Introduction to Differential Algebraic Equations

Dr. Abebe Geletu

Ilmenau University of Technology
Department of Simulation and Optimal Processes (SOP)

Winter Semester 2011/12

4.1 Definition and Properties of DAEs

A system of equations that is of the form

$$F(t, x, \dot{x}) = 0$$

is called a **differential algebraic equation** (DAE) if the Jacobian matrix $\frac{\partial F}{\partial \dot{x}}$ is singular (non-invertible); where, for each t, $x(t) \in \mathbb{R}^n$ and

$$F(t, x(t), \dot{x}(t)) = \begin{pmatrix} F_1(t, x(t), \dot{x}(t)) \\ F_2(t, x(t), \dot{x}(t)) \\ \vdots \\ F_n(t, x(t), \dot{x}(t)) \end{pmatrix}.$$

4.1 Definition and Properties of DAEs ...

Example: The system

$$x_1 - \dot{x}_1 + 1 = 0 \qquad (1)$$

$$\dot{x}_1 x_2 + 2 = 0 \qquad (2)$$

is a DAE. To see this, determine the Jacobian $\frac{\partial F}{\partial \dot{x}}$ of

$$F(t, x, \dot{x}) = \begin{pmatrix} x_1 - \dot{x}_1 + 1 \\ \dot{x}_1 x_2 + 2 \end{pmatrix}$$

with $\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}$, so that

$$\frac{\partial F}{\partial \dot{x}} = \begin{pmatrix} \frac{\partial F_1}{\partial \dot{x}_1} & \frac{\partial F_1}{\partial \dot{x}_2} \\ \frac{\partial F_2}{\partial \dot{x}_1} & \frac{\partial F_2}{\partial \dot{x}_2} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ x_2 & 0 \end{pmatrix}, \quad (\text{ see that, } det\left(\frac{\partial F}{\partial \dot{x}}\right) = 0).$$

$\Rightarrow$ the Jacobian is a singular matrix irrespective of the values of x_2.

Observe that: in this example the derivative $\dot{x}_2$ does not appear.

4.1 Definition and Properties of DAEs ...

Solving for $\dot{x}_1$ from the first equation $x_1 - \dot{x}_1 + 1 = 0$ we get $\dot{x}_1 = x_1 + 1$. Replace this for $\dot{x}_1$ in the second equation $\dot{x}_1 x_2 + 2 = 0$ to wire the DAE in equations (23) & (23) equivalently as:

$$
\begin{aligned}
\dot{x}_1 &= x_1 + 1 \qquad (3)\\
(x_1 + 1)x_2 + 2 &= 0 \qquad (4)
\end{aligned}
$$

In this DAE:

- equation (3) is a differential equation; while
- equation (4) is an algebraic equation.

$\Rightarrow$ There are several engineering applications that have such model equations.

4.1 Definition and Properties of DAEs ...

Suppose $F(t,x,\dot{x}) = A(t)\dot{x} + B(t)x + d(t)$.
Hence, for the system $F(t,x,\dot{x}) = 0$ the Jacobian will be $\frac{\partial F}{\partial \dot{x}} = A(t)$.

- If $A(t)$ is a non-singular (an invertible) matrix, then

$$
\begin{aligned}
&[A(t)]^{-1}\left(A(t)\dot{x}(t) + B(t)x(t) + d(t)\right) = [A(t)]^{-1}\,0 \\
&\Rightarrow \dot{x}(t) + [A(t)]^{-1}B(t)x(t) + [A(t)]^{-1}d(t)) = 0 \\
&\Rightarrow \dot{x}(t) = -[A(t)]^{-1}B(t)x(t) - [A(t)]^{-1}d(t)).
\end{aligned}
$$

This is an ordinary differential equation.

Remark

In general, if the **Jacobian matrix** $\frac{\partial F}{\partial \dot{x}}$ **is non-singular (invertible)**, then the system $F(t,x,\dot{x}) = 0$ can be transformed into an ordinary differential equation (ODE) of the form $\dot{x} = f(t,x)$. Some numerical solution methods for ODE models have been already discussed.

- Therefore, the most interesting case is when $\frac{\partial F}{\partial \dot{x}}$ is **singular**.

4.2. Some DAE models from engineering applications

• There are several engineering applications that lead DAE model equations.

Examples: process engineering, mechanical engineering and mechatronics (multibody Systems eg. robot dynamics, car dynamics, etc), electrical engineering (eg. electrical network systems, etc), water distribution network systems, thermodynamic systems, etc.

Frequently, DAEs arise from practical applications as:

▶ differential equations describing the dynamics of the process, plus

▶ algebraic equations describing:

• laws of conservation of energy, mass, charge, current, etc.

• mass, molar, entropy balance equations, etc.

• desired constraints on the dynamics of the process.

4.2. Some DAE models from engineering applications ... CSTR

An isothermal CSTR

$$A \rightleftharpoons B \to C.$$

Model equation:

$$\begin{aligned}
\dot{V} &= F_a - F && (5)\\
\dot{C}_A &= \frac{F_a}{V}(C_{A_0} - C_A) - R_1 && (6)\\
\dot{C}_B &= -\frac{F_a}{V} C_B + R_1 - R_2 && (7)\\
\dot{C}_C &= -\frac{F_a}{V} C_C + R_2 && (8)\\
0 &= C_A - \frac{C_B}{K_{eq}} && (9)\\
0 &= R_2 - k_2 C_B && (10)
\end{aligned}$$

4.2. Some DAE models from practical applications ...a CSTR...

- F_a-feed flow rate of A
- C_{A_0}-feed concentration of A
- R_1, R_2 - rates of reactions
- F - product withdrawal rate
- C_A, C_B, C_C - concentration of species A, B and C, resp., in the mixture.

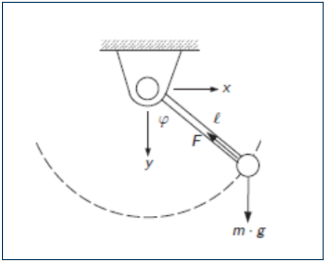
Defining

$$\begin{aligned} x &= (V, C_A, C_B, C_C)^\top \\ z &= (R_1, R_2)^\top \end{aligned}$$

the CSTR model equation can be written in the form

$$\begin{aligned} \dot{x} &= f(x, z) \\ 0 &= g(x, z). \end{aligned}$$

Some DAE models from practical applications...a simple pendulum



Newton's Law:

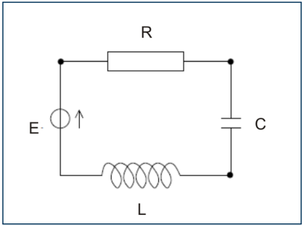
$m\ddot{x} = -\frac{F}{l}x$

$m\ddot{y} = mg\frac{F}{l}y$

Conservation of mechanical energy: $x^2 + y^2 = l^2$

$$(DAE) \qquad \begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= -\frac{F}{m\,l}x_1 \\ \dot{x}_4 &= g\frac{F}{l}x_2 \\ 0 &= x^2 + y^2 - l^2. \end{aligned}$$

Some DAE models from practical applications...an RLC circuit



Kirchhoff's voltage and current laws yield:

conservation of current:

$$i_E = i_R, i_R = i_C, i_C = i_L$$

conservation of energy:

$$V_R + V_L + V_C + V_E = 0$$

Ohm's Laws:

$$C\dot{V}_C = i_C, L\dot{V}_L = i_L, V_R = Ri_R$$

Some DAE models from practical applications...an RLC circuit...

After replacing i_R with i_E and i_C with i_L we get a reduced DAE:

$$
\begin{aligned}
\dot{V}_C &= \frac{1}{C} i_L && (11)\\
\dot{V}_L &= \frac{1}{L} i_L && (12)\\
0 &= V_R + R i_E && (13)\\
0 &= V_E + V_R + V_C + V_L && (14)\\
0 &= i_L - i_E && (15)\\
& && (16)
\end{aligned}
$$

Define $x(t) = (V_C, V_L, V_R, i_L, i_E)$

Some DAE models from practical applications...an RLC circuit...

The RLC system can be written as:

$$\dot{x} = \begin{pmatrix} \frac{1}{C} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} x \tag{17}$$

$$0 = \begin{pmatrix} 0 & 0 & 1 & 0 & R \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} V_E \tag{18}$$

which is of the form

$$\dot{x} = Ax \tag{19}$$

$$0 = Bx + Dz. \tag{20}$$

4.3. Classification of DAEs

• Frequently, DAEs posses mathematical structure that are specific to a given application area.

• As a result we have non-linear DAEs, linear DAEs, etc.

• In fact, a knowledge on the mathematical structure of a DAE facilitates the selection of model-specific algorithms and appropriate software.

Nonlinear DAEs:

In the DAE $F(t, x, \dot{x}) = 0$ if the function F is nonlinear w.r.t. any one of t, x or $\dot{x}$, then it is said to be a nonlinear DAE.

Linear DAEs: A DAE of the form

$$A(t)\dot{x}(t) + B(t)x(t) = c(t),$$

where $A(t)$ and $B(t)$ are $n \times n$ matrices, is linear. If $A(t) \equiv A$ and $B(t) \equiv B$, then we have time-invariant linear DAE.

4.3. Classification of DAEs...

Semi-explicit DAEs:

A DAE given in the form

$$\begin{aligned} \dot{x} &= f(t, x, z) & (21) \\ 0 &= g(t, x, z) & (22) \end{aligned}$$

- Note that the derivative of the variable z doesn't appear in the DAE.
- Such a variable z is called an **algebraic variable**; while x is called a **differential variable**.
- The equation $0 = g(t, x, z)$ called **algebraic equation** or a **constraint**.

Examples:

- The DAE model given for the RLC circuit, the CSTR and the simple pendulum are all semi-explicit form.

4.3. Classification of DAEs...

A fully-implicit DAEs:

The DAE

$$F(t, x, \dot{x}) = 0$$

is in **fully-implicit** form.

Examples:

(i) $F(t, x, \dot{x}) = A\dot{x} + Bx + b(t)$ is a fully-implicit DAE.

(ii) The equation (see equations (23) & (23))

$$\begin{aligned} x_1 - \dot{x}_1 + 1 &= 0 \\ \dot{x}_1 x_2 + 2 &= 0 \end{aligned}$$

is a fully-implicit DAE.

▶ Any fully-implicit DAE can be always transformed into a semi-explicit DAEs .

4.3. Classification of DAEs...

Example (transformation of a fully-implicit DAE into a semi-explicit DAE):

• Consider the linear time-invariant DAE $A\dot{x} + Bx + b(t) = 0$, where $\lambda A + B$ is nonsingular, for some scalar λ. Then there are non-singular $n \times n$ matrices G and H such that:

$$GAH = \begin{bmatrix} I_m & O \\ O & N \end{bmatrix} \text{ and } GBH = \begin{bmatrix} J & O \\ O & I_{n-m} \end{bmatrix} \tag{23}$$

where I_m the $m \times m$ identity matrix (here $m \leq n$), N is an $(n-m) \times (n-m)$ **nilpotent matrix**; i.e., there is a positive integer p such that $N^p = 0$, $J \in \mathbb{R}^{m \times m}$ and I_{n-m} is the $(n-m) \times (n-m)$ identity matrix.

▶ Now, we can write $A\dot{x} + Bx + b(t) = 0$ equivalently as

$$(GAH)\,(H^{-1})\dot{x} + (GBH)\,(H^{-1})x + Gb(t) = 0. \tag{24}$$

4.3. Classification of DAEs...

- Use the block decomposing given in equation (23) to write

$$\begin{bmatrix} I_m & O \\ O & N \end{bmatrix} (H^{-1})\dot{x} + \begin{bmatrix} J & O \\ O & I_{n-m} \end{bmatrix} (H^{-1})x + Gb(t) = 0. \tag{25}$$

- Use the **variable transformation** $w(t) = H^{-1}x(t)$ to write

$$\begin{bmatrix} I_m & O \\ O & N \end{bmatrix} \dot{w} + \begin{bmatrix} J & O \\ O & I_{n-m} \end{bmatrix} w + Gb(t) = 0. \tag{26}$$

- Decompose the vector $w(t)$ as $w(t) = \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix}$, with $w_1(t) \in \mathbb{R}^m$

and $w_2(t) \in \mathbb{R}^{n-m}$ and correspondingly the vector $Gb(t) = \begin{bmatrix} b_1(t) \\ b_2(t) \end{bmatrix}$ so that:

$$\dot{w}_1(t) + Jw_1(t) + b_1(t) = 0 \tag{27}$$

$$Nw_1(t) + w_2(t) + b_2(t) = 0. \tag{28}$$

4.3. Classification of DAEs...

• Use now the nilpotent property of the matrix N; i.e., multiply the second set of equation by N^{p-1} to get

$$\dot{w}_1(t) + Jw_1(t) + b_1(t) = 0 \tag{29}$$

$$N^p w_1(t) + N^{p-1} w_2(t) + N^{p-1} b_2(t) = 0. \tag{30}$$

From this it follows that (since $N^p w_1(t) = 0$)

$$\dot{w}_1(t) = -Jw_1(t) - b_1(t) \tag{31}$$

$$0 = -N^{p-1} w_2(t) - N^{p-1} b_2(t). \tag{32}$$

Therefore, we have transformed the fully-implicit DAE $A\dot{x} + Bx + b(t) = 0$ into a semi-explicit form.

• Similarly, using mathematician manipulations, any nonlinear fully-implicit DAE can be transformed into a semi-explicit DAE.

4.4. Index of a DAE

- DAEs are usually very complex and hard to be solved **analytically**.

$\Rightarrow$ DAEs are commonly solved by using **numerical methods**.

Question:

Is it possible to use numerical methods of ODEs for the solution of DAEs?

Idea:

Attempt to transform the DAE into an ODE.

- This can be chieved through repeated derivations of the algebraic equations $g(t, x, z) = 0$ **with respect to time** t.

Definition

The minimum number of differentiation steps required to transform a DAE into an ODE is known as the (differential) **index** of the DAE.

4.4. Index of a DAE...

Example:

$$\begin{aligned} (DAE) \qquad\qquad \dot{x}_1 &= x_1 + 1 \\ (x_1+1)x_2 + 2 &= 0. \end{aligned}$$

Here x_2 the algebraic variable (i.e. $z = x_2$). Differentiate $g(x_1, x_2) = 0$ to find a description for the time-derivative $\dot{x}_2$ of the algebraic variable. So

$$\frac{d}{dt}\left[g(x_1, x_2)\right] = \frac{d}{dt}[0] \Rightarrow \frac{d}{dt}\left[(x_1+1)x_2 + 2\right] = 0$$

$$\Rightarrow \dot{x}_1 x_2 + (x_1+1)\dot{x}_2 = 0.$$

$$\Rightarrow \dot{x}_2 = -\frac{\dot{x}_1 x_2}{(x_1+1)} = -\frac{(x_1+1)x_2}{(x_1+1)} = -x_2$$

▶ Only one differentiation step is required to describe $\dot{x}_2$. So, the DAE is of index 1.

4.4. Index of a DAE...

- The CSTR model is of index 2.
- The DAE model for the simple penudulum is of index 3.

▶ DAEs with index greater than 1 are commonly known as **higher index** DAEs.

▶ The higher the index, the more difficult will be the DAE to solve.

▶ Transformation of a higher index DAE into a lower index DAE (or to an ODE) is commonly known as **index reduction**. In general, index reduction for higher index DAE simplifies computational complexities.

Two serious issues to consider when solving DAEs

- The solutions of the lower index DAE may not be a solution of the original DAE. This is known as a **drift off** effect.
- Finding initial conditions that satisfy both the differential and algebraic parts of a DAE may not be trivial, known as **consistency of initial conditions**.

4.4. Index of a DAE...

Therefore, computational algorithms for a DAE should:

- be cable of identifying consistent initial conditions to the DAE; as well as,
- provide automatic index reduction mechanisms to simplify the DAE.

▶ Modern software like Sundials, Modelica, use strategies for index-reduction coupled with methods of consistent initialization.

In the following we consider only index 1 semi-explicit DAEs:

$$(DAE) \qquad \begin{aligned} \dot{x} &= f(t,x,z) \qquad (33) \\ 0 &= g(t,x,z). \qquad (34) \end{aligned}$$

Since $\frac{\partial g}{\partial t} + \frac{\partial g}{\partial x}\frac{dx}{dt} + \frac{\partial g}{\partial z}\frac{dz}{dt} = 0$. Hence, a semi-explicit DAE is of index 1 iff $\left[\frac{\partial g}{\partial z}\right]^{-1}$ exists. That is, one differentiation step yields the differential equation: $\frac{dz}{dt} = -\left[\frac{\partial g}{\partial z}\right]^{-1}\left[\frac{\partial g}{\partial x}\right](f(t,x,z)) - \left[\frac{\partial g}{\partial z}\right]^{-1}\frac{\partial g}{\partial t}$.

4.5. An overview of numerical Methods for DAEs

• Using mathematical principles and transformation of variables, fully-implicit DAEs can be transformed to semi-explicit DAEs.
• Note that, in the semi-explicit DAE

$$\begin{aligned} \dot{x} &= f(t, x, z) && (35) \\ 0 &= g(t, x, z). && (36) \end{aligned}$$

if Jacobian matrix $\left[\frac{\partial g}{\partial z}\right]$ singular (non-invertible), then the DAE is of higher index.
▶ Since many applications have model equations as semi-explicit DAEs, the discussion next is restricted to this form.

• In the DAE above, if both f and g do not explicitly depend on time t; i.e. $f(t, x, z) = f(x, z)$ and $g(t, x, z) = g(x, z)$, then the model is an **autonomous** DAE.

4.5. An overview of numerical Methods for DAEs...

▶ **BDF** and **collocation** methods are two most widely used methods for numerical solution of DAEs.

(I) **BDF** for DAEs:
Consider the initial value DAE

$$\begin{aligned} \dot{x} &= f(t, x, z), x(t_0) = x_0 && (37) \\ 0 &= g(t, x, z). && (38) \end{aligned}$$

Ideas of BDF:

▶ Select a time step h so that $t_{i+1} = t_i + h, i = 0, 1, 2, \ldots$

▶ Given $x_i = x(t_i)$ and $z_i = z(t_i)$, determine the value $x_{i+1} = x(t_{i+1})$ by using (extrapolating) values $x_i, x_{i-1}, \ldots, x_{i-m+1}$ of the current and earlier time instants of $x(t)$.

▶ Simultaneously compute $z_{i+1} = z(t_{i+1})$.

4.5. An overview of numerical Methods for DAEs...BDF

- There is a unique m-th degree polynomial P that interpolates the $m+1$ points

$$(t_{i+1}, x_{i+1}), (t_i, x_i), (t_{i-1}, x_{i-1}), \dots, (t_{i+1-m}, x_{i+1-m}).$$

- This interpolating polynomials P can be written as

$$P(t) = \sum_{j=0}^{m} x_{i+1-j} L_j(t)$$

with the Lagrange polynomial

$$L_j(t) = \prod_{\substack{l=0 \\ l \neq j}}^{m} \left[\frac{t - t_{i+1-l}}{t_{i+1-j} - t_{i+1-l}} \right], j = 0, 1, \dots, m.$$

4.5. ....BDF for DAEs...

- Observe that

$$P(t_{i+1-j}) = x_{i+1-j}, j = 0, 1, \ldots, m.$$

In particular $P(t_{i+1}) = x_{i+1}$.

- Thus, replace $\dot{x}_{i+1}$ by $\dot{P}(t_{i+1})$ to obtain

$$\dot{P}(t_{i+1}) = f(t_{i+1}, x_{i+1}). \qquad (*)$$

But

$$\dot{P}(t_{i+1}) = \sum_{j=0}^{m} x_{i+1-j}\dot{L}_j(t_{i+1}) = x_{i+1}\dot{L}_0(t_{i+1}) + \sum_{j=1}^{m} x_{i+1-j}\dot{L}_j(t)$$

Putting this into (*) we get:

$$x_{i+1} = -\sum_{j=1}^{m} x_{i+1-j}\frac{\dot{L}_j(t_{i+1})}{\dot{L}_0(t_{i+1})} + \frac{1}{\dot{L}_0(t_{i+1})} f(t_{i+1}, x_{i+1})$$

4.5. ....BDF for DAEs...

Define

$$a_j = \frac{\dot{L}_j(t_{i+1})}{\dot{L}_0(t_{i+1})}, \quad b_m = \frac{1}{h\dot{L}_0(t_{i+1})}.$$

- The values $a_j, j = 1, \ldots, m$ and b_m can be read from lookup tables.

An m-step BDF Algorithm (BDF m) for a DAE:

Given $a_1, \ldots, a_m$,b_m

$$\begin{aligned} x_{i+1} &= -\sum_{j=1}^{m} a_j x_{i+1-j} + b_m h f(t_{i+1}, x_{i+1}, z_{i+1}) && (39) \\ 0 &= g(t_{i+1}, x_{i+1}, z_{i+1}). && (40) \end{aligned}$$

- Each iteration of the BDF requires a Newton algorithm for the solution a system of nonlinear equations. Hence, the Jacobian $\frac{\partial g}{\partial w}$ needs to be well-conditioned, where $w = (t, x, z)$.

4.5. ....BDF for DAEs...

- Given $x_0 = x(t_0)$, BDF requires consistent initial conditions to be obtained by solving

$$g(t_0, x_0, z_0) = 0.$$

to determine $z_0 = z(t_0)$.

The m-step BDF algorithm converges if $m \leq 6$; i.e.

$$x_i - x(t_i) \leq O(h^m), \qquad z_i - z(t_i) \leq O(h^m)$$

for consistent initial conditions.

- BDF is commonly used to solve DAEs of index 1 or 2

Software based on BDF for DAEs:

$\gg$ **Matlab:** `ode15i`

$\gg$ **Open source:** `DASSL` ; CVODE, CVODES, and IDA (under Sundials); ODEPACK solvers, etc.

Orthogonal Collocation

• To collocate a function $x(t)$ through another function $p(t)$ = to captur the properties of $x(t)$ by using $p(t)$.

• In general, we use a simpler function $p(t)$ to collocate $x(t)$; eg., $p(t)$ can be a polynomial, a trigonometric function, etc.

$\Rightarrow$ polynomial and trigonometric functions are usually simpler to work with. (The discussion here is restricted to polynomilas)

Weirstraß' Theorem

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

$$\max_{a \leq t \leq b} |x(t) - p(t)| < \varepsilon$$

• Hence, we use the polynomial $p(t)$ instead of $x(t)$.

• However, Weirstraß' theorem doese not specifye how to construct the approximating polynomial $p(t)$.

• Suppose $p(t) = a_0 + a_1 t + \ldots + a_m t^m$ is the approximating polynomial to $x(t)$ on $[a, b]$.

Note: If the coefficients $a_0, a_1, \ldots, a_m$ are given, then the approximating polynomial is exactly known.

Question: How to determine $a_0, a_1, \ldots, a_m$?

Question: How to construct the apprximating polynomial?

• There are several ways to construct $p(t)$ to approximates $x(t)$ according to Weirstraß' theorem.

Here we require $p(t)$ to satisfy the following property:

• $\quad p(t_i) = x(t_i) =: x_i, i = 1, \ldots, N.$

for some *selected time instants* $t_1, t_2, \ldots, t_N$ from the interval $[a, b]$.

• This property relates $p(t)$ and $x(t)$ and is known as **interpolatory property**.

Uniqueness of an interpolating polynamial

There is a unique interpolatory polynomial $p(t)$ for the N data points (t_1, x_1),(t_2, x_2), …,(t_N, x_N) with degree $deg(p) = m = N - 1$.

- In the following we use $N = m + 1$.

Figure

According to the interpolatory property, we have

$$x_1 = p(t_1) \quad = \quad a_0 + a_1 t_1 + \ldots + a_m t_1^m \tag{41}$$
$$x_2 = p(t_2) \quad = \quad a_0 + a_1 t_2 + \ldots + a_m t_2^m \tag{42}$$
$$\vdots \tag{43}$$
$$x_{m+1} = p(t_{m+1}) \quad = \quad a_0 + a_1 t_{m+1} + \ldots + a_m t_{m+1}^m. \tag{44}$$

$$\Rightarrow \quad \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m+1} \end{pmatrix} = \begin{pmatrix} 1 & t_1 & t_1^2 & \ldots & t_1^m \\ 1 & t_2 & t_2^2 & \ldots & t_2^m \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & t_{m+1} & t_{m+1}^2 & \ldots & t_{m+1}^m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}$$

Thus, if we known $x_1, x_2, \ldots, x_{m+1}$ then we can compute $a_0, a_1, \ldots, a_m$ and vice-versa.

- But, since $x(t)$ is not yet known, $x_1, x_2, \ldots, x_{m+1}$ are unknown.

Hence, both $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m+1} \end{pmatrix}$ and $\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}$ are unkowns.

• To avoid working with two unknown vectors, we can define the polynamial $p(t)$ in a better way as:

$$p(t) = \sum_{i=1}^{m+1} x_i L_i(t) \tag{45}$$

where $L_i(t)$ are Lagrange polynomials given by

$$\begin{aligned} L_i(t) &= \prod_{\substack{j=1 \\ j \neq i}}^{m+1} \frac{t - t_j}{t_i - t_j} \\ &= \frac{(t-t_1)(t-t_2)\dots(t-t_{i-1})(t-t_{i+1})\dots(t-t_{m+1})}{(t_i-t_1)(t_i-t_2)\dots(t_i-t_{i-1})(t_i-t_{i+1})\dots(t_i-t_{m+1})}. \end{aligned}$$

Properties:

$$L_i(t_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

⋆ (satisfaction of the interpolatory property)

$$p(t_i) = x_i, i = 1, 2, \ldots, m+1.$$

⋆ (approximation of $x(t)$ by $p(t)$)
The polynomial $p(t)$ in equation (45) can be made to satisfy the Weirstraß' theorem by taking sufficiently large number of time instants $t_1, t_2, \ldots, t_{m+1}$ from $[a, b]$.

Question: What is the best choice for $t_1, t_2, \ldots, t_{m+1}$?

Determination of collocation Points

An idea to determine collocation points:

Select values $\tau_1, \tau_2, \ldots, \tau_N$ from the interval $[0, 1]$ and define the t_i's as follows:

$$t_i = a + \tau_i(b - a), i = 1, \ldots, N.$$

- This is a good idea, since the same values $\tau_1, \tau_2, \ldots, \tau_N$ from $[0, 1]$ can be used to generate collocation points on **various intervals** $[a, b]$.

Example:

(i) If $[a, b] = [2, 10]$, then $t_i = 2 + \tau_i\,(10 - 2)\,, i = 1, \ldots, N$;

(ii) If $[a, b] = [0, 50]$, then $t_i = 0 + \tau_i\,(50 - 0)\,, i = 1, \ldots, N$; etc.

Question: What is the best way to select the values $\tau_1, \tau_2, \ldots, \tau_N$ from $[0, 1]$?

Determination of collocation Points...

▶ Recall that: solution of a differential equation (or DAEs) is an integration process.
⇒ Numerical methods for one-dimensional integrals can give us information on how to select $\tau_1, \tau_2, \ldots, \tau_N$.
⇒ Gauss quadrature rules are one of the best numerical integration methods.
To approximate the integral

$$I[f] = \int_0^1 f(\tau) d\tau,$$

by the (iterpolatory) quadrature rule $Q_N[f] = \sum_{k=1}^{N} w_k f(\tau_k)$, where:
• the integration nodes $\tau_1, \tau_2, \ldots, \tau_N \in [0, 1]$
• weights $w_1, w_2, \ldots, w_N$
are constructed based on the interval $[0, 1]$.

Determination of collocation points...

Thus we would like to obtain the approximation

$$I[f] = \int_0^1 f(\tau)d\tau \approx Q_N[f] = \sum_{i=1}^{N} w_i f(\tau_i).$$

- Once a quadrature rule $Q_N[\cdot]$ is constructed it can be used to approximate integrals of various functions.

Question: How to determine the 2N unknowns $\tau_1, \tau_2, \ldots, \tau_N$, $w_1, w_2, \ldots, w_N$?

We require the quadrature rule to satisfy the equality:

$$\int_a^b W(\tau)p(\tau)d\tau = \sum_{i=1}^{N} w_i p(\tau_i)$$

for all polynomials p with degree $deg(p) \leq 2N$.

That is, $Q_N[\cdot]$ should integrate each of the polynomials

$$p(t) = 1, \tau, \tau^2, \ldots, \tau^{2N}$$

exactly. This implies

$$\int_0^1 1 d\tau = \sum_{i=1} w_i \tag{46}$$

$$\int_0^1 \tau d\tau = \sum_{i=1} w_i \tau_i \tag{47}$$

$$\int_0^1 \tau^2 d\tau = \sum_{i=1} w_i \tau_i^2 \tag{48}$$

$$\vdots \tag{49}$$

$$\int_0^1 \tau^{2N} d\tau = \sum_{i=1} w_i \tau_i^{2N}. \tag{50}$$

We need to solve the system of $2N$ nonlinear equations:

$$
\begin{aligned}
1 &= w_1 + w_2 + \ldots + w_N && (51)\\
\frac{1}{2} &= w_1\tau_1 + w_2\tau_2 + \ldots + w_N\tau_N && (52)\\
& && (53)\\
\frac{1}{3} &= w_1\tau_1^2 + w_2\tau_2^2 + \ldots + w_N\tau_N^2 && (54)\\
&\vdots && (55)\\
\frac{1}{2N} &= w_1\tau_1^{2N} + w_2\tau_2^{2N} + \ldots + w_N\tau_N^{2N}. && (56)
\end{aligned}
$$

From the equation $1 = w_1 + w_2 + \ldots + w_N$ we can solve for w_1 and replace for it in the remaining equations.
$\Rightarrow$ It remains to determine the $2N-1$ unknowns $\tau_2, \ldots, \tau_N$ and $w_1, w_2, \ldots, w_N$ from the reduced set of $2N-1$ equations.

- Unfortunately, this system of equations is difficult to solve.

Orthogonal polynomials

•There is a simpler way if we use **orthogonal polynomials**.

▶ The concept of orthogonality requires a definition of **scalar product**.

• The **scalar product of functions** h and g with respect on the interval $[0, 1]$ is

$$\langle h, g\rangle = \int_0^1 h(\tau)g(\tau)d\tau.$$

• Two functions h and g are **orthogonal** on the interval $[0, 1]$ if

$$\langle h, g\rangle = \int_0^1 h(\tau)g(\tau)d\tau = 0.$$

• In this lecture, we are only interested in **set of polynomials that are orthogonal to each other**.

Orthogonal polynomials...

- Orthogonal polynomials on $[0,1]$ (as defined above) are known as **shifted Legendre polynomials**.

Theorem (Three-term recurrence relation)

Suppose $\{p_0, p_1, \ldots\}$ the set of shifted Legendre orthogonal polynomials on $[0,1]$ with $deg(p_n) = n$ and leading coefficient equal to 1. The shifted Legendre polynomials are generated by the relation

$$p_{n+1}(\tau) = (\tau - a_n)p_n(\tau) - b_n p_{n-1}(\tau)$$

with $p_0(\tau) = 1$ and $p_{-1}(\tau) = 0$, where the recurrence coefficients are given as

$$\boxed{a_n = \frac{1}{2}, n = 0, 1, 2, \ldots}$$

$$\boxed{b_n = \frac{n^2}{4(4n^2-1)}, n = 0, 1, 2, \ldots}$$

Orthogonal polynomials...

The first four shifted Legendre orthogonal polynomials are $p_0(\tau) = 1, p_1(\tau) = 2\tau - 1, p_2(\tau) = 6\tau^2 - 6\tau + 1, p_3(\tau) = 20\tau^3 - 30\tau^2 + 12\tau - 1.$

Further important properties of orthogonal polynomials:

Given any set of orthogonal polynomials $\{p_0, p_1, p_2, \ldots\}$ the following hold true:

• Any finite set of orthogonal polynomials $\{p_0, p_1, \ldots, p_{N-1}\}$ is linearly independent.

• The polynomial P_N is orthogonal to each of $p_0, p_1, \ldots, p_{N-1}$.

• Any non-zero polynomial q with degree $deg(q) \leq N - 1$ can be written as a linear combination:

$$q(\tau) = c_0 p_0(\tau) + c_1 p_1(\tau) + \ldots, c_{N-1} p_{N-1}(\tau)$$

where at least one of the scalars $c_0, c_1, \ldots, c_{N-1}$ is non-zero.

Orthogonal polynomials...

• We demand the quadrature rule is exact for polynomials up to degree $2N-1$.

• Let $P(\tau)$ be any polynomial of degree $2N-1$.
Hence,

$$I[P] = Q_N[P] \Rightarrow \int_0^1 P(\tau)d\tau = \sum_{i=1}^{N} w_i P(\tau_i).$$

• Given the N-th degree orthogonal polynomial p_N, the polynomial P can be written as

$$P(\tau) = p_N(\tau)q(\tau) + r(\tau)$$

where $q(\tau)$ and $r(\tau)$ are polynomials such that $0 < deg(r) \leq N-1$ and

• $deg(q) = N-1$, since

$$2N-1 = deg(P) = deg(p_N) + deg(q) = N + deg(q).$$

Orthogonal polynomials...

- Now, from the equation $\int_0^1 P(\tau)d\tau = \sum_{k=1}^{N} w_k P(\tau_k)$ it follows that

$$\int_a^b \left(p_N(\tau)q(\tau) + r(\tau)\right) d\tau \;=\; \sum_{i=1}^{N} w_i \left(p_N(\tau_i)q(\tau_k) + r(\tau_i)\right).$$

$$\Rightarrow \underbrace{\int_a^b p_N(\tau)q(\tau)d\tau}_{=0} + \cancel{\int_a^b r(\tau)d\tau} \;=\; \sum_{i=1}^{N} w_i p_N(\tau_i)q(\tau_i) + \cancel{\sum_{i=1}^{N} w_i r(\tau_i)}.$$

Observe the following:

- Using polynomial exactness: $\int_a^b r(\tau)d\tau = \sum_{i=1}^{N} w_i\, r(\tau_i)$.
- Orthogonality implies $< p_N, p_k >= 0, k = 1, \ldots, N-1$. Thus,

$\int_a^b p_N(\tau)q(\tau) =$

$$\int_a^b p_N(\tau) \sum_{i=1}^{N-1} c_k p_k(\tau)d\tau = \sum_{k=1}^{N-1} c_k \underbrace{\int_a^b p_N(\tau)p_k(\tau)d\tau}_{=0} = 0.$$

Orthogonal polynomials...

It follows that $\sum_{k=1}^{N} w_k p_N(\tau_k) q(\tau_k) = 0.$ $(\star)$

Theorem

If the quadrature nodes $\tau_1, \tau_2, \ldots, \tau_N$ are zeros of the $N-$th degree shifted Legendre polynomial $p_N(\tau)$, then

- *all the roots $\tau_1, \tau_2, \ldots, \tau_N$ lie inside $(0, 1)$;*
- *the quadrature weights are determined from*

$$w_i = \int_0^1 L_i(\tau) d\tau,$$

where $L_i(\tau), i = 1, \ldots, N$ is the Lagrange function defined using $\tau_1, \tau_2, \ldots, \tau_N$ and $w_i > 0, i = 1, \ldots, N$.

- *The quadrature rule $Q_N[\cdot]$ integrates polynomials degree up to $2N - 1$ exactly.*

Hence, equation $(\star)$ holds true if $p_1(\tau_1) = p_N(\tau_2) = \ldots, p(\tau_N) = 0$.

Orthogonal polynomials...

- Therefore, the quadrature nodes $\tau_1, \tau_2, \ldots, \tau_N$ are chosen as the zeros of the N-th degree shifted Legendre orthogonal polynomial.

Question:

Is there a simple way to compute the zeros $\tau_1, \tau_2, \ldots, \tau_N$ of an orthogonal polynomial p_N and the quadrature weights $w_1, w_2, \ldots, w_N$?

The answer is give by a Theorem of Welsch & Glub (see next slide).

Orthogonal polynomials...

Theorem (Welsch & Glub 1969)

The quadrature nodes $\tau_1, \tau_2, \ldots, \tau_N$ and weights $w_1, w_2, \ldots, w_N$ can be computed from the spectral factorization of

$$J_N = V^\top \Lambda V; \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_N), VV^\top = I_N;$$

of the symmetric tri-diagonal **Jacobi matrix**

$$J_N = \begin{bmatrix} a_0 & \sqrt{b_1} & & & \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & & \\ & \sqrt{b_2} & \ddots & \ddots & \\ & & \ddots & a_{N-2} & \sqrt{b_{N-1}} \\ & & & \sqrt{b_{N-1}} & a_{N-1} \end{bmatrix}$$ where

$\boxed{a_0, a_n, b_n, n = 1, \ldots, N-1}$ are the known coefficients from the recurrence relation. In particular ...

Theorem ...

$$\tau_k = \lambda_k, k = 1, \ldots, N; \quad (57)$$

$$w_k = \left(e^\top V e_k\right)^2, k = 1, \ldots, N, \quad (58)$$

where e_1, e_k are the 1st and the $k-$the unit vectors of length N.

A Matlab program:

```
n = 5; format short
beta = .5./sqrt(1-(2*(1:n-1)).^(-2)); % recurr. coeffs
J = diag(beta,1) + diag(beta,-1)      % Jacobi matrix
[V,D] = eig(J);                       % Spectral decomp.
tau = diag(D); [tau,i] = sort(tau); % Quad. nodes
w = 2*V(1,i).^2;                      % Quad. weights
```

- There are standardised lookup tables for the integration nodes $\tau_1, \tau_2, \ldots, \tau_N$ and corresponding weights $w_1, w_2, \ldots, w_N$.

The collocation points $t_1, t_2, \ldots, t_N \in [a, b]$ can be determined by using the quadrature points $\tau_1, \tau_2, \ldots, \tau_N \in [0, 1]$ through the relation:

$$t_i = a + \tau_i(b - a).$$

Example: Suppose we would like to collocate $x(t)$ on $[a, b] = [0, 10]$ using a polynomial $\hat{x}(t)$ with $deg(\hat{x}) = m = 3$.

- Required number of collocation points $= N = m + 1 = 3 + 1 = 4$.

$\Rightarrow$ First, determine the zeros of the 4-th degree shifted Legendre polynomial $p_4(\tau)$; i.e., find $\tau_1, \tau_2, \tau_3, \tau_4$.

- These value can be determined to be:

$$\tau_1 = 0.0694318442, \qquad \tau_2 = 0.3300094783,$$
$$\tau_3 = 0.6699905218, \qquad \tau_4 = 0.9305681558.$$

- Next, determine the collocation points :

$t_1 = 0 + (10 - 0) * \tau_1 = 0.694318442,\quad t_2 = 0 + (10 - 0) * \tau_2 = 3.300094783,$

$t_3 = 0 + (10 - 0) * \tau_3 = 6.699905218,$

$t_4 = 0 + (10 - 0) * \tau_4 = 9.305681558.$

The collocation polynomial is $p(t) = \sum_i^4 x_i L_i(t)$. where

$$L_1(t) = \frac{(t-t_2)(t-t_3)(t-t_4)}{(t_1-t_2)(t_1-t_3)(t_1-t_4)} \tag{59}$$

$$L_2(t) = \frac{(t-t_1)(t-t_3)(t-t_4)}{(t_2-t_1)(t_2-t_3)(t_2-t_4)} \tag{60}$$

$$L_3(t) = \frac{(t-t_1)(t-t_2)(t-t_4)}{(t_3-t_1)(t_3-t_2)(t_3-t_4)} \tag{61}$$

$$L_4(t) = \frac{(t-t_1)(t-t_2)(t-t_3)}{(t_4-t_1)(t_4-t_2)(t_4-t_3)}. \tag{62}$$

Hence, we approximate $x(t)$ using the third degree polynomial

$$p(t) = \widehat{x}(t) = x_1 L_1(t) + x_2 L_2(t) + x_3 L_3(t) + x_4 L_4(t)$$

It remains to determine the coefficients x_1, x_2, x_3, x_4.

Discretization of DAEs using orthogonal collocation

Given the DAE:

$$\dot{x} = f(t, x, z), t_0 \leq t \leq t_f \tag{63}$$

$$0 = g(t, x, z) \tag{64}$$

where $x(t)^\top = (x_1(t), x_2(t), \ldots, x_n(t))$ and $z(t)^\top = (z_1(t), z_2(t), \ldots, z_m(t))$.

- Determine the a set of collocation points t_1, t_2, …, t_N.
- corresponding to each differential and algebraic variable define collocation polynomials:

$$x_k(t) \longrightarrow p^{(k)}(t) = \sum_{i=1}^{N} x_i^{(k)} L_i(t), k = 1, \ldots, n; \tag{65}$$

$$z_j(t) \longrightarrow p^{(j)}(t) = \sum_{i=1}^{N} z_i^{(j)} L_i(t), j = 1, \ldots, m. \tag{66}$$

... orthogonal collocation of DAEs

- In the DAE, replace $x_k(t)$ and $z_j(t)$ by $q^{(k)}(t)$ and $p^{(j)}(t)$ so that

$$
\begin{aligned}
\dot{p}^{(1)}(t) &= f_1(t, \left(p^{(1)}(t), p^{(2)}(t), \ldots, p^{(n)}(t)\right), \left(q^{(1)}(t), q^{(2)}(t), \ldots, q^{(m)}\right. \\
\dot{p}^{(2)}(t) &= f_2(t, \left(p^{(1)}(t), p^{(2)}(t), \ldots, p^{(n)}(t)\right), \left(q^{(1)}(t), q^{(2)}(t), \ldots, q^{(m)}\right. \\
&\vdots \\
\dot{p}^{(n)}(t) &= f_1(t, \left(p^{(1)}(t), p^{(2)}(t), \ldots, p^{(n)}(t)\right), \left(q^{(1)}(t), q^{(2)}(t), \ldots, q^{(m)}\right. \\
0 &= g_1(t, \left(p^{(1)}(t), p^{(2)}(t), \ldots, p^{(n)}(t)\right), \left(q^{(1)}(t), q^{(2)}(t), \ldots, q^{(m)}\right. \\
0 &= g_2(t, \left(p^{(1)}(t), p^{(2)}(t), \ldots, p^{(n)}(t)\right), \left(q^{(1)}(t), q^{(2)}(t), \ldots, q^{(m)}\right. \\
&\vdots \\
0 &= g_m(t, \left(p^{(1)}(t), p^{(2)}(t), \ldots, p^{(n)}(t)\right), \left(q^{(1)}(t), q^{(2)}(t), \ldots, q^{(m)}\right.
\end{aligned}
$$

... orthogonal collocation of DAEs

- Next discretize the system above using $t_1, t_2, \ldots, t_N$ to obtain:

$$\begin{aligned}
\dot{p}^{(1)}(t_i) &= f_1\left(t_i, \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right), \left(z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(m)}\right)\right) \\
\dot{p}^{(2)}(t_i) &= f_2\left(t_i, \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right), \left(z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(m)}\right)\right) \\
&\vdots \\
\dot{p}^{(n)}(t_i) &= f_n\left(t_i, \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right), \left(z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(m)}\right)\right) \\
0 &= g_1\left(t_i, \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right), \left(z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(m)}\right)\right) \\
0 &= g_2\left(t_i, \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right), \left(z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(m)}\right)\right) \\
&\vdots \\
0 &= g_m\left(t_i, \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right), \left(z_i^{(1)}, z_i^{(2)}, \ldots, z_i^{(m)}\right)\right), \\
& \quad i = 1, 2, \ldots, N.
\end{aligned}$$

... orthogonal collocation of DAEs

- Solve the system of $(n+m) \times N$ equations to determine the $(n+m) \times N$ unknwons $x_i^{(k)}, k = 1, \ldots, n$, $i = 1, \ldots, N$ and $z_i^{(j)}, j = 1, \ldots, m$, $i = 1, \ldots, N$.

Example: Use a four point orthogonal collocation to solve the DAE

$$
\begin{aligned}
\dot{x}_1 &= x_1 + 1, 0 \leq t \leq 1. \\
0 &= (x_1 + 1)x_2 + 2.
\end{aligned}
$$

Solution: we use the collocation polynomials

$$
p^{(1)}(t) = \sum_{i=1}^{4} x_i^{(1)}(t) L_i(t) \text{ and } p^{(2)}(t) = \sum_{i=1}^{4} x_i^{(2)}(t) L_i(t)
$$

to collocate $x_1(t)$ and $x_2(t)$, respectively.

Advantages and disadvantages collocation methods

- can be used for higher index DAEs.
- efficient for both initial value as well as boundary values DAEs.
- more accurate

Disadvantages:

- Can be computationally expensive
- The approximating polynomials may display oscillatory properties, impacting accuracy
- computational expenses become high if $t_1, t_2, \ldots, t_{N-1}$ are considered variables.

Resources - Literature

References:

- U. M. Ascher, L. R. Petzold : Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. SIAM 1998.
- K. E. Brenan, S. L. Campbell, and L.R. Petzold: (1996). Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations., SIAM, 1996.
- Campbell, S. and Griepentrog, E., Solvability of General Differential Algebraic Equations, SIAM J. Sci. Comput. 16(2) pp. 257-270, 1995.
- M. S. Erik and G. Sderlind: Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives. SIAM Journal on Scientific Computing, vol. 14, pp. 677-692, 1993.
- C. W. Gear : Differential-algebraic index transformation. SIAM J. Sci. Stat. Comp. Vol. 9, pp. 39 – 47, 1988.
- Walter Gautschi: Orthogonal polynomials (in Matlab). Journal of Computational and Applied Mathematics 178 (2005) 215– 234

Resources - Literature .. Software

• Pantelides Costas. (1988) "The consistent initialization of differential algebraic systems." SIAM Journal on Scientific and Statistical Computing, vol. 9: 2, pp. 213–231, 1988.

• L. Petzold: Differential/Algebraic Equations Are Not ODEs, SIAM J. Sci. Stat. Comput. 3(3) pp. 367-384, 1982.

Open source software:

• DASSL - - FORTRAN 77:
http://www.cs.ucsb.edu/ cse/software.html

• DASSL Matlab interface:
http://www.mathworks.com/matlabcentral/fileexchange/16917-dassl-mex-file-compilation-to-matlab-5-3-and-6-5

• Sundials - https://computation.llnl.gov/casc/sundials/main.html

• Modelica - https://modelica.org/

• Odepack - http://www.cs.berkeley.edu/ kdatta/classes/lsode.html

• Scicos- http://www.scicos.org/