Lecture 3

Numerical Integration of DAEs

COMASIC (M2)
January 25th, 2017

Khalil Ghorbal
khalil.ghorbal@inria.fr
Index Reduction

- Given a DAE $F(x, \dot{x}, t)$, we have seen how to perform a structural analysis to numerically compute $\dot{x}$ function of $x$. The structural nonsingularity ensures that, generically, one can perform the computation following the block order suggested by the BLT decomposition. Thus, one is able to compute the numerical values of the derivatives given a consistent state of the system and carry on with a standard numerical integration.

- Note: the structural index is not always equal to the differentiation index.
Index Reduction

- Given a DAE $F(x, \dot{x}, t)$, we have seen how to perform a structural analysis to numerically compute $\dot{x}$ function of $x$. The structural nonsingularity ensures that, generically, one can perform the computation following the block order suggested by the BLT decomposition. Thus, one is able to compute the numerical values of the derivatives given a consistent state of the system and carry on with a standard numerical integration.

- Note: the structural index is not always equal to the differentiation index.
Consider the following DAE

\[
\begin{align*}
\dot{z} - \dot{x}y - x\dot{y} + 2x + y - 3 &= 0 \\
z - xy &= 0 \\
x + y - 2 &= 0
\end{align*}
\]

- Pantelides: structural index 1
- Differentiation index is 0 (simple linear system)
- (Hidden) Cancellation problems are undecidable in general
Consider the following DAE

\[ \dot{z} - \dot{x}y - x\dot{y} + 2x + y - 3 = 0 \]
\[ z - xy = 0 \]
\[ x + y - 2 = 0 \]

- Pantelides: structural index 1
- Differentiation index is 0 (simple linear system)
- (Hidden) Cancellation problems are undecidable in general
Consider the following DAE

\[ \dot{z} - xy - xy + 2x + y - 3 = 0 \]
\[ z - xy = 0 \]
\[ x + y - 2 = 0 \]

- Pantelides: structural index 1
- Differentiation index is 0 (simple linear system)
- (Hidden) Cancellation problems are undecidable in general
Consider the following DAE

\[
\begin{align*}
\dot{z} - x\dot{y} - xy + 2x + y - 3 &= 0 \\
z - xy &= 0 \\
x + y - 2 &= 0
\end{align*}
\]

- Pantelides: structural index 1
- Differentiation index is 0 (simple linear system)
- (Hidden) Cancellation problems are undecidable in general
Index reduction transforms a fully implicit DAE to a semi-explicit DAE

\[ \dot{x} = f(x, y, t) \]
\[ 0 = g(x, y, t) \]

Integration Schemes

- Backward Differentiation Formula (BDF)
- Orthogonal Collocation
1 BDF Method

2 Collocation (Overview)

3 Brief Introduction to Modelica
Backward Differentiation Formula (BDF)

Fixed Step Size

- **Implicit**: next value not explicitly given.
- **Linear multistep**: the next value is linearly related to the immediate previous (backward) values (eventually more than one).
- For a fixed step size $\delta > 0$, let $t_n = t_0 + n\delta$, and $x_s$ the approximate of the exact $x(t_s)$.
- The general Backward Differentiation Formula:

  \[ \dot{x}_n = \text{linear combination of } x_n, x_{n-1}, \ldots, x_0 \]

- For a Cauchy problem $\dot{x} = f(x, t), x(t_0) = x_0$, one obtains an implicit equation for $x_n$:

  \[ x_n = \sum_{n=1}^{q} a_n x_{1-n} + \delta b_q f(x_n, t_n) \]

  where the $a_n$ and $b_q$ depends only on the order $q$.
- E.g., $q = 1$ (BDF1): $x_n = x_{n-1} + \delta f(x_n, t_n)$ (a.k.a. Backward Euler)
Backward Differentiation Formula (BDF)

Fixed Step Size

- **Implicit**: next value not explicitly given.
- **Linear multistep**: the next value is linearly related to the immediate previous (backward) values (eventually more than one).
- For a fixed step size $\delta > 0$, let $t_n = t_0 + n\delta$, and $x_s$ the approximate of the exact $x(t_s)$.
- The general Backward Differentiation Formula:
  \[ \dot{x}_n = \text{linear combination of } x_n, x_{n-1}, \ldots, x_0 \]

- For a Cauchy problem $\dot{x} = f(x, t)$, $x(t_0) = x_0$, one obtains an implicit equation for $x_n$:
  \[ x_n = \sum_{n=1}^{q} a_n \, x_{1-n} + \delta \, b_q \, f(x_n, t_n) \]
  where the $a_n$ and $b_q$ depends only on the order $q$.
- E.g., $q = 1$ (BDF1): $x_n = x_{n-1} + \delta \, f(x_n, t_n)$ (a.k.a. Backward Euler)
Backward Differentiation Formula (BDF)

Fixed Step Size

- **Implicit**: next value not explicitly given.
- **Linear multistep**: the next value is linearly related to the immediate previous (backward) values (eventually more than one).
- For a fixed step size $\delta > 0$, let $t_n = t_0 + n\delta$, and $x_s$ the approximate of the exact $x(t_s)$.
- The general Backward Differentiation Formula:
  \[
  \dot{x}_n = \text{linear combination of } x_n, x_{n-1}, \ldots, x_0
  \]
- For a Cauchy problem $\dot{x} = f(x, t), x(t_0) = x_0$, one obtains an implicit equation for $x_n$:
  \[
  x_n = \sum_{n=1}^{q} a_n x_{1-n} + \delta b_q f(x_n, t_n)
  \]
  where the $a_n$ and $b_q$ depends only on the order $q$.
- E.g., $q = 1$ (BDF1): $x_n = x_{n-1} + \delta f(x_n, t_n)$ (a.k.a. Backward Euler)
Implicit: next value not explicitly given.

Linear multistep: the next value is linearly related to the immediate previous (backward) values (eventually more than one).

For a fixed step size $\delta > 0$, let $t_n = t_0 + n\delta$, and $x_s$ the approximate of the exact $x(t_s)$.

The general Backward Differentiation Formula:

$$\dot{x}_n = \text{linear combination of } x_n, x_{n-1}, \ldots, x_0$$

For a Cauchy problem $\dot{x} = f(x, t), x(t_0) = x_0$, one obtains an implicit equation for $x_n$:

$$x_n = \sum_{n=1}^{q} a_n x_{1-n} + \delta b_q f(x_n, t_n)$$

where the $a_n$ and $b_q$ depends only on the order $q$.

E.g., $q = 1$ (BDF1): $x_n = x_{n-1} + \delta f(x_n, t_n)$ (a.k.a. Backward Euler)
**Backward Differentiation Formula (BDF)**

**Fixed Step Size**

- **Implicit**: next value not explicitly given.
- **Linear multistep**: the next value is linearly related to the immediate previous (backward) values (eventually more than one).
- For a fixed step size \( \delta > 0 \), let \( t_n = t_0 + n\delta \), and \( x_s \) the approximate of the exact \( x(t_s) \).
- The general Backward Differentiation Formula:

  \[
  \dot{x}_n = \text{linear combination of } x_n, x_{n-1}, \ldots, x_0
  \]

- For a Cauchy problem \( \dot{x} = f(x, t), x(t_0) = x_0 \), one obtains an implicit equation for \( x_n \):

  \[
  x_n = \sum_{n=1}^{q} a_n \ x_{1-n} + \delta \ b_q \ f(x_n, t_n)
  \]

  where the \( a_n \) and \( b_q \) depends only on the order \( q \).
- E.g., \( q = 1 \) (BDF1): \( x_n = x_{n-1} + \delta \ f(x_n, t_n) \) (a.k.a. Backward Euler)
Coefficients of BDFq
Taylor Series (Sundials Implementation)

\[ \mathcal{I} x_n = x_n \] (Identity)
\[ \mathcal{N} x_n = x_{n+1} \] (Forward Shift)
\[ \mathcal{N}^{-1} x_n = x_{n-1} \] (Backward Shift)
\[ \mathcal{D} x_n = \dot{x}_n \] (Differential)
\[ \Delta = \mathcal{I} - \mathcal{N} \] (Backward Operator)

Observe that \( \mathcal{N} = (\mathcal{I} - \Delta)^{-1} \) (Operator Algebra)

\[ \mathcal{N} x_n = x_{n+1} = x_n + \delta \mathcal{D} x_n + \frac{\delta^2}{2!} \mathcal{D}^2 x_n + \frac{\delta^3}{3!} \mathcal{D}^3 x_n + \cdots \]
\[ = (\mathcal{I} + \delta \mathcal{D} + \frac{\delta^2}{2!} \mathcal{D}^2 + \frac{\delta^3}{3!} \mathcal{D}^3 x_n + \cdots) x_n \]
\[ = e^{\delta \mathcal{D}} x_n \]

Thus: \( \mathcal{N} = e^{\delta \mathcal{D}} \)
\[ \delta D = \ln(N) = \ln \left( (I - \Delta)^{-1} \right) = \Delta + \frac{1}{2} \Delta^2 + \frac{1}{3} \Delta^3 + \cdots \]

\[ \delta \dot{x}_n = \delta D x_n = \Delta x_n + \frac{1}{2} \Delta^2 x_n + \frac{1}{3} \Delta^3 x_n + \cdots \]

BDFq: truncate at order \( q \), for instance for \( q = 2 \)

\[ \delta \dot{x}_n = \Delta x_n + \frac{1}{2} \Delta^2 x_n = (x_n - x_{n-1}) + \frac{1}{2} (x_n - 2x_{n-1} + x_{n-2}) \]

Thus

\[ \delta \dot{x}_n = \frac{3}{2} x_n - 2x_{n-1} + \frac{1}{2} x_{n-2} \]

\[ x_n = \frac{4}{3} x_{n-1} - \frac{1}{3} x_{n-2} + \frac{2}{3} \delta f(x_n, t_n) \]
BDF for Semi-Explicit DAE of index 1

\[ \dot{x} = f(x, y, t) \]
\[ 0 = g(x, y) \]

Numerical integration using the BDF2 Scheme

Suppose \( x_{n-1} \) and \( x_{n-2} \) are known, \( x_n \) and \( y_n \) are computed by numerically solving the following system (e.g. Newton’s methods) at each iteration:

\[
\begin{align*}
x_n &= \frac{4}{3} x_{n-1} - \frac{1}{3} x_{n-2} + \frac{2}{3} \delta f(x_n, y_n, t_n) \\
0 &= g(x_n, y_n)
\end{align*}
\]

- BDF converges if \( m \leq 6 \): \( x_i - x(t_i) = y_i - y_i(t_i) = o(\delta^m) \)
- BDF requires a consistent initial condition
\[ \dot{x} = f(x, y, t) \]
\[ 0 = g(x, y) \]

**Numerical integration using the BDF2 Scheme**

Suppose \( x_{n-1} \) and \( x_{n-2} \) are known, \( x_n \) and \( y_n \) are computed by numerically solving the following system (e.g. Newton’s methods) at each iteration:

\[
x_n = \frac{4}{3} x_{n-1} - \frac{1}{3} x_{n-2} + \frac{2}{3} \delta f(x_n, y_n, t_n)
\]

\[ 0 = g(x_n, y_n) \]

- BDF converges if \( m \leq 6 \):
  \[ x_i - x(t_i) = y_i - y_i(t_i) = o(\delta^m) \]
- BDF requires a consistent initial condition
Newton’s Methods: Generalization

Could be used to numerically approximate the roots of $F : \mathbb{R}^k \rightarrow \mathbb{R}^k$ ($F_i$ continuously differentiable functions):

- Multiply by the inverse of the Jacobian

  $$x_{n+1} = x_n - J_F^{-1}(x_n)F(x_n)$$

- Or solve the system of linear equations

  $$J_F(x_n)(x_{n+1} - x_n) = -F(x_n)$$

Under some assumptions, the method converges quadratically towards a root of $F$. 
Outline

1 BDF Method

2 Collocation (Overview)

3 Brief Introduction to Modelica
• Collocate means approximate/study an unknown function by means of other “simpler” functions
• For instance using polynomial or trigonometric functions

**Weierstrass Theorem (1885)**
If \( x(t) \) is a continuous function on \([a, b]\), then for any given \( \epsilon > 0 \), there exists a polynomial \( p(t) \) such that

\[
\max_{t \in [a,b]} |x(t) - p(t)| < \epsilon
\]

The theorem has a constructive proof using Bernstein polynomials.

**Integration by Collocation**
Unlike BDF, Collocation could be used to construct at once \( n \) points \((t_1, x_1), \ldots, (t_n, x_n)\) such that \( x_i \) approximates the function \( x(t) \) at \( t_i \) for all \( i \). Useful for Consistent Initialization.
Collocation

- Collocate means approximate/study an unknown function by means of other “simpler” functions
- For instance using **polynomial** or trigonometric functions

**Weierstrass Theorem** (1885)

If $x(t)$ is a continuous function on $[a, b]$, then for any given $\epsilon > 0$, there exists a polynomial $p(t)$ such that

$$\max_{t \in [a,b]} |x(t) - p(t)| < \epsilon$$

The theorem has a constructive proof using **Bernstein polynomials**.

**Integration by Collocation**

Unlike BDF, Collocation could be used to construct **at once** $n$ points $(t_1, x_1), \ldots, (t_n, x_n)$ such that $x_i$ approximates the function $x(t)$ at $t_i$ for all $i$. Useful for **Consistent Initialization**.
Collocation

- Collocate means approximate/study an unknown function by means of other “simpler” functions
- For instance using polynomial or trigonometric functions

**Weierstrass Theorem (1885)**

If \( x(t) \) is a continuous function on \([a, b]\), then for any given \( \epsilon > 0 \), there exists a polynomial \( p(t) \) such that

\[
\max_{t \in [a,b]} |x(t) - p(t)| < \epsilon
\]

The theorem has a constructive proof using Bernstein polynomials.

**Integration by Collocation**

Unlike BDF, Collocation could be used to construct at once \( n \) points \((t_1, x_1), \ldots, (t_n, x_n)\) such that \( x_i \) approximates the function \( x(t) \) at \( t_i \) for all \( i \). Useful for **Consistent Initialization**.
If $p(t)$ is a polynomial that approximates $x(t)$, solution of our semi-explicit DAE, such that $p(t_i) = x(t_i) = x_i$ for some known $t_i$, where $i = 1, \ldots, n$. Then, for each $i$, we can compute $x_i, y_i$ by solving

$$
\dot{p}(t_i) = f(p(t_i), y_i, t_i) \quad \quad 0 = g(p(t_i), y_i)
$$

- The collocation methods attempts to construct such $p$ as well as finding appropriate $t_i$.
- Observe that the polynomial $p$ may be dependent on $x_i$, the unknowns we want to determine.
- Again, Newton’s methods could be used to solve iteratively for $p(t_i)$ and hence $x_i$. 
Polynomial Interpolation

Suppose we have \( n \) points \((t_1, x_1), \ldots, (t_n, x_n)\), then we construct the polynomial \( p(t) \) such that:

\[
p(t_i) = x_i, \quad i = 1, \ldots, n
\]

- There exists a (unique) polynomial \( p \) of degree \( n \) interpolating the \( n \) points

\[
p(t) = \sum_{i=1}^{n} x_i L_i(t)
\]

- \( L_i \) are the Lagrange polynomials

\[
L_i(t) = \prod_{k=0, k\neq i}^{n} \frac{t - t_k}{t_i - t_k}, \quad i = 1, \ldots, n
\]

satisfying in particular \( L_i(t_j) = \delta_{ij} \) (Kronecker delta)
Determining the instant (nodes) $t_i$

We want to determine the instants $t_i \in [a, b]$ such that there exists positive weights $w_i$ which make the Gauss quadrature integral exact for the polynomial $p$, that is

$$I[p] = \int_a^b p(t) dt = \sum_{i=1}^{n} w_i p(t_i) = Q_n[p]$$

Without loss of generality, we can suppose that $a = -1$ and $b = 1$ since

$$\int_a^b f(t) dt = \frac{b - a}{2} \int_{-1}^{1} f \left( \frac{a + b}{2} + t \frac{b - a}{2} \right) dt .$$

**Gauss Quadrature Fundamental Theorem ($w(x) = 1$)**

If the $t_i$ are the zeros of the Legendre polynomial $\ell_n$, then there exist $n$ weights $w_i$ which make the Gauss quadrature integral exact for all polynomials of degree $2n - 1$ or less (in particular for our polynomial $p$ of degree $n$).
Legendre Polynomials

Rodrigues’ Formula

\[ \ell_n(t) = \frac{1}{2^n n!} \frac{d^n}{dt^n} (t^2 - 1)^n \]

- \( \ell_n \) has degree \( n \)
- Legendre polynomials are orthogonal

\[ <\ell_i, \ell_j> = \int_{-1}^{1} \ell_i(t) \ell_j(t) dt = \frac{2}{2n+1} \delta_{ij} \]

- \( \ell_0(t) = 1, \ell_1(t) = t, \ell_3(t) = \frac{1}{2} (5t^3 - 3t) \)
- There are effective methods to compute the zeros of \( \ell_n \) and \( w_i \) (Golub-Welsch algorithm)
- Thus we have the \( t_i \) which depend only on \( n \) and not on our original DAE system!
\[ \dot{x} = x + y \]
\[ 0 = x^2 + y^2 - 1 \]

- Suppose \([a, b] = [0, 0.1], \; n = 3\)
- The \(t_i\) are derived from \(\tau_i\) the roots of \(\ell_3(t): \; t_i = \frac{a + b}{2} + \tau_i \frac{b-a}{2}\)
- \(\tau_i \in \{-\sqrt{\frac{3}{5}}, 0, \sqrt{\frac{3}{5}}\}\)
- \(w_i = \int_a^b L_i(t) dt\) (Lagrange Polynomials)
- \(w_i \in \{\frac{1}{36}, \frac{2}{45}, \frac{1}{36}\}\)
- \(p(t) = x_1 L_1(t) + x_2 L_2(t) + x_3 L_3(t)\)
- \(\dot{p}(t) = x_1 \dot{L}_1(t) + x_2 \dot{L}_2(t) + x_3 \dot{L}_3(t) \quad (\dot{L}_i(t_j) \neq \delta_{ij})\)
System to Solve

\[
\begin{align*}
\dot{p}(t_1) &= x_1 + y_1 \\
\dot{p}(t_2) &= x_2 + y_2 \\
\dot{p}(t_3) &= x_3 + y_3 \\
0 &= x_1^2 + y_1^2 - 1 \\
0 &= x_2^2 + y_2^2 - 1 \\
0 &= x_3^2 + y_3^2 - 1
\end{align*}
\]

One solution for \([0, 0.1]\): \(x_1 = x_2 = x_3 = \frac{1}{\sqrt{2}}\) \(y_1 = y_2 = y_3 = \frac{-1}{\sqrt{2}}\)

Note that in this particular simple example, the system has a singularity at \((x, y) = (1, 0)\), so not all initial conditions and intervals \([a, b]\) work. In general, such problems occur and one wants to find an initial condition that has no singularities at least as long as \(t \in [a, b]\).
1 BDF Method

2 Collocation (Overview)

3 Brief Introduction to Modelica
Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous systems. It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic control and state machine subsystems, process oriented applications and generation and distribution of electric power. Models in Modelica are mathematically described by differential, algebraic and discrete equations. Modelica is designed such that available, specialized algorithms can be utilized to enable efficient handling of large models having more than one hundred thousand equations. Modelica is suited and used for hardware-in-the-loop simulations and for embedded control systems.
Modelica is a **programming language** not a tool.

- Domain Specific Language: modeling of physical systems
- Equation based (declarative, acausal, unlike State Flow Languages)
- Object Oriented (object: data + equations)
- Designed (and used) to model and simulate complex physical systems
- Textual and graphical editors

**Brief History**

- First design group meeting Fall 1996
- Modelica Association: open, non profit organization established in 2000
- Modelica conference held annually since 2000
(Nonexhaustive lists)

Commercial Environments

- Dymola (Dassault Systemes)
- LMS Imagine.Lab Amesim (Siemens)
- SystemModeler (Wolfram)
- MapleSim (Maplesoft)

Open/Free Environments

- OpenModelica (OSMC)
- JModelica.org
- Modelicac (Scilab)
- Hello World Example
- DAE Example
- Inheritance
- Connectors, Connections and Equations Generation
References

• computation.llnl.gov/projects/sundials/ida
• www.openmodelica.org