Lecture 5:
Solving Lumped-Parameter Propagation Problems

M.G. Lipsett
Department of Mechanical Engineering
University of Alberta
http://www.ualberta.ca/~mlipsett/ENGM541/ENGM541.htm

Solution Methods for Propagation Problems (Review)

- Summary of short time interval methods:
  - Picard’s Method
  - Taylor Series
- Residual methods for longer time intervals
  - Collocation
  - Subdomain
  - Least-Squares
  - Galerkin
Solution Methods for Propagation Problems

- Finite difference methods
- Errors introduced by discrete methods
- Step size extrapolation
- Recurrence formulae with higher-order truncation errors (Runge-Kutta)

“It's difficult to make predictions, especially about the future.”
- Yogi Berra

Finite Difference Methods

- The function
- Is a continuous formulation with respect to time. We now consider procedures for finding approximate solutions at discrete time intervals
- We replace the variables by their value at discrete times, e.g.
Finite Difference Methods (2)

- The derivative is defined as

- which we approximate as

- where variables subscripted I are variables defined at the discrete time \( r(I) \), and

\[
\text{Approximation of derivative:} \quad \frac{dx}{dt} \approx \frac{x(I+1) - x(I)}{\Delta t}
\]

Finite Difference Methods (3)

- Substitute in the governing equation

\[
\frac{dx}{dt} = f(x)
\]

- We want to find \( x(I+1) \) from this relationship, to extrapolate forward in time to new values of \( x \):

\[
\frac{x(I+1) - x(I)}{\Delta t} = f(x(I))
\]

- This expression allows us to extrapolate from position \((I)\) to position \((I+1)\)
Finite Difference Methods (4)

- We could approach this from a Taylor Series expansion about position I:

- Truncating the series after the first two terms means that there will be some amount of error

Finite Difference Methods Example

- For the system

Consider the extrapolation for constant time step \( h \)

<table>
<thead>
<tr>
<th>I</th>
<th>( t(I) )</th>
<th>( x(I) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Finite Difference Methods Example (2)

Let’s examine the solution at the time \( t(N) = N \ h \); expand using the binomial theorem.

By comparison, the exact solution is:

Which at \( t = Nh \) is:

Define the error as:

And get:

Finite Difference Methods Example (3)

If we choose \( h \) to be small, so that \( h^3 << h^2 \), then:

This is the truncation error:

Error decreases as \( h \) decreases (quadratically).

Error increases as \( N \) increases (linearly).

Let’s examine the potential sources of errors that can occur by discrete modeling of differential equations.
Errors Introduced by Discrete Methods

• For

• Define

• Consider a step from \( t \) to \( (t+h) \). For a numerical approach,

• The exact solution expressed in Taylor’s Series is

• There is a time \( \tau \) in the interval \( t \leq \tau \leq t+h \) which will truncate the series exactly, in which case we can write

\[
(\text{We don’t know what } \tau \text{ is, but we do know that it exists})
\]

Errors Introduced by Discrete Methods (2)

• Define the truncation error

• So

• The error at time \( t+h \) is:

\[
\text{The second term on the RHS is a measure of how different the derivative is when evaluated by the approximate (numerical) solution } X \text{ as opposed to the true solution } \Psi
\]
Errors Introduced by Discrete Methods (3)

- Write

- So

- We are concerned with the total solution error at some time $T$, where we have made $M$ integration steps of size $h$:

- The truncation error per step is
- The total truncation error after $M$ steps is

- So the truncation error is proportional to $h$.

Total Round-Off Error & Inherited Error

- The total round-off error is

- The inherited error can be shown to be insensitive to $h$, and depends only on $T$.

- $h_{\text{min}}$ is the $h$ that gives the minimum total error.
Step Size Extrapolation

- Euler’s method of numerical integration is

Which yields the numerical solution

Step Size Extrapolation (2)

- Truncation error $E_T$ for the derivative is of order $h$.
- This is the same as the order of truncation error at time $T$.

Solution at $T$

- Error varies with the size of $h$ that is chosen
- So by extrapolating back to $h = 0$ we can estimate the true solution
Central Difference Integration

- We can improve the truncation error by choosing a more appropriate estimate of the derivative.
- For example, from numerical solutions at two points:
  \[ t = t + h \quad \text{and} \quad t = t - h, \]

- We subtract to get

- So we can write an alternative expression for the derivative:

Central Difference Integration (2)

- In this case the truncation error is on the order of \( h^2 \) for the derivative

- The total error at time \( t \) will be smaller than for the Euler method, provided that \( h \) is relatively small
Central Difference Integration (3)

- This method is not self-starting. We need to know $X(t-h)$ in order to estimate $X(t+h)$.
- When $t = 0$ (initial conditions), $X(t-h)$ is undefined, and so we need to use Picard’s Method or Taylor Series to generate a solution out to $t=h$. Then we can start using central difference.

Recurrence Formulae with Higher-order Truncation Errors

- Given the problem
- We apply formulæ of the form
- where

$\text{So } x_s \text{ and } x_{s,q} \text{ are end points of a chord with a slope that is some weighted average of slopes given by the governing equation}$
### Three Categories of Methods

- There are three options for picking points in time when $f(x,t)$ is evaluated:
  - $f(x,t)$ evaluated at times previous to $t_s$ (e.g., $t_{s-2}$, $t_{s-1}$) using the weighted average of several previous slopes to extrapolate. (Euler’s method uses only the slope at the last point, so it is self-starting; but generally this method needs to be started by Taylor Series or Picard’s method.)
  - $f(x,t)$ evaluated at $t_s$ and at times previous to $t_s$ (e.g., $t_{s-2}$, $t_{s-1}$, $t_0$), requiring iteration to estimate $x_s$.
  - $f(x,t)$ evaluated at times between $t_{s-1}$ and $t_s$. These are self-starting methods. The Runge-Kutta methods fall into this category.

### Runge-Kutta Integration

- The standard Runge-Kutta method goes as follows:

  \[ x_{s+1} = x_s + \frac{1}{6} \left( k_1 + 2k_2 + 2k_3 + k_4 \right) \]

  where
  
  \[ k_1 = h f(x_s, t_s) \]
  \[ k_2 = h f(x_s + \frac{1}{2} k_1, t_s + \frac{1}{2} h) \]
  \[ k_3 = h f(x_s + \frac{1}{2} k_2, t_s + \frac{1}{2} h) \]
  \[ k_4 = h f(x_s + k_3, t_s + h) \]

  - This is the fourth-order RK method, with truncation error on the order of $h^5$. If we use time steps of 0.01 s, then $E_T$ is of order $10^{-16}$. 
Runge-Kutta Integration for Multiple Equations

- This method can be used for a set of governing equations \( \{f(x,t)\} \) in the following sequence:

- Current implementations can have truncation error \( O(h^{11}) \) (on the order of \( h^{11} \)) or even \( O(h^{13}) \), but accuracy comes with computational expense. A \( O(h^7) \) RK method requires 13 derivatives per step. For Euler method to be as accurate, one must use very small steps, because truncation error is \( O(h) \) (in which case round-off error may become important).

Adaptive Integration Methods

- So-called adaptive methods use time steps \( h \) of variable size
  - Evaluating truncation error in each integration step
  - Choosing \( h \) to satisfy some error tolerance set by the user.
- This is important when \( f(x,t) \) gets “interesting.”

- The program adjusts to meet error tolerance in several ways.
- One approach is to run two methods simultaneously, with truncation error varying by one order between them.
- The difference between the solutions is an estimate of the truncation error.
Adaptive Integration Methods (2)

- The difference between the solutions varies linearly with $h$:

- If the difference is too large (i.e., greater than the set tolerance) then $h$ can be reduced to an acceptable value using simple ratios.
- If the difference is too small, then $h$ is increased, which speeds up the solution while still keeping the error within tolerance.

Stiffness and Stability

- An initial value problem is called “stiff” if some terms in $\{x(t)\}$ vary much more rapidly than others.
- For differential equations of the form

  $$\Lambda \{x(t)\} = \{x'(t)\}$$

  where $\Lambda$ is a constant coefficient matrix, the system is stiff if there is a large range in the magnitudes of the positive eigenvalues of $\Lambda$.

  The step size $h$ needed for the numerical integration to be stable depends on the equations and the solution method.
- A stable solution has bounded error.
- An unstable solution accumulates errors exponentially, eventually causing numerical overflow.
- For Euler’s method to be stable, $h < \frac{2}{\lambda_{\text{max}}}$.
Stability Example

- For the governing equation

- We convert it to standard form using to get

- Which is of the form

- The eigenvalues of $\Lambda$ are $\lambda_1 = 1$ and $\lambda_2 = 1000$, so $\lambda_{\text{max}} = \lambda_2$

- If we want to use Euler’s method of numerical integration,

- for the solution to be stable.

Stiffness and Stability (2)

- Some numerical methods are designed to solve stiff problems; but the general rule is that stability can only be improved by reducing the order of the method (which increases the truncation error).

- In MATLAB, ODE45 is a reasonable method to use for non-stiff differential equations. It defaults to an adaptive step size based on the Runge-Kutta-Fehlberg formulæ:

$$
\{K_i\} = h\left\{f\left(x, i\right)\right\}
$$

$$
\{K_i\} = h\left\{f\left(x\right) + \sum_{j=1}^{i-1} B_j \{K_j\} t + A_i h\right\}, \quad i = 2,3,\ldots,6
$$

$$
\{x_5(t+h)\} = \{x(t)\} + \sum_{i=1}^{6} C_i \{K_i\}
$$

$$
\{x_4(t+h)\} = \{x(t)\} + \sum_{i=1}^{6} D_i \{K_i\}
$$
ODE45

• This method has an implicitly estimated truncation error of

\[
\text{error} = \frac{B_i}{A_i^2} + C_i + D_i
\]

where the coefficients are:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$A_i$</th>
<th>$B_{ij}$</th>
<th>$C_i$</th>
<th>$D_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{1}{5}$</td>
<td>$\frac{1}{5}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{3}{10}$</td>
<td>$\frac{9}{40}$</td>
<td>$\frac{250}{621}$</td>
<td>$\frac{18575}{48384}$</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{3}{5}$</td>
<td>$\frac{-9}{10}$</td>
<td>$\frac{125}{594}$</td>
<td>$\frac{13525}{55296}$</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>$\frac{70}{54}$</td>
<td>$\frac{35}{27}$</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>$\frac{7}{8}$</td>
<td>$\frac{1631}{55296}$</td>
<td>$\frac{575}{512}$</td>
<td>$\frac{44275}{13824}$</td>
</tr>
</tbody>
</table>

Example: Robot Contact Dynamics

• Robots typically move in free space, without contacting objects in the workspace. For robots to be more useful, however, they must manipulate objects. This complicates the dynamics, particularly during the transition phase when contact occurs.

• To illustrate this change in dynamic behaviour, consider a robot arm with a single degree-of-freedom:

  - Revolute joint motion
  - Force sensor at end effector
  - Linear stiffness
  - Linear damping (friction)
  - Small motions

• We will solve in terms of loop variables and choose node variables for force & moment admissibility relationships
**Example: Robot Contact Dynamics (2)**

- Proximity of the end effector to a workpiece is

![Diagram of robot contact dynamics](image)

**Example: Robot Contact Dynamics (3)**

- System block diagram using mass-spring damper elements:

![System block diagram](image)

- There are four nodes, including the ground (so $N = 3$)
- Frictional torque opposing the motion is
Example: Robot Contact Dynamics (4)

- **Free motion case** (no constraint), \( z > 0 \)
  - From the sum of moments on the robot arm, and applying (linear) constitutive relationships, first governing equation of motion is

  \[ \ldots \]

  - Similarly, other equations of motion come from force balances on the force sensor and on the workpiece:

- where

Example: Robot Contact Dynamics (5)

- **Constrained motion and contact force case**
  - Now the end effector is in contact with the workpiece, and so \( z = 0, \dot{z} = 0, \ddot{z} = 0 \)

  - But now \( F_c \) is generally non-zero, and the dynamics of the system include interactions between the robot and workpiece
  - The equations of motion are now

- And the contact force is
Example: Robot Contact Dynamics (6)

- If we think of a single model of the system, we must include an element between the end effector and the workpiece that incorporates the interesting behaviour of contact
- The constitutive relationship will look something like this:

\[
\begin{align*}
\text{Displacement} & \quad \Rightarrow \quad \text{Force} \\
\end{align*}
\]

- But this is just the relationship of displacement to force, (which we see is non-linear).
- The velocity and acceleration terms must also be included to reflect the real dynamics of contact

Example: Robot Contact Dynamics (7)

- This complex, non-linear dynamic element does not have a smooth function
- The sharp change in force with respect to displacement affects the stability of the system during contact, because small errors in displacement cause big force errors
- This system has a stiff set of equations of motion, and so an adaptive numerical integration method is required to simulate the solution
- A further uncertainty is what the characteristics of the workpiece might be
- Robots have control systems (part of the dynamics) that are “tuned” for free motion, and so they move slowly in the range of contact, and use compliant end effector grippers with low stiffness to absorb energy during contacts, thus avoiding instability.
Example: Automobile Steering Control

- Some systems are more easily described in terms of velocities (for example, current flows)

- Consider an automobile

\[ y \]

\[ x \]

We assume no forces in the direction of motion (coasting)

We describe the motions:

- Lateral acceleration:

- Angular acceleration:
Example: Automobile Steering Control (3)

- We will use node equations (forces and moments) expressed in terms of loop variables (velocities) to generate equations of motion

- We need to relate tire forces to velocities (and the steering angle). To get the constitutive relationships for these system elements, we have to conduct experiments.

Example: Automobile Steering Control (4)

- A tire with a lateral velocity is not in pure rolling motion.

  ![Diagram](image)

  - Experiments reveal this kind of behaviour:

  - This is well fit by a cubic relationship:
Example: Automobile Steering Control (5)

• For the rear tire

• For the front tire

Example: Automobile Steering Control (5)

• Substitute into equations 1 & 2 to yield two simultaneous differential equations in terms of velocities:

• and

• where

• To solve this system, we cast it into the form

• where
Example: Automobile Steering Control (5)

- Again, this system has non-linear elements
- Steering control changes spectacularly when the tire friction is compromised
- An adaptive solver is required for this stiff set of equations

Summary

- Different methods may be used to generate approximate solution to a set of differential equations at discrete points in time, by truncating the equations, expressing them as difference equations, and solving these equations progressively from the initial time at successively later points in time.
- Different methods have different amounts of truncation error. Round-off error and inherited error also contribute to the total error in a solution.
- Stability of the solution method will be affected by the stiffness of the set of equations.
- Adaptive solvers can help to keep the error bounded.
- This is important when the dynamics change dramatically.
Break Time: Which way is the bus going?