Lecture - Feb. 11 - 16.901

Topics:

* Reduction of systems to first-order
* Behavior of linear systems of eqns
* Stiffness
* Implicit vs. Explicit schemes

Reduction to First-order Systems

The canonical problem we will investigate will be coupled 1st order ODE's of the form:

$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \\ \vdots \\ \dot{u}_{N-1} \\ \dot{u}_N \end{bmatrix} = \begin{bmatrix} f_1(u_1, u_2, u_3 \cdots, u_{N-1}, u_N, t) \\ f_2(u_1, u_2, u_3 \cdots, u_{N-1}, u_N, t) \\ f_3(u_1, u_2, u_3 \cdots, u_{N-1}, u_N, t) \\ \vdots \\ f_{N-1}(u_1, u_2, u_3 \cdots u_{N-1}, u_N, t) \\ f_N(u_1, u_2, u_3 \cdots u_{N-1}, u_N, t) \end{bmatrix}$$

or, in vector form:

$$\vec{\dot{u}} = \vec{f}(\vec{u}, t)$$

With initial conditions on u_i, $i = 1 \to N$ at time $t = 0$

Many physical systems are originally in this form, however, many are not. A typical example of a problem which is not is a 2nd order ODE:

For example,

$$\underbrace{\ddot{v} + \dot{v}^3 + v = f(t)}_{\text{2nd order nonlinear ODE}} \qquad \underbrace{\begin{matrix} v(0) = \hat{v} \\ \dot{v}(0) = 0 \end{matrix}}_{\text{initial conditions}} \qquad (1)$$

Converting this to a 1st order system is easy by adding extra states for the higher derivatives. Say for example:

Define $v_1 = v$

$v_2 = \dot{v}$

Then, problem (1) can be written:

$$\begin{matrix} \dot{v}_2 + v_2^3 + v_1 = f(t) \\ \dot{v}_1 = v_2 \end{matrix} \quad \text{with} \quad \begin{matrix} v_1(0) = \hat{v} \\ v_2(0) = 0 \end{matrix}$$

Or, re-arranging:

$$\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} v_2 \\ -v_2^3 - v_1 + f(t) \end{bmatrix} \begin{matrix} \leftarrow f_1(v_1, v_2, t) \\ \leftarrow f_2(v_1, v_2, t) \end{matrix}$$

which is our canonical form.

Summarizing, to convert a higher order system to 1st order:

(1) Introduce states for all but highest derivatives

(2) Substitute in derivative states & re-arrange

Behavior of Linear Systems of ODE's

In some cases, we are interested in linear systems of ODE's, e.g.

* stability problems can often be dealt with by linearizing a system about it's mean or some other reasonable condition
* analysis can be done more readily allowing comparisons between analytical and numerical solution behavior
* a lot of insight into the problem's physics can be gained through a linearized system

For a canonical problem in a linear form we will consider:

$$\begin{bmatrix} \dot{U}_1 \\ \dot{U}_2 \\ \vdots \\ \dot{U}_{N-1} \\ \dot{U}_N \end{bmatrix} = A \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{N-1} \\ U_N \end{bmatrix} + \begin{bmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_{N-1}(t) \\ f_N(t) \end{bmatrix} \quad \text{w/ I.C.'s on } U_i\text{'s}$$

Or, in vector form:

$$\dot{\vec{U}} = A\vec{U} + \vec{f}(t) \quad \text{with } \vec{U}(0) = \vec{U}_0$$

While the forcing term leads to particular solutions, the homogeneous problem,

$$\vec{v}' = A\vec{v} \qquad \text{with } \vec{v}(0) = \vec{v}_0$$

Gives rise to the eigenvalues.

Specifically, assuming no repeated eigenvalues with degenerate eigenvectors. then the solution is of the form:

(Intentionally Blank)

$$\vec{u}(t) = \sum_{j=1}^{n} \alpha_j e^{\lambda_j t} \vec{r}_j$$

where α_j = constants

λ_j = jth eigenvalues of A

$\vec{r}_j$ = jth eigenvector of A

} all of these could be complex numbers!

For the eigenvalues, we can split them into real and imaginary parts:

$$\lambda_j = \lambda_j^{(r)} + i\lambda_j^{(i)}$$

and note that:

$$e^{\lambda_j t} = e^{\lambda_j^{(r)} t} e^{i\lambda_j^{(i)} t}$$

$$= e^{\lambda_j^{(r)} t} \left[\cos \lambda_j^{(i)} t + i \sin \lambda_j^{(i)} t \right]$$

$e^{\lambda_j^{(r)} t}$: amplitude controlled by $\lambda_j^{(r)}$

$\cos \lambda_j^{(i)} t + i \sin \lambda_j^{(i)} t$: oscillation frequency controlled by $\lambda_j^{(i)}$

Stiffness in ODE's

Stiffness is a general (somewhat fuzzy) term to describe systems of equations which exhibit phenomena at widely-varying scales. For the ODE's we are interested in currently, this means widely-varying timescales.

One way (very common) to measure the stiffness of a system is to linearize it and calculate the ratio of the largest magnitude eigenvalue to the smallest magnitude eigenvalue. This is known as the <u>spectral condition number</u>:

$$\text{spectral condition number} \equiv \frac{\max_{j=1:N} |\lambda_j|}{\min_{j=1:N} |\lambda_j|}$$

Typically, if this ratio is > 1000, the problem is considered stiff. Suppose we had the following system:

$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 0 & -1000 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

The eigenvalues of this matrix are -1 & -1000

$\Rightarrow$ SCN = 1000! $\Rightarrow$ stiff system

So what is happening? The problem is that one component of the system decays as e^{-t} while the other as a e^{-1000t}.

e^{-t} ↖ slow decay!

e^{-1000t} ↑ rapid decay!

As we have seen already (and will in more detail in the next few lectures), numerical stability is usually controlled by the fastest modes (i.e. shortest timescales). But, we may be more interested in long time behavior. So, this leads to:

λ_{max} → controls stability

λ_{min} → sets timescale of long time behavior

For our Forward Euler example from the first lecture, stability of the numerical algorithm requires

$$|\lambda_{max}\Delta t| < 2 \qquad \text{(assumes } \lambda_{max} < 0\text{)}$$

$$\Rightarrow \quad \Delta t < \frac{2}{|\lambda_{max}|} = \frac{1}{500}$$

$\Rightarrow$ 500 timesteps per characteristic time of the slow timescale (i.e. $|\lambda_{min}|$)

This is likely to be highly inefficient!

The forcing $\vec{f}(t)$ can also create long or short timescales. For example, consider the following problem:

Example: Stiffness through forcing

$$\dot{u} + 1000u = \sin t \qquad u(0) = 1$$

We see that the homogeneous problem

$$\dot{u}_h + 1000 u_h = 0$$

gives $u_h(t) = a e^{-1000t}$

$\Rightarrow \lambda = -1000 \Rightarrow$ "rapid" decay compared to external forcing

* This is very common in many aerospace applications.

* Example: acoustic speeds in structures compared to external forcing due to aircraft motion

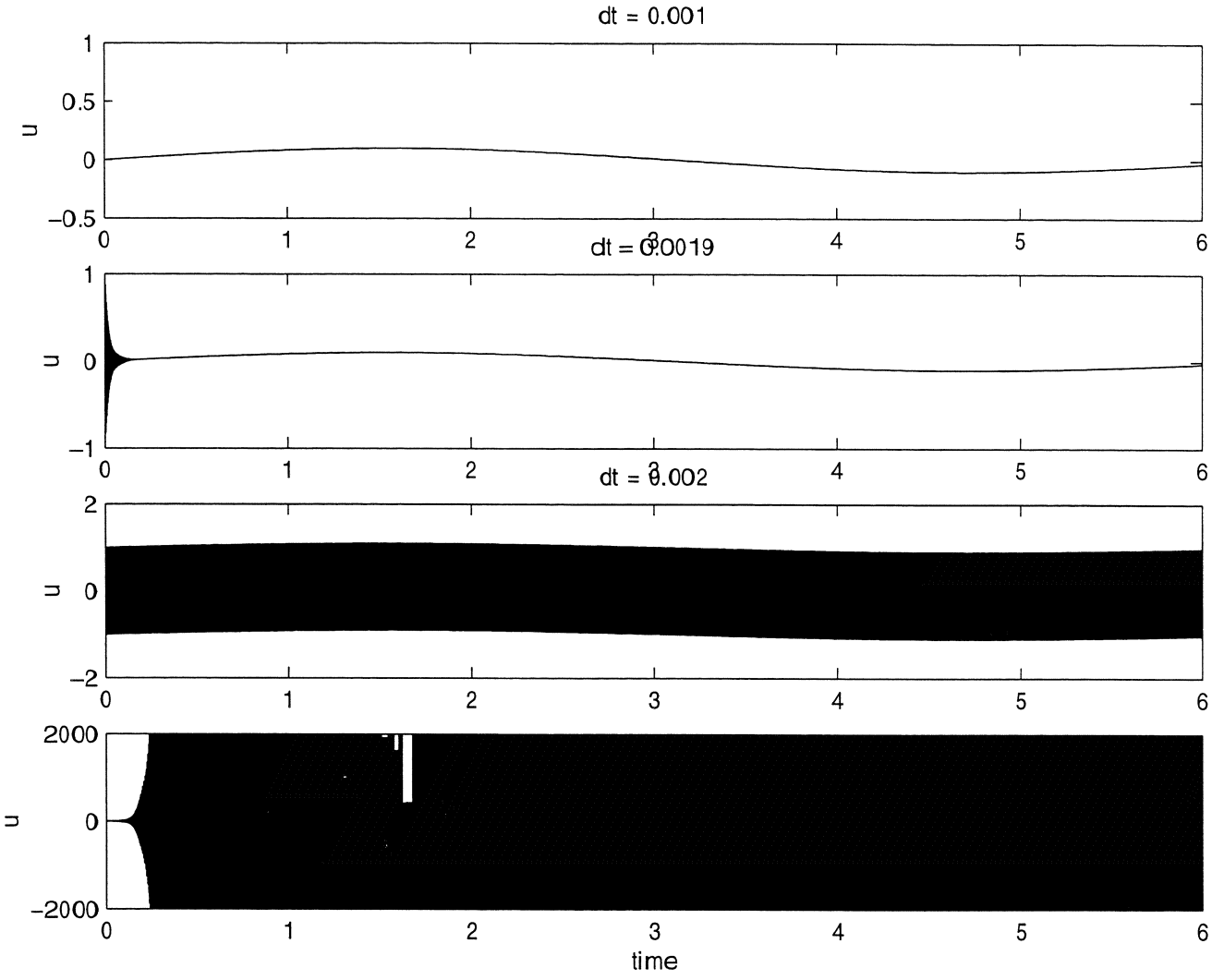
So, the above problem is also stiff!

A more general measure of stiffness for time dependent problems (which is also relevant for nonlinear problems) is:

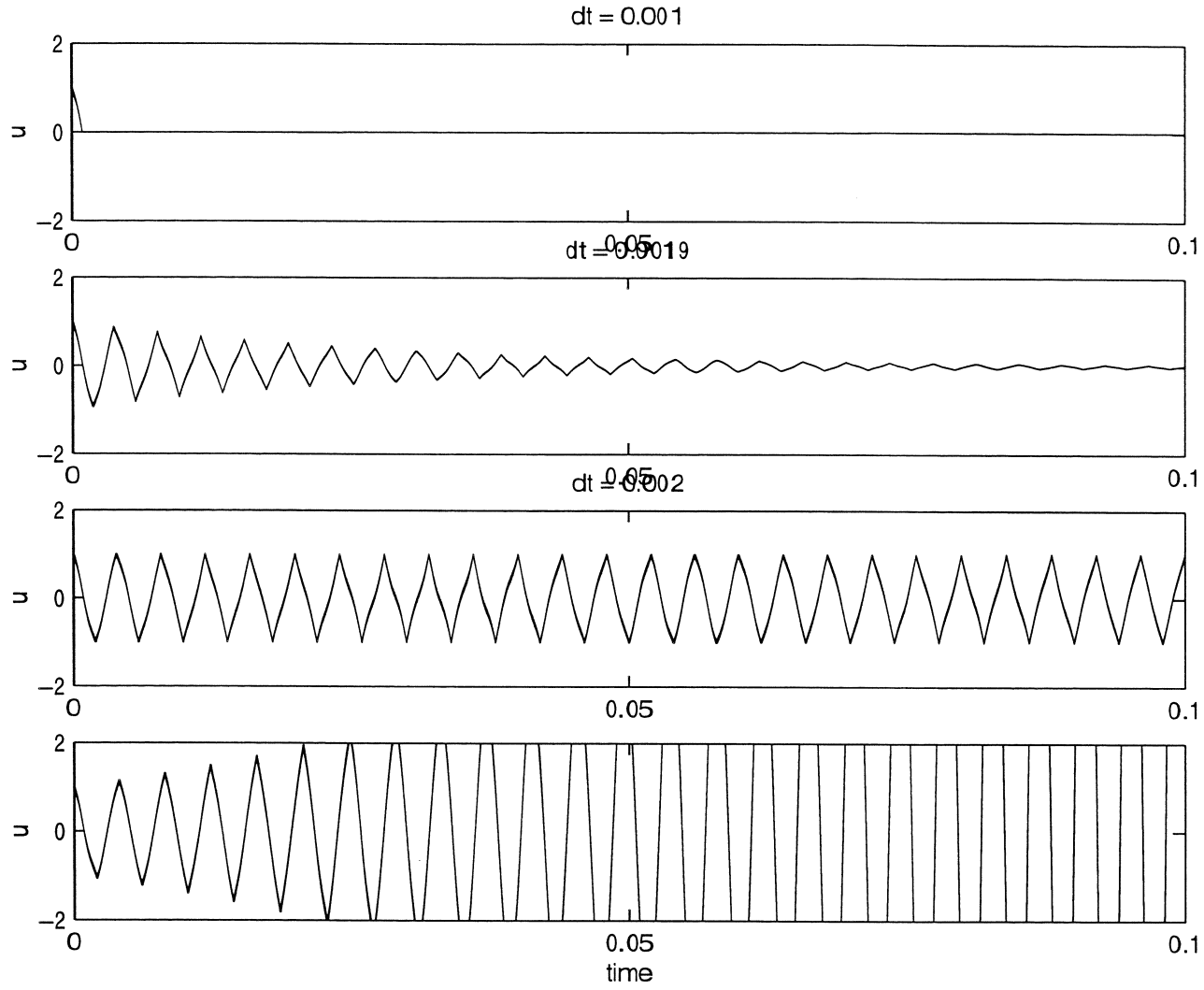
$$\text{Stiffness Measure} = \frac{\text{longest time scale}}{\text{shortest time scale}}$$

Forward Euler

$\dot{v} + 1000v = \sin \omega t \qquad v(0) = 1$



Forward Euler: zoom at $t \to 0$



Let's look at how forward Euler deals with the stiff forced problem.

(see attached figure on next page)

The results clearly show the same stability trends observed before. For $|\lambda \Delta t| \leq 2$, we have stability (this corresponds to $\Delta t \leq 0.002$). But, we also see oscillations for $1 < |\lambda \Delta t| \leq 2$, which corresponds to $0.001 < \Delta t \leq 0.002$.

What is happening here? Can we build some intuition about the problem and a possible cure?

Return to our generic problem for a scalar eqn:

$$\frac{du}{dt} = f(u,t) \qquad u(0) = u_0$$

forward Euler is:

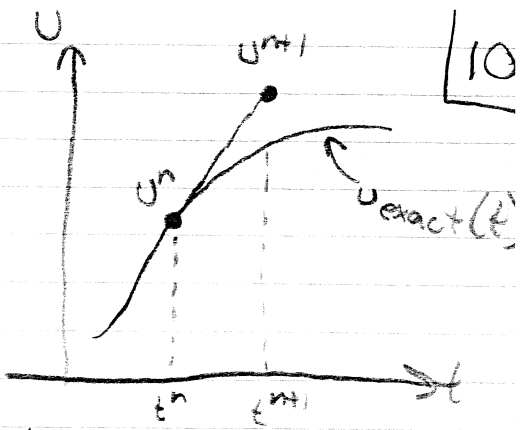
$$\frac{u^{n+1} - u^n}{\Delta t} = f(u^n, t^n)$$

This is an extrapolation which takes $\frac{du}{dt}$ at $t = t^n$ and extrapolates to find u^{n+1}:

$$\left.\frac{du}{dt}\right|_{t^n} = f(u^n, t^n)$$

$$u^{n+1} = u^n + \Delta t \left.\frac{du}{dt}\right|_{t^n}$$

Taylor series about $t = t^n$, first 2 terms



$$\Rightarrow u^{n+1} = u^n + \Delta t f(u^n, t^n)$$

One would expect that as Δt increased, this extrapolation would have problems.

A different approach would be interpolate back from t^{n+1}. For example, a Taylor series for u^n about u^{n+1} gives:

$$u^n = u^{n+1} - \Delta t \left.\frac{du}{dt}\right|_{t^{n+1}} + \frac{1}{2}\Delta t^2 \left.\frac{d^2u}{dt^2}\right|_{t^n}$$

drop this

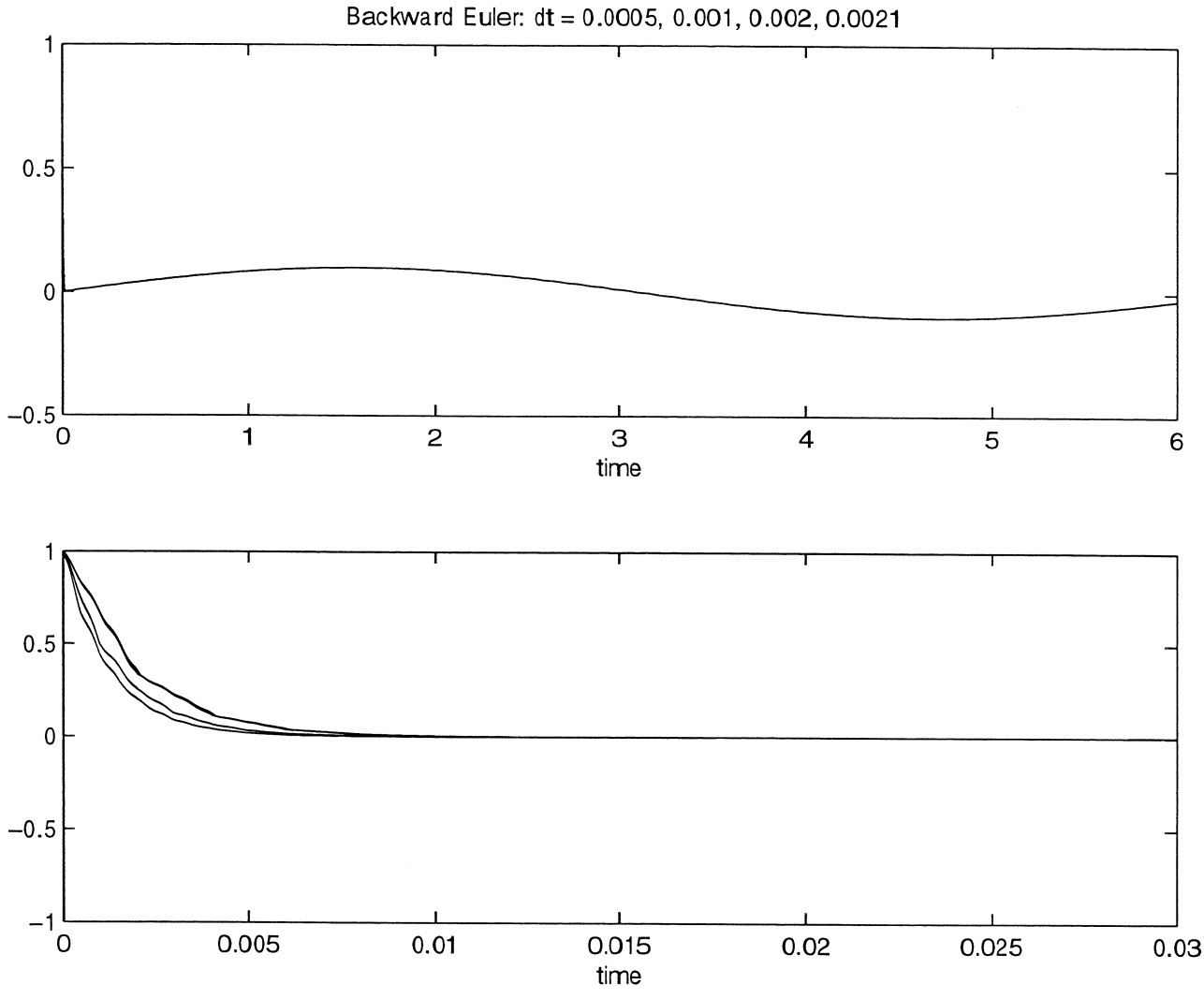
$$\Rightarrow u^{n+1} = u^n + \Delta t \left.\frac{du}{dt}\right|_{t^{n+1}}$$

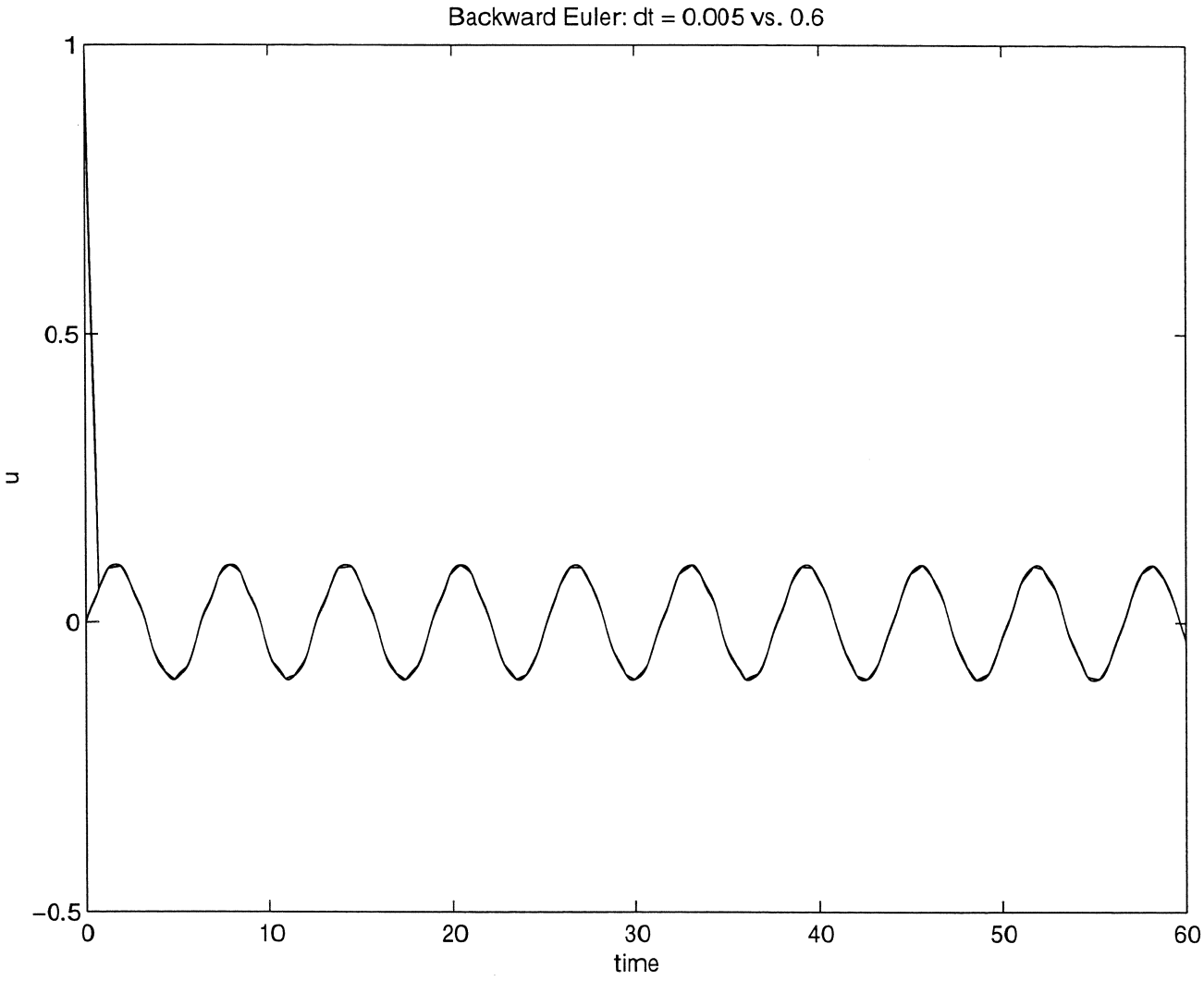
$$u^{n+1} = u^n + \Delta t f(u^{n+1}, t^{n+1})$$

Backward Euler

Let's look at how it works first...

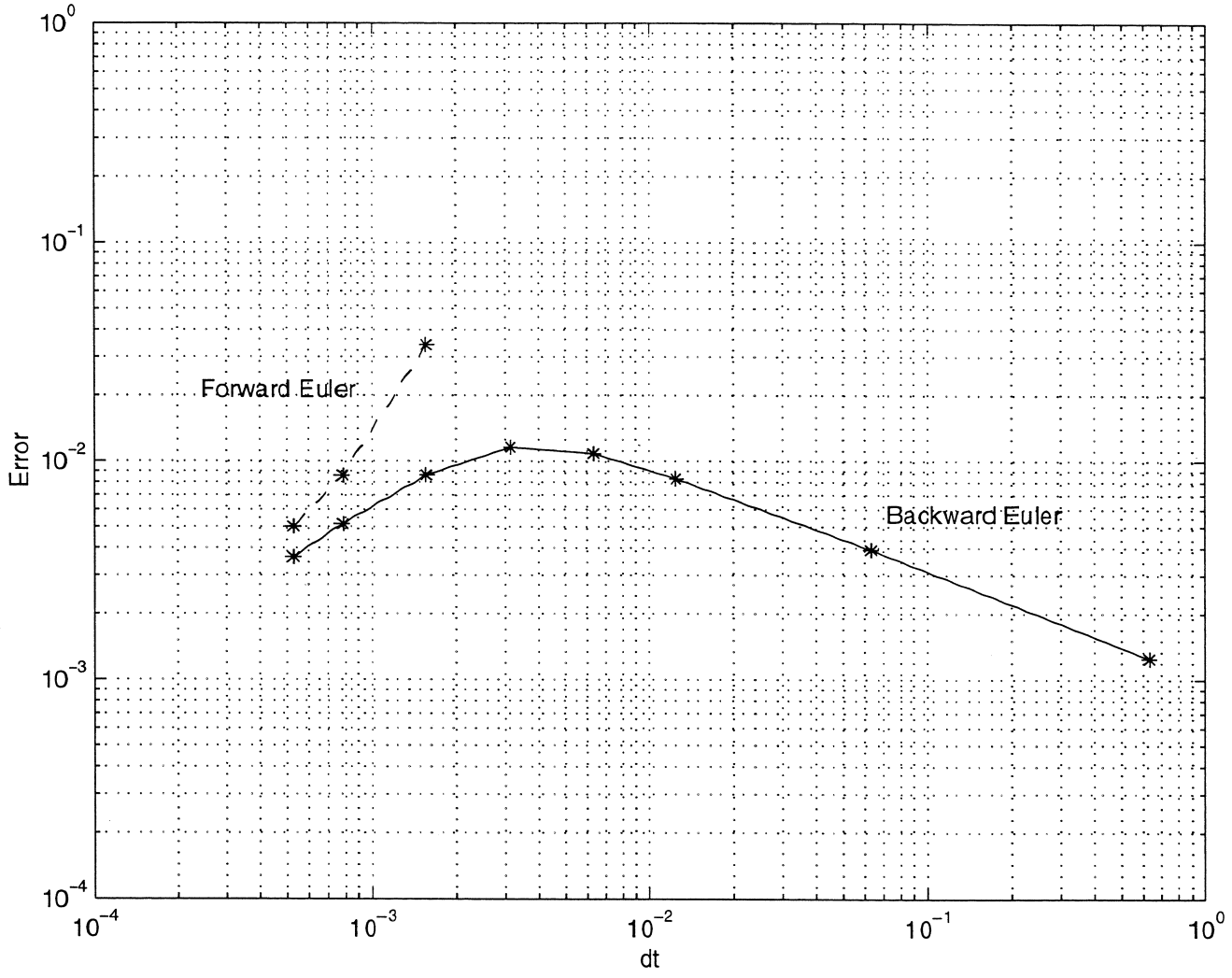
$$\dot{v} + 1000v = \sin t \qquad v(0) = 1$$





Recall: Forward Euler still unstable for $\Delta t = 0.005$!

$$E = \sqrt{\int_0^T (u_e - v)^2 \, dt}$$



Let's look at the stability a bit closer for Backward Euler. Reduce to simple problem:

$$\dot{v} = \lambda v$$

$$\Rightarrow \frac{v^{n+1} - v^n}{\Delta t} = \lambda v^{n+1}$$

$$\Rightarrow v^{n+1} = \frac{1}{1 - \lambda \Delta t} v^n$$

$$\Rightarrow \boxed{v^n = \left(\frac{1}{1 - \lambda \Delta t}\right)^n v^0}$$

If $\lambda < 0$, which physically is stable, then $0 < \frac{1}{1 - \lambda \Delta t} \leq 1 \Rightarrow$ amplitude of v^n cannot ever grow regardless of Δt size!

* What this means is Δt can be chosen
* based on required accuracy, not stability of numerics.

Great, but what's the cost? Nothing in life is free...

* Extra cost is a more expensive calculation each iteration.