h = 0.1f;
    float* theta = ExplicitEuler(c,theta0,u0,h,n);
    // use the output...
}
```

Figure 4.2 shows a plot of the output of the numerical method.

Figure 4.2 Euler's method applied to the simple pendulum problem. The image shows a plot of the pendulum angles over time.



Observe that the angles are becoming unbounded over time, contrary to how the physical solution should behave. The true solution should be periodic implying that the maximum angles are all the same and the minimum angles are all the same. Also, the time between two consecutive zeros should be a constant. The results should make you question whether choosing Euler's method without analysis was a good thing to do.

Euler's method is an explicit method. The next iterate is explicitly defined in terms of the previous iterates and time. Explicit methods tend to be conditionally stable-small steps sizes are required. Implicit methods tend to have better stability properties. Let's try an implicit Euler method and see what happens. The implicit method uses a backward difference to approximate a derivative:

$$\frac{d\mathbf{x}(t)}{dt} \doteq \frac{\mathbf{x}(t) - \mathbf{x}(t-h)}{h}.$$

The differential equation $d\mathbf{x}/dt = \mathbf{f}(t, \mathbf{x})$ is approximated as $\mathbf{x}(t) \doteq \mathbf{x}(t-h) + h\mathbf{f}(t, \mathbf{x}(t))$. Using the same notation as in the application of the explicit Euler's method, the iteration scheme is

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}(t_{i+1}, \mathbf{y}_{k+1}), \ \ i \geq 0.$$

Observe that $\mathbf{y}_{i+1}$ occurs on both sides of the equation, so it is an *implicit* term; generally it is not possible to solve for it explicitly. Having known values for $\mathbf{y}_i$, h, and t_{i+1}, we can formulate the problem instead as computing a vector $\mathbf{z}$ for which

$$\mathbf{G}(\mathbf{z}) = \mathbf{0}$$

where $\mathbf{G}(\mathbf{z}) = \mathbf{y}_i + h\mathbf{F}(t_{i+1}, \mathbf{z}) - \mathbf{z}$ That is, $\mathbf{z}$ is a root of the function $\mathbf{G}$. Newton's method can be applied to approximate a root.

For the problem at hand, $\mathbf{y}_i = (\theta_i, u_i)$ and the iteration equations are

$$\theta_{i+1} = \theta_i + hu_{i+1}, \ \ u_{i+1} = u_i - hc\,\sin\theta_{i+1}, \ \ i \geq 0$$

Rather than implementing Newton's method directly for two equations in two unknowns, we can combine the equations by substituting the u_{i+1} equation into the first and rearranging terms,

$$\theta_{i+1} + ch^2 \sin\theta_{i+1} - (\theta_i + hu_i) = 0.$$

Define $g(z) = z + ch^2 \sin z - (\theta_i + hu_i)$. The next iterate θ_{i+1} is a root to $g(z) = 0$. Given an initial guess z_0, the Newton iterates are

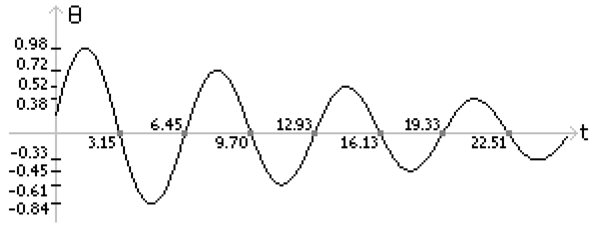
$$z_{m+1} = z_m - \frac{g(z_m)}{g'(z_m)}, \quad m \geq 0.$$

A reasonable initial guess is $z_0 = \theta_i$. The code for the implicit Euler's method is

```
float* ImplicitEuler (float c, float theta0, float u0, float h, int n)
{
    const int maxIterations = 32;  // can be increased if needed
    float a0 = c*h, a1 = h*a0;

    float* theta = new float[n];
    theta[0] = theta0;
    for (int i = 1; i < n; i++)
    {
        float a2 = theta0 + h*u0;
        float theta1 = theta0;
        for (int j = 0; j < maxIterations; j++)
        {
            float g = theta1 + a1*sin(fX1) - a2;
            float gderiv = 1.0f + a1*cos(fX1);
            theta1 -= g/gderiv;
        }
        float u1 = u0 - a0*sin(theta1);
        theta0 = theta1;
        u0 = u1;
        theta[i] = theta0;
    }
    return theta;
}

void Test ()
{
    int n = 256;
    float c = 1.0f, theta0 = 0.1f, u0 = 1.0f, h = 0.1f;
    float* theta = ImplicitEuler(c,theta0,u0,h,n);
    // use the output...
}
```

For simplicity, no convergence or stopping criterion is used in the inner loop that constructs the Newton's iterates; the loop just runs a fixed number of times. A more sophisticated loop with an eye towards minimizing inner loop cycles may certainly be tried. Figure 4.3 shows a plot of the output of the numerical method.

Figure 4.3 Implicit Euler's method applied to the simple pendulum problem. The image shows a plot of the pendulum angles over time.



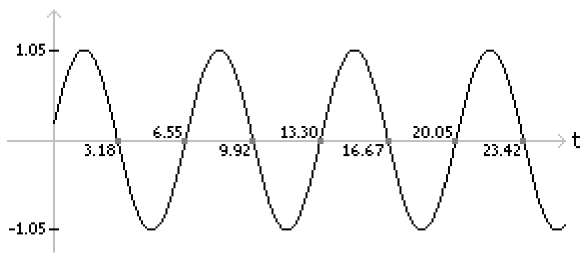
Now the angles are dampened over time, contrary to how the physical solution should behave, although in this case someone observing the numerical pendulum behavior might think the physical system has friction at the joint causing the oscillations to dampen. The time between two consecutive zeros in the Euler's method was significantly increasing over time. In the implicit Euler's method, the time between zeros is only gradually decreasing.

Finally, we try the Runge-Kutta fourth-order solver. The code is shown below. The input to the function is an array of two values, `p[0]` that corresponds to θ and `p[1]` that corresponds to $u = \theta'$.

```
float Function0 (float* p)
{
    return p[1];
}

float Function1 (float* p)
{
    const float c = 1.0f; // c = g/L in the model
    return -c*sin(p[0]);
}

float* RungeKutta (float c, float theta0, float u0, float h, int n)
{
    const int dim = 2;
    ODE::AutoFunction F[2] = { Function0, Function1 };
    RK4 solver(dim,h,F);

    float in[2] = { theta0, u0 }, out[2];
    float* theta = new float[n];
    theta[0] = theta0;
    for (int k = 1; k < n; k++)
    {
        kSolver.Update(in,out);
        in[0] = out[0];
        in[1] = out[1];
        theta[i] = in[0];
    }
    return theta;
}

void Test ()
{
    int n = 256;
    float c = 1.0f, theta0 = 0.1f, u0 = 1.0f, h = 0.1f;
    float* theta = RungeKutta(c,theta0,u0,h,n);
    // use the output...
}
```

Figure 4.4 shows a plot of the output of the numerical method.

Figure 4.4 Runge-Kutta fourth-order method applied to the simple pendulum problem. The image shows a plot of the pendulum angles over time.



The results appear to indicate that this method is stable. The zeros of $\theta(t)$ are evenly spaced and the four maximum values, in order of increasing time are 1.05289, 1.05117, 1.05285, and 1.05249. The four minimum values are -1.05232, -1.05291, -1.05221, -1.05293. The table below compares the times obtained for the zeros of $\theta(t)$.

explicit time	difference	implicit time	difference	Runke-Kutta time	difference
3.22	$--$	3.15	$--$	3.17	$--$
6.76	3.54	6.45	3.30	6.55	3.38
10.45	3.69	9.70	3.25	7.92	3.37
14.40	3.95	12.93	3.23	13.30	3.38
18.79	4.39	16.13	3.20	16.67	3.37
24.16	5.37	22.51	3.18	20.05	3.38
				23,42	3.37

4.2 Physical Stability for the Pendulum

The stability analysis of the simple pendulum problem will be done in a slightly more general form. The work required to understand the general form is no more than that of its special case. The initial value differential equation to be analyzed is

$$\ddot{\theta} + b\dot{\theta} + c\sin(\theta) = 0, \ \ \theta(0) = \theta_0, \dot{\theta}(0) = \dot{\theta}_0, \ \ t \geq 0$$

where $b \geq 0$ and $c > 0$ are known constants. The special case $b = 0$ corresponds to the simple pendulum problem. A choice of $b > 0$ is an attempt to add viscous friction to the system; the frictional force is proportional to the angular speed at the joint of the pendulum. The equation has two equilibrium solutions, $\theta(t) \equiv 0$ (the pendulum hangs straight down and is at rest) and $\theta(t) \equiv \pi$ (the pendulum stands straight up and is at rest, albeit unquite rest). At $\theta(t) \equiv 0$ the linearized equation is

$$\ddot{\theta} + b\dot{\theta} + c\theta = 0.$$

The corresponding characteristic equation is $\lambda^2 + b\lambda + c = 0$. At $\theta(t) \equiv \pi$ the linearized equation is

$$\ddot{\theta} + b\dot{\theta} + c(\pi - \theta) = 0.$$

The corresponding characteristic equation to the homogeneous equation is $\lambda^2 + b\lambda - c = 0$. An equilibrium solution is stable whenever the roots to its characteristic equation have only negative real parts. If both real parts are zero, you get marginal stability (as will be shown for the simple pendulum problem). If at least one root has positive real part, the solution is unstable. For $b > 0$, $\theta(t) \equiv 0$ is a stable solution since the roots to its characteristic equation are

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4c}}{2}.$$

Both roots are negative real numbers when $b^2 \geq 4c$ or are complex with negative real parts when $b^2 < 4c$. Regardless of choice of b, $\theta(t) \equiv \pi$ is unstable since the roots to its characteristic equation are

$$\lambda = \frac{-b \pm \sqrt{b^2 + 4c}}{2}.$$

Both roots are real-valued, but one is positive and one is negative.

4.3 Numerical Stability of the ODE Solvers

In this subsection let us look at the numerical stability of the various numerical methods relative to the equilibrium solution $(\theta(t), u(t)) \equiv (0, 0)$. The eigenvalues for the matrix in the linearized system are $\lambda = (-b \pm \sqrt{b^2 - 4c})/2$ where $b \geq 0$. The modal equation is $\dot{\mathbf{x}} = \lambda\mathbf{x}$.

Apply Euler's method to the modal equation,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\lambda\mathbf{x}_i = (1 + h\lambda)\mathbf{x}_i$$

The characteristic polynomial is

$$p(z) = z - (1 + h\lambda)$$

and has a single root $z = 1 + \lambda h$. For stability we need $|1 + \lambda h| \leq 1$.

The implicit Euler's method for the modal equation is

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\lambda\mathbf{x}_{i+1}$$

The characteristic polynomial is

$$p(z) = (1 - h\lambda)z - 1$$

and has a single root $z = 1/(1 - h\lambda)$. For stability we need $|1 - \lambda h| \geq 1$.

The Runge-Kutta fourth-order method to solve $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ in general is

$$\begin{aligned}
k_1 &= h\mathbf{f}(t_i, \mathbf{y}_i) \\
k_2 &= h\mathbf{f}(t_i + h/2, \mathbf{y}_i + k_1/2) \\
k_3 &= h\mathbf{f}(t_i + h/2, \mathbf{y}_i + k_2/2) \\
k_4 &= h\mathbf{f}(t_i + h, \mathbf{y}_i + k_3) \\
y_{i+1} &= y_i + (k_1 + 2k_2 + 2k_3 + k_4)/6
\end{aligned}$$

The linearized equation uses $\mathbf{f}(t, \mathbf{x}) = \lambda\mathbf{x}$. In this case,

$$\begin{aligned}
k_1 &= h\lambda y_i \\
k_2 &= h\lambda(1 + h\lambda/2)y_i \\
k_3 &= h\lambda(1 + (h\lambda/2)(1 + h\lambda/2))y_i \\
k_4 &= h\lambda(1 + h\lambda(1 + (h\lambda/2)(1 + h\lambda/2)))y_i
\end{aligned}$$

These combine to form

$$y_{i+1} = \left[1 + (h\lambda) + \frac{1}{2}(h\lambda)^2 + \frac{1}{6}(h\lambda)^3 + \frac{1}{24}(h\lambda)^4\right] y_i = q(h\lambda)y_i$$

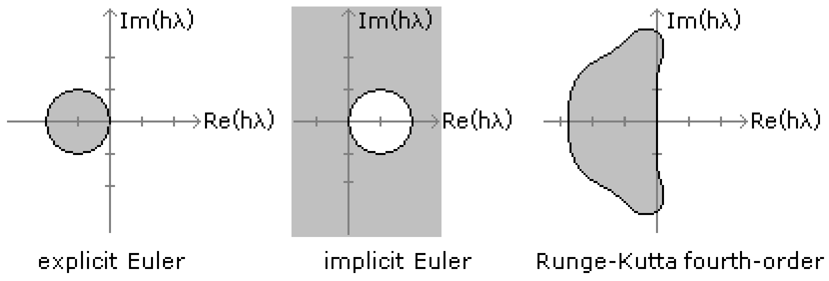
The characteristic polynomial is

$$p(z) = z - q(h\lambda)$$

and has a single root $z = q(h\lambda)$. For stability we need $|q(h\lambda)| \leq 1$.

Figure 4.5 shows the regions in the complex plane defined by the inequality constraints.

Figure 4.5 Left: The region for stability of the explicit Euler's method, $|1 + \lambda h| \leq 1$. Middle: The region for stability of the implicit Euler's method, $|1 - \lambda h| \geq 1$. Right: The region for stability of the Runge-Kutta fourth-order method.



The roots λ to the characteristic equation can be complex-valued which is why we are sketching the inequality regions in the complex plane. Also, the real parts must be nonnegative for stability (or marginal stability), so the regions to the left of the imaginary axis are the only ones of interest.

Now we are in a position to specify what the step size $h > 0$ must be in order that the methods be stable near the equilibrium solution $\theta(t) \equiv 0$ for $b = 0$ and $c = 1$ as we chose in the numerical experiments. The eigenvalues are $\lambda = \pm\imath$. The choice of h must guarantee that $|h\lambda|$ is inside the gray regions shown in Figure 4.5. Notice the $\pm h\imath$ are on the imaginary axes. For Euler's method, no imaginary axis points are inside the gray circle except zero (in which case $h = 0$), so this method is always unstable no matter how you choose your step size. For the implicit Euler's method, any choice of h leads to $\pm h\imath$ being in the gray region. The method is stable, but unfortunately not very accurate. For the Runge-Kutta method, the figure does not clearly show it, but the boundary of the gray region on the right is slightly to the right of the imaginary axis except at the origin which is on the boundary. An evaluation for $h = 0.1$, the choice we made in the numerical experiments, $|q(\pm 0.1\imath)| \doteq 0.999999986 < 1$. Thus, the Runge-Kutta method is stable. And it turns out to be quite accurate.