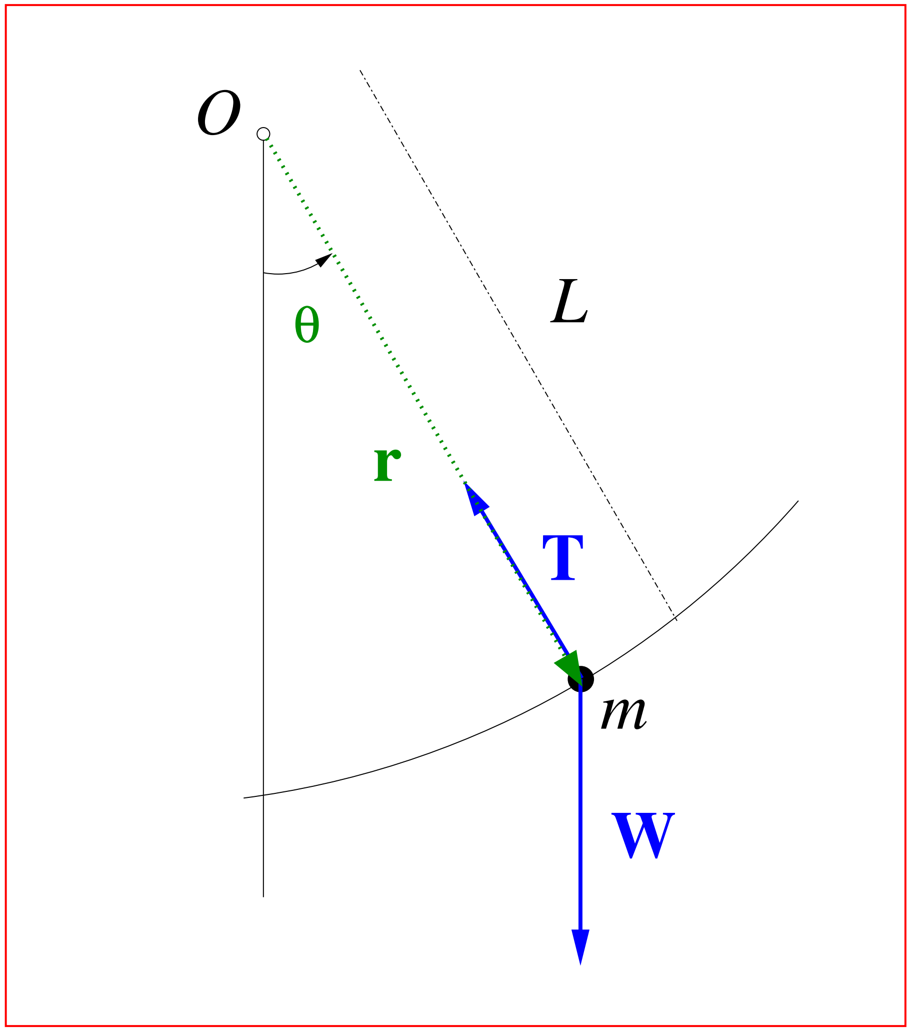
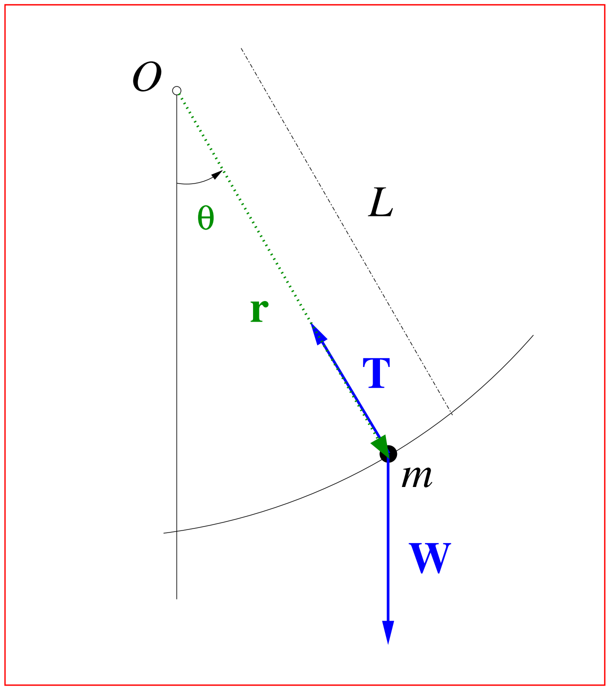
1. PHYSICAL & MATHEMATICAL FORMULATION



1.1 Derivation of the equation of motion



- Consider idealized pendulum:
  - Mass of bob, m
  - Infinitely rigid, massless pendulum rod, length L
  - Forces: Weight: $\mathbf{W}$, tension in rod: $\mathbf{T}$
  - No friction at pivot point O (origin of coordinate system)
  - Total mechanical energy (KE + PE) strictly conserved

- Consider bob's position vector, $\mathbf{r}(t)$
- $\mathbf{r}(t)$ makes an angle $\theta(t)$ with vertical, measured counter-clockwise
- The pendulum moves through two-dimensional space, but because the rod length, L, is fixed, its motion is completely determined by a knowledge of $\theta(t)$

- I.e. viewing $\theta(t)$ as a "generalized coordinate", the problem is effectively *one dimensional*, and we can use the laws of rotational dynamics, about the origin, O, to determine the *equation of motion* for the system

- Recall that the rotational analogues of mass, velocity and acceleration for translational motion are the moment of inertia, I, angular velocity, $\omega(t)$,

$$\omega(t) \equiv \frac{d\theta(t)}{dt}$$

  and angular acceleration $\alpha(t)$,

$$\alpha(t) \equiv \frac{d^2\theta(t)}{dt^2} = \frac{d\omega(t)}{dt}$$

- The rotational version of Newton's second law (torques) then tells us

$$I\alpha\,\mathbf{k} = \mathbf{r} \times \sum \mathbf{F}$$

  where $\mathbf{k}$ is the outwards-pointing unit vector normal to the plane of motion

- For the pendulum bob, we have $I = mL^2$. Noting that $\mathbf{r}$ and $\mathbf{T}$ are parallel, and that $\mathbf{r} \times \mathbf{W}$ points inwards, the torque equation gives us

$$mL^2\alpha\,\mathbf{k} = \mathbf{r} \times \mathbf{W} + \mathbf{r} \times \mathbf{T} = -Lmg\sin\theta\,\mathbf{k} + \mathbf{0}$$

so

$$\alpha = -\frac{g}{L}\sin\theta$$

- The equation of motion for the pendulum, written in the form of a second-order-in-time differential equation, is therefore

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta \qquad 0 \le t \le t_{\max} \tag{1}$$

where we have emphasized that we are interested in modeling the behaviour of the pendulum over some finite time interval, $0 \le t \le t_{\max}$

- Note that the mass of the pendulum bob does not appear in this equation (it “drops out” of the problem)

- This last equation governs the dynamics of the pendulum; it is not sufficient, however, to tell us precisely what the pendulum will do any time that is set in motion

- We know that the motion will be different if we release the pendulum from different locations (value of θ) or give it more or less of a push (value of ω) as we let it go

- That is, we must give *initial conditions* for the motion, which in this case are

$$\begin{aligned} \theta(0) &= \theta_0 \qquad \text{Initial angular displacement of pendulum} \\ \omega(0) &= \omega_0 \qquad \text{Initial angular velocity of pendulum} \end{aligned}$$

- Physically, as just mentioned, $\theta(0)$ and $\omega(0)$, describe from where and with what angular speed the pendulum is released; at least in an idealized world these are *arbitrary*

- Mathematically, the need to give θ_0 and ω_0 reflects the fact that we are solving a second-order-in-time differential equation, which has a unique solution if and only if we give *two* initial conditions

1.2 Non-dimensionalization

- Although this may seem strange, it is convenient to choose a system of units such that $g = 1$, $L = 1$; this makes the equation that we need to solve simpler

- Let's see how this would work for a specific set of initial units and length of rod: In particular, let's assume that we are working in *MKS* units so that $g = 9.8\,\mathrm{m/s^2}$, and that our pendulum has $L = 4.9\,\mathrm{m}$

- Then we first define a new unit of length that we will call a *rod* (not to be confused with the old Imperial unit having the same name!), so that

$$1 \text{ rod} \equiv 4.9 \text{ m}$$

- We then have

$$L = 1 \text{ rod}$$

- Now define a new unit of time, called the *tick*, so that

$$1 \text{ tick} \equiv \frac{1}{\sqrt{2}} \text{ s}$$

- This implies

$$1 \text{ s} \equiv \sqrt{2} \text{ tick}$$

- Now we have

$$g = 9.8 \, \frac{\text{m}}{\text{s}^2} = \frac{9.8 \text{ m}}{\text{s}^2} = \frac{2 \text{ rod}}{\text{s}^2} = \frac{2 \text{ rod}}{(\sqrt{2} \text{ tick})^2} = \frac{2 \text{ rod}}{2 \text{ tick}^2} = 1 \frac{\text{rod}}{\text{tick}^2}$$

- Therefore, in our new set of units we have (suppressing the units themselves)

$$L = g = 1$$

- Adopting these units, the simplified version of 1 that we want to solve is

$$\frac{d^2\theta}{dt^2} = -\sin\theta \qquad 0 \le t \le t_{\max} \tag{2}$$

with initial conditions

$$\theta(0) = \theta_0 \tag{3}$$

$$\omega(0) = \omega_0 \tag{4}$$

- Note: equation (2)

$$\frac{d^2\theta}{dt^2} = -\sin\theta \qquad 0 \le t \le t_{\max}$$

  is a *nonlinear* ordinary differential equation (ODE), since $\sin(\theta(t))$ is a nonlinear function of $\theta(t)$.

- Closed form (analytic) solution is possible, but is quite complex.

- **Numerical solution is no more difficult in principle/practice than it is for the linear case!**

- Note: No reason to expect $\theta(t)$ to be restricted in value, e.g. $-\pi \le \theta(t) \le \pi$; again, in principle, possible range of $\theta(t)$ is

$$-\infty < \theta(t) < \infty$$

  where we now have to interpret "angle from vertical" as "angle as measured from equilibrium state of pendulum", i.e. with $\theta = 0$ **and** $\omega = 0$

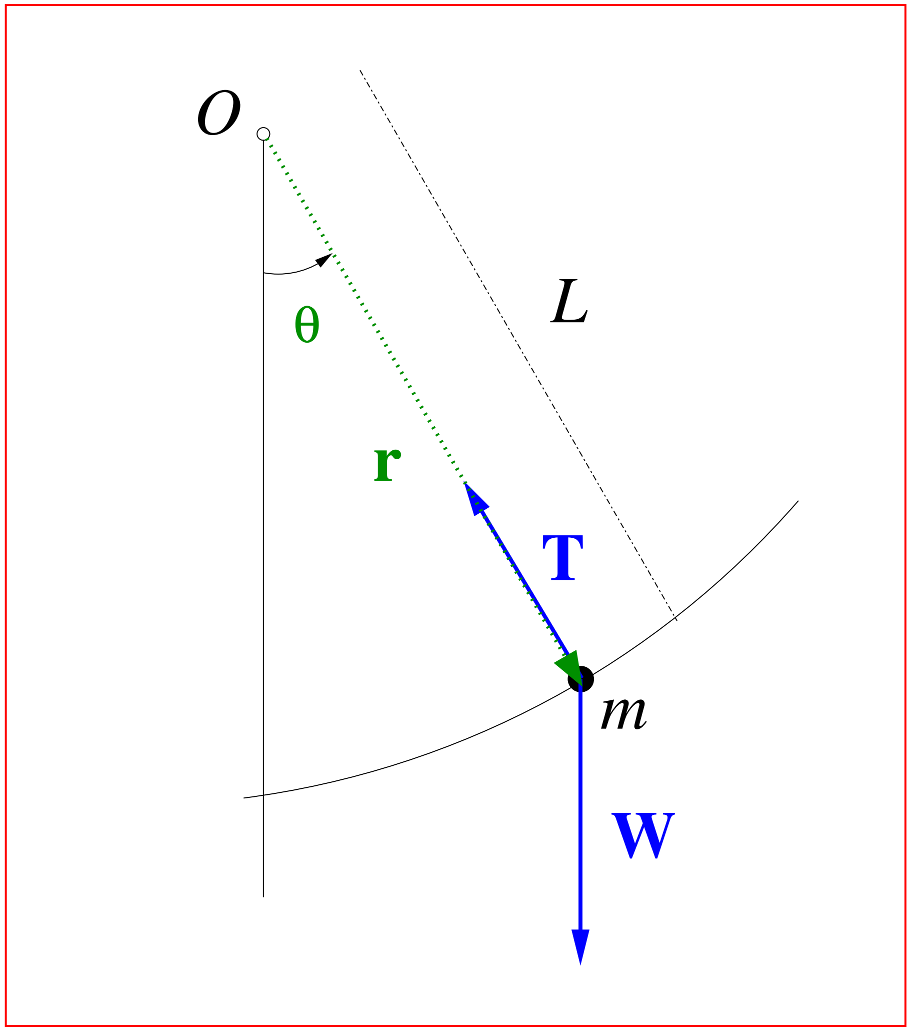
1.3 Linear limit

- Assume that the angular displacement $\theta(t)$ is very small, $\theta(t) \ll 1$
- Then
$$\sin(\theta) \approx \theta$$
and (2) becomes
$$\frac{d^2\theta}{dt^2} = -\theta \qquad 0 \le t \le t_{\max} \tag{5}$$
- which has the general solution
$$\theta(t) = A\sin(t + \delta)$$
where the constants A and δ are determined from the initial conditions (3) and (4).
- This is the usual "pendulum equation/solution" with which you should be familiar, except that here we are working with the special system of units such that $L = g = 1$

- *Key fact:* In linear case, oscillation frequency $\Omega = \sqrt{g/L} = 1$ with our choice of units (i.e. $\theta(t) = A\sin(\Omega t + \delta)$ with $\Omega = 1$), is *independent* of amplitude, A, of oscillation, or equivalently, independent of the initial conditions (2) and (3)

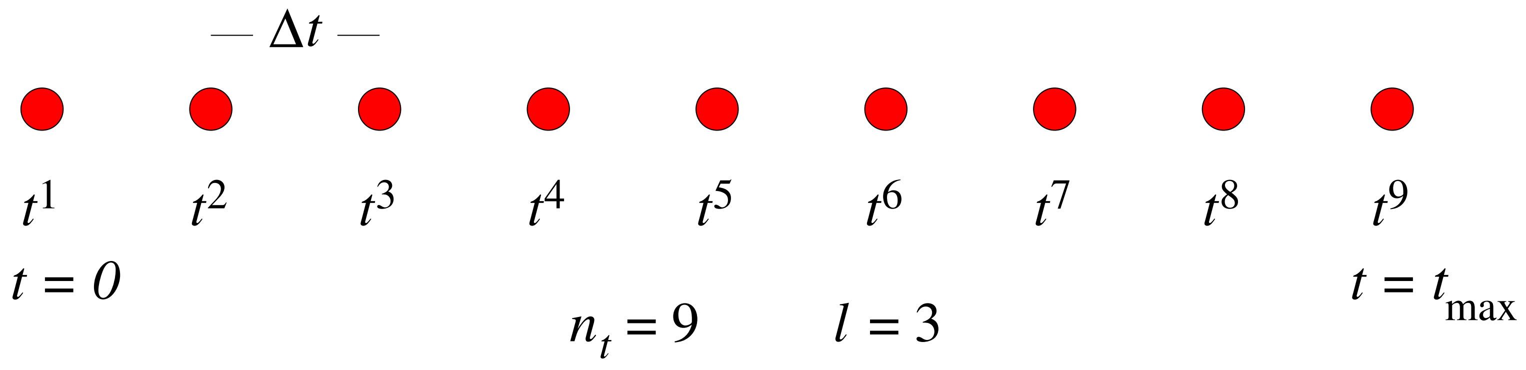
- This is *not* the case for the nonlinear pendulum!

2. SOLUTION VIA FINITE DIFFERENCE APPROXIMATION



2.1 Discretization: Step 1—Finite difference grid

- Continuum domain is $0 \le t \le t_{\max}$, replace with grid: example



- Grid characterized by number of grid points, n_t, and grid spacing, Δt: as discussed previously, will specify these implicitly in terms of integer-valued level parameter, ℓ

$$n_t = 2^\ell + 1$$
$$\Delta t = \frac{t_{\max}}{n_t - 1} = 2^{-\ell} t_{\max}$$
$$t^n = (n-1)\Delta t, \quad n = 1, 2, \ldots, n_t$$

2.2 Discretization: Step 2—Derivation of FDA

- Continuum equations $\rightarrow$ discrete equations
- Adopt usual finite difference notation for grid functions

$$\theta^n \equiv \theta(t^n) \equiv \theta((n-1)\Delta t))$$

(using a superscript, since it is conventional for a time-index, and subsequently being very careful not to misinterpret, θ^2, with θ-squared, etc.)

- Have one derivative to replace, use $O(\Delta t^2)$ accurate approximation derived earlier in class

$$\left.\frac{d^2\theta}{dt^2}\right|_{t=t^n} \rightarrow \frac{\theta^{n+1} - 2\theta^n + \theta^{n-1}}{\Delta t^2} \tag{6}$$

- **Important:** We view this formula as being applied ("centred") at the grid point t^n; we thus also make the replacement

$$\sin(\theta) \rightarrow \sin(\theta^n)$$

- Substituting the approximation (6) of the second derivative in

$$\frac{d^2\theta}{dt^2} = -\sin\theta \qquad 0 \le t \le t_{\max}$$

we get our desired FDA:

$$\frac{\theta^{n+1} - 2\theta^n + \theta^{n-1}}{\Delta t^2} = -\sin\theta^n \qquad n+1 = 3, 4 \ldots n_t \tag{7}$$

- Note that this formula expresses what values $n+1$ takes on. We write it in this way since we are going to view the equation as *defining* what the value of θ^{n+1} (called the "advanced" value) is in terms of θ^n and θ^{n-1} (called the "retarded" values)

2.3 Discretization: Step 3—Solution of FDA

- As just stated, we now view equation (7)

$$\frac{\theta^{n+1} - 2\theta^n + \theta^{n-1}}{\Delta t^2} = -\sin\theta^n \qquad n+1 = 3, 4, \ldots n_t$$

as an equation for θ^{n+1}, assuming that θ^n and θ^{n-1} are known

- Solving for θ^{n+1} we have

$$\theta^{n+1} = 2\theta^n - \theta^{n-1} - \Delta t^2 \sin\theta^n \qquad n+1 = 3, 4, \ldots n_t \tag{8}$$

- Since θ^{-2}, θ^{-1} and θ^0 are not defined, we can *not* use (8) to determine $\theta^1 = \theta(t=0)$ and $\theta^2 = \theta(t = \Delta t)$

- The first discrete time at which we can use (8) is $t^{n+1} = t^3$, and this is precisely why $n+1$ starts at 3 in the formula

- We thus need another way to determine θ^1 and θ^2

- θ^1 is given by the initial condition (3) (i.e. the angle of release)

$$\theta^1 = \theta(0) = \theta_0$$

- Determining θ^2 is a bit more involved. We state without proof that in order for the overall solution to be $O(\Delta t^2)$, we must determine $\theta^2 = \theta(\Delta t)$ up to and including terms of $O(\Delta t^2)$ (i.e. the leading order error term in $\theta(\Delta t)$ must be $O(\Delta t^3)$)

- Proceed via Taylor series expansion, and use the initial conditions $\theta^1 = \theta(0) = \theta_0$ and $\omega^1 = \omega(0) = \omega_0$ (we also freely choose the initial angular velocity)

$$\begin{aligned} \theta(\Delta t) &= \theta(0) + \Delta t \frac{d\theta}{dt}(0) + \frac{1}{2}\Delta t^2 \frac{d^2\theta}{dt^2}(0) + O(\Delta t^3) \\ &\approx \theta_0 + \Delta t\, \omega_0 + \frac{1}{2}\Delta t^2 \frac{d^2\theta}{dt^2} \end{aligned}$$

- We now the use the equation of motion (2) to eliminate $d^2\theta/dt^2$; i.e. $d^2\theta/dt^2 = -\sin\theta$, so we have

$$\theta(\Delta t) \approx \theta_0 + \Delta t\,\omega_0 - \frac{1}{2}\Delta t^2 \sin\theta_0$$

- Assembling results we have our complete and final set of FD equations

$$\theta^{n+1} = 2\theta^n - \theta^{n-1} - \Delta t^2 \sin\theta^n \qquad n+1 = 3, 4, \ldots n_t \tag{9}$$

$$\theta^1 = \theta_0 \tag{10}$$

$$\theta^2 = \theta_0 + \Delta t \omega_0 - \frac{1}{2}\Delta t^2 \sin\theta_0 \tag{11}$$

- Note that we have a total of n_t equations for the n_t unknowns,

$$\theta^n, \quad n = 1, 2, \ldots, n_t$$

which, of course, is necessary for us to be able to compute *all* of the θ^n

2.4 Convergence—Expected error behaviour (Extremely Important!!)

- We want to consider the behaviour of our finite difference (numerical) solution as $\Delta t \to 0$
- We first note that the following applies to essentially *any* differential equation that is solved with FDAs, not just the pendulum problem
- **Assumption:** (following L.F. Richardson, 1909) Let $u_\star(t)$ be the exact (continuum) solution of (2-4) Then the error, $e(t^n)$, in the numerical solution, $u(t^n)$

  $$e(t^n) \equiv u_\star(t^n) - u(t^n)$$

  takes the form

  $$\lim_{\Delta t \to 0} e(t^n) \equiv u_\star(t^n) - u(t^n) = \Delta t^2 e_2(t^n) + O(\Delta t^4) \qquad (12)$$

  so the leading order (dominant) term in the error is

  $$\Delta t^2 e_2(t^n)$$

- To emphasize, the leading order error term is *proportional to* Δt^2, so if, for example we reduce Δt in our calculations by a factor of 2, we should expect the error in the numerical solution to go down by a factor of 4.

- **Note:** e_2 is a *function*, just as the exact solution, u is; it is not something "random", as would be the case if we were analyzing the error in experimental data!

- You might ask: *Why should we expect* equation (12)

$$\lim_{\Delta t \to 0} e(t^n) \equiv u_\star(t^n) - u(t^n) = \Delta t^2 e_2(t^n) + O(\Delta t^4)$$

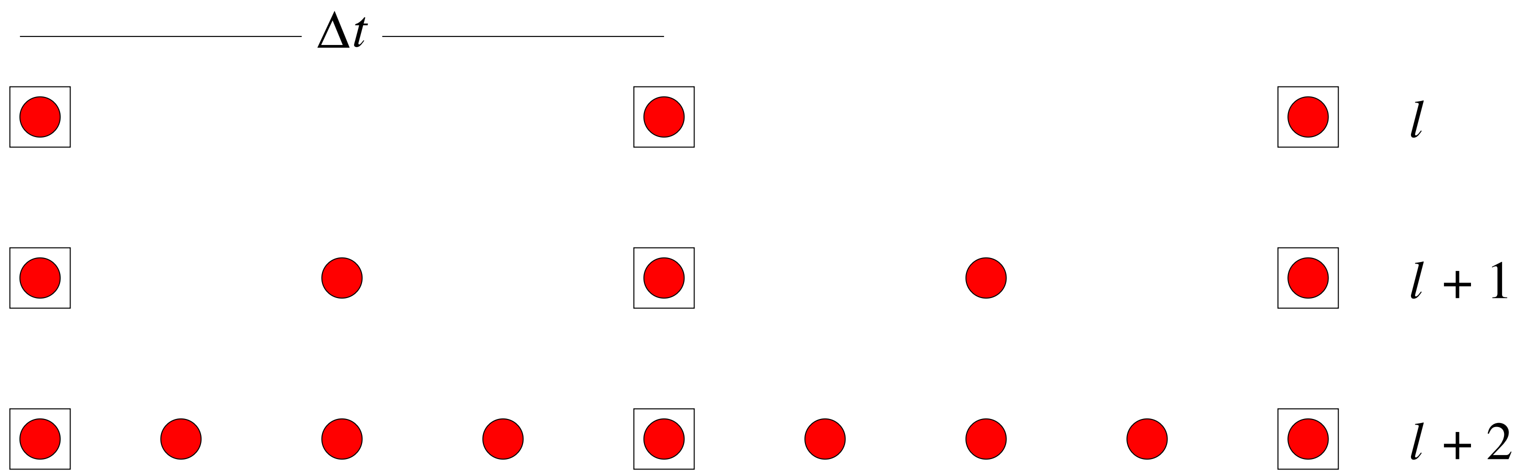
  to hold?

- This is a rather deep question and the full answer is beyond the scope of this course

- **However, the crucial observation for us is it can be VERIFIED empirically!**

2.5 Convergence testing (Also extremely important!!)

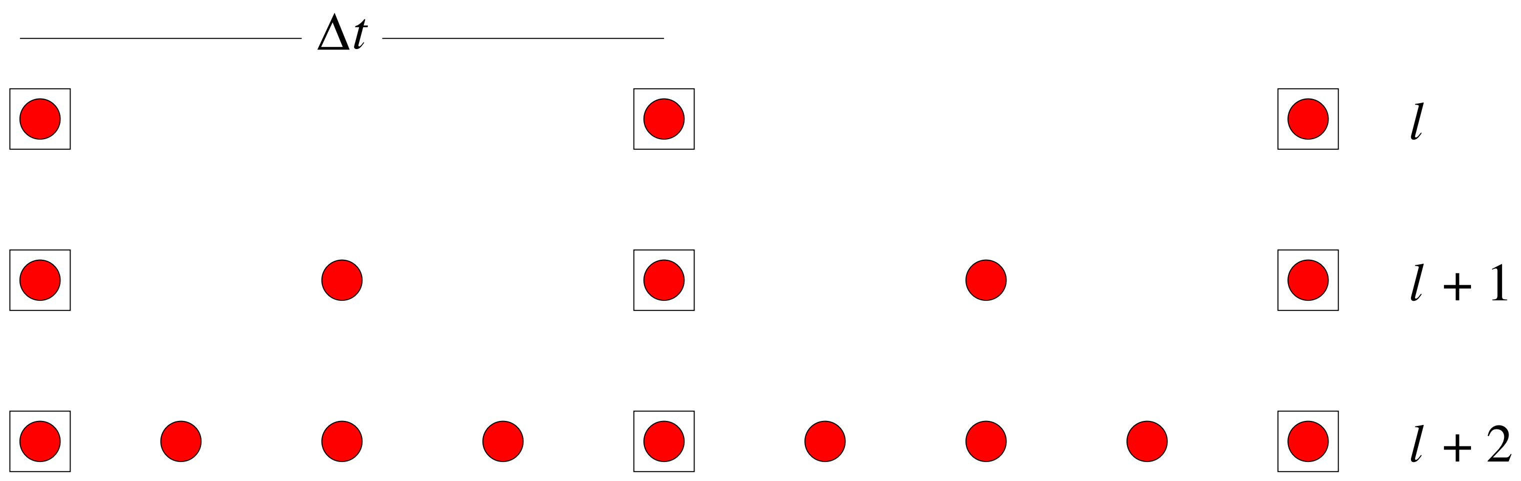
- Consider a sequence of three finite difference grids with spacings Δt, $\Delta t/2$ and $\Delta t/4$ (three levels of discretization)



- The key idea is to perform the same calculation, i.e. for pendulum problem, using the same initial values, θ_0 and ω_0, on the different grids

- Solutions will vary from level to level; expect more accuracy for smaller Δt, i.e. for increasing level, ℓ

- Note that the grid points in the squares are common to the three computations



- Define

$$u_\ell^n \equiv \text{soln solution at level } \ \ell$$
$$u_{\ell+1}^n \equiv \text{soln solution at level } \ \ell+1$$
$$u_{\ell+2}^n \equiv \text{soln solution at level } \ \ell+2$$

- As just described, as $\Delta t \to 0$ we expect

$$\text{Solution error } \ \equiv e(t^n) \equiv u_\star(t^n) - u(t^n) = \Delta t^2 e_2(t^n) + O(\Delta t^4)$$

or

$$u(t^n) \approx u_\star(t^n) - \Delta t^2 e_2(t^n)$$

- Repeating the last equation

$$u(t^n) \approx u_\star(t^n) - \Delta t^2 e_2(t^n)$$

- Now, this will hold for *any* sufficiently small Δt
- So, in particular, for the calculations on the level ℓ, $\ell+1$ and $\ell+2$ grids we have

$$u_\ell(t^n) \approx u_\star^n(t^n) - (\Delta t_\ell)^2 e_2(t^n)$$

$$u_{\ell+1}(t^n) \approx u_\star^n(t^n) - (\Delta t_{\ell+1})^2 e_2(t^n)$$

$$u_{\ell+2}(t^n) \approx u_\star^n(t^n) - (\Delta t_{\ell+2})^2 e_2(t^n)$$

where t^n is the common set of discrete times t_ℓ^n (the times on the coarsest grid)

- Now consider subtracting solutions on adjacent levels and remember that $\Delta t_{\ell+1} = \Delta t_\ell / 2$; then

$$\begin{aligned}
u_\ell(t^n) - u_{\ell+1}(t^n) &\approx -\left((\Delta t_\ell)^2 - (\Delta t_{\ell+1})^2\right) e_2(t^n) \\
&= \left((\Delta t_\ell)^2 - \frac{1}{4}(\Delta t_\ell)^2\right) e_2(t^n) = -\frac{3}{4}\Delta t_\ell^2 e_2(t^n) \\
u_{\ell+1}(t^n) - u_{\ell+2}(t^n) &\approx -\left((\Delta t_{\ell+1})^2 - (\Delta t_{\ell+2})^2\right) e_2(t^n) \\
&= -\frac{3}{4}(\Delta t_{\ell+1})^2 e_2(t^n) = -\frac{3}{16}(\Delta t_\ell)^2 e_2(t^n)
\end{aligned}$$

- This development leads to several observations that are crucial when we are analyzing the error in finite difference approximations

- **Observe:**
  - **First**, simply subtracting 2 solns computed on 2 different levels gives direct estimate of solution error (very general result!)
  - To see this, note that we have

$$e(t^n) = \Delta t^2 e_2(t^n) + \cdots$$

and we have just shown

$$u_\ell(t^n) - u_{\ell+1}(t^n) \approx -\frac{3}{4}(\Delta t_\ell)^2 e_2(t^n)$$

so

$$-\frac{4}{3}\left(u_\ell(t^n) - u_{\ell+1}(t^n)\right) \approx e(t^n)$$

- **Second**, if we consider the ratio

$$\frac{u_\ell(t^n) - u_{\ell+1}(t^n)}{u_{\ell+1}(t^n) - u_{\ell+2}(t^n)}$$

then in the limit $\Delta t_\ell \to 0$, should get

$$\frac{-(3/4)(\Delta t_\ell)^2\, e_2(t^n)}{-(3/16)(\Delta t_\ell)^2\, e_2(t^n)} = 4$$

- **Third, and most useful for labs/projects:** If we scale (multiply)

$$\begin{array}{rcc} u_\ell(t^n) - u_{\ell+1}(t^n) & \text{by} & 4^0 = 1 \\ u_{\ell+1}(t^n) - u_{\ell+2}(t^n) & \text{by} & 4^1 = 4 \\ u_{\ell+2}(t^n) - u_{\ell+3}(t^n) & \text{by} & 4^2 = 16 \end{array}$$

etc., and plot all the curves as a fcn of t^n, the curves should nearly coincide, and should become more coincident as $\Delta t \to 0$

- **Fourth**, we don't have to restrict convergence test to 3 levels, can use as many as possible, but *should always do 3-level test at a minimum!*
- **IMPORTANT!** We don't have to know what the error is to do the convergence test! (if we *did* know the error, what would be the point of doing the numerical calculation?)
  - Rather, we *assume* that the error is given by

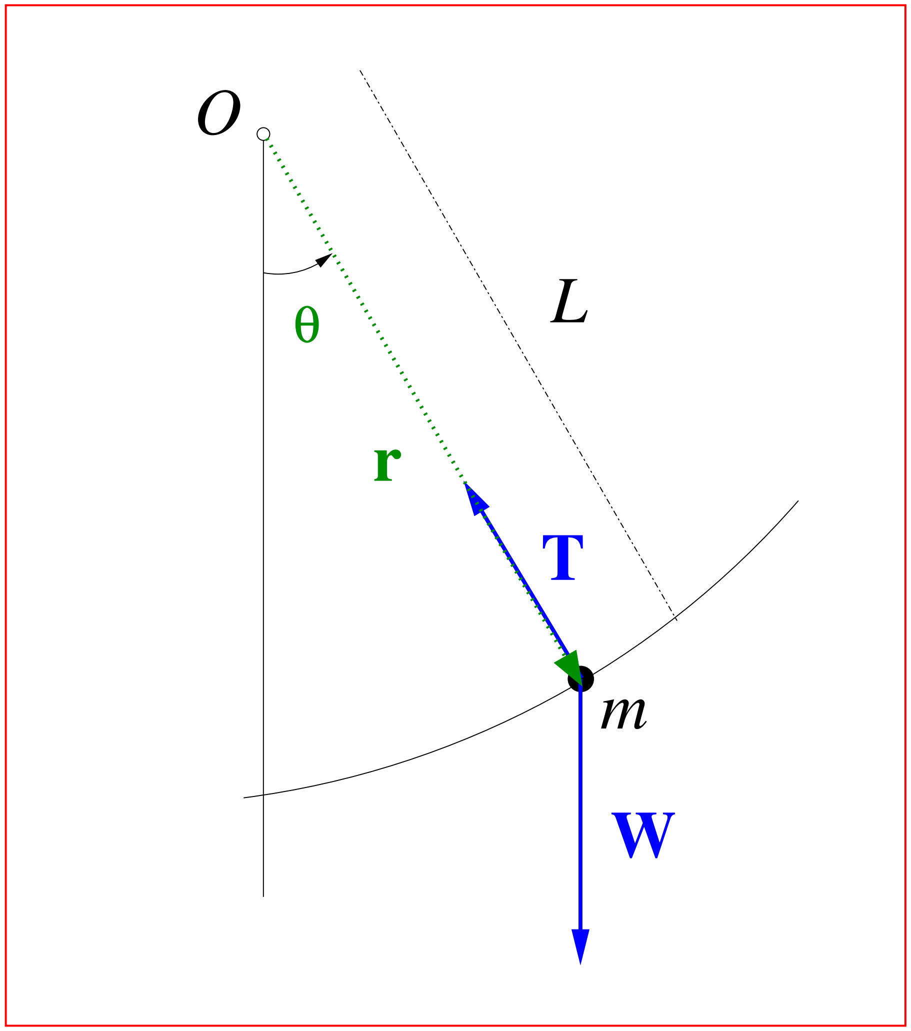
$$u_\star(t^n) - u(t^n) = \Delta t^2 e_2(t^n) + \ldots$$

    which should be the case provided that we have done the finite differencing properly, and have correctly implemented the solution of the resulting algebraic equations in our computer code

- We then convergence test by running simulation with different grid spacings (levels), and subtract the results computed at successive levels, and rescale as above
- If we *do* see near coincidence of the rescaled subtracted values, then we can have confidence in our numerical solutions, and will also have a good estimate of the error

- **ALSO IMPORTANT!** If we do *not* observe convergence, then we *know* that there is *some* problem with our implementation, and that we need to do some debugging!
- That is, convergence testing is an extremely powerful tool for developing and testing finite difference codes

2.6 Energy Quantities & Conservation



- For systems such as the nonlinear (linear) pendulum, where total (mechanical) energy is conserved, we can use conservation of energy as an additional check of the correctness and convergence of our numerical implementation

- For the nonlinear pendulum we have

$$\text{Kinetic energy} \equiv T(t) = \frac{1}{2}mv(t)^2 = \frac{1}{2}m\left(L\omega(t)\right)^2 \tag{13}$$

$$\text{Potential energy} \equiv V(t) = mgh(t) \tag{14}$$

  where $h(t)$ is the vertical displacement of the bob/mass relative to its stable equilibrium position.

- Basic trigonometry tells us that $h(t) = L(1 - \cos\theta(t))$ so we have

$$V(t) = mgh(t) = mgL\left[1 - \cos\theta(t)\right]$$

- Therefore

$$\text{Total energy} \equiv E(t) = T(t) + V(t) = \frac{1}{2}m\left(L\omega(t)\right)^2 + mgL\left[1 - \cos\theta(t)\right] \quad (15)$$

- Now, in our units, $g = L = 1$, and we will also take $m = 1$: can *always* do this via further choice of units (what would that choice be?), but in any case, since both terms are proportional to m, any particular choice of m is irrelevant to the central issue of how well the finite difference solution conserves energy

- Thus we have

$$E(t) = T(t) + V(t) = \frac{1}{2}\omega(t)^2 + \left[1 - \cos\theta(t)\right]$$

- In order to check for energy conservation, we can define the deviation in the total energy relative to the initial time

$$dE(t) \equiv E(t) - E(0)$$

- An obvious thing to do is to plot $dE(t)$ vs t to see whether the total energy "looks" conserved.

- However, a much better idea is to check the *convergence* of $dE(t)$ which, if our implementation is correct, should tend to 0 as $\Delta t \to 0$ like $O(\Delta t^2)$

- That is, we should ensure that our calculations display "convergence to conservation"

- Specifically, using the same type of analysis that we applied to the fundamental dynamical variable, θ, we can expect that in the limit $\Delta t \to 0$

$$dE(t^n) = dE^\star(t^n) - \Delta t^2 F_2(t^n) + \ldots$$

where $dE^\star(t^n)$ is the continuum (exact) value of dE and F_2 is the leading order error *function* for the energy deviation

- However, since the continuum total energy is precisely conserved, we have $dE^\star = 0$, so we should observe

$$dE(t^n) \approx -\Delta t^2 F_2(t^n)$$

- Thus, if we perform a convergence test, using fixed initial conditions and at least three levels of discretization, ℓ, $\ell+1$ and $\ell+2$, then plots of

$$dE_\ell(t^n), \quad 4 \times dE_{\ell+1}(t^n) \text{ and } 16 \times dE_{\ell+2}(t^n)$$

  should approach coincidence as ℓ increases ($\Delta t_\ell \to 0$)

- **Important!** Note that formulae (14) and (15) for the potential and total energies are *not* valid for the *linear* pendulum. We can get the correct formula by using the small angle approximation $\theta(t) \ll 1$ and keeping only the leading-order term.

- From Taylor series of $\cos\theta$ about $\theta = 0$ we have

$$\cos\theta = 1 - \frac{1}{2}\theta^2 + O(\theta^4) \approx 1 - \frac{1}{2}\theta^2$$

- So we have

$$E_{\text{linear}}(t) = T(t) + V(t) = \frac{1}{2}\omega(t)^2 + \frac{1}{2}\theta(t)^2$$

3. IMPLEMENTATION IN MATLAB (See pendulum.m and lpendulum.m)

- Discrete times, t^n, and grid functions such as θ^n, ω^n, T^n, V^n and E^n are all naturally represented as *row vectors* in Matlab

$$t^1, t^2, ..., t^{n_t} \;\rightarrow\; \texttt{t(1), t(2), ... t(nt)}$$

$$\theta^1, \theta^2, ..., \theta^{n_t} \;\rightarrow\; \texttt{theta(1), theta(2), ... theta(nt)}$$

  etc. I.e. the superscript label that identifies the discrete time step maps to the indexing/addressing of the Matlab vector

- With this "representation", the translation of the FDA into Matlab code is very straightforward; specifically

$$\theta^{n+1} = 2\theta^n - \theta^{n-1} - \Delta t^2 \sin\theta^n$$

  becomes

```
theta(n+1) = 2 * theta(n) - theta(n-1) - deltat^2 * sin(theta(n))
```

Bare-bones code: no comments, tracing

```
function [t theta omega] = pendulum(tmax, level, theta0, omega0)

   nt = 2^level + 1;
   t = linspace(0.0, tmax, nt);
   theta = zeros(1,nt);
   omega = zeros(1,nt);

   deltat = t(2) - t(1);

   theta(1) = theta0;
   theta(2) = theta0 + deltat * omega0 - 0.5 * deltat^2 * sin(theta0);

   omega(1) = omega0;

   for n = 2 : nt - 1
      theta(n+1) = 2 * theta(n) - theta(n-1) - deltat^2 * sin(theta(n));
      omega(n) = (theta(n+1) - theta(n-1)) / (2 * deltat);
   end

   omega(nt) = 2 * omega(nt-1) - omega(nt-2);

end
```

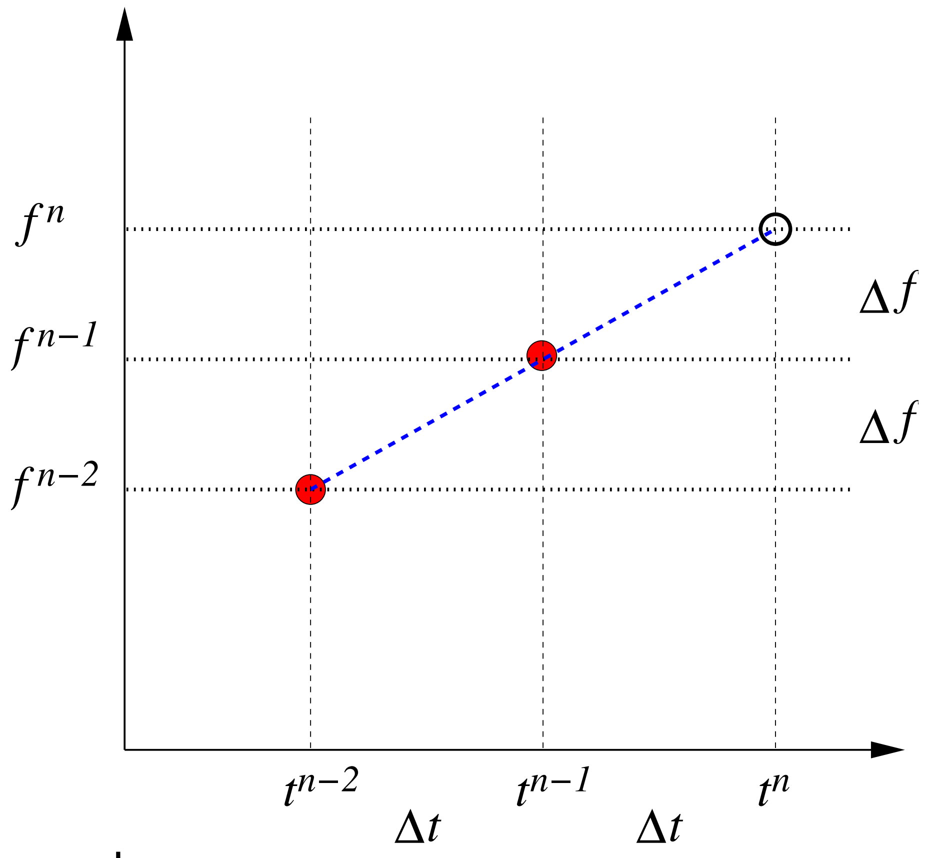
- **Note:** `pendulum` also computes and returns an approximation to the angular velocity, ω^n, using the $O(\Delta t^2)$ FDA for the first time derivative:

$$\omega^n \equiv \left.\frac{\partial \theta}{\partial t}\right|_{t=t^n} \to \frac{\theta^{n+1} - \theta^{n-1}}{2\Delta t}$$

- We can use this formula for $n = 2, 3, \ldots, n_t - 1$
- ω^1 is given from the initial condition, $\omega^1 = \omega(t = 0) = \omega_0$
- To compute a value for ω^{n_t} we can use *linear extrapolation* of the values ω^{n_t-1} and ω^{n_t-2}
- The formula is

$$\omega^{n_t} = 2\omega^{n_t-1} - \omega^{n_t-2}$$

- Derivation of linear extrapolation formula



- From the figure we see that

$$f^n = f^{n-1} + \Delta f = f^{n-1} + \left(f^{n-1} - f^{n-2}\right) = 2f^{n-1} - f^{n-2}$$

so, with $f \to \omega$, $n \to n_t$ we have

$$\omega^{n_t} = 2\omega^{n_t-1} - \omega^{n_t-2}$$

Full version of code: with comments and tracing

```
function [t theta omega] = pendulum(tmax, level, theta0, omega0)
%  pendulum Solves nonlinear pendulum equation using second order FDA
%  as discussed in class.
%
%  Input arguments
%
%      tmax:   (real scalar) Final solution time.
%      level:  (integer scalar) Discretization level.
%      theta0: (real scalar) Initial angular displacement of pendulum.
%      omega0: (real scalar) Initial angular velocity of pendulum.
%
%  Output arguments
%
%      t:      (real vector) Vector of length nt = 2^level + 1 containing
%              discrete times (time mesh).
%      theta:  (real vector) Vector of length nt containing computed
%              angular displacement at discrete times t(n).
%      omega:  (real vector) Vector of length nt containing computed
%              angular velocity at discrete times t(n).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   % trace controls "tracing" output.  Set 0 to disable, non-0 to enable.
   trace = 1;
   % tracefreq controls frequency of tracing output in main time step loop.
   tracefreq = 100;

   if trace
      fprintf('In pendulum: Argument dump follows\n');
      tmax, level, theta0, omega0
   end
```

```
% Define number of time steps and create t, theta and omega arrays of
% appropriate size for efficiency (rather than "growing" them element
% by element)
nt = 2^level + 1;
t = linspace(0.0, tmax, nt);
theta = zeros(1, nt);
omega = zeros(1, nt);

% Determine discrete time step from t array.
deltat = t(2) - t(1);

% Initialize first two values of the pendulum's angular displacement.
theta(1) = theta0;
theta(2) = theta0 + deltat * omega0 - 0.5 * deltat^2 * sin(theta0);

if trace
   fprintf('deltat=%g theta(1)=%g theta(2)=%g\n',...
           deltat, theta(1), theta(2));
end

% Initialize first value of the angular velocity.
omega(1) = omega0;

% Evolve the oscillator to the final time using the discrete equations
% of motion.  Also compute an estimate of the angular velocity at
% each time step.
for n = 2 : nt - 1
   % This generates tracing output every 'tracefreq' steps.
   if rem(n, tracefreq) == 0
      fprintf('pendulum: Step %g of %g\n', n, nt);
   end

   theta(n+1) = 2 * theta(n) - theta(n-1) - deltat^2 * sin(theta(n));

   omega(n) = (theta(n+1) - theta(n-1)) / (2 * deltat);
```

```
    end
    % Use linear extrapolation to determine the value of omega at the
    % final time step.
    omega(nt) = 2 * omega(nt-1) - omega(nt-2);
end
```