TenSyGrid

Pablo de Juan Vela [1]
[1]eRoots, Barcelona, Spain

1 Tensor Notation

A tensor is an N-way array where its values are accesed by N indeces. In this sense, a vector is an order 1 tensor and a matrix an order 2 tensor. A tensor of order n in the real numbers can de defined as $F \in \mathbb{R}^{I_1 \times ... \times I_n}$, where I_i is the dimension of the i_{th} order.

F be a tensor in $F \in \mathbb{R}^{\times_2 n}$ of order n, where each dimension has legth 2. remark

1.1 Tensor Operations

Tensor Product

Let $F \in \mathbb{R}^{j_1 \times ... \times j_m}$ and $G \in \mathbb{R}^{i_1 \times ... \times i_n}$. Then the tensor product $F \otimes G \in \mathbb{R}^{j_1 \times ... \times j_m, i_1 \times ... \times i_n}$ has the entries for $\forall a_k \in \{1, ..., j_k\}, b_k \in \{1, ..., i_k\}$:

$$(F \otimes G)_{a_1,...,a_m,b_1,...,b_n} := F_{a_1,...,a_m}.G_{b_1,...,b_m} \tag{1}$$

Inner Product

The inner product of two tensors of same order and dimension $F, G \in \mathbb{R}^{j_1 \times ... \times j_m}$ is a scalar that has the same properties of a vector product and is defined as:

$$< F, G >= \sum_{d_1=1}^{j_1} \sum_{d_2=1}^{j_2} \cdots \sum_{d_n=1}^{j_n} a_{d_1 d_2 ... d_n} b_{d_1 d_2 ... d_n} \tag{2}$$

This concept can be generalized for tensors of different dimensions. In this case the inner product can be interpreted as a form of order contraction.

Hadamard Product

The Hadamard product between 2 tensors of equal dimensionw is the point wise multiplication of those 2 tensors. Let $F, G \in \mathbb{R}^{i_1,...,i_n}$ then we define $F \circledast G \in \mathbb{R}^{i_1,...,i_n}$ the Hadamard product of F and G such that:

$$(F \circledast G)_{i_1,...,i_n} = F_{i_1,...,i_n} G_{i_1,...,i_n} \tag{3}$$

Monomial Tensor

Let $x \in \mathbb{R}^n$ a vector. Then $m(x) \in \mathbb{R}^{2^n}$ is a vector such that each entry is a unique multilinear monomial of coefficient 1. More specifically, $m(x)$ is defined as follows:

$$m(x) := \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \otimes ... \otimes \begin{pmatrix} 1 \\ x_n \end{pmatrix} \tag{4}$$

Let $x \in \mathbb{R}^n$ a vector. We define $M(x) \in \mathbb{R}^{\times_2 n}$ as the monomial tensor that is defined as follows. We can see that $M(x)$ is a reshaping of the monomial vector $m(x)$, $a_i \in \{0, 1\} \forall i$.

$$(M(x))_{a_1,...,a_n} := \prod_{i=1}^{n} x_i^{a_i} \tag{5}$$

1.2 Tensor decomposition and Properties

Let $F \in \mathbb{R}^{i_1,...,i_n}$ and $F_k \in \mathbb{R}^{i_k \times R}$

Let $F \in \mathbb{R}^{\times_2 n \times p}$ be a tensor of order $n+1$. And $\forall i \in \{0, .., n\}$ there exist $F_{x_i} \in \mathbb{R}^{2 \times r}$ and $F_\phi \in \mathbb{R}^{p \times r}$ such that, where $F_i(k)$ is the k_{th} column of F_{x_i}.

$$F = \sum_{k=0}^{r} F_{x_1}(k) \otimes ... \otimes F_{x_n}(k) \otimes F_\phi(k) \tag{6}$$

Then we say that $[F_{x_1}, ..., F_{x_n}, F_\phi]$ is a decomposition of rank r of F. Additionally, if we consider a vector $x \in \mathbb{R}^n$ we have that the following equation holds true:

$$< F, M(x) >= F_\phi(F_{x_1}^T \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \circledast \cdots \circledast F_{x_n}^T \begin{pmatrix} 1 \\ x_n \end{pmatrix}) \tag{7}$$

More generally, let $F \in \mathbb{R}^{i_1 \times ... i_n}$ then $[F_1, ..., F_n]$, where $F_k \in \mathbb{R}^{R \times i_k}$, is an exact rank R decomposition if the following equation is respected:

$$F = \sum_{r=0}^{R} F_{x_1}(r) \otimes ... \otimes F_{x_n}(r) \otimes F_\phi(r) \tag{8}$$

The rank R of a tensor F is the minimum integer so that there exist an exact decomposition. Finding the rank of a tensor is usually a hard problem. In order to overcome this hurdle, we fix an integer and find the best decomposition possible for that rank. This is usually done thoush an optimisation problem that minimizes the Frobenius norm between the original tensor and the approximate decomposition.

2 Multilinear models

A multi linear equation takes the form of:

$$p(x) = \sum_{i_n=0}^{i_n=1} ... \sum_{i_1=0}^{i_1=1} a_{i_1,...,i_n} x_1^{i_1} ... x_n^{i_n} \tag{9}$$

$$\tag{10}$$

We can stablish a relationship between the coefficients of a multilinear polinomial in $x\mathbb{R}^n$ and the entries of a tensor $F \in \mathbb{R}^{\times_2 n}$. The following equation describes the relationship between the multi linear polynomial $p(x)$ and a tensor F.

$$F_{i_1,...,i_n} = a_{i_1,...,i_n} \tag{11}$$

2.1 Multilinear Time Independent Models

Multilinear Time Independent models or (MTI) describes a system and its evolution over time. The values used to define the system's operations are the following:

- $x \in \mathbb{R}^n$ the state.
- $y \in \mathbb{R}^p$ the output.
- $u \in \mathbb{R}^m$ the input.

Additionally, we need to consider the system's equations. Using the previous tensor notation we can describe the system equations explicitly as follows, where $F \in \mathbb{R}^{\times_2 (n+m)\times n}$, $G \in \mathbb{R}^{\times_2 (n+m)\times p}$:

$$\dot{x} =< F|M(x,u) > \tag{12}$$

$$y =< G|M(x,u) > \tag{13}$$

Another way to define an MTI's evolution is through the use of an implicit multilinear (iMTI) equation. In this case the tensor $H \in \mathbb{R}^{\times_2 (2n+m+p)\times n+p}$ defines the state evolution. Additionnally, we would need to add the conditions $det(\partial_{\dot{x}} H) \neq 0$, $det(\partial_y H) \neq 0$ for the system to be properly defined (implicit function theorem). Otherwise, more equations would be needed to define the system completely.

$$< H|M(\dot{x},x,u,y) >= 0 \tag{14}$$

It is clear that the explicit formulation can be transformed into the implicit one such that the resulting system is: Where $\overline{H}$ is chosen appropriately to represent the $(n+p)$ multilinear equations $\dot{x} - < F|M(x,u) >= 0$ and $y - < G|M(x,u) >= 0$.

$$< \overline{H}|M(\dot{x},x,u,y) >= 0 \tag{15}$$

2.2 Differential Algebraic Equations

Differential Algebraic Equations (DAE) are a set of problems defined by differential equations and another set of algebraic equations. An explicit formulation of a DAE can be done as follows: Where $x \in C(\mathbb{R}^n, \mathbb{R}), y \in C(\mathbb{R}^n, \mathbb{R}), t \in \mathbb{R}$

$$\dot{x} = f(x, z, t) \tag{16}$$
$$0 = g(x, z, t) \tag{17}$$

It is clear from the previous equation that ODEs are a special case of DAEs where $g(x, z, t) = 0$. In general, DAEs are tighly linked with ODEs. One way to link them together is though the concept of index. The (differential) index of a DAE is the number of times the equations of the DAE need to be differentiated in order to recover an explicit ODE. However, it is important to note that the index is dependent on the formulation of the DAE as well as on the solution provided. Usually, a higher index means the DAE involves more complexity specially when solving it numerically.

The explicit form of a DAE is specifically useful since it enables To solve numerically the problem in a straighforward manner. First, considering the implicit function $aux(x, t)$ that yields a set of variables z such that $g(x, aux(x, t), t) = 0$. And then substituting in the first equation. In pseudo-code this yields:

Algorithm 1 Implicit Stepforward

```
Initialize x_0 ∈ X initial state.
Initialize z_0 ∈ Z initial auxiliary variables.
for t = 0 to T do
    Define aux(x, t)
    Solve f(x_{t+1}, aux(x_{t+1}, t + 1), t + 1) = 0 for variable x_{t+1}.
    x_{t+1} ← x*
end for
```

However, this method does not always yield good numerical results, specially with DAEs of higher index. For this purpose, we will explore different numerical methods and analyse how they compare to each other and, more specifically, which one fits the type of DAE that we are studying best.

3 Numerical Methods

3.1 Numerical Methods for ODE's

ODEs can be represented by the following implicit equation:

$$f(x, \dot{x}, t) = 0 \tag{18}$$

One specific method to solve a system of ODE are implicit methods. This is a type of family of methods that discretizes the time variable into a set of timesteps $\mathcal{T} := \{1, ..., T\}$. An initial value x_0 for x is given and then the following values are found by solving an equation involving the function in **??**. More specifically, we solve the equation $g(x_{t+1}, x_t) = 0$ where x_t is known, x_{t+1} is the unknown, anf g has the following expression:

$$g(x_{t+1}, x_t) = f(x_{t+1}, \frac{x_{t+1} - x_t}{\Delta t}, t + 1) \tag{19}$$

This method consists of solving iteratively an implicit system of equations. These type of systems are usually solved using Newton-Raphson methods. These methods demand a fixed number of evaluations of the function to be solved. This means that, if we define c as the number of calculations needed to evaluate the implicit function then the complexity of the algorithm is $\mathcal{O}(ct)$.

We want to adapt this method to solve an explicit DAE as in **??**. To do so we define a new function $\overline{F}(x, u)$ that can be used as the implicit function.

$$\overline{F}(\dot{x}, x, z, u) = \begin{pmatrix} \dot{x} - < F | M(x, z, u) > \\ < G | M(x, z, u) > \end{pmatrix} \tag{20}$$

$$\overline{G}(x_{t+1}, x_t, z_{t+1}, u_{t+1}) = \overline{F}(\frac{x_{t+1} - x_t}{\Delta t}, x_t, z_{t+1}, u_{t+1}) \tag{21}$$

We can describe the method in algorithmic form as follows:

Algorithm 2 Implicit Stepforward

1: Initialize $x_0 \in X$ initial state.
2: Initialize $z_0 \in Z$ initial auxiliary variables.
3: **for** $t = 0$ to T **do**
4: Solve for x_{t+1}, z_{t+1}, $\overline{G}(x_{t+1}, x_t, z_{t+1}, u_{t+1}) = 0$
5: $x_{t+1} \leftarrow x*$
6: $z_{t+1} \leftarrow z*$
7: **end for**

This method also consists of solving iteratively an implicit equation. In this case we are solving, a system of $(n + m)$ equations. If we define $c(n, m)$ as the number of calculations necessary to evaluate the function f.

Ill Conditioning & Solvability

A problem is said to be well posed if a solution to the problem exists and if said solution is unique. In the case of the proposed algorithms the well posedness of the problem depends on the well posedness of the intermediate Newton-Raphson methods. The most frequent problem that arise when dealing with

Let F, G the tensor defining a multilinear DAE in explicit formulation such that $det(G_z(x, z, t)) \neq 0 \forall x \in X, z \in Z, t \in \mathcal{T}$ then the algorithm **??** is well posed.

3.2 Tensor Construction

Let $F \in \mathbb{R}^{n,n,p}$ representing p multilinear equations of order 2. More specifically, $\forall k \in \{1, ..., k\}$ the tensor F represents the following polynomial $q_k(x)$:

$$q_k(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} (\frac{1 + \delta_{i,j}}{2}) x_i x_j F_{i,j,k} \tag{22}$$

The tensor F can be decomposed as follows:

$$F = \sum_{r=1}^{R} F_1(r) \otimes F_2(r) \otimes F_3(r) \tag{23}$$

$$F_{i,j,k} = \sum_{r=1}^{R} F_1(i,r) F_2(j,r) F_3(k,r) \tag{24}$$

We now consider the tensor $G \in \mathbb{R}^{\times_2 n \times p}$. representing the order 2 multilinear equations as a full tensor. It is clear that in this case the tensor G entries usually take 0 as a value. Indeed we have that:

$$G_{i_1,...,i_n} = \begin{cases} F_{i_k,i_l} & \text{if } i_k = i_l = 1, \sum_m i_m = 2 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

Therefore we want to take advantage of the sparsity of G to find its decomposition. Similarly as for F, the tensor G can also be decompose, and each entry is equal to:

$$G_{i_1,...,i_n} = \sum_{r=1}^{R} G_1(i_1, r) \otimes ... \otimes G_n(i_n, r) \tag{26}$$

We want to derive from this equation a relationship between both decompositon however this is not alwyas easy. Since we are dealing with an order 2, multilinear systems we can express the equation in

$$< F, x \otimes x > = F \circledast (x \otimes x) \tag{27}$$

$$= (\sum_r f_1(r) \otimes f_2(r)) \circledast (x \otimes x) \tag{28}$$

$$= \sum_r (f_1(r) \otimes f_2(r)) \circledast (x \otimes x) \tag{29}$$

$$= \sum_r < f_1(r), x >< f_2(r), x > \tag{30}$$

$$\tag{31}$$

When dealing with p equations this can be generalized to **??** where $f_1, f_2 \in \mathbb{R}^{n \times p}$. However, this equation excludes the lower order monomials, in this case

$$< F, x \otimes x >= \sum_{r=1}^{R} f_1(r)^T x \circledast f_2(r)^T x \tag{32}$$

If this is done for higher order polinomials of order q we can find:

$$< F, \otimes_{i=1}^{q} x >= \sum_{r=1}^{R} f_1(r)^T x \circledast ... \circledast f_q(r)^T x \tag{33}$$

Finally, by combining the previous equations we can express a multilinear function of maximum order q as follows, where $F_{expanded} \in \mathbb{R}^{\times_{n+1} q \times p}$. The new change includes all the lower order multinomial monomials.

$$F_{expanded} = [F_1, ..., F_q, F_\phi] \tag{34}$$

$$< F_{expanded}, \otimes_{i=1}^{q} \begin{pmatrix} 1 \\ x \end{pmatrix} > = F_\phi(F_0^T \begin{pmatrix} 1 \\ x \end{pmatrix} \circledast \cdots \circledast F_n^T \begin{pmatrix} 1 \\ x \end{pmatrix}) \tag{35}$$

This decomposition reduces the computational cost of the function evaluation from $\mathcal{O}(pn^q)$ to $\mathcal{O}(rnq)$. If we consider q a fixed parameter of the algorithm we are reducing the cost from a polynomial of order q to a linear cost, which is a sificant improvement.

$F \in \mathbb{R}^{\times_n q}$ be a symmetric tensor of order q we say that $[F_1, ..., F_n]$ with $f \in \mathbb{R}^{n \times R}$ is an R-rank symmetric decomposition if and only if $F_i = F_j =: a \quad \forall i, j \in \{1, \ldots, n\}$. Algebraically, this yields the following equation:

$$F = \sum_{r=1}^{R} (\bigotimes_{i=1}^{n} f(r)) \tag{36}$$

property

Given a symmetric tensor F we can further simplify the expression **??** to the following:

$$< F_{symmetric}, \otimes_{i=1}^{q} x > = \sum_{r=1}^{R} (f(r)^T x)^q \tag{37}$$

$$=< f(\bigodot_{i=1}^{q-1} f^T x) | x > \tag{38}$$

$$= (\bigodot_{i=1}^{q-1} x^T f) f^T x \tag{39}$$

With this simplication the number of computations goes from $\mathcal{O}(rnq)$ to $\mathcal{O}(r(n+q))$. This can enable us to simulate problems with a higher maximum degree. However, we remark that this decomposition is only handling an equation in one dimension. If we want to expand this decomposition to handle the whole DAE we would actually need to find a decomposition of $F \in \mathbb{R}^{\times_n (q+1)}$ such that, with $f_\phi \in \mathbb{R}^{n \times R}$ the factor related to last dimension.

$$F = [f, ..., f, f_\phi] \tag{40}$$

$$< F, \otimes_{i=1}^{q} x >= f_\phi(\bigotimes_{i=1}^{q} f^t x) \tag{41}$$

3.3 Auxiliary variables

WORK IN PROGRESS

Absolute value of a variable

Let v a variable from the original formulation, we add 3 auxiliary variable v^{abs}, v^+, v^-.

$$(v^{abs} - v)(v^{abs} + v) = 0 \tag{42}$$
$$v^+.v^- = 0 \tag{43}$$
$$v^+ + v^- = v^{abs} \tag{44}$$

By adding these 3 new equations we now can use non polynomial operations for the variable v operations. We remark that these new equations maintain the balance between the number of equations and the number of variables.

$$|v| = v^{abs} \tag{45}$$
$$\max(v, 0) = v^+ \tag{46}$$
$$\min(v, 0) = v^- \tag{47}$$

Let us consider the simple constraint for the variable v, $v_{min} \geq v \geq v^{max}$. Then we can define the the following equation so that v respects the previous constraints.

$$\frac{\partial (v - v_{\min})^+}{\partial t} = \begin{cases} 0 & \text{if } v \leq v_{\min} \\ 1 & \text{if } v \geq v_{\min} \end{cases} \qquad \frac{\partial (v - v_{\max})^-}{\partial t} = \begin{cases} -1 & \text{if } v \leq v_{\max} \\ 0 & \text{if } v \geq v_{\max} \end{cases} \tag{48}$$

Finally, by adding a new variable $v_{constrained}$ and combining it with the previous derivatives we can define a new equation that makes $v_{constrained}$

$$-v_{constrained} \frac{\partial (v - v_{\min})^+}{\partial t} \frac{\partial (v - v_{\max})^-}{\partial t} = v_{constrained} \tag{49}$$

$$v_{constrained} = \begin{cases} v & \text{if } v \in [v_{\min}, v_{\max}] \\ 0 & \text{otherwise} \end{cases} \tag{50}$$

3.3.1 Generalized Constraints

Let us consider a constraint of the type $f(\dot{x}(t), x(t)) \geq 0$ in our system. We want to transform this inequality into an equality that can be included in the DAE formulation. A simple way to transform this inequality into an equality is to consider the negative part of function f and then derivate it. By adding the following equality to the model we effectively ensure that the constraint is respected. Additionally, the DAE multi linear formulation is also maintained since it can be added by just adding new polynomial equalities as seen with dealing with the absolute value. This formulation is explored further in the case study 4.

$$f(\dot{x}(t), x(t)) \geq 0 \tag{51}$$

$$\longleftrightarrow \frac{\partial (f)^-}{\partial t}(\dot{x}(t), x(t)) = 0 \tag{52}$$

Trigonometric functions

Let θ a variable such that $cos(\theta), sin(\theta)$ appear in the model equations. We add 2 auxiliary variables u_θ, v_θ.

$$cos(\theta) = u_\theta \tag{53}$$

$$sin(\theta) = v_\theta \tag{54}$$

$$u_\theta^2 + v_\theta^2 = 1 \tag{55}$$

4 Case Study 1

For this case study we define a non linear EDO of the following form: Where $x \in \mathcal{C}(\mathbb{R}^n, \mathbb{R})$, $A, B, C \in \mathbb{R}^{n \times n}$.

$$\dot{x}(t) = Ax(t) + B(x(t) \odot x(t)) + x^T(t)Cx(t) \tag{56}$$

The equation represents an EDO with a linear component A and a quadratic component B. The objective is to transform this ODE into a multi linear DAE. For this purpose, we add n auxiliary variables in the form of the vector v as well as n algebraic equations.

$$B(x(t) \odot x(t)) \tag{57}$$

$$\Leftrightarrow \begin{cases} B(u(t) \odot x(t)) \\ u(t) = x(t) \end{cases} \tag{58}$$

This yields the following multi linear DAE system:

$$\dot{x}(t) = Ax(t) + B(u(t) \odot x(t)) \tag{59}$$
$$0 = u(t) - x(t) \tag{60}$$

$$A = \begin{pmatrix} -0.5 & 0.1 & 0 & 0 & 0 \\ 0.5 & -0.2 & 0.1 & 0 & 0 \\ 0 & 0 & -0.3 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & -0.1 \\ 0 & 0 & 0 & 0 & -0.2 \end{pmatrix},$$

$$\mathrm{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathrm{C} = \begin{pmatrix} 0 & 0.05 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.1 \end{pmatrix} \tag{61}$$

Finally we transform into the tenorsorized formulation of an iMTI. Where $F, G \in \mathbb{R}^{\times_2 2n \times n}$

$$\dot{x}(t) =< F|M(x,u) > \tag{62}$$
$$0 =< G|M(x,u) > \tag{63}$$

$$F_{i_1,...,i_n,j_1,...j_n,k} = \begin{cases} A_{k,l} & \text{if } i_l = 1 \text{ and } \sum_{m=1}^{n} i_m + j_m = 1 \\ B_{k,l} + & \text{if } i_l = 1, j_l = 1 \text{ and } \sum_{m=1}^{n} i_m + j_m = 2 \\ 0 & \text{otherwise} \end{cases} \tag{64}$$

$$G_{i_1,...,i_n,j_1,...j_n,k} = \begin{cases} 1 & \text{if } i_l = 1, l = k \text{ and } \sum_{m=1}^{2} i_m + j_m = 1 \\ -1 & \text{if } j_l = 1, l = k \text{ and } \sum_{m=1}^{2} i_m + j_m = 1 \\ 0 & \text{otherwise} \end{cases} \tag{65}$$

$$\tag{66}$$

5 Case Study 2

For this case study we define a non linear EDO of the following form: Where $x \in \mathcal{C}(\mathbb{R}^n, \mathbb{R})$, $A_i \in \mathbb{R}^{n \times n} \quad \forall i$.

$$\dot{x}_i(t) = x^T(t) A_i x(t) \quad \forall i \in \{1, ..., n\} \tag{67}$$

Using the same procedure done in the first case study we build the explicit multilinear tensor equations defining the model.

$$\dot{x}(t) =< F|M(x,u) > \tag{68}$$

$$0 =< G|M(x,u) > \tag{69}$$

$$F_{i_1,\ldots,i_n,j_1,\ldots j_n,k} = \begin{cases} A_{k,l,m} + A_{k,m,l} & \text{if } l \neq m, \quad i_l = 1, i_m = 1 \text{ and } \sum_{m=1}^{n} i_m + j_m = 2 \\ A_{k,l,l} & \text{if } i_l = 1, j_l = 1 \text{ and } \sum_{m=1}^{n} i_m + j_m = 2 \\ 0 & \text{otherwise} \end{cases} \tag{70}$$

$$G_{i_1,\ldots,i_n,j_1,\ldots j_n,k} = \begin{cases} 1 & \text{if } i_l = 1, l = k \text{ and } \sum_{m=1}^{n} i_m + j_m = 1 \\ -1 & \text{if } j_l = 1, l = k \text{ and } \sum_{m=1}^{n} i_m + j_m = 1 \\ 0 & \text{otherwise} \end{cases} \tag{71}$$

(72)

Let's us now analyze the DAE's index. By virtue of the equations being in explicit form the first equation are already a ODE so they are of differential index 0. On the other hand the algbraic equations consist exclusively of equations of the form $u_i = x_i$, therefore differentiating the n equations once already gives a EDO. This means that the DAE in this form is of index 1.