Order of MBD Algorithms

Concurrent Dynamics International

June 2017

Content

- Introduction
- MBD Order Definition
- Examples

Introduction

- The speed of execution of processes, algorithms (i.e. programs, subroutines) or computations involved in a multibody dynamics (MBD) problem is very important to analysts and engineers who write programs to solve these problems.

- We will use MBD processes, MBD algorithms and MBD computations interchangeably in the following discussions

- Speed of a MBD process is a measure of the efficiency of the underlying algorithms and can be quantified by the number of additions and multiplications (arithmetic operations) involved in the calculations

- Speed is also a function of the number of bodies (*N*) in the MBD problem

MBD Order Definition

- An MBD process is order(N) or O(N) if it takes aN arithmetic operations to execute it for some integer a.
- An MBD process is O(N^3) if it takes $aN+bN^2+cN^3$ arithmetic operations to execute it for some integers a, b and c.
- The proportionality constants a, b and c above are independent of how the MBD bodies are connected and they are independent of N
- An MBD process done with an O(N^3) procedure is significantly slower than that done with an O(N) procedure as N increases
- So, how can one tell the order of an algorithm?

- A typical procedure in a multibody dynamics MBD program involves an *N*-step do-loop as given by

```
for i=1:N  (or  for i=N:1)
    Algorithm_X(i,--)  ;  %  Alg. X takes some k arithmetic operations to compute
end
```

  - ❖ This procedure takes *kN* operations to execute => it is an O(*N*) procedure

- An *N*-step do-loop MBD Procedure is
  1. O(*N*) if Algorithm_X takes *k* arithmetic operations to execute over each body
  2. O(N^2) if Algorithm_X contains one or more *N*-step do-loops in series/parallel in it and none of them has nested *N*-step do-loops in them
  3. O(N^3) if Algorithm_X has two levels of nested *N*-step do-loops in it

- Each level of nested *j=1:N* loop increases the order by one

- An MBD algorithm generally consists of one or more *N*-step do-loops that are executed in series and/or in parallel. The algorithm is O(*N*) if every such do-loop is O(*N*). The algorithm is > O(*N*) if any such do-loop is > O(*N*), i.e. O(N^3) > O(*N*) for *N* > 8.

- An MBD solution process at each time, t, consists of kinematics computations given the state(t), controls computations, generalized accelerations determination, state propagation from t to t+dt. These computations are done serially. The order of the MBD process is the highest order of any of these computations.

- Published papers in MBD literature claim that $O(N)$ MBD solutions run much faster than $O(N^3)$ MBD solutions when $N > 8$

Examples

- Let A , B and C be *N* x *N* matrices. Let a, b and c be *N* x 1 vectors
  - c=a+b is O(*N*)
  - c=A*b is O(N^2)
  - C=A*B is O(N^3)
  - c=A*b is O(*N*) if A is diagonal or has one none-zero element per row
    => one multiply operation per element of c
  - C=A*B is O(*N*) if A and B are both diagonal
    => no need to compute off diagonal elements

- Gauss-Jordan elimination method or Gaussian elimination method used in solving generalized accelerations of an MBD problem is an O(N^3) procedure. This procedure has two levels of nested *N-do-loops* to put the inverse system mass matrix into a reduced row echelon form before solving for the generalized accelerations, i.e.
  - for *i=1:N => outer N-do-loop*
  - opsA(--) ; => k1 ops
  - for *j=i:N => level 1 nested N-do-loop*
  - opsB(--); => k2 ops
  - for *k=j+1:N = level 2 nested N-do-loop*
  - opsC(--); => k3 ops
  - end
  - end
  - end
  - Total ops= $aN+bN^2+cN^3$ ops for some constants a, b and c that are functions of k1, k2 and k3 => O(N^3)