§3.2.3 Leapfrog Scheme (again)

The numerical solution of the differential equations requires that they be transformed into algebraic form.

This is done by the now-familiar method of finite differences.

§3.2.3 Leapfrog Scheme (again)

The numerical solution of the differential equations requires that they be transformed into algebraic form.

This is done by the now-familiar method of finite differences.

The continuous variables are represented by their values at a finite set of points, and derivatives are approximated by differences between values at adjacent points.

§3.2.3 Leapfrog Scheme (again)

The numerical solution of the differential equations requires that they be transformed into algebraic form.

This is done by the now-familiar method of finite differences.

The continuous variables are represented by their values at a finite set of points, and derivatives are approximated by differences between values at adjacent points.

The atmosphere is partitioned into several discrete horizontal layers, and each layer is divided up into grid cells.

Then the variables are evaluated at the centre of each cell.

§3.2.3 Leapfrog Scheme (again)

The numerical solution of the differential equations requires that they be transformed into algebraic form.

This is done by the now-familiar method of finite differences.

The continuous variables are represented by their values at a finite set of points, and derivatives are approximated by differences between values at adjacent points.

The atmosphere is partitioned into several discrete horizontal layers, and each layer is divided up into grid cells.

Then the variables are evaluated at the centre of each cell.

Similarly, the time interval under consideration is sliced into a finite number of discrete time steps.

§3.2.3 Leapfrog Scheme (again)

The numerical solution of the differential equations requires that they be transformed into algebraic form.

This is done by the now-familiar method of finite differences.

The continuous variables are represented by their values at a finite set of points, and derivatives are approximated by differences between values at adjacent points.

The atmosphere is partitioned into several discrete horizontal layers, and each layer is divided up into grid cells.

Then the variables are evaluated at the centre of each cell.

Similarly, the time interval under consideration is sliced into a finite number of discrete time steps.

Thus, the continuous evolution of the variables is approximated by the change from step to step.

In a sense, the finite difference method corresponds to a reversal of history.

In a sense, the finite difference method corresponds to a reversal of history.

Lewis Fry Richardson described the procedure:

Although the infinitesimal calculus has been a splendid success, yet there remain problems in which it is cumbrous or unworkable. When such difficulties are encountered it may be well to return to the manner in which they did things before the calculus was invented, postponing the passage to the limit until after the problem has been solved for a moderate number of moderately small differences.

(Richardson, 1927)

The leapfrog time scheme

We consider again the method of advancing the solution in time known as the leapfrog scheme.

The leapfrog time scheme

We consider again the method of advancing the solution in time known as the leapfrog scheme.

Let U denote a typical dependent variable, governed by an equation of the form

$$\frac{dU}{dt} = F(U)\,.$$

The leapfrog time scheme

We consider again the method of advancing the solution in time known as the leapfrog scheme.

Let U denote a typical dependent variable, governed by an equation of the form

$$\frac{dU}{dt} = F(U)\,.$$

The continuous time domain t is replaced by a sequence of discrete moments $\{0, \Delta t, 2\Delta t, \dots\ , n\Delta t, \dots\ \}$.

The solution at these moments is denoted by $U^n = U(n\Delta t)$.

The leapfrog time scheme

We consider again the method of advancing the solution in time known as the leapfrog scheme.

Let U denote a typical dependent variable, governed by an equation of the form

$$\frac{dU}{dt} = F(U)\,.$$

The continuous time domain t is replaced by a sequence of discrete moments $\{0, \Delta t, 2\Delta t, \dots, n\Delta t, \dots\}$.

The solution at these moments is denoted by $U^n = U(n\Delta t)$.

If this solution is known up to time $t = n\Delta t$, the right-hand term $F^n = F(U^n)$ can be computed.

Thus, we can integrate the equation forward in time.

The advection equation is, again,

$$\frac{dU}{dt} = F(U)\,.$$

The advection equation is, again,

$$\frac{dU}{dt} = F(U)\,.$$

The time derivative is approximated by a centered difference

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = F^n\,,$$

The advection equation is, again,

$$\frac{dU}{dt} = F(U)\,.$$

The time derivative is approximated by a centered difference

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = F^n\,,$$

Thus, the forecast value U^{n+1} may be computed from the old value U^{n-1} and the tendency F^n:

$$U^{n+1} = U^{n-1} + 2\Delta t\, F^n\,.$$

The advection equation is, again,

$$\frac{dU}{dt} = F(U)\,.$$

The time derivative is approximated by a centered difference

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = F^n\,,$$

Thus, the forecast value U^{n+1} may be computed from the old value U^{n-1} and the tendency F^n:

$$U^{n+1} = U^{n-1} + 2\Delta t\, F^n\,.$$

This process of stepping forward from moment to moment is repeated a large number of times, until the desired forecast range is reached.

For the physical equation, a single initial condition U^0 is sufficient to determine the solution.

For the physical equation, a single initial condition U^0 is sufficient to determine the solution.

One problem with the leapfrog scheme (and other three-time-level schemes) is that two values of U are required to start the computation.

For the physical equation, a single initial condition U^0 is sufficient to determine the solution.

One problem with the leapfrog scheme (and other three-time-level schemes) is that two values of U are required to start the computation.

In addition to the *physical* initial condition U^0, a *computational* initial condition U^1 is required.

For the physical equation, a single initial condition U^0 is sufficient to determine the solution.

One problem with the leapfrog scheme (and other three-time-level schemes) is that two values of U are required to start the computation.

In addition to the *physical* initial condition U^0, a *computational* initial condition U^1 is required.

This cannot be obtained using the leapfrog scheme, so normally a simple non-centered step

$$U^1 = U^0 + \Delta t \; F^0$$

is used to provide the value at $t = \Delta t$.

For the physical equation, a single initial condition U^0 is sufficient to determine the solution.

One problem with the leapfrog scheme (and other three-time-level schemes) is that two values of U are required to start the computation.

In addition to the *physical* initial condition U^0, a *computational* initial condition U^1 is required.

This cannot be obtained using the leapfrog scheme, so normally a simple non-centered step

$$U^1 = U^0 + \Delta t\ F^0$$

is used to provide the value at $t = \Delta t$.

From then on, the leapfrog scheme can be used.

However, the errors of the first step will persist.

For the physical equation, a single initial condition U^0 is sufficient to determine the solution.

One problem with the leapfrog scheme (and other three-time-level schemes) is that two values of U are required to start the computation.

In addition to the *physical* initial condition U^0, a *computational* initial condition U^1 is required.

This cannot be obtained using the leapfrog scheme, so normally a simple non-centered step

$$U^1 = U^0 + \Delta t\ F^0$$

is used to provide the value at $t = \Delta t$.

From then on, the leapfrog scheme can be used.

However, the errors of the first step will persist.

The computational initial condition can be defined in several ways:

- **Set** $U^1 = U^0$**. Since** $u^1 = u^0 + u_t\Delta t + \cdots$**, this introduces errors of order** $O(\Delta t)$**, and is not recommended.**

- **Set $U^1 = U^0$. Since $u^1 = u^0 + u_t\Delta t + \cdots$, this introduces errors of order $O(\Delta t)$, and is not recommended.**
- **Use a <u>forward time scheme</u> for the first time step. Since the forward time step is only used once, the total error is still of $O(\Delta t)^2$.**

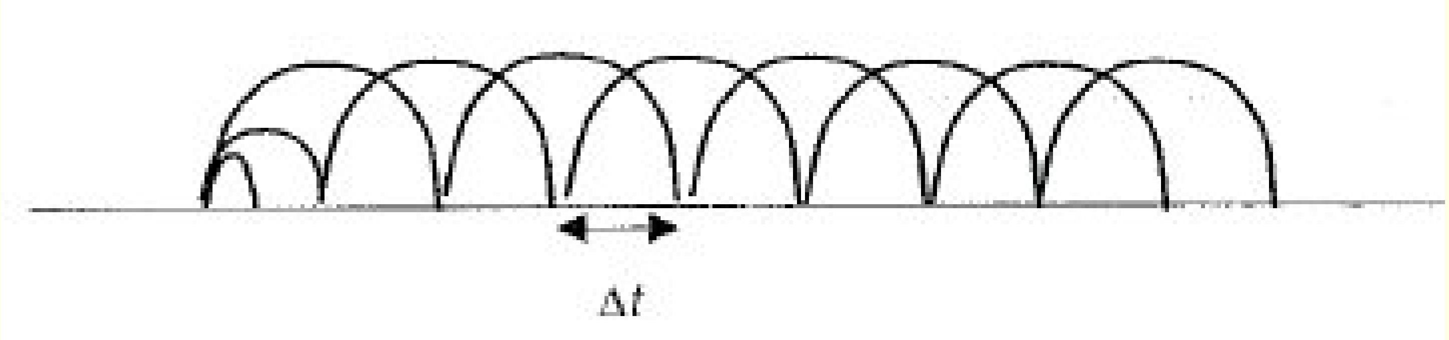
- **Set $U^1 = U^0$. Since $u^1 = u^0 + u_t \Delta t + \cdots$, this introduces errors of order $O(\Delta t)$, and is not recommended.**
- **Use a forward time scheme for the first time step. Since the forward time step is only used once, the total error is still of $O(\Delta t)^2$.**
- **An alternative is to use an Euler-backwards (Matsuno) scheme for the first time step.**

- **Set $U^1 = U^0$. Since $u^1 = u^0 + u_t\Delta t + \cdots$, this introduces errors of order $O(\Delta t)$, and is not recommended.**

- **Use a forward time scheme for the first time step. Since the forward time step is only used once, the total error is still of $O(\Delta t)^2$.**

- **An alternative is to use an Euler-backwards (Matsuno) scheme for the first time step.**

- **Use half of the initial time step for the forward time step, followed by leapfrog time steps. This will reduce the error introduced in the unstable first step.**



Schematic of the leapfrog scheme with a small starting step.

Computational Mode: Simple Case

We analyse the oscillation equation with $\omega = 0$:

$$\frac{dU}{dt} = 0$$

The true solution is $U = U^0$, constant.

Computational Mode: Simple Case

We analyse the oscillation equation with $\omega = 0$:

$$\frac{dU}{dt} = 0$$

The true solution is $U = U^0$, constant.

The leapfrog approximation to this is just

$$U^{n+1} = U^{n-1} \qquad \textbf{with the forward step} \qquad U^1 = U^0 \,.$$

Computational Mode: Simple Case

We analyse the oscillation equation with $\omega = 0$:

$$\frac{dU}{dt} = 0$$

The true solution is $U = U^0$, constant.

The leapfrog approximation to this is just

$$U^{n+1} = U^{n-1} \qquad \text{with the forward step} \qquad U^1 = U^0 \,.$$

We consider two particular choices of U^1.

First, suppose the exact value $U^1 = U^0$ is chosen. Then the numerical solution is $U^n = U^0$ for all n, which is exact.

Computational Mode: Simple Case

We analyse the oscillation equation with $\omega = 0$:

$$\frac{dU}{dt} = 0$$

The true solution is $U = U^0$, constant.

The leapfrog approximation to this is just

$$U^{n+1} = U^{n-1} \qquad \textbf{with the forward step} \qquad U^1 = U^0 \,.$$

We consider two particular choices of U^1.

First, suppose the exact value $U^1 = U^0$ is chosen. Then the numerical solution is $U^n = U^0$ for all n, which is exact.

Second, suppose $U^1 = -U^0$. The solution is $U^n = (-1)^n U^0$, which is comprised entirely of the computational mode.

Computational Mode: Simple Case

We analyse the oscillation equation with $\omega = 0$:

$$\frac{dU}{dt} = 0$$

The true solution is $U = U^0$, constant.

The leapfrog approximation to this is just

$$U^{n+1} = U^{n-1} \qquad \text{with the forward step} \qquad U^1 = U^0 \, .$$

We consider two particular choices of U^1.

First, suppose the exact value $U^1 = U^0$ is chosen. Then the numerical solution is $U^n = U^0$ for all n, which is exact.

Second, suppose $U^1 = -U^0$. The solution is $U^n = (-1)^n U^0$, which is comprised entirely of the computational mode.

This illustrates the importance of a careful choice of the computational initial condition.

Robert-Asselin time filter

The second problem is that, for nonlinear equations, the leapfrog scheme has a tendency to increase the amplitude of the computational mode with time.

This can separate the space dependence in a checkerboard fashion between the even and odd time steps.

Robert-Asselin time filter

The second problem is that, for nonlinear equations, the leapfrog scheme has a tendency to increase the amplitude of the computational mode with time.

This can separate the space dependence in a checkerboard fashion between the even and odd time steps.

The problem can be solved by restarting every 50 steps or so, or by applying a Robert–Asselin time filter.

Robert-Asselin time filter

The second problem is that, for nonlinear equations, the leapfrog scheme has a tendency to increase the amplitude of the computational mode with time.

This can separate the space dependence in a checkerboard fashion between the even and odd time steps.

The problem can be solved by restarting every 50 steps or so, or by applying a Robert–Asselin time filter.

After U^{n+1} is obtained, a slight time smoothing is applied to U^n:

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + \overline{U}^{n-1})$$

Robert-Asselin time filter

The second problem is that, for nonlinear equations, the leapfrog scheme has a tendency to increase the amplitude of the computational mode with time.

This can separate the space dependence in a checkerboard fashion between the even and odd time steps.

The problem can be solved by restarting every 50 steps or so, or by applying a Robert–Asselin time filter.

After U^{n+1} is obtained, a slight time smoothing is applied to U^n:

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + \overline{U}^{n-1})$$

Note that the added term is like smoothing in time, an approximation of an ideally time-centered smoother:

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

The smoother

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

reduces the amplitude of different frequencies ν **by a factor** $(1 - 4\gamma \sin^2(\nu \Delta t/2))$**.**

$\star \qquad \star \qquad \star$

The smoother

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

reduces the amplitude of different frequencies ν **by a factor** $(1 - 4\gamma \sin^2(\nu\Delta t/2))$**.**

$\star \qquad \star \qquad \star$

Exercise: Prove this. Hint: write $U^n = U^0 \exp(i\nu n\Delta t)$**.**

$\star \qquad \star \qquad \star$

The smoother

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

reduces the amplitude of different frequencies ν **by a factor** $(1 - 4\gamma \sin^2(\nu \Delta t/2))$**.**

$\star \quad \star \quad \star$

Exercise: Prove this. Hint: write $U^n = U^0 \exp(i\nu n \Delta t)$**.**

$\star \quad \star \quad \star$

The computational mode, whose period is $2\Delta t$**, is reduced by** $(1 - 4\gamma)$ **every time step.**

The smoother

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

reduces the amplitude of different frequencies ν by a factor $(1 - 4\gamma \sin^2(\nu\Delta t/2))$.

$\star \quad \star \quad \star$

Exercise: Prove this. Hint: write $U^n = U^0 \exp(i\nu n\Delta t)$.

$\star \quad \star \quad \star$

The computational mode, whose period is $2\Delta t$, is reduced by $(1 - 4\gamma)$ every time step.

Because the field at $t = (n-1)\Delta t$ is replaced by the already filtered value, the R-A filter introduces a slight distortion of the pure centered filter.

The smoother

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

reduces the amplitude of different frequencies ν by a factor $(1 - 4\gamma\sin^2(\nu\Delta t/2))$.

$\star \qquad \star \qquad \star$

Exercise: Prove this. Hint: write $U^n = U^0\exp(i\nu n\Delta t)$.

$\star \qquad \star \qquad \star$

The computational mode, whose period is $2\Delta t$, is reduced by $(1 - 4\gamma)$ every time step.

Because the field at $t = (n-1)\Delta t$ is replaced by the already filtered value, the R-A filter introduces a slight distortion of the pure centered filter.

A full analysis requires that the combined leapfrog scheme and time filter be analysed together (Asselin, 1972).

The smoother

$$\overline{U}^n = U^n + \gamma(U^{n+1} - 2U^n + U^{n-1})$$

reduces the amplitude of different frequencies ν by a factor $(1 - 4\gamma\sin^2(\nu\Delta t/2))$.

$\star \quad \star \quad \star$

Exercise: **Prove this.** **Hint:** **write $U^n = U^0 \exp(i\nu n\Delta t)$.**

$\star \quad \star \quad \star$

The computational mode, whose period is $2\Delta t$, is reduced by $(1 - 4\gamma)$ every time step.

Because the field at $t = (n-1)\Delta t$ is replaced by the already filtered value, the R-A filter introduces a slight distortion of the pure centered filter.

A full analysis requires that the combined leapfrog scheme and time filter be analysed together (Asselin, 1972).

This filter is widely used with the leapfrog scheme, with γ of the order of 0.01.

Time schemes for $dU/dt = F(U)$

(a) $\frac{U^{n+1}-U^{n-1}}{2\Delta t} = F(U^n)$	Leapfrog (good for hyperbolic equations, unstable for parabolic equations)
(a′) $\frac{U^{n+1}-\overline{U}^{n-1}}{2\Delta t} = F(U^n)$; $\overline{U}^n = U^n + \alpha(U^{n+1} - 2U^n + \overline{U}^{n-1})$	Leapfrog smoothed with the Robert–Asselin time filter; $\alpha \sim 1\%$
(b) $\frac{U^{n+1}-U^n}{\Delta t} = F(U^n)$	Euler (forward, good for diffusive terms, unstable for hyperbolic equations)
(c) $\frac{U^{n+1}-U^n}{\Delta t} = F\left(\frac{U^n+U^{n+1}}{2}\right)$	Crank–Nicholson or centered implicit
(c′) $\frac{U^{n+1}-U^n}{\Delta t} = F\left(\frac{\beta U^n+(1-\beta)U^{n+1}}{2}\right)$; $\beta < 0.5$	Implicit, slightly damping
(d) $\frac{U^{n+1}-U^n}{\Delta t} = F(U^{n+1})$	Fully implicit or backward

Time schemes for $dU/dt = F(U)$

(e) $\frac{U^* - U^n}{\Delta t} = F(U^n)$; $\frac{U^{n+1} - U^n}{\Delta t} = F(U^*)$

Euler-backward or Matsuno: good for damping high frequency waves

(f) $\frac{U^* - U^n}{\Delta t} = F(U^n)$;

$\frac{U^{n+1} - U^n}{\Delta t} = F\left(\frac{U^n + U^*}{2}\right)$

Another predictor–corrector scheme (Heun)

(g) $\frac{U^{n+1} - U^n}{\Delta t} = F\left(\frac{3}{2}U^n - \frac{1}{2}U^{n-1}\right)$

Adams–Bashford (second order in time).

(h) $\frac{U^{n+1/2^*} - U^n}{\Delta t/2} = F(U^n)$;

$\frac{U^{n+1/2^{**}} - U^n}{\Delta t/2} = F(U^{n+1/2^*})$;

$\frac{U^{n+1^*} - U^n}{\Delta t} = F(U^{n+1/2^{**}})$

$$\frac{U^{n+1} - U^n}{\Delta t} = \frac{1}{6}[F(U^n) + 2F(U^{n+1/2^*}) + 2F(U^{n+1/2^{**}}) + F(U^{n+1^*})]$$

Runge–Kutta (fourth order)

Time schemes for $dU/dt = F(U)$

(i)	$a = 0;\ b = 1/\Delta t$ $U^* \leftarrow (aU^* + F(U^n))/b$ $U^n \leftarrow U^n + U^*$ $a \leftarrow a - 1/(N\Delta t);\ b \leftarrow b - 1/(N\Delta t)$	N-times	Lorenz's N-cycle, $N =$ multiple of 4; Nth order
(j)	$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = F_1(U^n) + F_2\left(\frac{U^{n+1} + U^{n-1}}{2}\right)$		Semi-implicit
(k)	$\frac{U^* - U^n}{\Delta t} = F_1(U^n);\ \frac{U^{n+1} - U^*}{\Delta t} = F_2(U^*)$		Fractional steps

For schemes (j) and (k), the right hand side is split into two terms: $F(U) = F_1(U) + F_2(U)$.

Break here

Two Toy Equations

The following two *"toy equations"* serve as prototypes for dissipative and for wave-like process in the atmosphere.

Two Toy Equations

The following two *"toy equations"* serve as prototypes for dissipative and for wave-like process in the atmosphere.

- **The Friction Equation**

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0$$

with solution $U = U^0 \exp(-\kappa t)$ decaying with time.

Two Toy Equations

The following two *"toy equations"* serve as prototypes for dissipative and for wave-like process in the atmosphere.

- **The Friction Equation**

$$\frac{dU}{dt} = -\kappa U \,, \qquad \kappa > 0$$

with solution $U = U^0 \exp(-\kappa t)$ **decaying with time.**

- **The Oscillation Equation**

$$\frac{dU}{dt} = i\omega U \,,$$

with solution $U = U^0 \exp(i\omega t)$**, oscillating in time.**

Two Toy Equations

The following two *"toy equations"* serve as prototypes for dissipative and for wave-like process in the atmosphere.

- **The Friction Equation**

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0$$

with solution $U = U^0 \exp(-\kappa t)$ decaying with time.

- **The Oscillation Equation**

$$\frac{dU}{dt} = i\omega U\,,$$

with solution $U = U^0 \exp(i\omega t)$, oscillating in time.

If we substitute $U = \rho \exp(i\phi)$ in the oscillation equation, then

$$\frac{d\rho}{dt} = 0 \qquad \textbf{and} \qquad \phi = \omega$$

Two Toy Equations

The following two *"toy equations"* serve as prototypes for dissipative and for wave-like process in the atmosphere.

- **The Friction Equation**

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0$$

with solution $U = U^0 \exp(-\kappa t)$ decaying with time.

- **The Oscillation Equation**

$$\frac{dU}{dt} = i\omega U\,,$$

with solution $U = U^0 \exp(i\omega t)$, oscillating in time.

If we substitute $U = \rho \exp(i\phi)$ in the oscillation equation, then

$$\frac{d\rho}{dt} = 0 \qquad \textbf{and} \qquad \phi = \omega$$

Thus the solution U has a constant modulus, and a phase that increases or decreases linearly with time.

The Friction Equation

We consider now the friction equation:

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0 \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0$$

The Friction Equation

We consider now the friction equation:

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0 \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0$$

Of course, the analytical solution is $U(t) = U^0 \exp(-\kappa t)$**, which decays monotonically with time.**

The Friction Equation

We consider now the friction equation:

$$\frac{dU}{dt} = -\kappa U \,, \qquad \kappa > 0 \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0$$

Of course, the analytical solution is $U(t) = U^0 \exp(-\kappa t)$, which decays monotonically with time.

The simplest numerical scheme is the Euler forward method. The time derivative in the differential equation is approximated by a forward difference:

$$\frac{U^{n+1} - U^n}{\Delta t} = -\kappa U^n \,.$$

The Friction Equation

We consider now the friction equation:

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0 \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0$$

Of course, the analytical solution is $U(t) = U^0 \exp(-\kappa t)$, which decays monotonically with time.

The simplest numerical scheme is the Euler forward method. The time derivative in the differential equation is approximated by a forward difference:

$$\frac{U^{n+1} - U^n}{\Delta t} = -\kappa U^n\,.$$

It is easy to find the solution of this difference equation: $U^n = U^0(1 - \kappa\Delta t)^n$.

The Friction Equation

We consider now the friction equation:

$$\frac{dU}{dt} = -\kappa U\,, \qquad \kappa > 0 \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0$$

Of course, the analytical solution is $U(t) = U^0 \exp(-\kappa t)$, which decays monotonically with time.

The simplest numerical scheme is the Euler forward method. The time derivative in the differential equation is approximated by a forward difference:

$$\frac{U^{n+1} - U^n}{\Delta t} = -\kappa U^n\,.$$

It is easy to find the solution of this difference equation: $U^n = U^0(1 - \kappa\Delta t)^n$.

This solution decays monotonically in time provided

$$\kappa\Delta t < 1$$

Again, the solution decays monotonically in time provided

$$\kappa \Delta t < 1$$

Again, the solution decays monotonically in time provided

$$\kappa\Delta t < 1$$

The Euler scheme is *stable* for this friction equation.

Again, the solution decays monotonically in time provided

$$\kappa \Delta t < 1$$

The Euler scheme is *stable* for this friction equation.

However, the scheme is only first-order accurate: for a fixed time $\tau = n\Delta t$, the error is

$$\varepsilon(\tau) = |U^0(1-\kappa\Delta t)^n - U^0 \exp(-\kappa n \Delta t)| = \tfrac{1}{2}U^0 \tau \kappa^2 \Delta t + O(\Delta t^2)\,.$$

$\star \qquad \star \qquad \star$

Again, the solution decays monotonically in time provided

$$\kappa\Delta t < 1$$

The Euler scheme is *stable* for this friction equation.

However, the scheme is only first-order accurate: for a fixed time $\tau = n\Delta t$, the error is

$$\varepsilon(\tau) = |U^0(1-\kappa\Delta t)^n - U^0\exp(-\kappa n\Delta t)| = \tfrac{1}{2}U^0\tau\kappa^2\Delta t + O(\Delta t^2)\,.$$

$\star \qquad \star \qquad \star$

Exercise: Prove this.

$\star \qquad \star \qquad \star$

Again, the solution decays monotonically in time provided

$$\kappa \Delta t < 1$$

The Euler scheme is *stable* for this friction equation.

However, the scheme is only first-order accurate: for a fixed time $\tau = n\Delta t$, the error is

$$\varepsilon(\tau) = |U^0(1 - \kappa\Delta t)^n - U^0 \exp(-\kappa n \Delta t)| = \tfrac{1}{2} U^0 \tau \kappa^2 \Delta t + O(\Delta t^2)\,.$$

$\star \qquad \star \qquad \star$

Exercise: Prove this.

$\star \qquad \star \qquad \star$

We might attempt to obtain a more accurate solution by using a centered difference for the time derivative, as in the leapfrog scheme.

Let us look at this possibility now.

The leapfrog scheme for the Friction Equation is:

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = -\kappa U^n \,.$$

The leapfrog scheme for the Friction Equation is:

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = -\kappa U^n \,.$$

A solution in the form of a geometric progression, $U^n = U^0 \rho^n$ **for some constant** ρ**, will decrease monotonically provided** $0 < \rho < 1$**.**

The leapfrog scheme for the Friction Equation is:

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = -\kappa U^n \, .$$

A solution in the form of a geometric progression, $U^n = U^0 \rho^n$ **for some constant** ρ**, will decrease monotonically provided** $0 < \rho < 1$**.**

This is the condition for the character of the finite difference solution to resemble that of the continuous equation.

The leapfrog scheme for the Friction Equation is:

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = -\kappa U^n \,.$$

A solution in the form of a geometric progression, $U^n = U^0 \rho^n$ **for some constant** ρ**, will decrease monotonically provided** $0 < \rho < 1$**.**

This is the condition for the character of the finite difference solution to resemble that of the continuous equation.

Substituting this solution into the FDE, there are two possibilities:

$$\rho_+ = -\kappa\Delta t + \sqrt{1 + \kappa^2 \Delta t^2} \qquad \textbf{and} \qquad \rho_- = -\kappa\Delta t - \sqrt{1 + \kappa^2 \Delta t^2} \,.$$

The leapfrog scheme for the Friction Equation is:

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = -\kappa U^n \, .$$

A solution in the form of a geometric progression, $U^n = U^0 \rho^n$ **for some constant** ρ**, will decrease monotonically provided** $0 < \rho < 1$**.**

This is the condition for the character of the finite difference solution to resemble that of the continuous equation.

Substituting this solution into the FDE, there are two possibilities:

$$\rho_+ = -\kappa\Delta t + \sqrt{1 + \kappa^2 \Delta t^2} \qquad \textbf{and} \qquad \rho_- = -\kappa\Delta t - \sqrt{1 + \kappa^2 \Delta t^2} \, .$$

It is easy to see that $|\rho_+| < 1$ **for all** Δt **so that a decaying solution is obtained.**

$\star \qquad \star \qquad \star$

The leapfrog scheme for the Friction Equation is:

$$\frac{U^{n+1} - U^{n-1}}{2\Delta t} = -\kappa U^n \,.$$

A solution in the form of a geometric progression, $U^n = U^0 \rho^n$ **for some constant** ρ**, will decrease monotonically provided** $0 < \rho < 1$**.**

This is the condition for the character of the finite difference solution to resemble that of the continuous equation.

Substituting this solution into the FDE, there are two possibilities:

$$\rho_+ = -\kappa\Delta t + \sqrt{1 + \kappa^2 \Delta t^2} \qquad \textbf{and} \qquad \rho_- = -\kappa\Delta t - \sqrt{1 + \kappa^2 \Delta t^2} \,.$$

It is easy to see that $|\rho_+| < 1$ **for all** Δt **so that a decaying solution is obtained.**

$\star \qquad \star \qquad \star$

Exercise: **Prove this. Hint: if** $y = -x + \sqrt{1 + x^2}$**, then** $y(0) = 1$**;** $y > 0$**;** $y' < 0$ **so** $0 < y \leq 1$**.**

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ **for all** Δt**, so this solution grows without limit with** n**.**

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ for all Δt, so this solution grows without limit with n.

The first solution (ρ_+) gives the *physical mode*, which is similar in character to the solution of the differential equation.

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ for all Δt, so this solution grows without limit with n.

The first solution (ρ_+) gives the *physical mode*, which is similar in character to the solution of the differential equation.

The second solution (ρ_-) is called the *computational mode*. It appears as a result of replacing a first-order derivative by a second-order difference.

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ **for all** Δt**, so this solution grows without limit with** n**.**

The first solution (ρ_+**) gives the** ***physical mode*****, which is similar in character to the solution of the differential equation.**

The second solution (ρ_-**) is called the** ***computational mode*****. It appears as a result of replacing a first-order derivative by a second-order difference.**

The computational mode has no counterpart in the physical equation; it is a spurious numerical artifact.

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1 + \kappa^2\Delta t^2}] > 1$ **for all** Δt**, so this solution grows without limit with** n**.**

The first solution (ρ_+**) gives the** ***physical mode*****, which is similar in character to the solution of the differential equation.**

The second solution (ρ_-**) is called the** ***computational mode*****. It appears as a result of replacing a first-order derivative by a second-order difference.**

The computational mode has no counterpart in the physical equation; it is a spurious numerical artifact.

Since $\rho_- < 0$**, its sign alternates each time-step and, since it grows with time, it leads to numerical instability.**

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ for all Δt, so this solution grows without limit with n.

The first solution (ρ_+) gives the *physical mode*, which is similar in character to the solution of the differential equation.

The second solution (ρ_-) is called the *computational mode*. It appears as a result of replacing a first-order derivative by a second-order difference.

The computational mode has no counterpart in the physical equation; it is a spurious numerical artifact.

Since $\rho_- < 0$, its sign alternates each time-step and, since it grows with time, it leads to numerical instability.

Thus, despite its second-order accuracy, the leapfrog scheme produces unacceptable errors for the friction equation.

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ for all Δt, so this solution grows without limit with n.

The first solution (ρ_+) gives the *physical mode*, which is similar in character to the solution of the differential equation.

The second solution (ρ_-) is called the *computational mode*. It appears as a result of replacing a first-order derivative by a second-order difference.

The computational mode has no counterpart in the physical equation; it is a spurious numerical artifact.

Since $\rho_- < 0$, its sign alternates each time-step and, since it grows with time, it leads to numerical instability.

Thus, despite its second-order accuracy, the leapfrog scheme produces unacceptable errors for the friction equation.

The instability of the leapfrog scheme would appear to make it unsuitable for use.

However, $|\rho_-| = [\kappa\Delta t + \sqrt{1+\kappa^2\Delta t^2}] > 1$ for all Δt, so this solution grows without limit with n.

The first solution (ρ_+) gives the *physical mode*, which is similar in character to the solution of the differential equation.

The second solution (ρ_-) is called the *computational mode*. It appears as a result of replacing a first-order derivative by a second-order difference.

The computational mode has no counterpart in the physical equation; it is a spurious numerical artifact.

Since $\rho_- < 0$, its sign alternates each time-step and, since it grows with time, it leads to numerical instability.

Thus, despite its second-order accuracy, the leapfrog scheme produces unacceptable errors for the friction equation.

The instability of the leapfrog scheme would appear to make it unsuitable for use.

But we know better: read on!

The Oscillation Equation

If we apply the leapfrog scheme to the *oscillation equation*

$$\frac{dU}{dt} = i\omega U\,, \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0\,,$$

a markedly different result is obtained.

The Oscillation Equation

If we apply the leapfrog scheme to the ***oscillation equation***

$$\frac{dU}{dt} = i\omega U\,, \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0\,,$$

a markedly different result is obtained.

The analytical solution is $U(t) = U^0 \exp(i\omega t)$**.**

The Oscillation Equation

If we apply the leapfrog scheme to the *oscillation equation*

$$\frac{dU}{dt} = i\omega U\,, \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0\,,$$

a markedly different result is obtained.

The analytical solution is $U(t) = U^0 \exp(i\omega t)$.

Using the leapfrog scheme and again seeking a solution $U^n = U^0\rho^n$ for constant ρ, there are again two possibilities:

$$\rho_{\pm} = i\omega\Delta t \pm \sqrt{1 - \omega^2\Delta t^2}\,.$$

The Oscillation Equation

If we apply the leapfrog scheme to the *oscillation equation*

$$\frac{dU}{dt} = i\omega U\,, \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0\,,$$

a markedly different result is obtained.

The analytical solution is $U(t) = U^0 \exp(i\omega t)$.

Using the leapfrog scheme and again seeking a solution $U^n = U^0 \rho^n$ for constant ρ, there are again two possibilities:

$$\rho_\pm = i\omega\Delta t \pm \sqrt{1 - \omega^2 \Delta t^2}\,.$$

For $|\omega\Delta t| \le 1$, it is clear that $|\rho_\pm| = 1$ so that two oscillating solutions are obtained.

The Oscillation Equation

If we apply the leapfrog scheme to the *oscillation equation*

$$\frac{dU}{dt} = i\omega U \,, \qquad \textbf{with} \quad U = U^0 \quad \textbf{at} \quad t = 0 \,,$$

a markedly different result is obtained.

The analytical solution is $U(t) = U^0 \exp(i\omega t)$.

Using the leapfrog scheme and again seeking a solution $U^n = U^0 \rho^n$ for constant ρ, there are again two possibilities:

$$\rho_\pm = i\omega\Delta t \pm \sqrt{1 - \omega^2 \Delta t^2} \,.$$

For $|\omega\Delta t| \le 1$, it is clear that $|\rho_\pm| = 1$ so that two oscillating solutions are obtained.

For small $\omega\Delta t$ we have $\rho_+ \approx +1$ and $\rho_- \approx -1$.

Again, for small $\omega\Delta t$ we have $\rho_+ \approx +1$ and $\rho_- \approx -1$.

Again, for small $\omega\Delta t$ we have $\rho_+ \approx +1$ and $\rho_- \approx -1$.

The former approximates the analytical solution. The latter is the computational mode, which alternates in sign from step to step.

Again, for small $\omega\Delta t$ we have $\rho_+ \approx +1$ and $\rho_- \approx -1$.

The former approximates the analytical solution. The latter is the computational mode, which alternates in sign from step to step.

For $|\omega\Delta t| > 1$, either $|\rho_+| > 1$ or $|\rho_-| > 1$. Thus the leapfrog scheme is unstable in this case.

Again, for small $\omega\Delta t$ we have $\rho_+ \approx +1$ and $\rho_- \approx -1$.

The former approximates the analytical solution. The latter is the computational mode, which alternates in sign from step to step.

For $|\omega\Delta t| > 1$, either $|\rho_+| > 1$ or $|\rho_-| > 1$. Thus the leapfrog scheme is unstable in this case.

The leapfrog scheme is stable for the oscillation equation, provided

$$|\omega\Delta t| < 1 \,.$$

Again, for small $\omega\Delta t$ we have $\rho_+ \approx +1$ and $\rho_- \approx -1$.

The former approximates the analytical solution. The latter is the computational mode, which alternates in sign from step to step.

For $|\omega\Delta t| > 1$, either $|\rho_+| > 1$ or $|\rho_-| > 1$. Thus the leapfrog scheme is unstable in this case.

The leapfrog scheme is stable for the oscillation equation, provided

$$|\omega\Delta t| < 1 \,.$$

	Friction Equation	Oscillation Equation
Euler Scheme	Conditionally Stable	UNSTABLE
Leapfrog Scheme	UNSTABLE	Conditionally Stable

Matlab Exercises

- **Write a MATLAB program to solve the oscillation equation**

$$\frac{dU}{dt} = iU\,, \quad U^0 = 1 \qquad\qquad (\omega = 1)$$

the analytical solution of which is $U(t) = \exp(it)$**, using**

– **the Euler forward method**

– **the leapfrog method**

Draw conclusions about the stability of the two schemes.

Matlab Exercises

- **Write a MATLAB program to solve the oscillation equation**

$$\frac{dU}{dt} = iU\,, \quad U^0 = 1 \qquad\qquad (\omega = 1)$$

the analytical solution of which is $U(t) = \exp(it)$**, using**

- **the Euler forward method**
- **the leapfrog method**

Draw conclusions about the stability of the two schemes.

- **Write a MATLAB program to solve the friction equation**

$$\frac{dU}{dt} = -U\,, \quad U^0 = 1 \qquad\qquad (\kappa = 1)$$

the analytical solution of which is $U(t) = \exp(-t)$**, using**

- **the Euler forward method**
- **the leapfrog method**

Draw conclusions about the stability of the two schemes.

Conclusion of §3.2.3