Numerical Simulation of Dynamic Systems X

Prof. Dr. François E. Cellier
Department of Computer Science
ETH Zurich

March 26, 2013

Non-linear Velocity Models

Given the general second-derivative model:

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) \tag{1}$$

We need an algorithm to compute the velocity vector.

Non-linear Velocity Models

Given the general second-derivative model:

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) \tag{1}$$

We need an algorithm to compute the velocity vector.

- We can use any ODE solver to compute the velocity vector from the acceleration vector.

Non-linear Velocity Models

Given the general second-derivative model:

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) \tag{1}$$

We need an algorithm to compute the velocity vector.

- We can use any ODE solver to compute the velocity vector from the acceleration vector.
- As the velocity gets always multiplied by h, it suffices to use a second-order accurate algorithm.

Non-linear Velocity Models

Given the general second-derivative model:

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) \tag{1}$$

We need an algorithm to compute the velocity vector.

- We can use any ODE solver to compute the velocity vector from the acceleration vector.
- As the velocity gets always multiplied by h, it suffices to use a second-order accurate algorithm.
- As the **Godunov (GE3) algorithm** that we wish to employ for computing the position vector is explicit, we should use an explicit algorithm also for the velocity vector.

Non-linear Velocity Models

Given the general second-derivative model:

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) \tag{1}$$

We need an algorithm to compute the velocity vector.

- We can use any ODE solver to compute the velocity vector from the acceleration vector.
- As the velocity gets always multiplied by h, it suffices to use a second-order accurate algorithm.
- As the **Godunov (GE3) algorithm** that we wish to employ for computing the position vector is explicit, we should use an explicit algorithm also for the velocity vector.
- Let us use **AB2**:

$$\dot{\mathbf{x}}_{\mathbf{k+1}} = \dot{\mathbf{x}}_{\mathbf{k}} + \frac{h}{2} \cdot (3 \cdot \ddot{\mathbf{x}}_{\mathbf{k}} - \ddot{\mathbf{x}}_{\mathbf{k-1}})$$

Non-linear Velocity Models II

Let us apply this scheme to the linear second-derivative model:

$$\begin{aligned}
\mathbf{x_{k+1}} &= 2 \cdot \mathbf{x_k} - \mathbf{x_{k-1}} + h^2 \cdot \dot{\mathbf{v}}_\mathbf{k} &:& \quad \textit{GE3 solver} \\
\mathbf{v_{k+1}} &= \mathbf{v_k} + \frac{h}{2} \cdot (3 \cdot \dot{\mathbf{v}}_\mathbf{k} - \dot{\mathbf{v}}_\mathbf{k-1}) &:& \quad \textit{AB2 solver} \\
\dot{\mathbf{v}}_\mathbf{k} &= \mathbf{A}^2 \cdot \mathbf{x_k} + \mathbf{B} \cdot \mathbf{v_k} &:& \quad \textit{model equations}
\end{aligned}$$

Non-linear Velocity Models II

Let us apply this scheme to the linear second-derivative model:

$$\begin{aligned}
\mathbf{x_{k+1}} &= 2 \cdot \mathbf{x_k} - \mathbf{x_{k-1}} + h^2 \cdot \dot{\mathbf{v}}_\mathbf{k} \quad : \quad \textit{GE3 solver} \\
\mathbf{v_{k+1}} &= \mathbf{v_k} + \frac{h}{2} \cdot (3 \cdot \dot{\mathbf{v}}_\mathbf{k} - \dot{\mathbf{v}}_\mathbf{k-1}) \quad : \quad \textit{AB2 solver} \\
\dot{\mathbf{v}}_\mathbf{k} &= \mathbf{A}^2 \cdot \mathbf{x_k} + \mathbf{B} \cdot \mathbf{v_k} \quad : \quad \textit{model equations}
\end{aligned}$$

Plugging the model equations into the two sets of solver equations, we obtain:

$$\begin{aligned}
\mathbf{x_{k+1}} &= 2 \cdot \mathbf{x_k} - \mathbf{x_{k-1}} + (\mathbf{A} \cdot h)^2 \cdot \mathbf{x_k} + (\mathbf{B} \cdot h) \cdot (h \cdot \mathbf{v_k}) \\
(h \cdot \mathbf{v_{k+1}}) &= (h \cdot \mathbf{v_k}) + \frac{3}{2} \cdot (\mathbf{A} \cdot h)^2 \cdot \mathbf{x_k} + \frac{3}{2} \cdot (\mathbf{B} \cdot h) \cdot (h \cdot \mathbf{v_k}) \\
&\quad - \frac{1}{2} \cdot (\mathbf{A} \cdot h)^2 \cdot \mathbf{x_{k-1}} - \frac{1}{2} \cdot (\mathbf{B} \cdot h) \cdot (h \cdot \mathbf{v_{k-1}})
\end{aligned}$$

Non-linear Velocity Models III

This can be rewritten in a matrix/vector form:

$$\begin{pmatrix} \mathbf{x_k} \\ h \cdot \mathbf{v_k} \\ \mathbf{x_{k+1}} \\ h \cdot \mathbf{v_{k+1}} \end{pmatrix} = \mathbf{F} \cdot \begin{pmatrix} \mathbf{x_{k-1}} \\ h \cdot \mathbf{v_{k-1}} \\ \mathbf{x_k} \\ h \cdot \mathbf{v_k} \end{pmatrix}$$

where:

$$\mathbf{F} = \begin{pmatrix} \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{I}^{(n)} & \mathbf{Z}^{(n)} \\ \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{I}^{(n)} \\ -\mathbf{I}^{(n)} & \mathbf{Z}^{(n)} & \left[2 \cdot \mathbf{I}^{(n)} + (\mathbf{A} \cdot h)^2\right] & \mathbf{B} \cdot h \\ -\frac{1}{2} \cdot (\mathbf{A} \cdot h)^2 & -\frac{1}{2} \cdot (\mathbf{B} \cdot h) & \frac{3}{2} \cdot (\mathbf{A} \cdot h)^2 & \left[\mathbf{I}^{(n)} + \frac{3}{2} \cdot (\mathbf{B} \cdot h)\right] \end{pmatrix}$$

Non-linear Velocity Models III

This can be rewritten in a matrix/vector form:

$$\begin{pmatrix} \mathbf{x_k} \\ h \cdot \mathbf{v_k} \\ \mathbf{x_{k+1}} \\ h \cdot \mathbf{v_{k+1}} \end{pmatrix} = \mathbf{F} \cdot \begin{pmatrix} \mathbf{x_{k-1}} \\ h \cdot \mathbf{v_{k-1}} \\ \mathbf{x_k} \\ h \cdot \mathbf{v_k} \end{pmatrix}$$

where:

$$\mathbf{F} = \begin{pmatrix} \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{I}^{(n)} & \mathbf{Z}^{(n)} \\ \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{I}^{(n)} \\ -\mathbf{I}^{(n)} & \mathbf{Z}^{(n)} & \left[2 \cdot \mathbf{I}^{(n)} + (\mathbf{A} \cdot h)^2\right] & \mathbf{B} \cdot h \\ -\frac{1}{2} \cdot (\mathbf{A} \cdot h)^2 & -\frac{1}{2} \cdot (\mathbf{B} \cdot h) & \frac{3}{2} \cdot (\mathbf{A} \cdot h)^2 & \left[\mathbf{I}^{(n)} + \frac{3}{2} \cdot (\mathbf{B} \cdot h)\right] \end{pmatrix}$$

When plotting the stability domain of the **GE3/AB2 algorithm**, the elements of the **B**-matrix cannot be chosen independently of those of the **A**-matrix. They must be chosen such that the overall system has its eigenvalues located on the unit circle.

Non-linear Velocity Models III

This can be rewritten in a matrix/vector form:

$$\begin{pmatrix} \mathbf{x_k} \\ h \cdot \mathbf{v_k} \\ \mathbf{x_{k+1}} \\ h \cdot \mathbf{v_{k+1}} \end{pmatrix} = \mathbf{F} \cdot \begin{pmatrix} \mathbf{x_{k-1}} \\ h \cdot \mathbf{v_{k-1}} \\ \mathbf{x_k} \\ h \cdot \mathbf{v_k} \end{pmatrix}$$

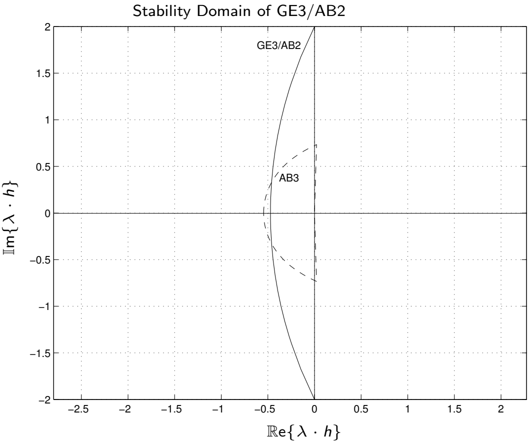
where:

$$\mathbf{F} = \begin{pmatrix} \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{I}^{(n)} & \mathbf{Z}^{(n)} \\ \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{Z}^{(n)} & \mathbf{I}^{(n)} \\ -\mathbf{I}^{(n)} & \mathbf{Z}^{(n)} & \left[2 \cdot \mathbf{I}^{(n)} + (\mathbf{A} \cdot h)^2\right] & \mathbf{B} \cdot h \\ -\frac{1}{2} \cdot (\mathbf{A} \cdot h)^2 & -\frac{1}{2} \cdot (\mathbf{B} \cdot h) & \frac{3}{2} \cdot (\mathbf{A} \cdot h)^2 & \left[\mathbf{I}^{(n)} + \frac{3}{2} \cdot (\mathbf{B} \cdot h)\right] \end{pmatrix}$$

When plotting the stability domain of the **GE3/AB2 algorithm**, the elements of the **B**-matrix cannot be chosen independently of those of the **A**-matrix. They must be chosen such that the overall system has its eigenvalues located on the unit circle.

Since this is a third-order accurate linear explicit multi-step method similar in scope to AB3, we decided to plot the stability domain of AB3 on top of the stability domain of GE3/AB2.

Non-linear Velocity Models IV



Non-linear Velocity Models V

- This time around, we hit the mark. The **GE3/AB2 algorithm** beats AB3 by leaps and bounds when simulating marginally stable second-derivative systems, i.e., second-derivative systems with their dominant eigenvalues located either on or at least in the vicinity of the imaginary axis.

Non-linear Velocity Models V

- This time around, we hit the mark. The **GE3/AB2 algorithm** beats AB3 by leaps and bounds when simulating marginally stable second-derivative systems, i.e., second-derivative systems with their dominant eigenvalues located either on or at least in the vicinity of the imaginary axis.

- Although this algorithm is restricted to the simulation of second-derivative systems, these are so frequent in practice that this could turn out to be a significant discovery.

Non-linear Velocity Models V

- This time around, we hit the mark. The **GE3/AB2 algorithm** beats AB3 by leaps and bounds when simulating marginally stable second-derivative systems, i.e., second-derivative systems with their dominant eigenvalues located either on or at least in the vicinity of the imaginary axis.

- Although this algorithm is restricted to the simulation of second-derivative systems, these are so frequent in practice that this could turn out to be a significant discovery.

- Especially for the case of real-time simulation of oscillatory mechanical systems, the GE3/AB2 algorithm could offer a highly efficient and therefore attractive alternative to traditional ODE solvers.

Other Godunov Methods

- Classes of both explicit and implicit integration algorithms of different orders of approximation accuracy for second-derivative systems can be derived using *Newton-Gregory polynomials*.

Other Godunov Methods

- Classes of both explicit and implicit integration algorithms of different orders of approximation accuracy for second-derivative systems can be derived using *Newton-Gregory polynomials*.
- To this end, we develop $\mathbf{x}(t)$ into a Newton-Gregory backward polynomial around t_{k+1}. We then compute the second derivative of the Newton-Gregory polynomial. Evaluating this second derivative polynomial for $s = -1$, we obtain the class of *explicit Godunov schemes*. Evaluating the second derivative polynomial for $s = 0$, we obtain the class of *implicit Godunov methods*.

Other Godunov Methods

- Classes of both explicit and implicit integration algorithms of different orders of approximation accuracy for second-derivative systems can be derived using *Newton-Gregory polynomials*.

- To this end, we develop $\mathbf{x}(t)$ into a Newton-Gregory backward polynomial around t_{k+1}. We then compute the second derivative of the Newton-Gregory polynomial. Evaluating this second derivative polynomial for $s = -1$, we obtain the class of *explicit Godunov schemes*. Evaluating the second derivative polynomial for $s = 0$, we obtain the class of *implicit Godunov methods*.

- We shall denote the explicit Godunov scheme of order n as $\mathbf{GE_n}$, and the implicit Godunov algorithm of the same order as $\mathbf{GI_n}$. The enhanced algorithms, that also compute the velocity vector, are denoted as $\mathbf{GE_n/AB_{n-1}}$ and $\mathbf{GI_n/BDF_{n-1}}$, respectively.

Other Godunov Methods II

The resulting algorithms are:

$$GE3: \quad \mathbf{x_{k+1}} = 2 \cdot \mathbf{x_k} - \mathbf{x_{k-1}} + h^2 \cdot \mathbf{\ddot{x}_k}$$

$$GE4: \quad \mathbf{x_{k+1}} = \frac{20}{11} \cdot \mathbf{x_k} - \frac{6}{11} \cdot \mathbf{x_{k-1}} - \frac{4}{11} \cdot \mathbf{x_{k-2}} + \frac{1}{11} \cdot \mathbf{x_{k-3}} + \frac{12}{11} \cdot h^2 \cdot \mathbf{\ddot{x}_k}$$

$$GE5: \quad \mathbf{x_{k+1}} = \frac{3}{2} \cdot \mathbf{x_k} + \frac{2}{5} \cdot \mathbf{x_{k-1}} - \frac{7}{5} \cdot \mathbf{x_{k-2}} + \frac{3}{5} \cdot \mathbf{x_{k-3}} - \frac{1}{10} \cdot \mathbf{x_{k-4}} + \frac{6}{5} \cdot h^2 \cdot \mathbf{\ddot{x}_k}$$

$$GI2: \quad \mathbf{x_{k+1}} = 2 \cdot \mathbf{x_k} - \mathbf{x_{k-1}} + h^2 \cdot \mathbf{\ddot{x}_{k+1}}$$

$$GI3: \quad \mathbf{x_{k+1}} = \frac{5}{2} \cdot \mathbf{x_k} - 2 \cdot \mathbf{x_{k-1}} + \frac{1}{2} \cdot \mathbf{x_{k-2}} + \frac{1}{2} \cdot h^2 \cdot \mathbf{\ddot{x}_{k+1}}$$

$$GI4: \quad \mathbf{x_{k+1}} = \frac{104}{35} \cdot \mathbf{x_k} - \frac{114}{35} \cdot \mathbf{x_{k-1}} + \frac{56}{35} \cdot \mathbf{x_{k-2}} - \frac{11}{35} \cdot \mathbf{x_{k-3}} + \frac{12}{35} \cdot h^2 \cdot \mathbf{\ddot{x}_{k+1}}$$

$$GI5: \quad \mathbf{x_{k+1}} = \frac{154}{45} \cdot \mathbf{x_k} - \frac{214}{45} \cdot \mathbf{x_{k-1}} + \frac{52}{15} \cdot \mathbf{x_{k-2}} - \frac{61}{45} \cdot \mathbf{x_{k-3}} + \frac{2}{9} \cdot \mathbf{x_{k-4}} + \frac{12}{45} \cdot h^2 \cdot \mathbf{\ddot{x}_{k+1}}$$

Other Godunov Methods III

The algorithms can be summarized using the following α-vectors and β-matrices:

$$\alpha_{\mathrm{GE}} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ \frac{12}{11} \\ \frac{6}{5} \end{pmatrix} \quad ; \quad \beta_{\mathrm{GE}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & 0 \\ \frac{20}{11} & -\frac{6}{11} & -\frac{4}{11} & \frac{1}{11} & 0 \\ \frac{3}{2} & \frac{2}{5} & -\frac{7}{5} & \frac{3}{5} & -\frac{1}{10} \end{pmatrix}$$

$$\alpha_{\mathrm{GI}} = \begin{pmatrix} 0 \\ 1 \\ \frac{1}{2} \\ \frac{12}{35} \\ \frac{12}{45} \end{pmatrix} \quad ; \quad \beta_{\mathrm{GI}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & 0 \\ \frac{5}{2} & -2 & \frac{1}{2} & 0 & 0 \\ \frac{104}{35} & -\frac{114}{35} & \frac{56}{35} & -\frac{11}{35} & 0 \\ \frac{154}{45} & -\frac{214}{45} & \frac{52}{15} & -\frac{61}{45} & \frac{2}{9} \end{pmatrix}$$

Other Godunov Methods IV

Unfortunately, all of these methods have $\mathbf{F}$-matrices that are *even functions* in $\mathbf{A} \cdot h$. Thus, none of these methods can be expected to offer an asymptotic region for eigenvalues located along the real axis. In fact, all of the above techniques are unstable everywhere in the vicinity of the origin, with the exception of a section of the imaginary axis in the vicinity of the origin, where they exhibit marginal stability, as they should.

Other Godunov Methods V

Let us plot the damping properties of some of these algorithms up and down along the imaginary axis:

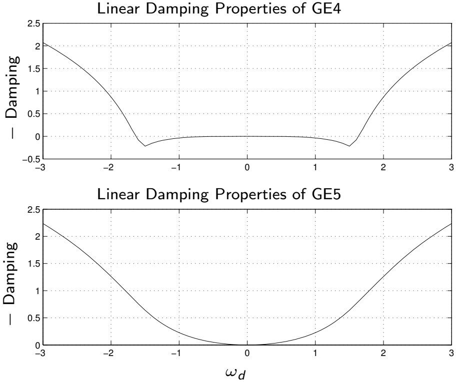


Figure: Damping properties of GE4 and GE5 along the imaginary axis

Other Godunov Methods VI

We notice that:

Other Godunov Methods VI

We notice that:

- All $\mathbf{GE_i}$ algorithms can be used for *linear conservation laws* only.

Other Godunov Methods VI

We notice that:

- All **GE$_{\text{i}}$** algorithms can be used for *linear conservation laws* only.
- The asymptotic regions of these algorithms up and down the imaginary axis are shrinking for increasing orders. Whereas **GE3** has an asymptotic region $\in [-2j, +2j]$, **GE4** has an asymptotic region $\in (-j, +j)$, and **GE5** has an asymptotic region $\in (-0.2j, +0.2j)$. This is disappointing.

Other Godunov Methods VI

We notice that:

- All **GE$_i$** algorithms can be used for *linear conservation laws* only.

- The asymptotic regions of these algorithms up and down the imaginary axis are shrinking for increasing orders. Whereas **GE3** has an asymptotic region $\in [-2j, +2j]$, **GE4** has an asymptotic region $\in (-j, +j)$, and **GE5** has an asymptotic region $\in (-0.2j, +0.2j)$. This is disappointing.

- Engineers will likely shrug these algorithms off anyway, because there aren't many real-life engineering applications that call for the simulation of linear conservation laws.

Other Godunov Methods VII

Let us now discuss the *implicit Godunov schemes*. Their damping properties along the imaginary axis are:

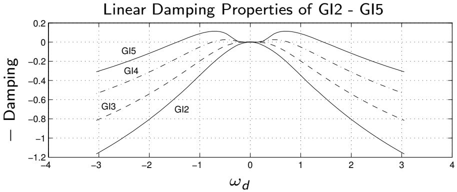


Figure: Damping properties of GI2 . . . GI5 along the imaginary axis

Other Godunov Methods VIII

- All of these algorithms could be used for the simulation of *linear conservation laws* as well, but there is no good reason, why we would ever want to do so. These algorithms have no advantages over their explicit brethren. They are only less efficient.

Other Godunov Methods VIII

- All of these algorithms could be used for the simulation of *linear conservation laws* as well, but there is no good reason, why we would ever want to do so. These algorithms have no advantages over their explicit brethren. They are only less efficient.

- These methods exhibit positive damping, i.e., they have stable regions. Thus, we might consider using them for the simulation of *damped mechanical systems*.

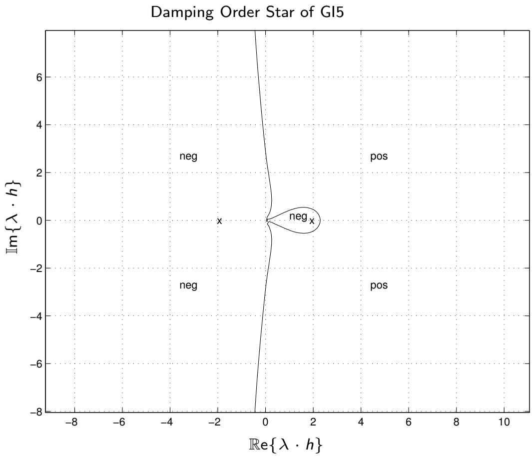
Other Godunov Methods VIII

- All of these algorithms could be used for the simulation of *linear conservation laws* as well, but there is no good reason, why we would ever want to do so. These algorithms have no advantages over their explicit brethren. They are only less efficient.

- These methods exhibit positive damping, i.e., they have stable regions. Thus, we might consider using them for the simulation of *damped mechanical systems*.

- Unfortunately, we cannot do this either. To understand why, we may look at the damping order star of e.g. **GI5**.

Other Godunov Methods IX



Other Godunov Methods X

- The locus of zero damping error approximates the imaginary axis, which is nice.

Other Godunov Methods X

- The locus of zero damping error approximates the imaginary axis, which is nice.
- Unfortunately, all of the algorithms of the GI class have a pole pair symmetric to the imaginary axis. Thus, all of these algorithms have a pole on the negative real axis.

Other Godunov Methods X

- The locus of zero damping error approximates the imaginary axis, which is nice.
- Unfortunately, all of the algorithms of the GI class have a pole pair symmetric to the imaginary axis. Thus, all of these algorithms have a pole on the negative real axis.
- A pole on the negative real axis means that there is a region around that pole with very strong negative damping, i.e., any eigenvalue that "falls into this hole" will cause the simulation to blow up.

Other Godunov Methods X

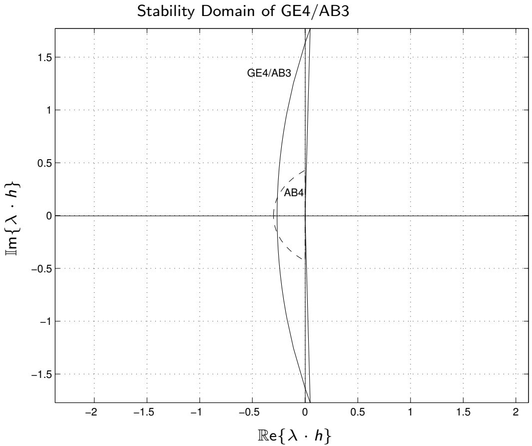
- The locus of zero damping error approximates the imaginary axis, which is nice.
- Unfortunately, all of the algorithms of the GI class have a pole pair symmetric to the imaginary axis. Thus, all of these algorithms have a pole on the negative real axis.
- A pole on the negative real axis means that there is a region around that pole with very strong negative damping, i.e., any eigenvalue that "falls into this hole" will cause the simulation to blow up.
- For this reason, **poles on the negative real axis are something we must shun away from**.

Other Godunov Methods X

- The locus of zero damping error approximates the imaginary axis, which is nice.
- Unfortunately, all of the algorithms of the GI class have a pole pair symmetric to the imaginary axis. Thus, all of these algorithms have a pole on the negative real axis.
- A pole on the negative real axis means that there is a region around that pole with very strong negative damping, i.e., any eigenvalue that "falls into this hole" will cause the simulation to blow up.
- For this reason, **poles on the negative real axis are something we must shun away from**.

Let us now extend the idea of the **GE3/AB2 algorithm** to higher orders of approximation accuracy.

Other Godunov Methods XI



Other Godunov Methods XII

- Also **GE4/AB3** beats AB4 by leaps and bounds when simulating second-derivative systems with their dominant eigenvalues located either on or at least in the vicinity of the imaginary axis.

Other Godunov Methods XII

- Also **GE4/AB3** beats AB4 by leaps and bounds when simulating second-derivative systems with their dominant eigenvalues located either on or at least in the vicinity of the imaginary axis.

- The **GE5/AB4 algorithm** is unfortunately unstable.

Other Godunov Methods XII

- Also **GE4/AB3** beats AB4 by leaps and bounds when simulating second-derivative systems with their dominant eigenvalues located either on or at least in the vicinity of the imaginary axis.
- The **GE5/AB4 algorithm** is unfortunately unstable.
- The **GE5/ABM4 algorithm** turns out to be unstable as well.

Other Godunov Methods XIII

- We can also try to enhance the *implicit Godunov schemes* in the same fashion.

Other Godunov Methods XIII

- We can also try to enhance the *implicit Godunov schemes* in the same fashion.
- It makes sense to pair them with BDF algorithms of one order lower.

Other Godunov Methods XIII

- We can also try to enhance the *implicit Godunov schemes* in the same fashion.
- It makes sense to pair them with BDF algorithms of one order lower.
- For example, the **GI3/BDF2 algorithm** can be written as follows:

$$\begin{aligned}
\mathbf{x_{k+1}} &= \frac{5}{2} \cdot \mathbf{x_k} - 2 \cdot \mathbf{x_{k-1}} + \frac{1}{2} \cdot \mathbf{x_{k-2}} + \frac{h^2}{2} \cdot \ddot{\mathbf{x}}_{\mathbf{k+1}} \\
h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} &= \frac{4 \cdot h}{3} \cdot \dot{\mathbf{x}}_{\mathbf{k}} - \frac{h}{3} \cdot \dot{\mathbf{x}}_{\mathbf{k-1}} + \frac{2 \cdot h^2}{3} \cdot \ddot{\mathbf{x}}_{\mathbf{k+1}}
\end{aligned}$$

Other Godunov Methods XIII

- We can also try to enhance the *implicit Godunov schemes* in the same fashion.
- It makes sense to pair them with BDF algorithms of one order lower.
- For example, the **GI3/BDF2 algorithm** can be written as follows:

$$\begin{aligned}
\mathbf{x_{k+1}} &= \frac{5}{2} \cdot \mathbf{x_k} - 2 \cdot \mathbf{x_{k-1}} + \frac{1}{2} \cdot \mathbf{x_{k-2}} + \frac{h^2}{2} \cdot \ddot{\mathbf{x}}_{\mathbf{k+1}} \\
h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} &= \frac{4 \cdot h}{3} \cdot \dot{\mathbf{x}}_{\mathbf{k}} - \frac{h}{3} \cdot \dot{\mathbf{x}}_{\mathbf{k-1}} + \frac{2 \cdot h^2}{3} \cdot \ddot{\mathbf{x}}_{\mathbf{k+1}}
\end{aligned}$$

- Some of these algorithms are A-stable, but none are L-stable.

Other Godunov Methods XIII

- We can also try to enhance the *implicit Godunov schemes* in the same fashion.
- It makes sense to pair them with BDF algorithms of one order lower.
- For example, the **GI3/BDF2 algorithm** can be written as follows:

$$\begin{aligned} \mathbf{x_{k+1}} &= \frac{5}{2} \cdot \mathbf{x_k} - 2 \cdot \mathbf{x_{k-1}} + \frac{1}{2} \cdot \mathbf{x_{k-2}} + \frac{h^2}{2} \cdot \ddot{\mathbf{x}}_{\mathbf{k+1}} \\ h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} &= \frac{4 \cdot h}{3} \cdot \dot{\mathbf{x}}_{\mathbf{k}} - \frac{h}{3} \cdot \dot{\mathbf{x}}_{\mathbf{k-1}} + \frac{2 \cdot h^2}{3} \cdot \ddot{\mathbf{x}}_{\mathbf{k+1}} \end{aligned}$$

- Some of these algorithms are A-stable, but none are L-stable.
- Unfortunately, none of these algorithms gives rise to an asymptotic region all around the origin.

The Newmark Algorithm

The idea of developing second-derivative solvers that do not contain a first derivative in the formula was reasonable when dealing with linear conservation laws. Yet, when dealing with general second-derivative models, this turned out to be an unnecessary restriction, as the first derivative term got reinserted into the discrete-time model through the model equations.

The Newmark Algorithm

The idea of developing second-derivative solvers that do not contain a first derivative in the formula was reasonable when dealing with linear conservation laws. Yet, when dealing with general second-derivative models, this turned out to be an unnecessary restriction, as the first derivative term got reinserted into the discrete-time model through the model equations.

One second-derivative method that lets go of this unnecessary restriction is *Newmark's algorithm*, a general-purpose second-derivative solver that has been known since 1959. It can be written as follows:

$$\begin{aligned}
\mathbf{x_{k+1}} &= \mathbf{x_k} + h \cdot \dot{\mathbf{x}}_\mathbf{k} + \frac{h^2}{2} \cdot [(1-\vartheta_1) \cdot \ddot{\mathbf{x}}_\mathbf{k} + \vartheta_1 \cdot \ddot{\mathbf{x}}_\mathbf{k+1}] \\
h \cdot \dot{\mathbf{x}}_\mathbf{k+1} &= h \cdot \dot{\mathbf{x}}_\mathbf{k} + h^2 \cdot [(1-\vartheta_2) \cdot \ddot{\mathbf{x}}_\mathbf{k} + \vartheta_2 \cdot \ddot{\mathbf{x}}_\mathbf{k+1}]
\end{aligned}$$

The Newmark Algorithm II

- The method is clearly second-order accurate, as the solution for $\mathbf{x_{k+1}}$ approximates the Taylor-Series directly up to the quadratic term, whereas the solution for $\dot{\mathbf{x}}_{\mathbf{k+1}}$ approximates the Taylor-Series up to the linear term. Since the velocity vector gets always multiplied by the step size, h, the overall method must be second-order accurate.

The Newmark Algorithm II

- The method is clearly second-order accurate, as the solution for $\mathbf{x_{k+1}}$ approximates the Taylor-Series directly up to the quadratic term, whereas the solution for $\dot{\mathbf{x}}_{\mathbf{k+1}}$ approximates the Taylor-Series up to the linear term. Since the velocity vector gets always multiplied by the step size, h, the overall method must be second-order accurate.

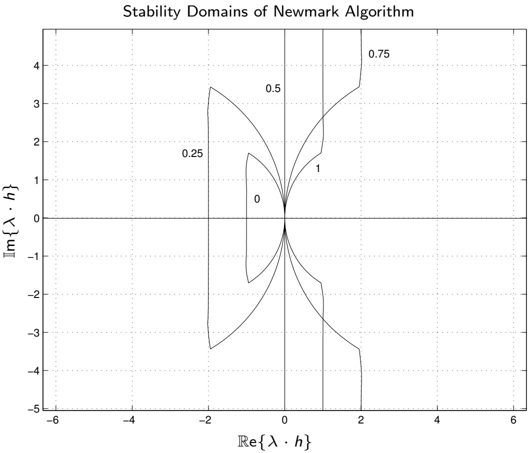
- It is a *ϑ-method* with two fudge parameters, ϑ_1 and ϑ_2. For $\vartheta_1 = \vartheta_2 = 0$, the method is explicit; for all other combinations of ϑ_1 and ϑ_2, the method is implicit.

The Newmark Algorithm II

- The method is clearly second-order accurate, as the solution for $\mathbf{x_{k+1}}$ approximates the Taylor-Series directly up to the quadratic term, whereas the solution for $\dot{\mathbf{x}}_{\mathbf{k+1}}$ approximates the Taylor-Series up to the linear term. Since the velocity vector gets always multiplied by the step size, h, the overall method must be second-order accurate.
- It is a *ϑ-method* with two fudge parameters, ϑ_1 and ϑ_2. For $\vartheta_1 = \vartheta_2 = 0$, the method is explicit; for all other combinations of ϑ_1 and ϑ_2, the method is implicit.

Let us plot the stability domains of the Newmark algorithm for $\vartheta_1 = \vartheta_2 = \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

The Newmark Algorithm III



The Newmark Algorithm IV

- The stability domains of the algorithms are symmetric to $\vartheta_1 = \vartheta_2 = 0.5$. This is evident from the symmetry of the formulae themselves.

The Newmark Algorithm IV

- The stability domains of the algorithms are symmetric to $\vartheta_1 = \vartheta_2 = 0.5$. This is evident from the symmetry of the formulae themselves.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0$ is *explicit*. This can be an interesting algorithm for *real-time simulation of mechanical systems*, and the algorithm is often used for just that purpose. The algorithm exhibits a stable region in the left half plane that looks like an ascending half moon. The stable region is limited by $\mathbb{R}\mathrm{e}\{\lambda \cdot h\} \geq -1.0$.

The Newmark Algorithm IV

- The stability domains of the algorithms are symmetric to $\vartheta_1 = \vartheta_2 = 0.5$. This is evident from the symmetry of the formulae themselves.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0$ is *explicit*. This can be an interesting algorithm for *real-time simulation of mechanical systems*, and the algorithm is often used for just that purpose. The algorithm exhibits a stable region in the left half plane that looks like an ascending half moon. The stable region is limited by $\mathbb{R}\mathrm{e}\{\lambda \cdot h\} \geq -1.0$.

- The algorithms with $\vartheta_1 = \vartheta_2 \in (0, 0.5)$ are probably not of much interest, as they are implicit, yet neither F-stable nor A-stable.

The Newmark Algorithm IV

- The stability domains of the algorithms are symmetric to $\vartheta_1 = \vartheta_2 = 0.5$. This is evident from the symmetry of the formulae themselves.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0$ is *explicit*. This can be an interesting algorithm for *real-time simulation of mechanical systems*, and the algorithm is often used for just that purpose. The algorithm exhibits a stable region in the left half plane that looks like an ascending half moon. The stable region is limited by $\mathbb{R}\mathrm{e}\{\lambda \cdot h\} \geq -1.0$.

- The algorithms with $\vartheta_1 = \vartheta_2 \in (0, 0.5)$ are probably not of much interest, as they are implicit, yet neither F-stable nor A-stable.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0.5$ is *F-stable*.

The Newmark Algorithm IV

- The stability domains of the algorithms are symmetric to $\vartheta_1 = \vartheta_2 = 0.5$. This is evident from the symmetry of the formulae themselves.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0$ is *explicit*. This can be an interesting algorithm for *real-time simulation of mechanical systems*, and the algorithm is often used for just that purpose. The algorithm exhibits a stable region in the left half plane that looks like an ascending half moon. The stable region is limited by $\mathbb{R}\mathrm{e}\{\lambda \cdot h\} \geq -1.0$.

- The algorithms with $\vartheta_1 = \vartheta_2 \in (0, 0.5)$ are probably not of much interest, as they are implicit, yet neither F-stable nor A-stable.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0.5$ is *F-stable*.

- The algorithms with $\vartheta_1 = \vartheta_2 \in (0.5, 1]$ are *A-stable*.

The Newmark Algorithm IV

- The stability domains of the algorithms are symmetric to $\vartheta_1 = \vartheta_2 = 0.5$. This is evident from the symmetry of the formulae themselves.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0$ is *explicit*. This can be an interesting algorithm for *real-time simulation of mechanical systems*, and the algorithm is often used for just that purpose. The algorithm exhibits a stable region in the left half plane that looks like an ascending half moon. The stable region is limited by $\mathbb{R}\mathrm{e}\{\lambda \cdot h\} \geq -1.0$.

- The algorithms with $\vartheta_1 = \vartheta_2 \in (0, 0.5)$ are probably not of much interest, as they are implicit, yet neither F-stable nor A-stable.

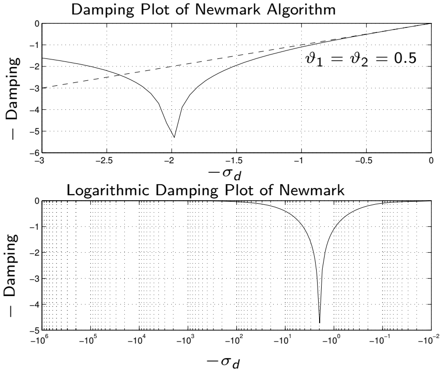
- The algorithm with $\vartheta_1 = \vartheta_2 = 0.5$ is *F-stable*.

- The algorithms with $\vartheta_1 = \vartheta_2 \in (0.5, 1]$ are *A-stable*.

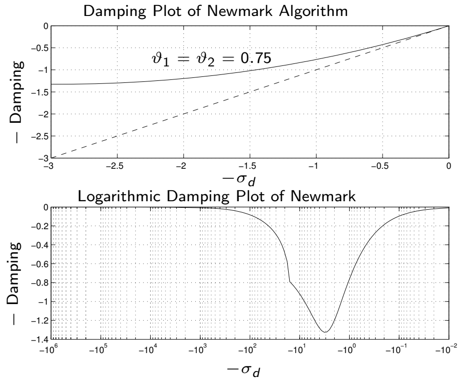
- As marginal stability of all algorithms with $\vartheta_1 = \vartheta_2 > 0.5$ reaches all the way to infinity, none of these algorithms can be *L-stable*. Their damping at infinity is exactly zero.

The Newmark Algorithm V

Let us look at some damping plots:



The Newmark Algorithm VI



The Newmark Algorithm VII

- Due to the poor damping characteristics far to the left, the Newmark algorithms aren't well suited for the simulation of stiff systems.

The Newmark Algorithm VII

- Due to the poor damping characteristics far to the left, the Newmark algorithms aren't well suited for the simulation of stiff systems.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0.5$ may be a good choice for simulating conservative mechanical systems.

The Newmark Algorithm VII

- Due to the poor damping characteristics far to the left, the Newmark algorithms aren't well suited for the simulation of stiff systems.

- The algorithm with $\vartheta_1 = \vartheta_2 = 0.5$ may be a good choice for simulating conservative mechanical systems.

- The algorithms with $\vartheta_1 = \vartheta_2 \geq 0.5$ can be used for the simulation of mechanical systems exhibiting oscillatory behavior, such as earthquakes or elastic systems.

Conclusions

- In the last two presentation, we have talked about linear multi-step algorithms, specifically designed for the simulation of second derivative systems, i.e., models that contain the second derivatives of the partial state vector explicitly in their model equations.

Conclusions

- In the last two presentation, we have talked about linear multi-step algorithms, specifically designed for the simulation of second derivative systems, i.e., models that contain the second derivatives of the partial state vector explicitly in their model equations.

- The results obtained were rather modest.

Conclusions

- In the last two presentation, we have talked about linear multi-step algorithms, specifically designed for the simulation of second derivative systems, i.e., models that contain the second derivatives of the partial state vector explicitly in their model equations.
- The results obtained were rather modest.
- We still don't have any higher-order algorithms for simulating second-derivative systems.

Conclusions

- In the last two presentation, we have talked about linear multi-step algorithms, specifically designed for the simulation of second derivative systems, i.e., models that contain the second derivatives of the partial state vector explicitly in their model equations.
- The results obtained were rather modest.
- We still don't have any higher-order algorithms for simulating second-derivative systems.
- The class of *Newmark algorithms*, which is almost as trivial as forward Euler and has been known for more than half a century already, is still state-of-the-art, is being used in engineering practice, especially for the real-time simulation of mechanical systems, and is still being discussed in papers.

Conclusions II

- The reason for this somewhat surprising fact is that the class of second-derivative systems has been almost entirely ignored by the applied mathematicians.

Conclusions II

- The reason for this somewhat surprising fact is that the class of second-derivative systems has been almost entirely ignored by the applied mathematicians.

- As applied mathematicians have worked long and hard on developing numerical ODE solvers for state-space descriptions, it has become quite difficult to still obtain new results. This research area has been ploughed rather well.

Conclusions II

- The reason for this somewhat surprising fact is that the class of second-derivative systems has been almost entirely ignored by the applied mathematicians.
- As applied mathematicians have worked long and hard on developing numerical ODE solvers for state-space descriptions, it has become quite difficult to still obtain new results. This research area has been ploughed rather well.
- We were able to present some exciting new results in previous presentations, such as the new BDF algorithms of orders 7 ... 9, but we had to work very hard to find those, and we were actually lucky to be able to still make a mark.

Conclusions II

- The reason for this somewhat surprising fact is that the class of second-derivative systems has been almost entirely ignored by the applied mathematicians.
- As applied mathematicians have worked long and hard on developing numerical ODE solvers for state-space descriptions, it has become quite difficult to still obtain new results. This research area has been ploughed rather well.
- We were able to present some exciting new results in previous presentations, such as the new BDF algorithms of orders $7 \ldots 9$, but we had to work very hard to find those, and we were actually lucky to be able to still make a mark.
- **In contrast, the field of numerical ODE solvers for second-derivative systems is in its infancy. It should be easy to improve the state of the art, and therefore, this is an excellent ongoing research area for Ph.D. students interested in advancing simulation technology.**

References

1. Beamis, Christopher Paul (1990), *Solution of Second Order Differential Equations Using the Godunov Integration Method*, MS Thesis, Dept. of Electrical & Computer Engineering, University of Arizona, Tucson, AZ.