第 XX 卷第 XX 期 XXXX 年 XX 月
1672-6553/XXXX/XX(XX)/001-28

动力学与控制学报
JOURNAL OF DYNAMICS AND CONTROL

Vol.XX No.XX
XX XXXX

多体系统动力学常用积分器的算法 *

任辉† 周平

（哈尔滨工业大学航天学院，黑龙江 150001）

摘要 本文将以单步法中的广义 α 族积分器和多步法中的BDF族积分器为主要讨论对象，详细介绍大型多体系统动力学软件中常见类型的积分器的算法细节.每族积分器都给出了不止一套计算公式，而且其对应求解微分代数方程组（DAE）的index可以为1、2或者3.除此以外，本文还着重介绍了微分代数方程组的误差估计、变阶变步长策略等关键技术；并讨论了大型DAE问题求解过程中的初始条件分析、Jacobian矩阵复用等重要环节的算法实现；对于BDF积分器族，文中还详细描述了高阶格式的非绝对稳定性、速度变量的误差估计等瓶颈问题的解决方案.全文以多体系统动力学软件的积分器程序实现为目标，强调在满足给定精度的条件下，如何提高计算效率和保证仿真运行的鲁棒性.另外，本文也简要介绍了在某些应用场合中有很大潜力的显式积分器族.通过分析和比较，文中还将指出各种算法的优缺点以及可能的改进方向，希望能够为研究人员和程序开发者提供一定的参考.由于篇幅限制，本文只列出了几个标准的算例比较，作为文中内容的补充；并给出了几种积分器性能比较的一般性结论.文中几乎所有方法都经由作者程序实现、测试和比较，并且相关算法的实现细节也都已尽量列出，可以很容易地编程实现并应用到实际问题的求解中去.

关键词 微分代数方程组， 积分器， 向后差分公式， 广义 α 方法， 显式积分器

DOI：10.6052/1672-6553-2020-098

引言

多体系统动力学得到的动力学方程组是微分代数方程组（Differential-Algebraic Equations，简称DAEs），其一般形式特别复杂.为了简单起见，本文以最典型的index-3的完整约束DAEs：

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q},t)\ddot{\boldsymbol{q}}+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}=\boldsymbol{Q}(\boldsymbol{q},\dot{\boldsymbol{q}},t)\\&\boldsymbol{C}(\boldsymbol{q},t)=\boldsymbol{0}\end{aligned}\tag{1}$$

为主要对象，介绍一下常见的通用型软件积分器的常用算法与实现细节.通常来说，通用型多体动力学软件的积分器主要强调以下几个方面的特性：

（1）精确性.通过控制积分过程的局部误差，可以大致保证计算结果的正确性和精度；

（2）高效性.在满足给定的误差条件下，自适应地选择尽可能大的积分步长，保证计算速度；

（3）鲁棒性.积分器应当可以稳定地求解所有常见类型的多体系统动力学方程组.

通用型软件中的动力学方程组在形式上要比方程组（1）繁琐得多.除了完整约束方程组之外，很多实际系统中还包含非完整约束方程组、控制器方程组（一阶常微分方程组或代数方程组）、离散时间方程组等等.但从数值方法的角度来看，最复杂也最难求解的还是类似方程组（1）的index-3的微分代数方程组.因此，本文主要以方程组（1）为分析对象，而只在文中简要讨论一下其他情况，主要是带非完整约束的DAEs的求解方法.

本文只强调适合大规模多体系统动力学仿真软件的通用型积分器技术，而不详细讨论针对某些专门问题的特殊积分器技巧.在通用型软件中，积分器一般是作为独立模块开发的，主要用来宏观控制求解进程，而尽量少与其他模块在底层交互.软件中的方程组模型一般是提前设定好的.当软件的积分器模块开发调试完成后，一般很少再作大的修改.随着求解问题领域和规模的扩大，当软件架构需要扩展升级时，积分器模块往往也需要整体重新设计和开发，此时亟需整理和消化前人开发过程中发展出来的技术.因此积分器模块中求解技术的维

2019-5-30 收到第1稿，2020-11-13 收到修改稿.
* 国家自然科学基金资助项目（11772101）
† 通讯作者 E-mail：renhui@hit.edu.cn

护和传承非常重要.还应该指出的是,尽管积分器的理论对程序的开发有绝对的指导作用,但好的积分器中的绝大部分有效的技术细节往往都来自实践中的反复测试、改进和提供,而非理论公式推导本身.

本文尽量整理了作者在科研和程序开发中学习和体会到的积分器的一些技术细节.文中绝大部分积分器都经作者亲自开发并测试过,希望能对我国自主研发多体系统动力学软件有所帮助.

动力学方程组的特性分类

动力学过程是非线性时变的.积分器必须能够自适应地追踪系统的时变特性,并选择合适的时间步长来复现真实的物理过程.要做到既不能错过重要的物理现象,也没必要采用过密的时间节点而浪费大量计算资源.尽管待求解的方程组都是方程(1)的形式,但在仿真计算过程中,由于具体系统的动力学过程千差万别,积分器的表现也是非常不同的.从动力学特性上来说,多体系统仿真中常见的DAE系统大致可以分为以下几类:

(1) 刚体为主导的系统.这类系统在大型多体系统动力学中最为常见.由于建模方法不同,所得方程组的数学特征也稍有不同.

• 通过递归(recursive)公式建立的动力学方程组,其主要特点是:方程的非线性很强,表达式相当复杂;其Jacobian 矩阵经常为满阵,但其规模相对较小.

• 由绝对坐标描述,通过拉格朗日方程建立的方程组,其主要特点是:约束方程的个数很多,可能与动力学方程的个数为同一量级;刚体之间的相互作用绝大部分由作用力和约束方程来描述.

一般来说,多刚体系统方程的刚性问题往往并不严重,而且光滑性比较好,往往可以用高阶格式并结合比较大的时间步长进行数值积分.

(2) 柔性体动力学为主导的系统.一般来说,在这种多体系统中,刚体的个数仍然远多于柔性体的个数.但每个刚体只有6个自由度,而每个柔性体的模态坐标个数可能比较多.更重要的是,模态坐标对应的固有频率可能很高,会在DAE系统中引入比较强的刚性,导致Jacobian矩阵的条件数比较大.实践中发现,积分器本身的耗散往往不能有效地衰减模态坐标对应的高频振动,因此推荐在模态坐标中引入少许线性阻尼或结构阻尼以提高仿真计算效率.

(3) 大变形体为主导的系统.通常包含用几何精确方法或者绝对节点坐标方法建立的大变形有限元模型,需要引入几何非线性和/或材料非线性的影响.这类系统的动力学方程个数通常远远多于约束方程个数,且方程组主要由二阶动力学方程组成,具有特殊的稀疏格式.一般说来,从结构动力学中发展出来的方法,例如广义 α 方法或者HHT方法,在求解这类问题中被证明是很有效的.

(4) 接触碰撞过程为主导的系统.在散体颗粒系统或者传动系统的动力学仿真中,经常会遇到这一类系统.首先,由于方程结构的变拓扑性,稀疏矩阵求解器的符号LU分解不再能复用,而根据具体条件进行矩阵LU分解是计算中所必需的.其次,常用的罚函数方法得到的接触碰撞动力学方程组的刚性和高频振荡问题都比较严重.最后,这类系统仿真过程对于接触体的几何形状和接触模型比较敏感,采用不同的算法求出的接触力可能差异比较大.

(5) 有特殊要求的一些多体系统.例如在虚拟现实仿真中,要求做到实时仿真,需要用到实时积分;高速旋转系统中,需要对转动进行有效精确描述;长时间运行的无耗散系统,对能量守恒要求比较高,需要用到守恒格式的积分器;对最优控制设计得到的系统,可能需要求解的并非初值问题,而是边值问题等等.这些问题将不在本文中详细讨论.

实际中遇到的大型多体系统仿真问题,往往是上述几种类型的组合.因此实践中需要的通用型积分器,要求能同时求解所有这些类型的方程组.此外,具体的应用场合也非常重要,而不同场合考验的是积分器不同方面的特性.例如,很多设计过程需要反复进行大量的动力学仿真以优化产品的动力学性能;而在另一些场合中,实时仿真的要求可能更加重要.一个优秀的通用型积分器应该在求解常见应用问题的时候具有高效性,而同时在不常见的特殊应用中也应该具有鲁棒性.

常见通用型DAE积分器简介

通用型DAE积分器需要有能力做到:

(1) 自适应地选择积分步长;对于变阶积分器,还需要自适应地选择阶数;

(2) 有效地进行误差估计,既可用于保证计算精确性,又可用于变阶变步长;

(3) 不连续性和不稳定性现象的检测与处理,

自动进行算法调整和事件解决;

(4) 有效处理由于数值原因导致的速度、加速度、拉格朗日乘子曲线的不光滑性;

(5) 在奇异位形、不光滑点处积分器遇到无法解决的问题时可以自动重启.

所有的DAE积分器都源自相应的常微分方程(Ordinary Differential Equations,简称ODEs)积分器.但DAEs与ODEs在数值特性上有很大的不同[1],而求解算法的理论和程序实现细节也不同.

常用的通用型DAE积分器大部分都是隐式格式的,其主流算法通常认为主要有三大类:基于向后差分公式(Backward Difference Formula,简称BDF)的积分器族,基于广义α(Generalized-α)方法的积分器族,与基于隐式龙格库塔(Implicit Runge-Kutta,简称IRK)方法的积分器族.这些方法可以直接求解DAE方程组(1)或其变形.此外,将DAEs转化为ODEs并采用ODE积分器来进行计算也是重要的DAE求解思路.

BDF族积分器包含了一系列变阶变步长积分器.其具体的公式有好几种变形格式[2-4];求解的DAE方程组的index可以为1、2或者3;截断误差的阶数可以从1阶到5阶.对于光滑性比较好的多体系统,例如没有接触碰撞的多刚体系统,可以用高阶格式结合大的时间步长进行积分,求解效率非常高.这类方法的缺点是其低阶格式对低频运动的阻尼比较大,因此如果系统的光滑性不太好,其动力学响应可能会有比较大的衰减.Gear[5]最早将刚性ODEs求解中常用的BDF方法引入到index-1的微分代数方程组的求解中,并将其应用于电路系统和化学过程的动力学仿真.Orlandea等[6]将其进一步推广到了多体系统动力学的应用场合.经过深入研究,Brenan等[7]从数学角度证明了BDF方法在一般的index-1和index-2的DAE系统中的收敛性;同时指出[8],对于一般的index-3的DAE系统,该方法的收敛性不能从理论上得到保证.但大型商业软件的实践证明,在一定限制条件下[9],BDF方法完全可以用来求解index-3的多体动力学方程组(1),而且到目前为止该方法在所有方法中似乎是相当高效的.工业界中,BDF方法仍然是一些大型商业软件的缺省求解器,甚至在很多结构动力学的应用场合,其性能仍然非常卓越.直接用BDF积分器求解方程(1)遇到的最大问题是速度变量的误差估计比较困难[10-11].这是因为Jacobian矩阵的条件数很大,求解带来的舍入误差会严重污染速度变量的截断误差,往往会导致对速度变量无法进行有效的误差估计.这一问题有几种解决方案,投影滤波方法[11]即是其中之一.Gear等[12]证明,index-3的多体动力学系统可以等价地转化为数值特性更好的index-2的DAE系统,因此可以更稳定地利用BDF方法求解.实践表明,这样求解的结果精度更高,尤其是速度和加速度的连续性和光滑性更好.但同时,得到的index-2的DAE系统的方程规模会更大,因此计算量会稍微大一些.BDF积分器在实践中还有很严重的问题;三阶以上的BDF格式不是绝对稳定的,因此在求解宽频动力学问题时很容易有频率进入不稳定性区域,导致阶数和步长的估计不再可靠.在某些应用场合甚至需要手动设置最大阶数为2才能顺利计算下去,非常不方便.如何自适应地选择积分器的阶数来避免不稳定性问题,这一方面的研究也已取得了一些进展[13-14].作者认为,BDF族积分器的性能还存在着一定的提升空间.

近年来,广义α族方法[15],包括作为同类格式的HHT方法[16],受到了更多人的研究和使用.这类方法是传统Newmark方法的高精度扩展,具有统一的二阶截断误差精度.与BDF方法不同,这种方法计算中用到了位置、速度与加速度之间的关系,可以直接离散二阶动力学方程组;计算中修正的量本质上是加速度和拉格朗日乘子.这类方法具有低频不衰减而高频衰减的优良特性[17-18],鲁棒性很好,特别适合求解结构动力学问题或者柔性多体系统动力学问题等.此外,这种方法程序设计比较简单,且可以推广到index-2格式[19].尽管经过推广,这种方法也被应用到了一阶动力学方程组,但其数值精度有所损失.最后,该族积分器的精度保持为一阶或二阶,对于光滑的动力学系统,计算效率偏低.

隐式龙格库塔算法[20-21]也可以用来求解大型多体系统.与广义α方法类似,这种方法也是单步法,其自动选步长的算法比较简单;同时,这种方法可以实现变阶,对于光滑的动力学系统,理论上也可以采用高阶格式结合大的时间步长来计算[22].隐式龙格库塔算法既可以直接求解index-3的DAE系统[23]例如方程组(1),又可以求解index-2的DAE系统[24],很适合多体系统动力学仿真的应用[25].这种方法的缺点是,高阶格式的方程规模比较大,对于大型系统计算效率不高.因此实践中的隐式龙格库塔算法大多采用对角格式(Diagonally Implicit

Runge-Kutta，简称 DIRK）[26-27]. 数值实践发现，大部分高阶隐式龙格库塔格式在实际计算中事实上达不到理论上的阶数[28]，而具有阶数饱和现象[29]. 最近这方面的研究有一些新的进展[30]，其有效性还有待进一步的测试. 到目前为止，大型通用型软件很少选择隐式龙格库塔方法作为主要的积分器算法，因此本文中也不再详细讨论其计算细节.

以上积分器求解 DAEs 时都会遇到如下困难：

（1）Jacobian 矩阵的相关问题. 例如在约束系统的奇异位形下，Jacobian 矩阵可能是奇异的或近似奇异的；在刚性问题中，可能由于 Jacobian 矩阵条件数太大[31]，带来计算精度的降低、误差估计的失败或其他数值问题；

（2）对方程组（1）的直接计算仿真过程中，隐含速度约束往往不能得到有效地满足，而加速度和拉格朗日乘子的曲线往往会有强烈的不连续性（spikes 现象）. 此外，约束方程本身的不光滑性也可能会带来数值计算中的严重困难；

（3）微分代数方程解的连续性往往弱于外激励的连续性[1]，高频激励常常会导致计算结果出现一些数值不连续现象；而输入变量或者作用力在时间上的不光滑性会对积分器的鲁棒性带来很大的挑战；在接触碰撞等问题中，不连续性与高频激励同时出现，甚至会导致计算无法顺利开展下去；

（4）大多数积分器的误差估计和自适应步长选择策略都是基于动力学响应解的光滑性假设. 在各种非光滑情形，步长估计可能无效，往往引起大量的计算失败，大大增加整体计算量.

除了上面的直接方法以外，将 DAEs 转化为 ODEs 并用 ODE 积分器来进行仿真的方法也由来已久. 通常认为这类方法计算效率不高，鲁棒性也较差，因此很少应用于商业软件的实践. 如果约束方程组足够光滑，Baumgarte 降阶方法[32-33]常常计算效果也不错. 其它的一些降阶方法要么结合积分步长选择罚因子[34]，要么将结果在约束流形上投影[35]或用罚函数拉回[36]，要么采用修正的动力学方程组[37]，更详细的内容可以在综述文章[38]中找到. 通常这些降阶方法会引入一些高频的自由度，需要用刚性积分器或者特殊的显式积分器[39-40]来求解.

坐标分解方法也是一种相当有效的方法，而且计算精度较高. 利用约束方程将一部分广义坐标用和自由度个数相等的其他广义坐标表示出来，从而将动力学方程组（1）转化为 ODEs[41]，进而进行求解. 一般的多体动力学系统很难找到在仿真过程中一直适用的广义坐标，必须在某些时刻进行坐标转换. 其广义坐标的选择可以采用流形方法[42]或者投影方法[43]. Shabana 教授团队提出的两步隐式稀疏矩阵积分法[44-46]（two-loop implicit sparse matrix numerical integration，简称 TLISMNI 方法），在每一步积分中都要进行坐标分解，并保证其同时满足几何约束、速度约束和加速度约束. 这类积分器计算精度很高，但每步计算量太大. 如果能证实其积分步长可以取得相当大，足以摊平每步的计算成本，该方法也将适合用于通用型积分器. 作者尚未对该类积分器的效果进行测试，因此这部分内容也将不在本文中多做讨论.

1 微分代数方程组的初始条件

与 ODEs 不同，DAEs 的初始条件不能任意给出. 理论分析表明[8,10,21]，DAEs 的初始条件下约束方程需要非常高精度地满足，计算才能有效地开展下去. 方程组（1）中的约束方程组为：

$$\boldsymbol{C}(\boldsymbol{q},t)=\boldsymbol{0} \tag{2}$$

隐含速度约束

$$\dot{\boldsymbol{C}}(\boldsymbol{q},t)=\boldsymbol{C}_q(\boldsymbol{q},t)\dot{\boldsymbol{q}}+\boldsymbol{C}_t(\boldsymbol{q},t)=\boldsymbol{0} \tag{3}$$

和加速度条件 $\ddot{\boldsymbol{C}}(\boldsymbol{q},\mathrm{t})=0$，即

$$\boldsymbol{C}_q(\boldsymbol{q},t)\ddot{\boldsymbol{q}}=\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},t) \tag{4}$$

其中 $(\dot{\bullet})$ 代表对时间的全微分；而

$$\boldsymbol{\gamma}(\boldsymbol{q},\boldsymbol{v},t)=-(\boldsymbol{C}_q\boldsymbol{v})_q\boldsymbol{v}-2\boldsymbol{C}_{tq}\boldsymbol{v}-\boldsymbol{C}_{tt} \tag{5}$$

实际问题提供的初始条件 $\boldsymbol{q}_0$ 和 $\dot{\boldsymbol{q}}_0$ 不一定满足这些约束方程，其原因可能有以下几点：

（1）$\boldsymbol{q}_0$ 和 $\dot{\boldsymbol{q}}_0$ 无法准确给出，或者由于人为原因导致错误的初值设定；

（2）实践中，初值往往是测量出来的. 而一般测量误差远大于数值计算中要求的误差量级，导致初始约束方程不能满足计算的精度要求；

（3）设定的初始数据，其中有些量可能相对比较精确，而其他的可能并不精确. 因而需要调整不精确的初始变量以满足约束方程；

（4）求解过程中可能只确保了几何约束（2）能精确地满足，而速度约束和加速度约束不一定满足，导致积分器重启后的初始条件不合适.

初始条件分析的目的是优化初始数据，使各种约束条件都能精确满足，并算出初始加速度和约束力. 在此之前，首先要确认约束方程组是适定的.

冗余约束分析

约束方程可能是冗余的,甚至是互相矛盾的.在建模过程中,有时也可能故意用冗余约束的方法来保证系统安装的完整性.冗余约束分析的目的是找出数值特性最好的一组约束方程组,而删除所有其他的冗余约束.冗余约束分析的基本方法是对$\boldsymbol{C}_q(\boldsymbol{q}_0,0)$进行全选主元的LU分解,并选择一组最大程度线性无关的约束方程组,而其余的约束方程即被认为是冗余约束而舍弃.

初始奇异位形检测

多体系统动力学仿真偶尔会陷入奇异位形中.在一般的计算过程中,由于舍入误差的存在,多体系统滑入奇异位形的可能性比较小,一般不需要专门的处理.更常见的情形是,多体系统的初始位形本身就是奇异位形,导致在冗余约束分析中,有些非冗余约束可能会被误判为冗余约束而舍去.此时,仿真计算的模型并不是需要的物理模型,因此有必要进行初始奇异位形检测,从而将这些非冗余约束捡回系统.

初始奇异位形检测的算法是对初始位形进行摄动,然后重新进行冗余约束分析,验证得到的独立约束个数是否与原来相同.如果不同,说明初始位形有奇异性.发生了这种情况,需要给出警告,要求用户输入更多信息来选择初始位置.如果用户不能提供其他信息,就随机选择若干个与初始位形非常接近的摄动位形分别出发进行仿真计算,并验证它们的最终计算结果是否相近.

初始位形分析

给出初始广义坐标$\boldsymbol{q}_0$,希望求得适合约束方程的初始位形.这可以简单地通过求解优化问题:

$$\begin{aligned}&\min \mathbf{W}^q(\boldsymbol{q}-\boldsymbol{q}_0)\\&\boldsymbol{C}(\boldsymbol{q},0)=\mathbf{0}\end{aligned}\tag{6}$$

而得出.其中$\mathbf{W}^q$为对角权重矩阵,其权重可根据$\boldsymbol{q}_0$的先验知识而调整.一般对已知精确的分量,其对角权重取为10^6,而对于不确定的分量,其权重取为1.从方程(6)得到的优化方程组为

$$\begin{aligned}&\mathbf{W}^q(\boldsymbol{q}-\boldsymbol{q}_0)+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},0)\boldsymbol{\lambda}=\mathbf{0}\\&\boldsymbol{C}(\boldsymbol{q},0)=\mathbf{0}\end{aligned}\tag{7}$$

这可以用牛顿迭代法求解,初始迭代向量取为$\boldsymbol{q}_0$.迭代收敛后,用计算出来的$\boldsymbol{q}$取代初始变量$\boldsymbol{q}_0$.初始位形分析一般只涉及完整约束和/或状态变量本身的约束表达式,而不需要用到非完整约束等其他形式的约束方程.

初始速度分析

给出初始广义坐标$\dot{\boldsymbol{q}}_0$,为了求得适合速度约束方程的初始速度,可以简单地求解优化问题

$$\begin{aligned}&\min \mathbf{W}^v(\dot{\boldsymbol{q}}-\dot{\boldsymbol{q}}_0)\\&\boldsymbol{C}(\boldsymbol{q}_0,0)\dot{\boldsymbol{q}}+\boldsymbol{C}_t(\boldsymbol{q}_0,0)=\mathbf{0}\end{aligned}\tag{8}$$

其中$\mathbf{W}^v$为对角速度权重矩阵,其权重系数也可根据先验知识进行调整.得到的优化方程组为

$$\begin{pmatrix}\mathbf{W}^v & \boldsymbol{C}_q^{\mathrm{T}}\\ \boldsymbol{C}_q & \mathbf{0}\end{pmatrix}\begin{pmatrix}\ddot{\boldsymbol{q}}_0\\ \boldsymbol{\lambda}_0\end{pmatrix}=\begin{pmatrix}\mathbf{W}\dot{\boldsymbol{q}}_0\\ -\boldsymbol{C}_t\end{pmatrix}\tag{9}$$

它完全是线性方程组,可以直接求解出满足速度约束方程组的初始速度.然后,用计算出来的$\dot{\mathbf{q}}$取代初始变量$\dot{\boldsymbol{q}}_0$.应该强调的是,在一般情况下,方程(8)中也应该包括动力学系统所有非完整约束方程,因此通常也需要通过迭代才能求解.

初始加速度和约束力计算

很多积分器需要提供初始加速度和拉格朗日乘子的值,这可以由方程(1)和(4)计算出来:

$$\begin{pmatrix}\mathbf{M} & \boldsymbol{C}_q^{\mathrm{T}}\\ \boldsymbol{C}_q & \mathbf{0}\end{pmatrix}\begin{pmatrix}\ddot{\boldsymbol{q}}_0\\ \boldsymbol{\lambda}_0\end{pmatrix}=\begin{pmatrix}\boldsymbol{Q}\\ \boldsymbol{\gamma}\end{pmatrix}\tag{10}$$

通过初始值分析,既精简了约束方程,又得到了更合适的初始坐标$\boldsymbol{q}_0$、速度$\dot{\boldsymbol{q}}_0$、加速度$\ddot{\boldsymbol{q}}_0$和拉氏乘子$\boldsymbol{\lambda}_0$,是开展动力学仿真前的基础准备.

2　运动学问题求解方法

运动学问题是动力学系统的自由度为0的特殊情形.此时,约束方程组(2)足以决定整个系统的运动学.另外,此时Jacobian矩阵$\boldsymbol{C}_q(\boldsymbol{q},0)$为方阵,且除了个别奇异位形以外是可逆的.运动学问题实际上就是用Newton迭代求解方程(2)

$$\boldsymbol{q}_n^{(k+1)}=\boldsymbol{q}_n^{(k)}-\left[\boldsymbol{C}_q\left(\boldsymbol{q}_n^{(k)},t_n\right)\right]^{-1}\boldsymbol{C}\left(\boldsymbol{q}_n^{(k)},t_n\right)\tag{11}$$

求解的过程要注意以下几点:

(1)　初始估计$\boldsymbol{q}_n^{(0)}$是用上一步的位置$\boldsymbol{q}_{n-1}$、速度$\dot{\boldsymbol{q}}_{n-1}$和加速度$\ddot{\boldsymbol{q}}_{n-1}$估计得出

$$\boldsymbol{q}_n^{(0)}=\boldsymbol{q}_{n-1}+h\dot{\boldsymbol{q}}_{n-1}+\frac{1}{2}h^2\ddot{\boldsymbol{q}}_{n-1}\tag{12}$$

(2) 理论上,迭代时间步长h可以任意选取,但考虑到初始估计(12)必须足够精确以保证Newton迭代(11)的收敛性,h也不能太大.

(3) 由于Jacobian矩阵的生成和LU分解计算都比较昂贵,实用中常用拟Newton方法迭代.

计算得到了t_n时刻的$\boldsymbol{q}_n$之后,可以进而求出t_n时刻的速度,只需代入方程(3),即

$$\boldsymbol{C}_q(\boldsymbol{q}_n,t_n)\dot{\boldsymbol{q}}_n=-\boldsymbol{C}_t(\boldsymbol{q}_n,t_n)\tag{13}$$

求解得出$\dot{\boldsymbol{q}}_n$.更进一步,要求解t_n时刻的加速

度,需要用到方程(4),即

$$\boldsymbol{C}_q(\boldsymbol{q}_n, t_n)\ddot{\boldsymbol{q}}_n = \boldsymbol{\gamma}(\boldsymbol{q}_n, \dot{\boldsymbol{q}}_n, t) \tag{14}$$

最后,很多应用中还需要算出约束力,只需求解

$$\boldsymbol{C}_q(\boldsymbol{q}_n, t_n)\boldsymbol{\lambda}_n = \boldsymbol{Q}(\boldsymbol{q}_n, \dot{\boldsymbol{q}}_n, t_n) - \mathbf{M}(\boldsymbol{q}_n)\ddot{\boldsymbol{q}}_n \tag{15}$$

即可求出任意时刻的拉格朗日乘子 $\boldsymbol{\lambda}_n$,进而得到约束力.方程(13)、(14)和(15)中可使用同一 $\boldsymbol{C}_q$ 矩阵的LU分解.当然,运动学仿真在实用中只占很少份额,绝大部分问题还是需要动力学仿真方法,这将是下文的主要内容.

3 广义 α 族积分器

广义 α 方法最早是在结构动力学仿真中提出来的[17],它与稍早提出的HHT方法[18]实际上是等价的,而它们都是传统的Newmark方法的推广.这些方法都是单步法,其一般提法为:

问题 1 已知 t_n 时刻的状态变量 $\boldsymbol{q}_n$、$\dot{\boldsymbol{q}}_n$、$\ddot{\boldsymbol{q}}_n$ 和 $\boldsymbol{\lambda}_n$,及步长的估计 h 之后,求出 $t_{n+1} = t_n + h$ 时刻对应的变量 $\boldsymbol{q}_{n+1}$、$\dot{\boldsymbol{q}}_{n+1}$、$\ddot{\boldsymbol{q}}_{n+1}$ 以及 $\boldsymbol{\lambda}_{n+1}$,并作出误差估计以判断其是否满足误差条件要求.如果满足,则估计下一步的时间步长;如果不满足,则估计更合适的时间步长,并重新开始计算.

传统Newmark方法可用来求解常微分方程

$$\mathbf{M}(\boldsymbol{q}, t)\ddot{\boldsymbol{q}} = \boldsymbol{Q}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \tag{16}$$

假设已知 t_n 时刻的 $\boldsymbol{q}_n$、$\dot{\boldsymbol{q}}_n$ 和 $\ddot{\boldsymbol{q}}_n$,则在估计出步长 h 之后,只需将 $\ddot{\boldsymbol{q}}_{n+1}$ 当作未知量,取近似

$$\begin{aligned}\dot{\boldsymbol{q}}_{n+1} &= \dot{\boldsymbol{q}}_n + h\left(1-\boldsymbol{\gamma}\right)\ddot{\boldsymbol{q}}_n + h\boldsymbol{\gamma}\ddot{\boldsymbol{q}}_{n-1}\\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + h\dot{\boldsymbol{q}}_n + \left(\frac{1}{2}-\beta\right)h^2\ddot{\boldsymbol{q}}_{n-1} + \beta h^2\ddot{\boldsymbol{q}}_{n-1}\end{aligned} \tag{17}$$

其中参数 β 和 γ 用来调节计算精度和稳定性,且

$$\beta = \frac{1}{4}\left(\frac{1}{2}+\boldsymbol{\gamma}\right)^2 \tag{18}$$

将 $\ddot{\boldsymbol{q}}_n$ 作为 $\ddot{\boldsymbol{q}}_{n+1}$ 的初步估计 $\ddot{\boldsymbol{q}}^{(0)}_{n+1}$,令更新量(innovations)$\boldsymbol{x} = \ddot{\boldsymbol{q}}_n - \ddot{\boldsymbol{q}}^{(0)}_{n+1}$,则

$$\dot{\boldsymbol{q}}_{n+1} = \dot{\boldsymbol{q}}^{(0)}_{n+1} + h\boldsymbol{\gamma}\boldsymbol{x}, \qquad \boldsymbol{q}_{n+1} = \boldsymbol{q}^{(0)}_{n+1} + \beta h^2\boldsymbol{x} \tag{19}$$

其中

$$\begin{aligned}\dot{\boldsymbol{q}}^{(0)}_{n+1} &= \dot{\boldsymbol{q}}_n + h\left(1-\boldsymbol{\gamma}\right)\ddot{\boldsymbol{q}}_n + h\boldsymbol{\gamma}\ddot{\boldsymbol{q}}^{(0)}_{n+1}\\ \boldsymbol{q}^{(0)}_{n+1} &= \boldsymbol{q}_n + h\dot{\boldsymbol{q}}_n + \left(\frac{1}{2}-\beta\right)h^2\ddot{\boldsymbol{q}}_{n-1} + \beta h^2\ddot{\boldsymbol{q}}^{(0)}_{n+1}\end{aligned}$$

代入方程(16),得到关于 $\boldsymbol{x}$ 的非线性方程组

$$\mathbf{M}(\boldsymbol{q}_{n+1}, t_{n+1})\ddot{\boldsymbol{q}}_{n+1} = \boldsymbol{Q}(\boldsymbol{q}_{n+1}, \dot{\boldsymbol{q}}_{n+1}, t_{n+1}) \tag{20}$$

用Newton-Raphson方法迭代求解出修正量 $\boldsymbol{x}$,进而代入(19)中得到 $\boldsymbol{q}_{n+1}$、$\dot{\boldsymbol{q}}_{n+1}$ 和 $\ddot{\boldsymbol{q}}_{n+1}$.

传统Newmark方法的缺点在于,除非 $\gamma = 1/2$,否则算法的精度只有一阶,达到所需精度所需要的计算步数太多;而 $\gamma = 1/2$ 时数值稳定性又不太好,高频振荡的误差无法得到有效衰减,也会影响积分器的效率.为了提高计算的精确性和高效性,需要保证格式是二阶的而且是稳定的.引入一个自由参数 $\alpha \in [0, \frac{1}{3}]$,即可推导得到HHT积分器[16],该积分器的其他参数为

$$\gamma = \frac{1}{2} + \alpha, \qquad \beta = \frac{1}{4}\left(\frac{1}{2}+\gamma\right)^2 = \frac{1}{4}(1+\alpha)^2 \tag{21}$$

而 t_{n+1} 时刻的状态变量假设为

$$\begin{aligned}\dot{\boldsymbol{q}}_{n+1} &= \dot{\boldsymbol{q}}_n + h\left(1-\boldsymbol{\gamma}\right)\boldsymbol{a}_n + h\boldsymbol{\gamma}\boldsymbol{a}_{n-1}\\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + h\dot{\boldsymbol{q}}_n + \left(\frac{1}{2}-\beta\right)h^2\boldsymbol{a}_{n-1} + \beta h^2\boldsymbol{a}_{n-1}\end{aligned} \tag{22}$$

其中公式(17)中的加速度替代为稍早时刻的值

$$\boldsymbol{a}_n = \ddot{\boldsymbol{q}}_{n-\alpha}, \qquad \boldsymbol{a}_{n+1-\alpha} = \ddot{\boldsymbol{q}}_{n+1-\alpha} \tag{23}$$

而动力学方程组在较早的时间节点 $t_{n+1-\alpha}$ 上离散

$$\begin{aligned}&\mathbf{M}_{n+1-\alpha}\boldsymbol{a}_{n+1} + (1-\alpha)\boldsymbol{Q}_{n+1} + \alpha\boldsymbol{Q}_n = \mathbf{0}\\ &\frac{1}{\beta h^2}\boldsymbol{C}(\boldsymbol{q}_{n+1}, t_{n+1}) = \mathbf{0}\end{aligned} \tag{24}$$

其中未知量为 $\boldsymbol{a}_{n+1} = \ddot{\boldsymbol{q}}_{n+1-\alpha}$,而

$$\begin{aligned}\boldsymbol{Q}_{n+1} &= \boldsymbol{C}_q^T(\boldsymbol{q}_{n+1})\boldsymbol{\lambda}_{n+1} - \boldsymbol{Q}(\boldsymbol{q}_{n+1}, \dot{\boldsymbol{q}}_{n+1})\\ \boldsymbol{Q}_n &= \boldsymbol{C}_q^T(\boldsymbol{q}_n)\boldsymbol{\lambda}_n - \boldsymbol{Q}(\boldsymbol{q}_n, \dot{\boldsymbol{q}}_n)\end{aligned} \tag{25}$$

HHT积分器直接求解的并不是方程组(1),而是其变形格式(24),而且求解时间节点也不完全在 t_{n+1} 时刻,利用 $\boldsymbol{a}_{n+1}$ 近似地插值得到 t_{n+1} 时刻的加速度,就可以直接求解 t_{n+1} 时刻的动力学方程组(1),得到的即为广义 α 积分器.这两种积分器的公式稍有不同,广义α 积分器的参数稳定区域大于HHT积分器.除此之外,二者的各项性能都极其接近,误差估计与变步长策略也完全相同.本文将以广义 α 积分器作为主要讨论对象.

在广义 α 积分器中,每步需求解的未知变量仍然是公式(23)中的 $\boldsymbol{a}_{n+1}$,但此时假设

$$(1-\alpha_m)\boldsymbol{a}_{n+1} + \alpha_m\boldsymbol{a}_n = (1-\alpha_f)\ddot{\boldsymbol{q}}_{n+1} + \alpha_f\ddot{\boldsymbol{q}}_n \tag{26}$$

其中 α_m 和 α_f 两个参数一般并不相互独立,可以用一个自由参数 ρ 表示出来,取值 $\rho \in [0,1]$.令

$$\alpha_m = \frac{2\rho-1}{1+\rho}, \alpha_f = \frac{\rho}{1+\rho}, \eta = \frac{1-\alpha_m}{1-\alpha_f}, \gamma = \frac{1}{2} +$$

$$\alpha_m + \alpha_f \tag{27}$$

从方程(22)和(26)得出t_{n+1}时刻的状态变量

$$\begin{aligned}\ddot{\boldsymbol{q}}_{n+1} &= \frac{\alpha_m}{1-\alpha_f}\boldsymbol{a}_n - \frac{\alpha_f}{1-\alpha_f}\ddot{\boldsymbol{q}}_n + \frac{1-\alpha_m}{1-\alpha_f}\boldsymbol{a}_{n+1}\\ \dot{\boldsymbol{q}}_{n+1} &= \boldsymbol{q}_n + h(1-\gamma)\boldsymbol{a}_n + h\gamma\boldsymbol{a}_{n+1}\\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + h\dot{\boldsymbol{q}}_n + (\frac{1}{2}-\beta)h^2\boldsymbol{a}_n + \beta h^2\boldsymbol{a}_{n+1}\end{aligned} \tag{28}$$

取估计值$\boldsymbol{a}_{n+1}^{(0)} = \ddot{\boldsymbol{q}}_{n+1}$，更新量$\boldsymbol{x} = \boldsymbol{a}_{n+1} - \boldsymbol{a}_{n+1}^{(0)}$,则

$$\begin{aligned}\ddot{\boldsymbol{q}}_{n+1} &= \ddot{\boldsymbol{q}}_{n+1}^{(0)} + \eta\boldsymbol{x}\\ \dot{\boldsymbol{q}}_{n+1} &= \dot{\boldsymbol{q}}_{n+1}^{(0)} + h\gamma\boldsymbol{x}\\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_{n+1\,n}^{(0)} + \beta h^2\boldsymbol{x}\end{aligned} \tag{29}$$

其中

$$\begin{aligned}\ddot{\boldsymbol{q}}_{n+1}^{(0)} &= \frac{\alpha_m}{1-\alpha_f}\boldsymbol{a}_n + \frac{1-\alpha_m-\alpha_f}{1-\alpha_f}\ddot{\boldsymbol{q}}_n\\ \dot{\boldsymbol{q}}_{n+1}^{(0)} &= \dot{\boldsymbol{q}}_n + h(1-\gamma)\boldsymbol{a}_n + h\gamma\ddot{\boldsymbol{q}}_n\\ \boldsymbol{q}_{n+1}^{(0)} &= \boldsymbol{q}_n + h\dot{\boldsymbol{q}}_n + (\frac{1}{2}-\beta)h^2\boldsymbol{a}_n + \beta h^2\ddot{\boldsymbol{q}}_n\end{aligned} \tag{30}$$

此外,假设乘子$\boldsymbol{\lambda}$具有一定的连续性,即取

$$\boldsymbol{\lambda}_{n+1}^{(0)} = \boldsymbol{\lambda}_n, \qquad \boldsymbol{\lambda}_{n+1} = \boldsymbol{\lambda}_{n+1}^{(0)} + z \tag{31}$$

将表达式(29)和(31)代入方程(1),得到关于更新量$\boldsymbol{x}$和z的非线性方程组

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q}_{n+1})\ddot{\boldsymbol{q}}_{n+1} + \boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\boldsymbol{\lambda}_{n+1} = \boldsymbol{Q}\\ &\frac{1}{\beta h^2}\boldsymbol{C}(\boldsymbol{q}_{n+1}, t_{n+1}) = \mathbf{0}\end{aligned} \tag{32}$$

用 Newton-Raphson 方法求解,其 Jacobian 矩阵为

$$\mathbf{J}_n = \begin{pmatrix}\eta\mathbf{M} + \beta h^2\left(\sum_i \lambda_i C_{qq}^i - \boldsymbol{Q}_q\right) - h\gamma\boldsymbol{Q}_v & \boldsymbol{C}_q^{\mathrm{T}}\\ \boldsymbol{C}_q & \mathbf{0}\end{pmatrix} \tag{33}$$

更新量$\boldsymbol{x}$可以用来进行误差估计和步长选择.借鉴 Negrut 等[16]的推导,可以得出该方法的截断误差主项为$ch^2\boldsymbol{x}$;其中c为一个量级为 1 的常数.此外,由于广义 α 方法是二阶精度的,其广义坐标的截断误差正比于h^3;而步长的选择应该满足每个分量的相对误差都小于但接近于设定的误差限,即

$$\Xi = \sqrt[3]{h^2\|\boldsymbol{x}\|} \leqslant 1 \tag{34}$$

本文中的范数$\|\cdot\|$定义如下

$$\|\boldsymbol{x}\| = \max_k \frac{|x_k|}{W_q^k} \tag{35}$$

其中$W_q^k = RTOL\cdot|q_k| + ATOL_k$;其中$q_k$为第$k$个广义坐标对应的参考变量;$RTOL$为相对误差限,而$ATOL_k$为第$k$项的绝对误差限,一般有量纲.

如果$\Xi > 1$,则说明误差条件没有满足,需要减小步长;而如果Ξ太小,则说明计算步长太小,会降低计算效率,最好放大步长.一个合适的步长选择策略是

$$h_{new} = \frac{0.9}{\Xi}h \tag{36}$$

其中系数0.9是安全系数.由于这里的计算是用本步得出的后验误差来估计下一步的积分步长,其正确性依赖于系统参数的缓变特性,因此安全系数是必要的.如果预测失败,即$\Xi > 1$,则可以利用新得出的Ξ和方程(36)设置本步的计算步长重新进行计算.由于该估计误差为后验误差,得到的h_{new}一般能很好地满足误差条件.如果仍不满足误差要求,直接将时间步长减半重新计算;如果再不满足,说明在t_n和$t_n + h$之间出现了某种间断现象,需要进行间断检测,发现间断位置$t_n + h^*$,运行到该时刻然后重启积分器.广义 α 方法保持二阶精度,而且稳定性很好.以上的自适应步长选择策略失败率低,鲁棒性好,对很多间断情形也不需要积分器重启.由于间断性检测程序较复杂,在附录的算法 3 中将暂时忽略.

上述方法计算得出的加速度曲线和拉格朗日乘子曲线上经常会出现很多尖刺(Spikes),这种现象是非物理的随机误差,与所求解的方程组是index-3的DAEs有关.方程组(1)等价于

$$\begin{aligned}&\dot{\boldsymbol{q}} - \boldsymbol{v} = \mathbf{0}\\ &\mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}, t)\boldsymbol{\lambda} = \boldsymbol{Q}(\boldsymbol{q}, \boldsymbol{v}, t)\\ &\boldsymbol{C}(\boldsymbol{q}, t) = \mathbf{0}\end{aligned}$$

该方程需要在整个约束流形上满足,但数值离散格式不是在流形上开展的,而是在扩展的相空间$(\boldsymbol{q}, \boldsymbol{v})$上进行的,这将导致如下问题:

1 即使方程$\dot{\boldsymbol{q}} - \boldsymbol{v} = 0$在约束流形的切丛上是点点满足的,该方程的离散格式往往已经稍稍脱离了约束流形,而进入了相空间;

2. 尽管约束方程的引入强制保证了计算出的$\dot{\boldsymbol{q}}$近似位于约束流形上,但隐含的速度约束方程(3)却没有保证会被满足.

数值上来说,广义坐标上量级为ε的极小误差对应于速度变量上量级为ε/h的误差以及加速度变量上量级为ε/h^2的误差,而后者可能相当大,表现为尖刺现象.

将$\dot{\boldsymbol{q}}$和$\boldsymbol{v}$作为独立变量;根据流形上的常微分方程理论,可以将方程$\dot{\boldsymbol{q}} - \boldsymbol{v} = 0$换为约束流形上的等价格式;并将隐含的速度约束显式地引入方程

组,得到新的动力学方程组

$$\begin{aligned}&\dot{\boldsymbol{q}}-\boldsymbol{v}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\mu}=\boldsymbol{0}\\&\mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{v}}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}=\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{v},t)\\&\boldsymbol{C}(\boldsymbol{q},t)=\boldsymbol{0}\\&\boldsymbol{C}_q(\boldsymbol{q},t)\boldsymbol{v}+\boldsymbol{C}_t(\boldsymbol{q},t)=\boldsymbol{0}\end{aligned}\tag{37}$$

其未知量为 $\boldsymbol{q}$、$\boldsymbol{v}$、$\boldsymbol{\lambda}$ 和 $\boldsymbol{\mu}$. 该方程组是 index-2 的微分代数方程组,其与方程组(1)的等价性早已经被 Gear 等人[12]所证明. 下面用广义 α 方法来离散方程组(37). 此时,方程(26)和(28)中的速度项仍然被离散为

$$\begin{aligned}&(1-\alpha_m)\mathbf{a}_{n+1}+\alpha_m\mathbf{a}_n=(1-\alpha_f)\dot{\boldsymbol{v}}_{n+1}+\alpha_f\dot{\boldsymbol{v}}_n\\&\boldsymbol{v}_{n+1}=\boldsymbol{v}_n+h(1-\gamma)\mathbf{a}_n+h\gamma\mathbf{a}_{n+1}\end{aligned}$$

此时速度项 $\boldsymbol{v}_{n+1}$ 与 $\dot{\boldsymbol{q}}_{n+1}$ 不再相同,但二者差别不应该很大,不妨假设

$$\begin{aligned}&\dot{\boldsymbol{q}}_{n+1}=\boldsymbol{v}_n+h(1-\boldsymbol{\gamma})\mathbf{a}_n+h\boldsymbol{\gamma}\boldsymbol{a}_{n+1}\\&\boldsymbol{q}_{n+1}=\boldsymbol{q}_n+h\boldsymbol{v}_n+(\frac{1}{2}-\beta)h^2\mathbf{a}_n+\beta h^2\boldsymbol{a}_{n+1}\end{aligned}$$

计算中,取 $\boldsymbol{a}_{n+1}$ 与 $\mathbf{a}_{n+1}$ 其预测值相同,且假设

$$\mathbf{a}_{n+1}=\mathbf{a}_{n+1}^{(0)}+\boldsymbol{y},\qquad h\boldsymbol{a}_{n+1}=h\boldsymbol{a}_{n+1}^{(0)}+\boldsymbol{x}$$

进而类似方程(29)可以得出,

$$\begin{aligned}&\dot{\boldsymbol{v}}_{n+1}=\ddot{\boldsymbol{q}}_{n+1}^{(0)}+\eta\boldsymbol{y},\qquad \boldsymbol{v}_{n+1}=\dot{\boldsymbol{q}}_{n+1}^{(0)}+h\gamma\boldsymbol{y}\\&\dot{\boldsymbol{q}}_{n+1}=\dot{\boldsymbol{q}}_{n+1}^{(0)}+\gamma\boldsymbol{x},\qquad \boldsymbol{q}_{n+1}=\boldsymbol{q}_{n+1}^{(0)}+\beta h\boldsymbol{x}\end{aligned}$$

代入方程(37)得到

$$\begin{aligned}&\gamma\boldsymbol{x}-h\gamma\boldsymbol{y}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\boldsymbol{\mu}=\boldsymbol{0}\\&\mathbf{M}(\boldsymbol{q}_{n+1})\dot{\boldsymbol{v}}_{n+1}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\boldsymbol{\lambda}_{n+1}=\boldsymbol{Q}\\&\frac{1}{\beta h^2}\boldsymbol{C}(\boldsymbol{q}_{n+1},t_{n+1})=\boldsymbol{0}\\&\frac{1}{\boldsymbol{\gamma}h}\left\{\boldsymbol{C}_q(\boldsymbol{q}_{n+1})\boldsymbol{v}_{n+1}+\boldsymbol{C}_t(\boldsymbol{q}_{n+1})\right\}=\boldsymbol{0}\end{aligned}\tag{38}$$

其未知量为 $\boldsymbol{x}$、$\boldsymbol{y}$、$\boldsymbol{\lambda}$ 和 $\boldsymbol{\mu}$,迭代 Jacobian 矩阵为

$$\mathbf{J}=\begin{pmatrix}\boldsymbol{\gamma}\mathbf{I}-\beta h\sum_i\mu_iC_{qq}^i & -\boldsymbol{\gamma}h\mathbf{I} & \boldsymbol{0} & \boldsymbol{C}_q^{\mathrm{T}}\\ \eta\mathbf{M}-\boldsymbol{\gamma}h\boldsymbol{Q}_v & \beta h\mathbf{K} & \boldsymbol{C}_q^{\mathrm{T}} & \boldsymbol{0}\\ \boldsymbol{C}_q & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \dfrac{\beta}{\gamma}\left[(\boldsymbol{C}_q\boldsymbol{v})_q+\boldsymbol{C}_{tq}\right] & \boldsymbol{C}_q & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\tag{39}$$

其中

$$\mathbf{K}=(\mathbf{M}\dot{\boldsymbol{v}})_q+\sum_i\lambda_iC_{qq}^i-\boldsymbol{Q}_q$$

由方程(38)可以看出,$\boldsymbol{x}$ 是与 $\boldsymbol{\mu}$ 同量级的量. 用截断误差分析可以证明[47],该积分格式是一阶的. 该方法关于 $\boldsymbol{q}$ 的截断误差近似为 $h\boldsymbol{x}$,而关于 $\boldsymbol{v}$ 的截断误差近似为 $h\boldsymbol{y}$. 同样记

$$\Xi\equiv\sqrt{h\max\left(\|\boldsymbol{x}\|,\|\boldsymbol{y}\|\right)}$$

则 $\Xi\leqslant 1$ 为误差判定条件;而新步长估计方法仍然基于公式(36). Jay 和 Negrut 的分析表明[47],如果采用等时间步长积分,求解 index-2 方程组(37)的格式(38)是二阶精度的;但如果采用变时间步长积分,则离散格式(38)只有一阶精度. 另外,他们还提出一种修正方法,可以将对应的 HHT 格式变为二阶精度的,而且该方法似乎也很容易推广到广义 α 方法. 但作者还没有亲自测试证实,尚不知该方法的实践效果如何.

4 BDF 族积分器

BDF 族积分器有好几种公式实现[8]. 它一般用来求解常微分方程组和 index-1、index-2 的微分代数方程组[48],且结合一些特别的技巧后也能用来计算 index-3 的微分代数方程组(1). 与广义 α 方法不同,BDF 方法是多步法,而且是一类变阶数、变步长的方法. 多步法的一般提法为:

问题 2 已知之前 $k+1$ 个相继时刻 t_{n-i} $(i=0,1,\cdots,k)$ 对应的广义坐标 $\boldsymbol{q}_{n-i}$、速度 $\dot{\boldsymbol{q}}_{n-i}$、加速度 $\ddot{\boldsymbol{q}}_{n-i}$ 以及拉格朗日乘子 $\boldsymbol{\lambda}_{n-i}$,在给出时间步长 h 之后,求出 $t_{n+1}=t_n+h$ 时刻对应的变量 $\boldsymbol{q}_{n+1}$、$\dot{\boldsymbol{q}}_{n+1}$、$\ddot{\boldsymbol{q}}_{n+1}$ 以及 $\boldsymbol{\lambda}_{n+1}$,并给出计算误差的估计,判断计算结果是否满足误差条件. 如果满足,则估计下一步的阶数和时间步长;如果不满足,则更新本步阶数和时间步长,并重新进行计算.

下面以常微分方程为例,介绍两套常用的 BDF 公式. 常微分方程组的形式为:

$$\mathbf{A}(\boldsymbol{x},t)\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x},t)\tag{40}$$

准步长公式

假设给定一系列等步长为 h 的时间节点 $t_{n-k},t_{n-k+1},\cdots,t_n,t_{n+1}$,以及其对应的数据 $\boldsymbol{x}_{n-k},\boldsymbol{x}_{n-k+1},\cdots,\boldsymbol{x}_n,\boldsymbol{x}_{n+1}$. 根据在以 h 为步长的均匀时间网格上,后差分算子 ∇ 如下递归定义:

$$\begin{aligned}&\nabla^0\boldsymbol{x}_{n+1}=\boldsymbol{x}_{n+1}\\&\nabla^{j+1}\boldsymbol{x}_{n+1}=\nabla^j\boldsymbol{x}_{n+1}-\nabla^j\boldsymbol{x}_n\end{aligned}\tag{41}$$

其中 $j=0,1,2,\cdots$. 用中值定理可以证明,存在某个 ξ 满足 $t_{n+1}-kh\leqslant\xi\leqslant t_{n+1}$,使得

$$\nabla^k\boldsymbol{x}_{n+1}=h^k\frac{\boldsymbol{d}^k\boldsymbol{x}}{\boldsymbol{d}t^k}(\xi)\approx h^k\boldsymbol{x}_{n+1}^{(k)}\tag{42}$$

给定 $k\geqslant 1$,记

$$\boldsymbol{\delta}=\nabla^{k+1}\boldsymbol{x}_{n+1}\tag{43}$$

递归地使用(41),可得 $\nabla^k\boldsymbol{x}_{n+1}=\nabla^k\boldsymbol{x}_n+\boldsymbol{\delta}$,$\nabla^{k-1}\boldsymbol{x}_{n+1}=\nabla^{k-1}\boldsymbol{x}_n+\nabla^k\boldsymbol{x}_n+\boldsymbol{\delta},\cdots$,其通项为

$$\nabla^j\boldsymbol{x}_{n+1}=\sum_{i=j}^k\nabla^i\boldsymbol{x}_n+\boldsymbol{\delta},\qquad j=0,1,...,k\tag{44}$$

特别当 $j=0$ 时有

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_{n+1}^{(0)} + \boldsymbol{\delta} \tag{45}$$

其中

$$\boldsymbol{x}_{n+1}^{(0)} = \boldsymbol{x}_n + \sum_{j=1}^{k} \nabla^j \boldsymbol{x}_n \tag{46}$$

另外,可以直接验证[49],多项式

$$\boldsymbol{p}_h(t) = \boldsymbol{x}_n + \sum_{j=1}^{k} \frac{1}{j!\ h!} \nabla^k \boldsymbol{x}_n \prod_{a=0}^{j-1} (t - t_{n-a}) \tag{47}$$

为唯一满足 $\boldsymbol{p}_h\left(t_{n-j}\right) = x_{n-j}$ 的 k 阶多项式,其中j= 0, 1, 2,…, k;对此方程求微分可得

$$h\dot{\boldsymbol{x}}_n \approx h\dot{\boldsymbol{p}}_h(t_n) = \sum_{j=1}^{k} \frac{1}{j} \nabla^j \boldsymbol{x}_n \tag{48}$$

将近似表达式(48)中的n替换为$n+1$,得到

$$h\dot{\boldsymbol{x}}_{n+1} \approx \sum_{j=1}^{k} \frac{1}{j} \nabla^j \boldsymbol{x}_{n+1}$$

其截断误差主项为$1/(k+1)\nabla^{k+1}\boldsymbol{x}_{n+1}$.利用公式(43)

$$h\dot{\boldsymbol{x}}_{n+1} \approx h\dot{\boldsymbol{x}}_{n+1}^{(0)} + c_k \boldsymbol{\delta} \tag{49}$$

其截断误差主项为$c_e^k\boldsymbol{\delta}$;其中

$$h\dot{\boldsymbol{x}}_{n+1}^{(0)} = \sum_{j=1}^{k} c_j \nabla^j \boldsymbol{x}_n \tag{50}$$

其中的常数为

$$c_j = \sum_{i=1}^{j} \frac{1}{i}, \qquad c_e^k = \frac{1}{k+1}$$

为了与后文符号的统一,记

$$\boldsymbol{c}_k = (c_1, c_2, ..., c_k)^T, \qquad \boldsymbol{\omega}_k = (1, 1, ..., 1)^T \tag{51}$$

以及

$$\mathbf{D}_n^k = \left(\nabla \boldsymbol{x}_n, \nabla^2 \boldsymbol{x}_n, ..., \nabla^k \boldsymbol{x}_n\right) \tag{52}$$

则方程(46)和(50)可以改为

$$\boldsymbol{x}_{n+1}^{(0)} = \boldsymbol{x}_n + \mathbf{D}_n^k \boldsymbol{\omega}_k, \qquad h\dot{\boldsymbol{x}}_{n+1}^{(0)} = \mathbf{D}_n^k \boldsymbol{c}_k \tag{53}$$

最后,应该指出的是,差分矩阵$\mathbf{D}_n^k$是基于时步长为h的均匀网格得到的.当步长h变化为新的步长$\overline{h}$时,需要求出新步长下的差分矩阵$\overline{\mathbf{D}}_n^k$.这一变换可以用一个简单的计算[48]来实现:

$$\overline{\mathbf{D}}_n^k = \mathbf{D}_n^k(\mathbf{R}^k\mathbf{U}^k) \tag{54}$$

其中矩阵$\mathbf{R}^k$和$\mathbf{U}^k$为$k \times k$的矩阵,其(i, j)分量分别为

$$R_{ij}^k = \frac{1}{i!} \prod_{a=0}^{i-1} (a - j\zeta), \qquad U_{ij}^k = \frac{1}{i!} \prod_{a=0}^{i-1} (a - j) \tag{55}$$

其中$\zeta = \overline{h}/h$为步长缩放比.矩阵$\mathbf{R}^k$和$\mathbf{U}^k$的规模都很小,且很好计算,因此变步长操作很简便.公式(54)中的计算原理如下[49]:两种步长下得到的k阶插值多项式应该完全等价,即

$$\boldsymbol{p}_h(t) = \boldsymbol{p}_{\overline{h}}(t) \tag{56}$$

参考公式(47),分别将$t = t_n - i\zeta h$(其中$i = 0, 1, \cdots, k$)代入方程(56)得到k个线性方程:

$$\sum_{j=1}^{k} \nabla^k \boldsymbol{x}_n \frac{1}{j!} \prod_{a=0}^{j-1} (a - i\zeta) = \sum_{j=1}^{k} \overline{\nabla}^k \boldsymbol{x}_n \frac{1}{j!} \prod_{a=0}^{j-1} (a - i)$$

写成矩阵形式,即为 $\mathbf{D}_n^k\mathbf{R}^k=\overline{\mathbf{D}}_n^k\mathbf{U}^k$.进而利用如下恒等式①:$(\mathbf{U}^k)^2 = \mathbf{I}$可以得出(54).

计算流程:准定步长积分器的步进循环计算为:已知$\mathbf{D}_n^{k+1}$,以及估计得到的新步长h.

(1) 利用公式(53)得出 $\boldsymbol{x}_{n+1}$ 和 $h\dot{\boldsymbol{x}}_{n+1}$ 的初始估计,然后将(45)和(49)代入微分方程组得到离散格式,用Newton迭代求出更新量$\boldsymbol{\delta}$;

(2) 判断后验误差 $c_e^k\boldsymbol{\delta}$ 是否满足误差要求.如果不满足,重新估计阶数k和步长h,并利用公式(54)更新 $\mathbf{D}_n^{k+1}$,回到1;否则,进入3;

(3) 用递推公式(41)计算差分矩阵$\mathbf{D}_{n+1}^{k+2}$如下:

a) $\nabla^{k+1}\boldsymbol{x}_{n+1} = \boldsymbol{\delta}$ 已经求得;

b) $\nabla^{k+2}\boldsymbol{x}_{n+1} = \boldsymbol{\delta} - \nabla^{k+1}\boldsymbol{x}_n$;

c) 依次计算 $\nabla^j\boldsymbol{x}_{n+1} = \nabla^j\boldsymbol{x}_n + \nabla^{j+1}\boldsymbol{x}_{n+1}$,其中$j$= $k, k-1, \cdots, 1$;

(4) 矩阵 $\mathbf{D}_{n+1}^{k+2}$ 中包含了所有$1 \leqslant j \leqslant k+1$阶的后验误差$c_e^j\nabla^{j+1}\boldsymbol{x}_{n+1}$;依次选择下一步的阶数$\overline{k}$和步长$\overline{h}$,并利用公式(54)计算$\overline{\mathbf{D}}_{n+1}^{\overline{k}+1}$;

(5) 更新 $k \leftarrow \overline{k}$、$h \leftarrow \overline{h}$、$\mathbf{D}_{n+1}^{k+1} \leftarrow \overline{\mathbf{D}}_{n+1}^{\overline{k}+1}$ 以及 $n \leftarrow n+1$,回到起始步1开始下一循环.

代入初值$\boldsymbol{x}_0$依次计算出来.在以上计算过程中,只需要自适应地更新矩阵$\mathbf{D}_n^{k+1}$,而无需实际生成对应定步长网格.因此这一算法称为准定步长公式.

变步长公式

在这套公式下,均匀网格是不需要的,但理论推导要用到Newton插值公式:$[\boldsymbol{x}_n] = \boldsymbol{x}_n$

$$\left[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_{n-j}\right] = \frac{\left[\boldsymbol{x}_n, ..., \boldsymbol{x}_{n-j+1}\right] - \left[\boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_{n-j}\right]}{t_n - t_{n-j}} \tag{57}$$

其中j= 1, 2, 3,….可以用数学归纳法和中值定理证明,存在某个ξ,满足$t_{n-k} \leqslant \xi \leqslant t_n$

$$\left[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_{n-k}\right] = \frac{1}{k!} \frac{d^k \boldsymbol{x}}{dt^k}(\boldsymbol{\xi})$$

为了符号方便,对j= 1,2,…,k+1,记

①直接用公式(55)中的表达式计算可证:$\Sigma_l U_{il}U_{lj} = \delta_{il}$

$$\tau_j^n = t_n - t_{n-j}, \qquad \omega_j^n = \prod_{i=1}^{j} \frac{\tau_i^{n+1}}{\tau_i^n} \tag{58}$$

并且记(注意它与公式(41)中的定义没有关系)

$$\nabla^j \boldsymbol{x}_n = \left(\prod_{i=1}^{j} \tau_i^n\right)\left[\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_{n-j}\right]$$

用中值定理可以证明此时近似公式(42)仍然成立.公式(57)两边同时乘以 $\Pi_{i=1}^{j}\tau_i^n$,可以得出递推公式 $\nabla^j \boldsymbol{x}_n = \omega_j^{n-1}\nabla^j \boldsymbol{x}_{n-1} + \nabla^{j+1}\boldsymbol{x}_n$. 为了应用的方便,进行替换 $n \leftarrow \overline{n}$,得到

$$\nabla^j \boldsymbol{x}_{n+1} = \omega_j^n \nabla^j \boldsymbol{x}_n + \nabla^{j+1}\boldsymbol{x}_{n+1} \tag{59}$$

利用这个公式递归可得

$$\nabla^j \boldsymbol{x}_{n+1} = \sum_{i=j}^{k} \omega_i^n \nabla^i \boldsymbol{x}_n + \nabla^{k+1}\boldsymbol{x}_{n+1} \tag{60}$$

以及

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \sum_{j=1}^{k} \omega_j^n \nabla^j \boldsymbol{x}_n + \nabla^{k+1}\boldsymbol{x}_{n+1} \tag{61}$$

可以证明①,插值公式

$$\boldsymbol{p}(t) = \sum_{j=0}^{k}\left(\prod_{i=0}^{j-1}(t - t_{n+1-i})\right)\left[\boldsymbol{x}_{n+1}, \boldsymbol{x}_n, ..., \boldsymbol{x}_{n-j+1}\right]$$

是唯一满足 $\boldsymbol{p}\left(t_{n+1-m}\right) = \boldsymbol{x}_{n+1-m}$ 的 k 阶多项式,其中.直接微分上式可得

$$\dot{\boldsymbol{p}}(t_{n+1}) = \sum_{j=1}^{k} \frac{1}{\tau_j^{n+1}} \nabla^j \boldsymbol{x}_{n+1} \tag{62}$$

将公式(60)代入方程(62)中,简化得到

$$h\dot{\boldsymbol{x}}_{n+1} \approx h\dot{\boldsymbol{p}}(t_{n+1}) = \sum_{j=1}^{k} c_j \omega_j^n \nabla^j \boldsymbol{x}_n + c_k \nabla^{k+1}\boldsymbol{x}_{n+1} \tag{63}$$

且该表达式的截断误差主项为 $c_e^k \nabla^{k+1}\boldsymbol{x}_{n+1}$,其中

$$c_e^k = \frac{h}{\tau_{k+1}^{n+1}}, \qquad c_j = \sum_{i=1}^{j} \frac{h}{\tau_i^{n+1}} \tag{64}$$

为了符号统一,同样记 $\boldsymbol{\delta}$ 和 $\mathbf{D}_n^k$ 如公式(43)和(52),且定义由历史计算时间节点组成的向量

$$\boldsymbol{\tau}_{k+1}^n = \left(\tau_1^n, \tau_2^n, ..., \tau_{k+1}^n\right)^T \tag{65}$$

给定步长 h,可计算出下一时刻的 $\boldsymbol{\tau}_{k+2}^{n+1}$,其分量为

$$\tau_1^{n+1} = h, \qquad \tau_{j+1}^{n+1} = h + \tau_j^n \tag{66}$$

其中 $j = 1, 2, \cdots, k+1$. 进一步,记

$$\boldsymbol{\omega}_k = (\omega_1^n, \omega_2^n, ..., \omega_k^n)^T$$
$$\boldsymbol{c}_k = \tag{67}$$

于是公式(61)和(63)可以写成

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_{n+1}^{(0)} + \boldsymbol{\delta}, \qquad h\dot{\boldsymbol{x}}_{n+1} \approx h\dot{\boldsymbol{x}}_{n+1}^{(0)} + c_k\boldsymbol{\delta} \tag{68}$$

而后者的截断误差主项为 $c_e^k\boldsymbol{\delta}$,其中

$$\boldsymbol{x}_{n+1}^{(0)} = \boldsymbol{x}_n + \mathbf{D}_n^k \boldsymbol{\omega}_k, \qquad h\dot{\boldsymbol{x}}_{n+1}^{(0)} = \mathbf{D}_n^k \boldsymbol{c}_k \tag{69}$$

计算结束后,如果误差符合要求,即 $\nabla^{k+1}\boldsymbol{x}_{n+1} = \boldsymbol{\delta}$ 被接受,就可以进一步用公式(59)依次更新 $\nabla^k \boldsymbol{x}_{n+1}$、$\nabla^{k-1}\boldsymbol{x}_{n+1}$、$\nabla \boldsymbol{x}_{n+1}$. 此外为了对 $k+1$ 阶公式进行误差估计,还需要估计出 $\nabla^{k+2}\boldsymbol{x}_{n+1} = \boldsymbol{\delta} - \omega_{k+1}^n \nabla^{k+1}\boldsymbol{x}_n$,这些量组合成差分矩阵 $\mathbf{D}_{n+1}^{k+2}$.

计算流程:变步长积分器的步进计算为:

已知 $\mathbf{D}_n^{k+1}$,时间向量 $\boldsymbol{\tau}_{k+1}^n$ 以及新步长 h

(1) 利用公式(66)计算新的时间向量 $\boldsymbol{\tau}_{k+2}^{n+1}$,进而用公式(58)和(64)算出 $\boldsymbol{\omega}_k$、$\boldsymbol{c}_k$ 和 c_e^k;

(2) 利用公式(69)得出 $\boldsymbol{x}_{n+1}$ 和 $h\dot{\boldsymbol{x}}_{n+1}$ 的初始估计,然后将(68)代入微分方程组得到其离散格式,再用 Newton 迭代求出更新量 $\boldsymbol{\delta}$;

(3) 判断后验误差 $c_e^k\boldsymbol{\delta}$ 是否满足误差要求. 如果不满足,重新估计阶数 k 和步长 h,回到 1 重新开始本步计算;如满足,进入下一步 4;

(4) 更新矩阵 $\mathbf{D}_{n+1}^{k+2}$,并估计从 1 阶到 k+1 阶的后验误差,进而选择下一步的阶数和步长;

(5) 令 $n \leftarrow n+1$,进入下一步循环的起始步 1.

初始条件 $\mathbf{D}_0^1 = h\dot{\boldsymbol{x}}_0$. 第一步采用定阶定步长;而从第二步才开始正式进入以上计算循环.

方程组求解细节

将表达式(45)和(49)或者(68)代入方程(40)得到关于 $\boldsymbol{\delta}$ 的非线性方程组

$$\mathbf{A}(\boldsymbol{x}_{n+1}^{(0)} + \boldsymbol{\delta})\left[\hat{h}\dot{\boldsymbol{x}}_{n+1}^{(0)} + \boldsymbol{\delta}\right] - \hat{h}\boldsymbol{f}\left(\boldsymbol{x}_{n+1}^{(0)} + \boldsymbol{\delta}\right) = \mathbf{0} \tag{70}$$

用拟 Newton 方法求解,并且计算过程可以用近似

Jacobian 矩阵 $\mathbf{A} - h\hat{}\boldsymbol{f}_x$ 来进行迭代;其中

$$\hat{h} = h/c_k \tag{71}$$

求解出更新量 $\boldsymbol{\delta} = \nabla^{k+1}\boldsymbol{x}_{n+1}$ 之后,就直接得出 k 阶截断误差主项为 $c_e^k\nabla^{k+1}\boldsymbol{x}_{n+1}$;同时可以更新 $\mathbf{D}_{n+1}^{k+2}$ 矩阵,得到 $j = 1, 2, \cdots, k+1$ 等各阶的后验误差估计 $c_e^j\nabla^{j+1}\boldsymbol{x}_{n+1}$,并基于这些信息选择下一步积分的阶数与时间步长.

求解方程组(1)用到 $\boldsymbol{q}$ 和 $\boldsymbol{v} \triangleq \dot{\boldsymbol{q}}$ 的差分矩阵

$$\boldsymbol{Q}_n^k = \left(\nabla \boldsymbol{q}_n, ..., \nabla^k \boldsymbol{q}_n\right), \qquad \boldsymbol{V}_n^k = \left(\nabla \boldsymbol{v}_n, ..., \nabla^k \boldsymbol{v}_n\right) \tag{72}$$

①证明用到 $\prod_{i=0}^{j-1}\tau_{m-i}^{n-i} + \tau_j^{n+1}\prod_{i=0}^{j-2}\tau_{m-i}^{n-i} = \prod_0^{j-1}\tau_{m+1-i}^{n+1-i}$,然后从 t_{n+1} 往下用归纳法证明

从而这两个差分矩阵可以算出变量的估计表达式

$$\begin{aligned}\boldsymbol{q}_{n+1}^{(0)}&=\boldsymbol{q}_n+\boldsymbol{Q}_n^k\boldsymbol{\omega}_k,\qquad \dot{\boldsymbol{q}}_{n+1}^{(0)}=\frac{1}{h}\boldsymbol{Q}_n^k\boldsymbol{c}_k\\ \boldsymbol{v}_{n+1}^{(0)}&=\boldsymbol{v}_n+\boldsymbol{V}_n^k\boldsymbol{\omega}_k,\qquad \dot{\boldsymbol{v}}_{n+1}^{(0)}=\frac{1}{h}\boldsymbol{V}_n^k\boldsymbol{c}_k\end{aligned}\tag{73}$$

因此,广义坐标的BDF预测-校正公式为

$$\boldsymbol{q}_{n+1}=\boldsymbol{q}_{n+1}^{(0)}+\boldsymbol{\delta}\tag{74}$$

而对应速度变量的BDF预测-校正公式为

$$\dot{\boldsymbol{q}}_{n+1}\approx\dot{\boldsymbol{q}}_{n+1}^{(0)}+\frac{1}{\hat{h}}\boldsymbol{\delta},\qquad \boldsymbol{v}_{n+1}=\boldsymbol{v}_{n+1}^{(0)}+\boldsymbol{\varepsilon}\tag{75}$$

为了求解方程组(1),可以将$\dot{\boldsymbol{q}}=\boldsymbol{v}$的离散格式$\dot{\boldsymbol{q}}_{n+1}^{(0)}+\boldsymbol{\delta}/\hat{h}\approx\boldsymbol{v}_{n+1}^{(0)}+\boldsymbol{\varepsilon}$化简,得到

$$\boldsymbol{\varepsilon}\approx\dot{\boldsymbol{q}}_{n+1}^{(0)}-\boldsymbol{v}_{n+1}^{(0)}+\frac{1}{\hat{h}}\boldsymbol{\delta}\tag{76}$$

于是加速度$\ddot{\boldsymbol{q}}_{n+1}^{(0)}=\dot{\boldsymbol{v}}_{n+1}\approx\dot{\boldsymbol{v}}_{n+1}^{(0)}+\boldsymbol{\varepsilon}/\hat{h}$可表达为

$$\ddot{\boldsymbol{q}}_{n+1}\approx\dot{\boldsymbol{v}}_{n+1}^{(0)}+\frac{1}{\hat{h}}(\dot{\boldsymbol{q}}_{n+1}^{(0)}-\boldsymbol{v}_{n+1}^{(0)})+\frac{1}{\hat{h}^2}\boldsymbol{\delta}\tag{77}$$

将公式(74)、(75)和(77)代入方程组(1),得到关于$\boldsymbol{\delta}$和$\hat{\boldsymbol{\lambda}}=\hat{h}^2\boldsymbol{\lambda}$的离散格式的非线性方程组

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q}_{n+1})\ddot{\boldsymbol{q}}_{n+1}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\boldsymbol{\lambda}_{n+1}=\boldsymbol{Q}\\ &\frac{1}{\hat{h}^2}\boldsymbol{C}(\boldsymbol{q}_{n+1},t_{n+1})=\mathbf{0}\end{aligned}\tag{78}$$

可以用Newton迭代法求解,其Jacobian矩阵为

$$\mathbf{J}=\frac{1}{\hat{h}^2}\begin{pmatrix}\hat{\mathbf{M}}&\boldsymbol{C}_q^{\mathrm{T}}\\ \boldsymbol{C}_q&\mathbf{0}\end{pmatrix}\tag{79}$$

其中

$$\hat{\mathbf{M}}=\mathbf{M}-\hat{h}\boldsymbol{Q}_v+\hat{h}^2\left(\sum_i\lambda_i\boldsymbol{C}_{qq}^i-\boldsymbol{Q}_q\right)$$

求解得到的$\boldsymbol{\delta}$可以直接用来估计$\boldsymbol{q}$的误差,但由方程(76)得出的$\boldsymbol{\varepsilon}$用来估计$\dot{\boldsymbol{q}}$的误差却可能会被严重放大[10],需特殊处理后才能应用.这里介绍一种更新误差估计中的$\boldsymbol{\varepsilon}$的方法,即投影滤波法.深入分析[11]发现,方程(76)计算出来的速度变量的误差并不是沿所有方向都会被放大;而误差真正被放大的方向与约束流形的切空间正交.如果将$\boldsymbol{\varepsilon}$投影到约束流形切空间内,即

$$\hat{\boldsymbol{\varepsilon}}=\boldsymbol{\varepsilon}-\mathbf{P}_\perp\boldsymbol{\varepsilon}$$

其中

$$\mathbf{P}_\perp=\hat{\mathbf{M}}^{-1}\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{C}_q\hat{\mathbf{M}}^{-1}\boldsymbol{C}_q^{\mathrm{T}})^{-1}\boldsymbol{C}_q$$

可以直接验证,$\boldsymbol{C}_q\hat{\boldsymbol{\varepsilon}}=0$,即$\hat{\boldsymbol{\varepsilon}}$为$\boldsymbol{\varepsilon}$在约束流形切平面的投影.投影滤波法即用这部分分量$\hat{\boldsymbol{\varepsilon}}$来进行误差估计.虽然投影得到的误差偏小,但其在数量级上是正确的,可以用于步长选择.但要注意的是,$\hat{\boldsymbol{\varepsilon}}$的应用仅限于误差估计,而非数值计算.容易验证,通过求解

$$\mathbf{J}\begin{pmatrix}\hat{\boldsymbol{\varepsilon}}\\ \boldsymbol{\mu}\end{pmatrix}=\frac{1}{\hat{h}^2}\begin{pmatrix}\hat{\mathbf{M}}\boldsymbol{\varepsilon}\\ \mathbf{0}\end{pmatrix}\tag{80}$$

即可得出$\hat{\boldsymbol{\varepsilon}}$,其中$\mathbf{J}$即为公式(79)中的Jacobian矩阵,其LU分解已经在前面计算出来,因此投影滤波过程并不会额外增加多少计算量.

计算直接得到量$\nabla^{k+1}\boldsymbol{x}_{n+1}$(即$\boldsymbol{\delta}$)和$\nabla^{k+1}\boldsymbol{v}_{n+1}$(即$\boldsymbol{\varepsilon}$),进而利用$\boldsymbol{Q}_n^{k+1}$和$\boldsymbol{V}_n^{k+1}$由公式(41)或者(59)递归得出$\boldsymbol{Q}_{n+1}^{k+2}$和$\boldsymbol{V}_{n+1}^{k+2}$,供下一步计算使用.与广义α方法一样,求解得出的$\boldsymbol{q}$是准确的;但$\boldsymbol{v}$的精度稍差,可能会出现一些不连续的情形;而加速度$\dot{\boldsymbol{v}}$和拉格朗日乘子$\boldsymbol{\lambda}$的误差可能相当大,经常会出现一些强烈的不连续现象,表现为一些非物理的尖刺.但这种方法的求解方程规模小,计算速度快,是实际仿真中最常用的办法.

为了改进速度变量的误差估计,并减小加速度和拉氏乘子的计算误差,可以将方程组(1)转化为与其等价的index-2的方程组[12]

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{v}}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}=\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{v},t)\\ &\dot{\boldsymbol{q}}-\boldsymbol{v}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\mu}=\mathbf{0}\\ &\boldsymbol{C}(\boldsymbol{q},t)=\mathbf{0}\\ &\boldsymbol{C}_q(\boldsymbol{q},t)\boldsymbol{v}+\boldsymbol{C}_t(\boldsymbol{q},t)=\mathbf{0}\end{aligned}\tag{81}$$

然后用BDF积分器进行求解.该方程组的规模比直接求解方程组(1)大了一倍,所需的计算量也大了很多.具体计算步骤为:将变量的离散格式

$$\begin{aligned}&\boldsymbol{q}_{n+1}=\boldsymbol{q}_{n+1}^{(0)}+\boldsymbol{\delta},\qquad \boldsymbol{v}_{n+1}=\boldsymbol{v}_{n+1}^{(0)}+\boldsymbol{\varepsilon}\\ &\hat{h}\boldsymbol{\lambda}_{n+1}=\hat{h}\boldsymbol{\lambda}_{n+1}^{(0)}+\hat{\boldsymbol{\lambda}},\qquad \hat{h}\boldsymbol{\mu}_{n+1}=\hat{\boldsymbol{\mu}}\end{aligned}\tag{82}$$

以及时间导数项的离散格式

$$\hat{h}\dot{\boldsymbol{q}}_{n+1}\approx\hat{h}\dot{\boldsymbol{q}}_{n+1}^{(0)}+\boldsymbol{\delta},\qquad \hat{h}\dot{\boldsymbol{v}}_{n+1}\approx\hat{h}\dot{\boldsymbol{v}}_{n+1}^{(0)}+\boldsymbol{\varepsilon}\tag{83}$$

代入方程组(81),得到离散代数方程组为

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q}_{n+1})\hat{h}\dot{\boldsymbol{v}}_{n+1}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\hat{h}\boldsymbol{\lambda}_{n+1}=\hat{h}\boldsymbol{Q}(\boldsymbol{q}_{n+1},\boldsymbol{v}_{n+1})\\ &\hat{h}\dot{\boldsymbol{q}}_{n+1}-\hat{h}\boldsymbol{v}_{n+1}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\hat{\boldsymbol{\mu}}_{n+1}=\mathbf{0}\\ &\boldsymbol{C}(\boldsymbol{q}_{n+1},t_{n+1})=\mathbf{0}\\ &\boldsymbol{C}_q(\boldsymbol{q}_{n+1},t_{n+1})\boldsymbol{v}_{n+1}+\boldsymbol{C}_t(\boldsymbol{q}_{n+1},t_{n+1})=\mathbf{0}\end{aligned}\tag{84}$$

其未知量为$\boldsymbol{\varepsilon}$、$\boldsymbol{\delta}$、$\hat{\boldsymbol{\lambda}}$和$\hat{\boldsymbol{\mu}}$,可以用Newton-Raphson迭代求解,其迭代Jacobian矩阵为

$$\mathbf{J}_n=\begin{pmatrix}\mathbf{M}-\hat{h}\boldsymbol{Q}_v&\hat{h}\left(\sum_i\lambda_i\boldsymbol{C}_{qq}^i-\boldsymbol{Q}_q\right)&\mathbf{0}&\boldsymbol{C}_q^{\mathrm{T}}\\ -\hat{h}\mathbf{I}&\mathbf{I}-\hat{h}\sum_i\mu_i\boldsymbol{C}_{qq}^i&\boldsymbol{C}_q^{\mathrm{T}}&\mathbf{0}\\ \mathbf{0}&\boldsymbol{C}_q&\mathbf{0}&\mathbf{0}\\ \boldsymbol{C}_q&(\boldsymbol{C}_q\boldsymbol{v})_q+\boldsymbol{C}_{tq}&\mathbf{0}&\mathbf{0}\end{pmatrix}$$

在求解以上 Index-2 微分代数方程组并得到 $\boldsymbol{\varepsilon}$ 和 $\boldsymbol{\delta}$ 后，就可以同时判断 $\boldsymbol{\varepsilon}$ 和 $\boldsymbol{\delta}$ 相对于 $\boldsymbol{q}$ 和 $\boldsymbol{v}$ 的误差是否满足计算要求. 计算得出的 $\boldsymbol{q}_{n+1}$ 和 $\boldsymbol{v}_{n+1}$ 都是准确的，但拉格朗日乘子 $\boldsymbol{\lambda}_{n+1}$ 和加速度

$$\dot{\boldsymbol{v}}_{n+1} \approx \dot{\boldsymbol{v}}_{n+1}^{(0)} + \frac{1}{\hat{h}}\boldsymbol{\varepsilon}$$

可能还是误差较大. 为了进一步提高计算精度，可以将 $\boldsymbol{q}$、$\boldsymbol{v}$ 和 $\boldsymbol{a} = \dot{\boldsymbol{v}}$ 都变成 index-1 的变量，即方程(81)修改为

$$\begin{aligned}
&\boldsymbol{M}(\boldsymbol{q})\boldsymbol{a} + \boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda} = \boldsymbol{Q}(\boldsymbol{q},\boldsymbol{v},t)\\
&\dot{\boldsymbol{v}} - \boldsymbol{a} + \boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\tau} = \boldsymbol{0}\\
&\dot{\boldsymbol{q}} - \boldsymbol{v} + \boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\mu} = \boldsymbol{0}\\
&\boldsymbol{C}(\boldsymbol{q},t) = \boldsymbol{0}\\
&\boldsymbol{C}_q(\boldsymbol{q},t)\boldsymbol{v} + \boldsymbol{C}_t(\boldsymbol{q},t) = \boldsymbol{0}\\
&\boldsymbol{C}_q(\boldsymbol{q},t)\boldsymbol{a} - \boldsymbol{\gamma}(\boldsymbol{q},\boldsymbol{v},t) = \boldsymbol{0}
\end{aligned}$$

该方程组仍然是 index-2 的 DAEs，但其对应的 index-2 的变量为较次要的 $\boldsymbol{\lambda}$、$\boldsymbol{\mu}$ 和 $\boldsymbol{\tau}$，而所有重要的变量 $\boldsymbol{q}$、$\boldsymbol{v}$ 和 $\boldsymbol{a}$ 都是 index-1 的变量，在计算中都可以精确地计算出来并进行误差控制.

变步长变阶

BDF 方法是变阶变步长的，可以根据本步计算的后验误差进行下一步的阶数和步长的估计. 早期的 BDF 族 ODE 积分器，例如 DIFSUB [50] 或者 LSODI [51] 中，通常建议 k 阶格式等步长运行 $k+2$ 步之后再变阶变步长. 但在通用型 DAE 积分器中，为了及时捕捉系统的时变特性，建议每一步都要变阶和变步长. 为了方便讨论，记

$$e^j = \left\|\nabla^{j+1}\boldsymbol{x}_{n+1}\right\|, \qquad \varepsilon_j = \sqrt[j+1]{c_e^j e^j}$$

其中范数 $\|\cdot\|$ 的定义与方程(35)相同. BDF 方法变阶变步长的原理是将本步的 j 阶后验误差

$$\varepsilon_j \approx h^{j+1}\sqrt{c_e^j\left\|\boldsymbol{x}^{(j+1)}\right\|}$$

作为下一步计算步长仍为 h 时对应的 j 阶格式的误差预测，利用某种规则选择下一步最合适的阶数和时间步长. 计算所需的相关数据都含在矩阵

$$\mathbf{D}_{n+1}^{k+2} = \left(\nabla\boldsymbol{x}_{n+1}, \nabla^2\boldsymbol{x}_{n+1}, \ldots, \nabla^{k+2}\boldsymbol{x}_{n+1}\right)$$

中，其中备选的阶数为 $j = 1, 2, \cdots, k+1$. 因此，每步计算中可以随时降多阶，但至多只能升 1 阶. 这样一来，积分器既可以在处理意外(不光滑)事件过程中做到自适应降阶；又可以在求解光滑的动力学过程中根据需要将阶数逐渐增加，提高仿真效率；达到了高效性与鲁棒性之间的有效平衡.

阶数和步长的选择有很多方法，最简单的思路是选择阶数使下一步的预测步长最大，即选择

$$k_{new} = \arg\min_{j=1,2,\ldots,k+1}\varepsilon_j$$

和对应步长 $h_{new} = 0.8\cdot h/\varepsilon_{k_{new}}$，其中 0.8 为安全系数. 这种方法实现容易，其各种变形或修正格式在通用型软件的积分器中仍被广泛采用.

一般来说，BDF 低阶格式计算效率低，而高阶格式(特别是 $k \geqslant 3$ 的格式)的稳定性较差. 此外，如果保持阶数和步长，则上一步的 Jacobian 矩阵往往可以复用，会节省不少计算量. 因此在新的阶数和步长的选择中，一般采用如下原则：尽量不变阶也不变步长；升阶时候尽量保守. 例如：

(1) 在 Gear 的程序 DIFSUB[49] 中，每步要求升/降阶次都不超过 1. 如果本步的阶数为 k，在进行下一步阶数选择时只考虑 $k-1$、k、$k+1$ 这三种情况，从中选择可以在下一步取最大步长的阶数. 在该程序中，上述选阶原则是通过调整安全系数来实现的：

$$h_{k-1} = \frac{h}{1.3\varepsilon_{k-1}}, h_k = \frac{h}{1.2\varepsilon_k}, h_{k+1} = \frac{h}{1.4\varepsilon_{k+1}} \tag{85}$$

(2) 在 Petzold 等的程序 DASSL[52] 及其后续版本 DASPK[8] 中，该原则是这样体现的：采用 $1/\sqrt[k+1]{2}$ 作为 k 阶的保险系数，使程序更倾向于高阶格式；同时对于非绝对稳定的阶数(即 $k>2$ 时)必须满足条件 $\left\|\nabla^{k-1}\boldsymbol{x}\right\| \geqslant \left\|\nabla^{k}\boldsymbol{x}\right\| \geqslant \left\|\nabla^{k+1}\boldsymbol{x}\right\| \geqslant \left\|\nabla^{k+2}\boldsymbol{x}\right\|$ 才允许升阶. 这样做不但保证了计算效率，而且同时解决了高阶格式的不稳定问题.

还要指出的是，如果变步长[53]或者变阶[54]算法不合适，也可能带来稳定性问题，因此通常步长和阶数不能增加得太快. 作者经过测试发现，Gear 等[53]给出的每一阶的最大步长增加量偏于保守，而 Skelboe[13] 给出的步长增加量在实践中更为合理，其步长增加上界 f_k 如表 1 所示.

表 1 BDF 格式变步长的最大稳定增加量

Table 1 The maximum stable increment of the variable step-size BDF format

Order k	1	2	3	4	5
$h_{n+1}/h_n \leqslant f_k$	∞	2.6	1.9	1.5	1.2

当然，矩阵 $\mathbf{D}_{n+1}^{k+2}$ 包含了远多于上述变阶变步长策略所需的信息，这给了变阶变步长算法更多自由度，而文献中也提供了一些其他思路[55]. 商业软件中的步长和阶数选择还要复杂得多，以有效地应

对各种意外情形.

高阶格式稳定性检测与自适应处理

不同阶数的BDF格式具有不同的稳定区域,其中只有前两阶是绝对稳定的,而更高阶的算法都在虚轴上有不稳定区域.在宽频动力学系统,特别是在含有柔性体或者大变形体的多体系统的仿真中,某些频率可能会滑到这些不稳定区域内,造成误差的严重放大.此时上文的误差估计方法就不再正确,且阶数和步长选择策略也不再具有鲁棒性,在计算中表现为一种积分阶数的频繁跳跃现象,并伴随出现大量的积分步失败.为了避免这一问题,很多程序(例如LSODI[51])中建议手动设置最高阶数为2.为了进一步解决这一问题,实践中还发展出了几种其他解决方案:

(1) Skelboe[13]经过稳定性分析给出了如下判据:如果$s_k h^k \left\| \boldsymbol{x}^{(k)} \right\| \leqslant h^{k+1} \left\| \boldsymbol{x}^{(k+1)} \right\|$时,意味着$k$阶格式失稳发生了,其中$k \geqslant 3$的$s_k$由表2给出.在实际计算中只需监测:

表2　BDF高阶格式的失稳判断参数

Table 2　Instability parameters of high-order BDF format

Order k	3	4	5
S_k	0.59	0.65	0.89

若$s_k \left\| \nabla^k \boldsymbol{x} \right\| \leqslant \left\| \nabla^{k+1} \boldsymbol{x} \right\|$,则说明$k$阶格式发生失稳,程序自动降低到$<k$阶格式进行仿真计算;

(2) 在DASSL[51]及其后续版本DASPK[8]中,采用了另一种稳定性判据:当$k \geqslant 2$时,要求至少满足$\left\| \nabla^{k-1} \boldsymbol{x} \right\| \geqslant \left\| \nabla^k \boldsymbol{x} \right\| \geqslant \left\| \nabla^{k+1} \boldsymbol{x} \right\| \geqslant \left\| \nabla^{k+2} \boldsymbol{x} \right\|$时才允许升阶,否则只能保持原阶或者降阶;

(3) Stewart[14]进一步考虑了步长变化的影响,提出当第n步结束时,如果下列条件满足:

$$\ell \left\| \nabla^{k+2} \boldsymbol{x} \right\| \geqslant \max\left(\left\| \nabla^{k+1} \boldsymbol{x} \right\|, 0.9 \left\| \nabla^k \boldsymbol{x} \right\| \right)$$

则下一步暂时不考虑升阶,其中

$$\ell = \frac{h_n}{h_{n-1}} \frac{h_n + h_{n-1}}{h_{n-1} + h_{n-2}}$$

作者经过测试发现,将Skelboe和Stewart的方案结合起来,虽然最有效地克服了失稳问题,但在实践中似乎偏于保守,会影响计算效率.DASSL采用的策略虽然也可以在一定程度上避免失稳现象,但却不能保证不会出现失稳.其工作原理大致如下:如果$\boldsymbol{x}(t)$至少k+1阶光滑可导,一般有

$$h \left\| \dot{\boldsymbol{x}} \right\| \geqslant h^2 \left\| \ddot{\boldsymbol{x}} \right\| \geqslant \cdots \geqslant h^{k+1} \left\| \boldsymbol{x}^{(k+1)} \right\| \tag{86}$$

这是因为若系统可以用Taylor展开表达式逐阶近似,高阶Taylor项应当比低阶项更小.系统的数值光滑度可定义为选择最大的k满足如下条件

$$\left\| \nabla \boldsymbol{x} \right\| \geqslant \left\| \nabla^2 \boldsymbol{x} \right\| \geqslant \cdots \geqslant \left\| \nabla^{k+1} \boldsymbol{x} \right\|$$

如果$\boldsymbol{x}(t)$达不到k阶光滑度,就不满足BDF算法的基本假设,只能采用$<k$阶的BDF格式来求解.作者认为上述条件过于严苛,而且没有理由认为低阶项会影响到高阶连续性,只要条件

$$\left\| \nabla^k \boldsymbol{x} \right\| \geqslant \left\| \nabla^{k+1} \boldsymbol{x} \right\| \geqslant \left\| \nabla^{k+2} \boldsymbol{x} \right\|$$

满足,即可认为符合从k阶升到k+1阶的必要条件.这一判据可以和Skelboe的方法联合起来使用.这种解决方案效果比较明显,一般情况下只需设定最大计算阶数为5,在一般的高阶不稳定情形,程序自动会跳回2阶,而不会再出现阶数跳跃和步长估计失败的问题,保证了计算效率.

5　隐式积分器的其他计算细节

隐式积分器每步的计算量比较大,但其时间步长较大,总体来说计算效率较高;其稳定特性很好,可以很容易地处理方程组的刚性问题.但是,时间步长太大,可能会在计算中不能有效地处理局部的一些突变问题,而需要一些特殊的技巧.

5.1　拟Newton迭代与Jacobian复用

在求解常微分方程(20)、(70)或者微分代数方程(32)、(38)、(78)和(84)的过程中,甚至在后文中显式积分器的约束流形投影过程中,都需要用到Newton迭代或者拟Newton迭代.动力学仿真中的非线性方程组的求解具有以下特点:

(1) **多**.每个时间步都要用拟Newton迭代求解;

(2) **快**.状态变量的预测值通常非常准确,往往只需2－3次拟Newton迭代即可收敛;

(3) **大**.待求解方程组的规模比较大;

(4) **规**.Jacobian矩阵往往具有特定的稀疏性.

Jacobian矩阵的相关计算往往是整个仿真计算的瓶颈问题,这是由以下两个原因导致的:

(1) Jacobian矩阵生成过程的计算量很大,无论是用解析表达式还是数值差分方法;

(2) Jacobian矩阵的LU分解计算复杂度比其他计算过程要高,即使采用稀疏矩阵计算也是如此.

一般情况下多体动力学计算仿真中都用的是拟Newton迭代,以避免大量的Jacobian矩阵计算,并在所有地方都尽量复用Jacobian矩阵的LU分解.

实践证明,这样做往往可以将计算速度提高5到10倍.还应该指出的是,几乎在所有的仿真过程中,总会遇到少数拟Newton迭代收敛很慢甚至失败的情形,给整个仿真带来问题.如果这些问题不解决,会导致整个计算的中止.而拟Newton迭代的失败,大多是由于Jacobian矩阵不再精确造成的.因此,为了解决这一问题,可以设一个最大迭代步数(例如4到5次),如果达到最大迭代步数还不收敛,就可以认为拟Newton迭代失败,需要重新生成Jacobian矩阵并继续进行迭代.

拟Newton迭代的一般过程为,为了求解非线性方程组 $\boldsymbol{F}(\boldsymbol{x})=0$,采用迭代格式

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-v\hat{\mathbf{J}}^{-1}\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)$$

直到修正量或残差足够小为止,其中 $\hat{\mathbf{J}}$ 为近似Jacobian矩阵,v 为松弛因子.根据压缩不动点定理,迭代收敛的充分条件是存在某一常数 σ 满足

$$\left\|\mathbf{I}-v\hat{\mathbf{J}}^{-1}\frac{\partial\boldsymbol{F}}{\partial\boldsymbol{x}}\left(\boldsymbol{x}^{(k)}\right)\right\|\leqslant\sigma<1 \tag{87}$$

如果上述条件满足,则由压缩映射的性质得到 $\left\|\boldsymbol{x}^{(k+1+j)}-\boldsymbol{x}^{(k+j)}\right\|\leqslant\sigma^{j}\left\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\right\|$,其中 $j=1,2,\cdots$,进而对所有 $j\geqslant1$ 项求和得出

$$\left\|\boldsymbol{x}^{*}-\boldsymbol{x}^{(k+1)}\right\|\leqslant\frac{\sigma}{1-\sigma}\left\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\right\|$$

其中 $\boldsymbol{x}^{*}$ 为真实解.该公式可以用来判断迭代是否收敛,因为右边项可以计算,其中 σ 可以由下式估计得出

$$\sigma=\lim_{x\to\infty}\left\|\frac{\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}}{\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}}\right\|$$

为了保证拟Newton迭代快速收敛,需要监测收敛速度 σ.计算中如果发现 $\sigma>0.9$,最好重新计算Jacobian并估计 σ.另外,实际计算中,条件(87)可能是不满足的,特别是在以下情形中:

(1) 所采用的Jacobian矩阵 $\hat{\mathbf{J}}$ 可能是奇异的或者近于奇异的.例如在机械系统的奇异位形附近,约束矩阵 $\boldsymbol{C}_q$ 近于冗余,而此时 $\hat{\mathbf{J}}^{-1}$ 的范数很大,条件(87)可能不再满足;

(2) 强刚性问题中,Jacobian矩阵的条件数很大,近似Jacobian与真实Jacobian矩阵之间很小的差异,也会导致条件(87)不满足[31];

(3) 在很多问题中,真实解析Jacobian矩阵的生成太过烦琐,或者其稀疏性不太好,往往用近似Jacobian计算,可能不满足条件(87);

(4) 在某些场合,非线性问题本身的收敛域可能很小,而这种情况是完全无法提前预知的.

检测到拟Newton迭代不收敛时需要立即重新计算Jacobian矩阵,并进行LU分解.一般说来,拟Newton迭代的收敛速度低于Newton迭代,但计算量远小于后者.实际计算中需要结合二者的优点,尽可能地加快计算速度.

总体说来,广义 α 方法计算步长较小,拟Newton方法的收敛性基本上是有保证的;除非在某些情形,Jacobian矩阵接近奇异,或者出现某些不连续现象.而BDF族方法一般步长较大,因此拟Newton迭代不收敛,或者收敛速度太慢的问题也更加严重.作者建议如果在5步以内拟Newton迭代不收敛的话,最好重新计算Jacobian矩阵.

另一方面,由于一般动力学系统的质量矩阵、刚度矩阵和约束矩阵等随时间变化比较慢.因此,如果步长变化不大,上一步用到的Jacobian矩阵就可以在下一步被重复使用.作者经过比较[8,56],建议采用如下判据:如果新步长与原步长相比,

$$\sigma\frac{\hat{h}_{new}}{\hat{h}_{old}}+\left|\frac{\hat{h}_{new}}{\hat{h}_{old}}-1\right|<\frac{1}{3}$$

则可以重复使用Jacobian矩阵,并采用松弛因子

$$v=\frac{2\hat{h}_{new}}{\hat{h}_{new}+\hat{h}_{old}}$$

对于广义 α 族方法,$\hat{h}=h$;而对于BDF族方法,$\hat{h}=h/c_k$,如方程(71)所示.实践表明,这一复用往往至少能把计算时间减少一半以上;如果动力学过程比较光滑,计算速度可以提升超过3倍.

最后,即使拟Newton迭代本身收敛,但得到的修正更新值也可能不满足预设的误差条件.总体说来,误差判断失败的原因可能有以下几点:

(1) 动力学中遇到非光滑事件(接触碰撞、机构奇点、参数或模型突变等),导致状态预测值不准确;

(2) 由于参数变化,用上一步仿真的后验误差估计得到的时间步长 h 太大,不满足误差限要求;

(3) 系统刚性太强,或者积分步长太小,导致Jacobian矩阵条件数很高,舍入误差太大.

其解决方案就是用得到的后验修正更新量重新设置迭代步长,并在必要的情况下降低积分器阶数.这一过程和变步长变阶策略类似,只是不会用到升阶,因而不用进行 $k+1$ 阶的误差估计.

5.2　数值奇点检测与处理

数值奇点是仿真中经常遇到的问题.与物理奇异点不同,数值奇点指的是因为数值原因导致的奇异位形,例如用欧拉角等参数描述旋转矩阵时遇到的奇点.数值奇异性的处理流程为:

(1) 奇点检测:对于每种可能遇到的奇异性都要设置接近于奇点处的报警功能;

(2) 奇点消除技术:选择新的坐标系统将数值奇点处的位形变换到新坐标系统下的非奇异位形;

(3) 积分器重启技术:在新的坐标下重启积分器.

数值奇点检测与处理的缺点在于随着部件数量的增加,不但自由度个数会增加,而且奇点出现的机率也会增加,而积分器重启的可能性也会增加.自由度个数增加,本来计算量就会增加;而积分器重启的机率也会增加,给仿真带来的效率损失也会增加.而这种损失并非物理系统本身的特性导致的,是由于所选取坐标系统的数值特性不合适导致的,只能用数值方法来解决.

数值奇点附近光滑性保持问题对于单步法是比较简单的,例如对于广义 α 方法,只要在奇点附近换一套局部非奇异坐标,换算出新坐标下的 $\boldsymbol{q}$、$\dot{\boldsymbol{q}}$、$\ddot{\boldsymbol{q}}$ 和 $\boldsymbol{a}$ 就可以直接开始下一步的计算.积分步长只是局部发生了一些变化,而不会影响整体的仿真效果.而对于多步法,情况比较复杂.一种思路是总是从最低阶(1阶)用很小的步长重启并开始计算,其缺点是重启过程太慢,一般需要 10 –20 步才能恢复正常仿真阶数和步长.对于很多部件组成的系统,奇点出现机会可能比较高,反复重启可能速度太慢,影响仿真效率.另一种思路是在奇点严重影响仿真效率之前,尽早进行坐标变换,并将前几步得到的结果也进行同样变换,保持高阶积分格式.

5.3　不光滑性检测与处理

不光滑性出现一般有如下几个原因:接触碰撞、模型拓扑变化、不连续外力作用等等.其具体表现为某些物理量或者其时间导数、高阶导数的不连续性.不连续性问题在很多研究者看来似乎不甚重要,因为很多自适应步长的积分器往往可以通过多次试错突破不连续性的障碍,但其代价一般还是比较大的.好在对于绝大部分问题来说,不连续点的位置是比较少的,因此这些代价相当于整个计算成本来说还是比较小的.但事实上,确实有不少工业界遇到的问题,是由于不连续性太强而求解失败的.因此作为商业软件级别的积分器,还是应该包含不光滑性自动检测和定位的功能,从而可以保证更好的效率和鲁棒性.Gear 等[57]提出一种 ODE 的不连续的检测和定位方法,可以用来将积分器的时间节点置于定位得到的不连续点,而不连续性问题可以有效地通过积分器重启得到解决.这一处理不连续性的方法也可以推广到 DAE 中去[58],并可利用以下性质:

(1) 质量矩阵 $\mathbf{M}$ 随时间的变化一般是缓慢光滑的,而外力项 $\boldsymbol{Q}$ 可能有一些不连续现象,而 $\boldsymbol{Q}$ 的不光滑性检测可以给下一步的预测提供信息;

(2) 除了在有限个间断点之外,约束 $\boldsymbol{C}(\boldsymbol{q},\mathrm{t})$ 都是至少二阶连续的.事实上,由于在所有计算过程中,都希望做到 $\boldsymbol{C}\approx 0$、$\dot{\boldsymbol{C}}\approx 0$ 以及 $\ddot{\boldsymbol{C}}\approx 0$,因此不光滑性测试不能直接作用在它们上面,而选取实际中最关键的矩阵 $\boldsymbol{C}_q$ 中的非零元素作为不光滑性测试的量.

作者认为,BDF 族方法得到的 $\mathbf{D}_{n+1}^{k+2}$ 矩阵很好地描述了动力学在 t_n 至 t_{n+1} 之间的数值连续性,可以用于不光滑性检测,以及间断处理算法设计.

6　显式积分器简介

一般说来,显式积分器不适合用于求解代数微分动力学系统,但少数情形例外.在用递归方法得到的动力学方程组中,约束方程的个数远少于自由度个数;同时,如果系统的刚性问题不太严重,或者高频振动不能有效地得到衰减(例如对于大规模的接触碰撞问题的仿真),此时显式积分器的计算性能可能会优于隐式积分器.另外, 显式积分器的计算量基本可控,因此比较适用于实时仿真.还有,由于显式积分器无需迭代,在多积分器合作仿真中实现起来方便得多.

约束修正法

从形式上看,约束修正法有些类似直接求解 index-3 的微分代数方程组.在这种意义下,大部分显式常微分方程积分器都可以用来求解方程组(1).首先,将方程组(1)改写成常微分方程组

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}=\boldsymbol{Q}(\boldsymbol{q},\dot{\boldsymbol{q}},t)\\&\boldsymbol{C}_q(\boldsymbol{q},t)\ddot{\boldsymbol{q}}=\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},t)\end{aligned}\tag{88}$$

其中 $\boldsymbol{\gamma}$ 和方程(5)中的一样.直接求解方程(88)会引起约束违约现象,因为公式(88)的第二个方程并不等价于公式(1)中的约束方程.方程(88)是常微分方程组,可以用高阶显式的常微分方程积分器来

从 t_n 时刻到 t_{n+1} 时刻进行计算，所用积分器既可以是单步法又可以是多步法.计算得到的解不一定满足约束，因此需要进行修正，将每一步求得的 $\boldsymbol{q}$ 和 $\dot{\boldsymbol{q}}$ 在约束流形上投影.显式积分器求解方程(88)的计算过程需要用到

$$\mathbf{J}(\boldsymbol{q},t)=\begin{pmatrix}\mathbf{M}(\boldsymbol{q}) & \boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\\ \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t) & \mathbf{0}\end{pmatrix} \tag{89}$$

在多体系统动力学中，由于 $\mathbf{M}(\boldsymbol{q})$ 和 $\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t)$ 变化缓慢.因而 $\mathbf{J}$ 也是时间缓变的，可以采用各种近似和复用来提高计算效率.求解常微分方程的计算过程可以用算子在形式上表示为

$$\left(\bar{\boldsymbol{v}}_{n+1},\bar{\boldsymbol{q}}_{n+1}\right)=\mathbf{T}\left(\boldsymbol{q}_n,\dot{\boldsymbol{q}}_n,h\right)$$

得到的 $\bar{\boldsymbol{q}}$ 和 $\bar{\boldsymbol{v}}$ 可能不适合于约束方程，要进一步作约束流形投影修正，包括位置修正

$$\mathbf{M}\left[\boldsymbol{q}_{n+1}-\bar{\boldsymbol{q}}_{n+1}\right]+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q}_{n+1},t_{n+1})\boldsymbol{\mu}=\mathbf{0}$$
$$\boldsymbol{C}(\boldsymbol{q}_{n+1},t_{n+1})=\mathbf{0}$$

和速度修正

$$\mathbf{M}\left[\dot{\boldsymbol{q}}_{n+1}-\bar{\boldsymbol{v}}_{n+1}\right]+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q}_{n+1},t_{n+1})\boldsymbol{\mu}=\mathbf{0}$$
$$\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q}_{n+1},t_{n+1})\dot{\boldsymbol{q}}_{n+1}+\boldsymbol{C}_{t}(\boldsymbol{q}_{n+1},t_{n+1})=\mathbf{0}$$

这两种修正计算都需要通过迭代完成，计算中可以复用方程(89)中的 $\mathbf{J}$，以节省计算量.

罚函数法

形式上，罚函数法有些类似求解 index-2 的微分代数方程组(81)，只是做了一些修改

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{v}}+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}=\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{v},t)\\&\mathbf{M}(\boldsymbol{q})\left[\dot{\boldsymbol{q}}-\boldsymbol{v}\right]+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\mu}=\mathbf{0}\\&\boldsymbol{C}(\boldsymbol{q},t)=\mathbf{0}\\&\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t)\boldsymbol{v}+\boldsymbol{C}_{t}(\boldsymbol{q},t)=\mathbf{0}\end{aligned} \tag{90}$$

罚函数的意思是利用很大的正数 χ（罚因子)将

index-2 的微分代数方程组中的约束方程组

$$\boldsymbol{C}(\boldsymbol{q},t)=\mathbf{0},\qquad \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t)\boldsymbol{v}+\boldsymbol{C}_{t}(\boldsymbol{q},t)=\mathbf{0}$$

中形如 $\boldsymbol{f}=0$ 的约束改写成方程 $\dot{\boldsymbol{f}}+\chi\boldsymbol{f}=0$ 的形式，得到

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{q}}+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\mu}=\boldsymbol{M}(\boldsymbol{q})\boldsymbol{v}\\&\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t)\dot{\boldsymbol{q}}=-\boldsymbol{C}_{t}(\boldsymbol{q},t)-\chi\boldsymbol{C}(\boldsymbol{q},t)\end{aligned} \tag{91}$$

和

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{v}}+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}=\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{v},t)\\&\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t)\dot{\boldsymbol{v}}=\boldsymbol{\gamma}(\boldsymbol{q},\boldsymbol{v},t)-\chi\left\{\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{q},t)\boldsymbol{v}+\boldsymbol{C}_{t}(\boldsymbol{q},t)\right\}\end{aligned} \tag{92}$$

可以从中求解出 $\dot{\boldsymbol{q}}(\boldsymbol{q},\boldsymbol{v},t)$ 和 $\dot{\boldsymbol{v}}(\boldsymbol{q},\boldsymbol{v},t)$，而且求解过程用到方程(89)中的 $\mathbf{J}$.得到的常微分方程组可以用显式积分器进行仿真，当然并非所有常微分方程积分器都适合开展罚函数法，有一类特殊的 Runge-Kutta-Chebyshev 类型的积分器[39-40]，其在负半实轴上稳定域很长(见图 1)，因此适合用来显式求解方程组(91)和(92).本文推荐采用四阶的 RKC 积分器，其稳定区域是递归扩展的[40]，且负半轴上稳定区域近似与递归阶的平方成正比.

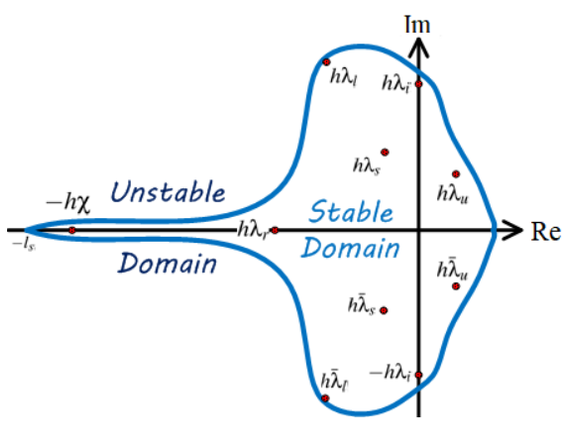

图 1 RKC 积分器的稳定区域示意图

Fig. 1 Schematic of the Stable Domain for RKC Integrators

该积分器在虚轴上收敛区域有限，不适合用来求解刚性问题，但对一般非刚性的多刚体系统动力学的递归格式，显式积分格式求解(91)和(92)的效果非常好.要指出的是，在计算之前可以从系统最大特征值基本确定罚因子 χ，因为 χ 必须远远大于系统的最大特征值，进而可以定出递归阶.此外，提高计算效率的关键在于如何有效地从微分方程组(91)和(92)中求解出 $\dot{\boldsymbol{q}}$ 和 $\dot{\boldsymbol{v}}$，可以根据所求问题的特点加进这一过程.

7 非完整约束 DAEs 的积分技术

一般来说，非完整约束的 DAEs 比完整约束的 DAEs 简单，因为对应的变量一般是 index-2 的.带非完整约束的 DAEs 的一般格式为[46]

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q},t)\ddot{\boldsymbol{q}}+\boldsymbol{C}_{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}+\boldsymbol{G}_{\dot{\boldsymbol{q}}}^{\mathrm{T}}(\boldsymbol{q},\dot{\boldsymbol{q}},t)\boldsymbol{\mu}=\boldsymbol{Q}(\boldsymbol{q},\dot{\boldsymbol{q}},t)\\&\boldsymbol{C}(\boldsymbol{q},t)=\mathbf{0},\qquad \boldsymbol{G}(\boldsymbol{q},\dot{\boldsymbol{q}},t)=\mathbf{0}\end{aligned} \tag{93}$$

该方程组可以分别被上面介绍的几种方法离散：

(1) **广义 α 族方法**.同样假设 $\boldsymbol{q}$、$\dot{\boldsymbol{q}}$ 和 $\ddot{\boldsymbol{q}}$ 的表达式为方程(29)和(30)，代入方程组(93)，得到关于 $\boldsymbol{x}$、$\boldsymbol{\lambda}$ 和 $\boldsymbol{\mu}$ 的代数方程组

$$\mathbf{M}(\boldsymbol{q}_{n+1})\ddot{\boldsymbol{q}}_{n+1}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\boldsymbol{\lambda}+\boldsymbol{G}_{\dot{q}}^{\mathrm{T}}(\boldsymbol{q}_{n+1},\dot{\boldsymbol{q}}_{n+1})\boldsymbol{\mu}-\boldsymbol{Q}(\boldsymbol{q}_{n+1},\dot{\boldsymbol{q}}_{n+1})=\mathbf{0}$$

$$\frac{1}{\beta h^2}\boldsymbol{C}(\boldsymbol{q}_{n+1})=\mathbf{0},\qquad \frac{1}{\gamma h}\boldsymbol{G}(\boldsymbol{q}_{n+1},\dot{\boldsymbol{q}}_{n+1})=\mathbf{0}$$

可以用Newton-Raphson方法或者其他迭代方法求解,其对应Jacobian矩阵为

$$\mathbf{J}=\begin{pmatrix}\mathbf{K} & \boldsymbol{C}_q^{\mathrm{T}} & \boldsymbol{G}_{\dot{q}}^{\mathrm{T}}\\ \boldsymbol{C}_q & \mathbf{0} & \mathbf{0}\\ \boldsymbol{G}_{\dot{q}}+\dfrac{\beta}{\gamma}h\boldsymbol{G}_q & \mathbf{0} & \mathbf{0}\end{pmatrix}$$

其中

$$\mathbf{K}=\eta\mathbf{M}+\gamma h\left(\sum\nolimits_i\mu_i\boldsymbol{G}^i_{\dot{q}\dot{q}^{\mathrm{T}}}-\boldsymbol{Q}_{\dot{q}}\right)+\beta h^2\left(\sum\nolimits_i\left(\lambda_i\boldsymbol{C}^i_{qq^{\mathrm{T}}}+\mu_i\boldsymbol{G}^i_{q\dot{q}^{\mathrm{T}}}\right)-\boldsymbol{Q}_q\right)$$

(2) **BDF族方法**. 将BDF方法的公式(74)、(75)和(77)代入方程组(93)得到离散格式

$$\mathbf{M}(\boldsymbol{q}_{n+1})\hat{h}^2\ddot{\boldsymbol{q}}_{n+1}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q}_{n+1})\hat{h}^2\boldsymbol{\lambda}+\boldsymbol{G}_{\dot{q}}^{\mathrm{T}}(\boldsymbol{q}_{n+1},\dot{\boldsymbol{q}}_{n+1})\hat{h}^2\boldsymbol{\mu}-\hat{h}^2\boldsymbol{Q}(\boldsymbol{q}_{n+1},\dot{\boldsymbol{q}}_{n+1})=\mathbf{0}$$

$$\boldsymbol{C}(\boldsymbol{q}_{n+1})=\mathbf{0},\qquad \hat{h}\boldsymbol{G}(\boldsymbol{q}_{n+1},\dot{\boldsymbol{q}}_{n+1})=\mathbf{0}$$

其对应Jacobian矩阵为

$$\mathbf{J}=\begin{pmatrix}\mathbf{K} & \boldsymbol{C}_q^{\mathrm{T}} & \boldsymbol{G}_{\dot{q}}^{\mathrm{T}}\\ \boldsymbol{C}_q & \mathbf{0} & \mathbf{0}\\ \boldsymbol{G}_{\dot{q}}+\dfrac{\beta}{\gamma}h\boldsymbol{G}_q & \mathbf{0} & \mathbf{0}\end{pmatrix}$$

其中

$$\mathbf{K}=\mathbf{M}+\hat{h}\left(\sum\nolimits_i\mu_i\boldsymbol{G}^i_{\dot{q}\dot{q}^{\mathrm{T}}}-\boldsymbol{Q}_{\dot{q}}\right)+\hat{h}^2\left(\sum\nolimits_i\left(\lambda_i\boldsymbol{C}^i_{qq^{\mathrm{T}}}+\mu_i\boldsymbol{G}^i_{q\dot{q}^{\mathrm{T}}}\right)-\boldsymbol{Q}_q\right)$$

在求解出位置修正量$\boldsymbol{\delta}$后,可以计算出速度修正量$\boldsymbol{\varepsilon}$;同样的,$\boldsymbol{\varepsilon}$不能直接用来估计计算误差,而需要对几何约束作投影滤波修正.

带非完整约束的index-2格式的DAEs为

$$\begin{aligned}&\dot{\boldsymbol{q}}-\boldsymbol{v}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\xi}=\mathbf{0}\\&\mathbf{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}}+\boldsymbol{C}_q^{\mathrm{T}}(\boldsymbol{q},t)\boldsymbol{\lambda}+\boldsymbol{G}_v^{\mathrm{T}}(\boldsymbol{q},\boldsymbol{v},t)\boldsymbol{\mu}=\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{v},t)\\&\boldsymbol{C}(\boldsymbol{q},t)=\mathbf{0}\\&\boldsymbol{C}_q(\boldsymbol{q},t)\boldsymbol{v}+\boldsymbol{C}_t(\boldsymbol{q},t)=\mathbf{0}\\&\boldsymbol{G}(\boldsymbol{q},\boldsymbol{v},t)=\mathbf{0}\end{aligned}\tag{94}$$

该方程组同样可以用广义α族方法和BDF族方法来进行积分,可以避免spikes现象,计算出更光滑更高精度的速度、加速度和约束力曲线.

此外,在实际应用中,多体动力学系统往往和控制算法结合起来应用,因此在仿真中也要将控制律方程组同时引入进行仿真.控制方程组往往由一阶微分方程组和代数方程组组成,因此可以通过一阶微分方程积分器,例如BDF方法等进行有效地时间离散.另外,动力学仿真中往往还包含一些离散动力学方程组,其积分方法大致有两种,一种是将离散系统连续化,另一种是交互仿真技术.前者比较简单,已经普遍应用于商业软件可以解决自适应步长与离散动力学时间节点的不协调问题;后者需要用到积分器与离散系统动力学之间的交互仿真技术,在本文中将不做赘述.

8　典型算例

下面用两个典型多体系统动力学测试算例来说明文中讨论的一些技术细节.应该指出的是,不同族的积分器的局部误差估计在仿真精度控制实践中有着不同的意义,这既与积分器的阶数有关又与局部误差到全局误差的传导机制有关.仿真中的整体计算精度其实是由全局误差来控制的,但全局误差是很难估计的.大量测试发现,广义α族积分器的局部相对误差限需要比BDF族积分器低两个数量级才能达到类似的整体仿真精度.例如在商业软件ADAMS中[59],HHT积分器的缺省相对误差限为10^{-5},而BDF族积分器的对应误差限为10^{-3},才能保证二者的整体计算效果大致接近.在以下算例计算中,对广义α族积分器,取局部相对误差限为10^{-6};而对BDF族积分器,取局部相对误差限为10^{-4}.计算中,选择index-3的积分器作为广义α族积分器中的代表;选择变步长公式的index-2和index-3格式的积分器作为BDF族积分器中的代表;而选择四阶显式积分器作为显式积分器族中的代表开展计算.

8.1　七刚体机构

七刚体模型[60]是多刚体系统动力学的标准测试程序,其机构简图为图2.该机构与地面有四个连接点:点O位于原点;点A的坐标为$(x_{\mathrm{A}},\ y_{\mathrm{A}})$;点B的坐标为$(x_{\mathrm{B}},\ y_{\mathrm{B}})$;以及点C的坐标为$(x_{\mathrm{C}},\ y_{\mathrm{C}})$.每个刚体的几何形状如图1所示,其质心坐标系用箭头标识出来;定常驱动力矩τ作用于原点;C点连接的弹簧原长为l_0,其刚度系数为k_0.这些机构的几何参数及驱动力矩见表3,而每个刚体的惯性参数见表4.

上述模型分别用广义α积分器、显式积分器、

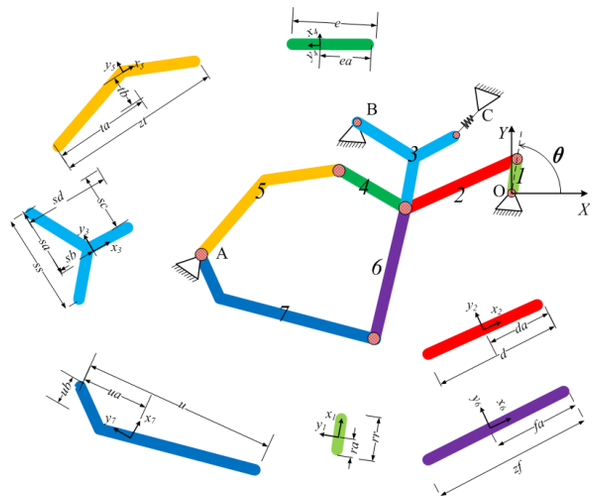

图2 七刚体机构系统

Fig. 2 Schematic of the seven body mechanism

BDF 格式的 index−3 积分器进行仿真，其计算结果如图 3 所示.

仿真过程中各种积分器自适应计算的步数和时间比较见表 5. 其结果显示，对于上述多刚体系统仿真来说，BDF 积分器一般要比广义 α 积分器快；而对于非刚性问题，显式积分器也比广义 α 积分器快. 这主要是由于七刚体模型满足高阶格式的连续性条件，可以采用大的时间步长进行积分. 显式积分器基本不能用于一般的刚性问题求解，因为此时决定其积分步长的是稳定性而非精度，而此时稳定性条件导致显式积分器的步长非常小.

表3 七刚体机构的参数

Table 3 Parameters of the seven body mechanism

Parameters (m)						
d	ea	rr	sa	sd	tb	ub
0.028	0.01421	0.007	0.01874	0.02	0.00916	0.00449
da	zf	ra	sb	zt	u	c_0
0.0115	0.02	0.00092	0.01043	0.04	0.04	4530
e	fa	ss	sc	ta	ua	l_0
0.02	0.01421	0.035	0.018	0.02308	0.01228	0.07785
x_A	y_A	x_B	y_B	x_C	y_C	τ
−0.06934	−0.00277	−0.036335	0.03273	0.014	0.072	0.033

表4 七刚体机构的惯性参数表

Table 4 Inertia parameters of the seven body mechanism

Index of rigid body	1	2	3	4	5	6	7
Mass (10^{-2})	4.325	0.365	2.373	0.706	7.050	0.706	5.498
Moment of inertia (10^{-6})	2.194	0.441	5.255	0.5667	11.69	0.5667	19.12

图4给出了计算中 BDF 积分器的阶数变化，证实了上述模型的仿真中，BDF 积分器在整个计算过程中其自适应选择的格式的阶数基本都高于二阶，这也是其可以采用大的时间步长来计算的主要原因.

8.2 曲柄滑块机构

曲柄滑体机构模型是测试积分器性能的柔性多体系统动力学模型，因为实践表明，三阶以上的 BDF 积分器会在该模型中遇到稳定性的考验.

模型描述如图 5 所示，系统由刚性杆 1、柔性杆 2 以及滑块 3 组成. 刚性杆 1 的质量为 0.36 kg，质心转动惯量为 0.002727 kg · m^2，长度 l_1 = 0.15 m；柔性杆的密度为 7570 kg/m^3，截面为 8 × 8 mm^2 的矩形，杨氏模量为 200Gpa，未变形时长度 0.30m；滑块 3 的质量为 0.7555 kg. 忽略重力作用，假设给定旋转驱动为 $\phi(t) = \omega t$，其中 ω = 150 rad/s. 计算中取柔性杆前 3 阶横向振动模态和第 1 阶纵向振动模态的模态坐标来描述其变形. 假设机构的初始位形为 ϕ = 0 和 x = 0.45 m，以及横向振动模态坐标 $q_1 = q_2 = 0$，$q_3 = 1.033 \times 10^{-5}$ 和纵向振动模态坐标 $q_4 = 1.69 \times 10^{-5}$；速度初始值由 $\dot{\phi} = \omega$ 可以由 DAE 初始条件分析算出其他广义初速度.

该模型具有很强的刚性，只能用隐式积分器来计算. 用广义 α 积分器、BDF 格式的 Index−2 和 Index−3 积分器的计算结果如图 6 所示. 从中可以看出，几个积分器的结果基本完全吻合；另外，直接求解 index−3 方程组得到的加速度曲线图中发生了误差放大的尖刺现象.

由于 BDF 积分器里使用了自适应稳定性算法，在各阶频率组合导致高阶格式可能会发生不稳定性的场合，程序自适应地找到最合适的低阶格式（一般为二阶格式，如图 7 所示）来进行仿真，而无需手动修改设置，这样做为程序的使用带来了很大方便. 另外，即使在如此不利的情况下，BDF 积分器的计算效率也与广义 α 方法大致接近，如表 6 所示，这也验证了 BDF 族积分器的高效性和鲁棒性.

9 几种积分器的性能比较

积分器的性能比较应该是在同样的计算精度下进行. 在同样的误差要求下，index−2 的 DAE 积分

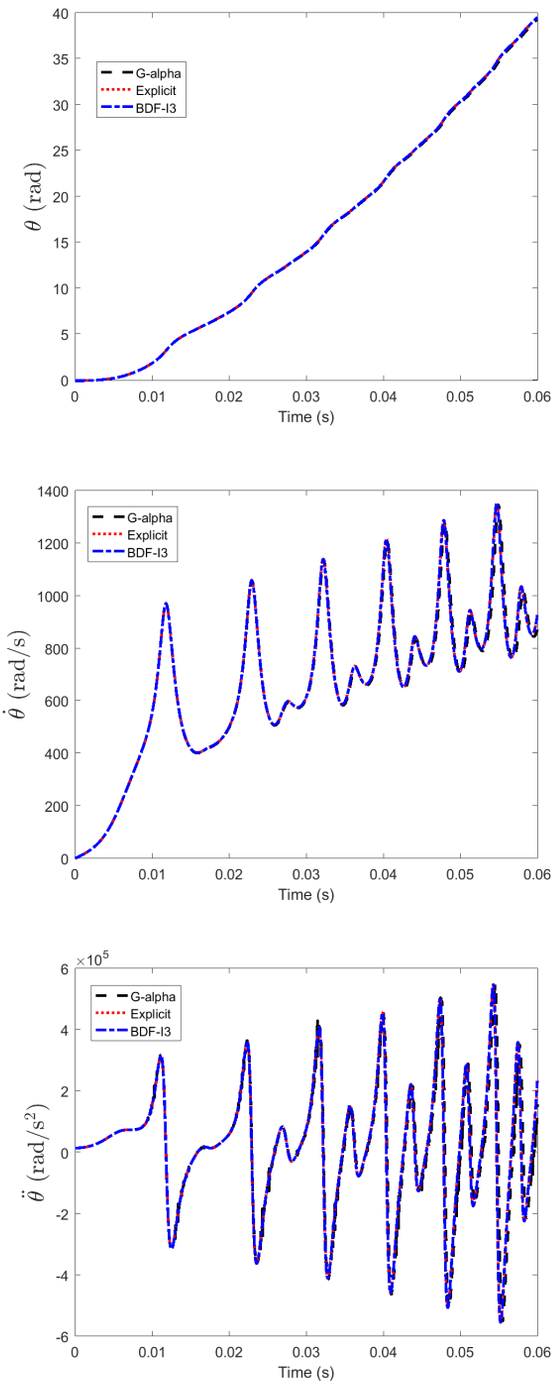

图3　七刚体机构仿真中的曲线图

Fig. 3　Diagram for the simulations of the seven body mechanism

表5　七刚体机构仿真结果比较

Table 5　The result comparison of the seven body mechanism

Model	Seven body mechanism		
Integrator	Generalized α	Explicit	BDF
steps	3148	946	568
Time(s)	2.81	1.56	0.48

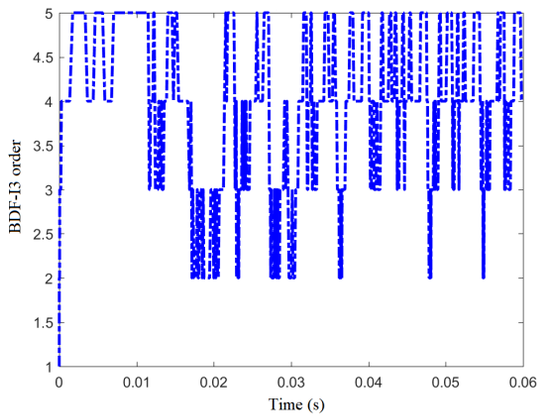

图4　七刚体机构仿真中的BDF阶数曲线图

Fig. 4　BDF order profiles for the simulations of the Seven Body Mechanism

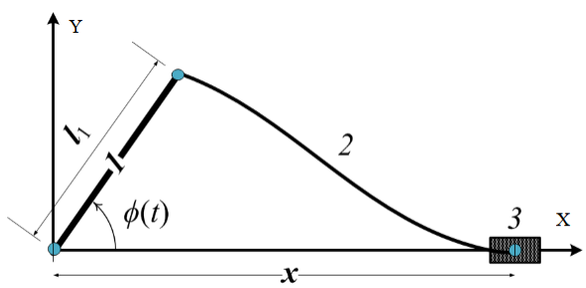

图5　曲柄-滑体机构系统

Fig. 5　Schematic of the crank slider mechanism

器的速度变量计算结果比index-3积分器的结果更准确;而且,index-2的DAE积分器可以解决index-3积分器在加速度和拉格朗日乘子曲线上的尖刺状误差.此外,index-1的DAE积分器计算得到的加速度曲线更光滑,而且可以对它们进行误差控制.其次从计算规模上来说,index-1的方程组规模大于index-2的方程组,更大于index-3的方程组;而其Jacobian矩阵的规模和复杂程度也是同样顺序.因此从计算效率上来说,index-3的DAE积分器大于index-2的DAE积分器,更大于index-1的DAE积分器.最后,从计算鲁棒性来说,大部分时候BDF族的index-2格式积分器优于index-3格式积分器;而index-3的广义α积分器的鲁棒性似乎更佳.

从计算速度上来说,一般认为情况下在同样的精度要求下,BDF方法快于广义α方法.显式积分器不能求解强刚性问题,而根据目前的测试结果,它在弱刚性问题和一些中等刚性问题(刚性< 10^4)的动力学仿真中比广义α方法要快.从程序设计角度来说,显式积分器最容易编程实现,因为无需

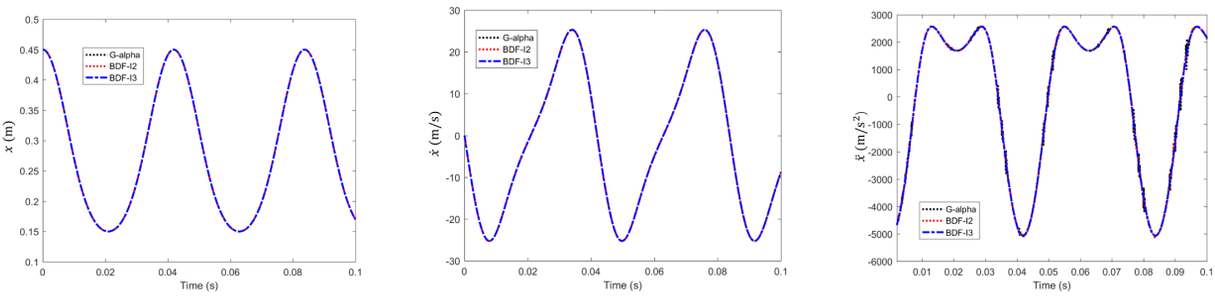

图6 曲柄滑体机构仿真中的曲线图

Fig. 6 Diagram for the simulations of the crank slider mechanism

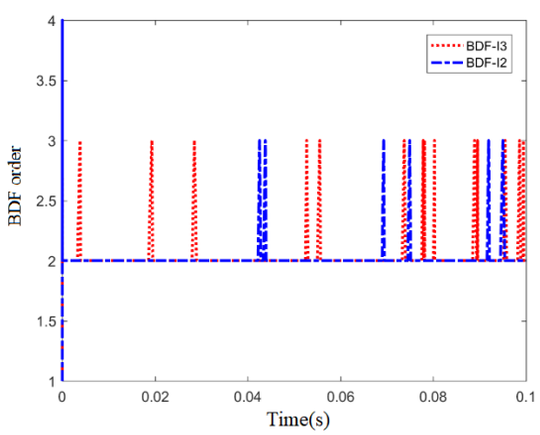

图7 曲柄滑块机构仿真中的BDF阶数曲线图

Fig. 7 BDF order profiles for the simulations of the crank slider mechanism

表6 曲柄滑块机构用各种积分器仿真的效率比较

Table 6 Comparison of simulation efficiency of various integrators for the crank slider mechanism

Integrator	Generalized α	BDF-I3	BDF-I2
Steps	494	436	444
Time(s)	2.38	2.91	3.17

迭代求解非线性方程组;广义 α 族积分器其次,而且稳定性比较好;BDF族积分器的结构最复杂,因为既需要实现变阶变步长,还要特别处理高阶格式的稳定性问题;特别是对于index-3的BDF族积分器还需要特殊处理速度变量的误差估计问题.从计算鲁棒性来说,显式积分器强于隐式积分器,而广义 α 方法优于BDF方法,因为大步长积分可能在连续性较差的问题中会引起计算失败.另外,广义 α 方法除了几个简单的参数调节以外,可以改进的地方很少,而BDF积分器中的 $\mathbf{D}_{n+1}^{k+2}$ 矩阵所含信息量很大,如果能更好地利用可以有效提高其计算性能和稳定性.最后,显式积分器无需迭代求解大规模非线性方程组,在连续性较差的问题和并行仿真上应用潜力似乎更大.

就BDF方法的两种公式来说,一般认为,准定步长公式稍快于变步长公式,但由于后者直接采用原始数据而无需进行插值或者步长转换,在程序设计方便程度和鲁棒性方面优于前者.另外,广义 α 方法的index-2格式尽管可以完全满足速度约束,但在很多实际问题应用中计算效率似乎不够高,值得进一步深入研究.

10 展望

微分代数方程积分器是多体系统动力学软件的核心模块.在上世纪70至90年代,伴随多体系统动力学软件的开发热潮,积分器技术的发展进入了黄金时期.很多著名的研究团队,通过大量的理论分析和数值实验,提出了很多思路来解决动力学仿真中的各种数值问题.这些研究结果也大量被商业软件采用、验证和改进.同时,为了使自己的软件能更有效地求解各种工程问题,各大商业公司也投入成本,展开了大量的测试研究,从工程实践中提炼出了很多需求,并在软件升级中一一解决.目前可以认为,在比较有名的通用型软件中,传统单个积分器的技术已经比较成熟.

这几年,李群积分器和辛积分器的研究引起了很多学者的重视.李群积分器在求解高速旋转问题中有无与伦比的优势,其计算结果的精确性、算法的高效性和鲁棒性得到研究者们的赞赏.辛积分器在长时间仿真的应用中性能卓越.完善这些积分器的功能并开发出高效计算程序是目前正在开展的重要研究课题.

随着硬件技术的进步、计算规模的扩大和学科的交叉,多体系统动力学中的积分器技术也面临着新的挑战和机遇.积分器的计算复杂度一般在 $O(n^2)$ 与 $O(n^4)$ 之间,取决于系统的物理特性,其中 n

为系统的自由度.因此,积分器技术的一个重要研究方向是如何尽可能地降低通用型大规模多体动力学仿真中的计算复杂度.

本文讨论的都是动力学仿真中的初值问题.在最优控制的动力学系统仿真中,往往生成DAE的边值问题,即对某些状态变量,初始值给定;而对其他状态变量,最终值给定.一般多体系统动力学的边值问题要比初值问题复杂得多,其Newton迭代可能不收敛.发展适合大型多体系统动力学边值问题的积分器,也是需要重点研究的方向.

多物理场仿真目前来看是多体系统动力学的重点研究领域,而工业需求为积分器的研究提出了新的研究课题.如何将多体系统动力学中发展出来的积分器和其他学科的积分器整合起来求解多物理场问题,并要保证计算过程的精确性、高效性与鲁棒性,也是积分器技术的重要研究方向.

声明与致谢

多体系统动力学积分器的开发需要理论的指导,更需要详实的细节实现.在工作和学习中,作者阅读和研究过从Gear的DIFSUB到Petzold等人的DASPK等十几种ODE和DAE积分器的代码,从中受益非浅.本文主要以广义 α 族和BDF族积分器为对象,并结合作者实践中的一些体会,尽量系统化地介绍积分器程序开发中的一些关键技术.这些技术也可以用于其他族积分器的实现.作者声明,在本文内绝不包含任何非开源代码的技术细节,以尊重商业软件的专利和知识产权.

本文初稿完成后,北京理工大学胡海岩院士,清华大学任革学教授,青岛大学潘振宽教授,美国同事宋培林博士等阅读了相关内容,并给作者提出了宝贵的修改意见.作者深表感谢,也尽量按照几位前辈的意见作了相应的修改.作者在完成和修改本文过程中,曾就文中很多细节和北京理工大学刘铖研究员和哈工大魏承副教授反复交流讨论,最终才得以定稿,在此一并致谢.另外,还要感谢我的几个学生,杨凯、杨思铭、袁腾飞等,最初本文只是团队内部的简单讨论笔记,各位同学向作者指出了初稿中的一些含糊不清之处,进而帮助作者完善了本文中的一些细节内容.

参　考　文　献

1　Petzold L R. Differential/algebraic equations are not ODEs. *SIAM Journal on Scientific computing*, 1982, 3(3):367~384

2　Van Wyk R. Variable mesh multistep methods for ordinary differential equations. *Journal of Computational Physics*, 1970, 5(2): 244~264

3　Van Bokhoven W M G. Linear implicit differentiation formulas of variable steps and order. *IEEE Transactions on Circuits and Systems*, 1975, 22(2):109~115

4　Jackson K R, Sacks Davis R. An alternative implementation of variable stepsize multistep formulas for stiff ODEs. *ACM Transactions on Mathematical Software*, 1980, 6(3):295~318

5　Gear C W. Simultaneous numerical solution of differential algebraic equations. *IEEE Transactions on circuit theory*, 1971, 18(1):89~95

6　Orlandea N, Chace M A, Calahan D A. A sparsity-oriented approach to the design of mechanical systems, Part I and II. *Paper NO*. 76-DET-19 and 76-DET-20, *presented at ASME Mechanical Conferences*, Montreal, *Quebec*, Canada, *October*, 1976

7　Brenan K E, Engquist B. Backward differentiation approximations of nonlinear differential/ algebraic systems. *Mathematics of Computation*, 1988, 51(184):659~676

8　Brenan K E, Campbell S L, Petzold L R. Numerical solution of intial value problems in differential algebraic equations, 2nd edition, *SIAM Classics in applied Mathematics*, 1997

9　Petzold L R. Numerical solution of differential-algebraic equations in mechanical systems simulation. Physica D: Nonlinear Phenomena, 1995, 60:269~279

10　Loetstedt P L, Petzold L R. Numerical solution of nonlinear differential equations with algebraic constraints I: Convergence Results for Backward Differentiation Formulas. *Mathematics of Computation*, 1986, 46(174): 491~516

11　Petzold L R, Loetstedt P. Numerical solution of nonlinear differential equations with algebraic constraints II: Practical Implementations. *SIAM Journal on Scientific computing*, 1986, 7(3):720~733

12　Gear C W, Gupta G A, Leimkuhler B. Automatic integration of Euler-lagrange equations with constraints. *Journal of Computational and Applied Mathematics*, 1985, 12-13: 77-90

13　Skelboe S. The control of order and step length for backward difference methods. *BIT Numerical Mathematics*, 1977, 17: 269-279

14　Stewart K. Avoiding stability-induced inefficiencies in BDF methods. *Journal of Computational and Applied Mathematics*, 1990, 29(3):357~367

15 Arnold M, Bruels O. Convergence of the generalized-alpha scheme for constrained mechanical systems. *Multibody System Dynamics*, 2007, 18(2):185~202

16 Negrut D, Rampalli R, Ottarsson G, Sajdak A. On an implementation of the Hilber-Hughes-Taylor method in the context of index 3 differential-algebraic equations of multibody dynamics. *Journal of Computational Nonlinear Dynamics*, 2007, 2(1):73~85

17 Chung J, Hulbert G M. A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized alpha method. *ASME Journal of Applied Mechanics*, 1993, 60:371~375

18 Hilber H M, Hughes T J R, Taylor R L. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering & Structural Dynamics*, 1977, 5(3):283~292

19 Negrut D, Jay L O, Khude N. A discussion of low-order numerical integration formulas for rigid and flexible multibody dynamics. *Journal of Computational Nonlinear Dynamics*, 2008, 4(2):149~160

20 Hairer E, Roche L, Lubich C. The numerical solution of differential-algebraic systems by Runge-Kutta methods. *SpringerVerlag*, *Berlin Heidelberg*, 1989

21 Hairer E, Wanner G. Solving ordinary differential equations ii: stiff and differential-algebraic problems. 2nd Edition, Springer-Verlag, *Berlin Heidelberg*, 1996

22 Roche L. Implicit Runge-Kutta methods for differential algebraic equations. *SIAM Journal on Numerical Analysis*, 1989, 26(4): 963~975

23 Jay L. Convergence of Runge-Kutta methods for differential-algebraic systems of index 3. *Applied Numerical Mathematics*, 1995, 17:97~118

24 Jay L. Convergence of a Class of Runge-Kutta methods for differential-algebraic systems of index 2. *BIT Numerical Mathematics*, 1993, 33:137~150

25 Skvortsov L M. Runge-Kutta collocation methods for differential-algebraic equations of indices 2 and 3. *Computational Mathematics and Mathematical Physics*, 2012, 52(10):1373~1383

26 Negrut D, Haug E J, German H C. An implicit Runge-Kutta method for integration of differential algebraic equations of multibody dynamics. *Multibody System Dynamics*, 2003, 9(2):121~142

27 Skvortsov L M. Diagonally implicit Runge-Kutta methods for differential-algebraic equations of indices 2 and 3. *Computational Mathematics and Mathematical Physics*, 2010, 50(6):1047~1059

28 Verwer J H, Hundsdorfer W H, Sommeijer B P. Convergence properties of the Runge-Kutta-Chebyshev method. *Numerische Mathematik*, 1990, 57:157~178

29 Burrage K, Petzold L R. On order reduction for Runge-Kutta Methods to differential/algebraic systems and to stiff systems of ODEs. *SIAM Journal on Numerical Analysis*, 1990, 27(2):447~456

30 Skvortsov L M. How to avoid accuracy and order reduction in Runge-Kutta methods as applied to stiff problems. *Computational Mathematics and Mathematical Physics*, 2017, 57(7):1126~1141

31 Baumgarte J. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Method in Applied Mechanics and Engineering*, 1972, 1(1):1~16

32 Shampine L F. Evaluation of implicit formulas for the solutions of ODEs. *BIT Numerical Mathematics*, 1979, 19: 495~502

33 Ostermeyer G P. On Baumgarte stabilization for diferential algebraic equations. In E. J. Haug and R. C. Deyo, *editors*, *Real-Time Integration Methods for Mechanical System Simulation*, *1990*, 193~207

34 Park K C, Chiou J C. Stabilization of computational procedures for constrained dynamical systems. *Journal Guidance Control and Dynamics*, 1988, 11(4):365~370

35 Ascher U M, Chin H, Petzold L R, Reich S. Stabilization of constrained mechanical systems with DAEs and invariant manifolds. *ACSE Journal Engineering Mechanics*, 1994, 23(2):135~157

36 Gear C W. Towards explicit methods for differential algebraic equations. *BIT Numerical Mathematics*, 2005, 46: 505~514

37 Braun D J, Goldfarb M. Simulation of constrained mechanical systems: Part I: an equation of motion, and Part II: explicit numerical integration. *ASME Journal of Applied Mechanics*, 2012, 79(4): 041017~041018

38 Bauchau O A, Laulusa A. Review of contemporary approaches for constraint enforcement in multibody systems. *Journal of Computational Nonlinear Dynamics*, 2008, 3(1):011005

39 Abdulle A, Medovikov A A. Second order Chebyshev methods based on orthogonal polynomials. *Numerische Mathematik*, 2001, 90:1~18

40 Abdulle A. Fourth order Chebyshev methods with recurrence relation. *SIAM Journal on Scientific computing*, 2002, 25:2041~2054

41 Haug E J, Yen J. Generalized coordinate partitioning methods for numerical integration of differential-algebraic equations of dynamics. In E. J. Haug and R. C. Deyo, *editors*, *Real-Time Integration Methods for Mechanical System Simulation*, *1990*, 97~114

42 Potra F A, Rheinbold W C. On the numerical solution of

Euler-Lagrange equations. *Mechanics of Structures and Machines*, 1991, 19(1):1~18

43 Rabier P J, Rheinbold W C. On the numerical solution of Euler-Lagrange equations. *SIAM Journal on Numerical Analysis*, 1995, 32(1):318~329

44 Shabana A A, Hussein B A. A two-loop sparse matrix numerical integration procedure for the solution of differential/algebraic equations: application to multibody systems. *Journal of Sound and Vibration*, 2009, 327(3-5): 557~563

45 Hussein B A, Shabana A A. Sparse matrix implicit numerical integration of the stiff differential/algebraic equations: implementations. *Nonlinear Dynamics*, 2011, 65: 369~382

46 Aboubakr A K, Shabana A A. Efficient and robust implementation of the TLISMNI method. *Journal of Sound and Vibration*, 2015, 353(29):220~242

47 Jay O L, Negrut D. A second order extension of the generalized alpha method for constrained systems in mechanics. In: *Bottasso*, C.(L. (eds) *Multibody Dynamics. Computational Methods in Applied Sciences*, 12. Springer, *Dordrecht*, 2009)

48 Gear C W, Petzold L R. ODE Methods for the solution of differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 1984, 21(4):716~728

49 Shampine L F, Reichelt M W. The MATLAB ODE suite. *SIAM Journal on Scientific computing*, 1997, 18(1): 1~22

50 Gear C W. Numerical initial value problems in ordinary differential equations, *Prentice-Hall*, *Englewood Cliffs*, *NJ*, 1971

51 Hindmarsh A C. LSODE and LSODI: two new initial value ordinary differential equation solvers. *ACM SIGNUM Newsletters*, 1980, 15:10~11

52 Petzold L R. A description of DASSL: A differential/algebraic system solver. In: *The 10th IMACS World Congress on System Simulation and Scientific Computation*, 1982

53 Gear C W, Tu K W. The effect of variable mesh size on the stability of multistep methods. *SIAM Journal on Numerical Analysis*, 1974, 11(5):1025~1043

54 Gear C W, Watanabe D S. Stability and convergence of variable order multistep methods. *SIAM Journal on Numerical Analysis*, 1974, 11(5):1044~1058

55 Moore P K, Petzold L R. A stepsize control strategy for stiff systems of ordinary differential equations. *Applied Numerical Mathematics*, 1994, 15(4):449~463

56 Shampine L F. Implementations of implicit formulas for the solutions of ODEs. *SIAM Journal on Scientific computing*, 1980, 1(1):103~118

57 Gear C W, Osterby O. Solving ordinary differential equations with discontinuities. *ACM Transactions on Mathematical Software*, 1984, 10(1):23~44

58 Mao G, Petzold L R. Efficient integration over discontinuities for differential-algebraic systems. *Computers & Mathematics with Application*, 2002, 43(1-2): 65~79

59 MSC. Software Corporation, *ADAMS 2019 Online Help. MSC. Software Corporation*, *Ann Arbor*, *Michigan*, USA

60 Schiehlen W. Multibody Systems Handbook, Springer-Verlag, *Berlin Heidelberg*, 1990

IMPLEMENTATION DETAILS OF DAE INTEGRATORS FOR MULTIBODY SYSTEM DYNAMICS *

Ren Hui† Zhou Ping

(*School of Astronautics*, *Harbin Institute of Technology*, *Heilongjiang* 150001, *China*)

Abstract The governing equations of multibody system (MBS) dynamics are differential algebraic equations (DAEs), and those equations must be solved by DAE integrators. In this work, the details of two families of DAE integrators, the backward difference formula (BDF) family and the generalized α method family, were amply described and discussed. At least two formulations were provided in the description of each family of integrators; the numerical schemes of each formulation for index-3, index-2 and index-1 DAEs were depicted in details; and the accuracy, efficiency, and robustness of all those integrators are discussed. Adaptive step-size techniques were implemented in all the integrators, based on error estimations, and adaptive order technique is implemented in the BDF integrators. The index-3 integrators were usually more efficient than the other schemes, but the calculated accelerations and Lagrange multipliers in those index-3 integrators endure spike phenomena, while the corresponding results in index-2 and index-1 schemes were usually smooth. Critical computational procedures, such as the initial condition analysis and reutilizations of Jacobian matrices, were discussed. Moreover, for BDF integrators, an error filtering method was introduced to improve the error estimations

for velocity variables in the index-3 integrators, and several stabilization techniques were introduced to solve the issues caused by high order (⩾ 3) BDF schemes, which were not absolutely stable. Furthermore, explicit DAE integrators were briefly introduced, which do not require iterations and might be significant in some specific applications. Two benchmark examples were calculated using those integrators, and the pros and cons of each integrator were depicted and discussed. Typical integrator algorithms were provided in details in the appendix, which can be directly adopted to practical problems.

Key words differential algebraic equation, integrator, backward difference formula, generalized α method, explicit DAE integrator

Received XXXX-XX-XX, Revised XXXX-XX-XX

* the project supported by the national natural science foundation of china (11772101)

† Corresponding author renhui@hit.edu.cn

附录

附录中给出两个有代表性的隐式积分器的简单版本的算法细节,分别是 index-3 的广义 α 积分器和变步长公式的 index-3 的 BDF 积分器.作者相信,读者可以类似地写出 index-2 的广义 α 积分器、变步长公式的 index-2 的 BDF 积分器、准定步长公式的 index-3 和 index-2 的 BDF 积分器,还可以很容易开发出 index-1 的各类积分器.这两个积分器的共同特征是都需要初值处理程序,并且其 Newton 迭代过程有相似之处,因此将这两部分单独拿出来作为子程序.另外,程序中忽略了由于间断性导致积分器中断以及重启的步骤,这在商业软件级别的积分器中是非常关键的.最后,关于算法格式符号书写的几点说明:

1 ← 用来表示赋值语句;

2. = 用来判断两个值相等.

算法 1 初始条件求解(IC Analysis):

输入:初始广义坐标和速度的估计值 $\boldsymbol{q}_0$ 和 $\dot{\boldsymbol{q}}_0$;

可选输入:对角权重矩阵 $\mathbf{W}^q$ 和 $\mathbf{W}^v$,若没有用户相应输入,计算时缺省取为单位矩阵;

输出:满足约束方程的初始广义坐标 $\boldsymbol{q}_0$、广义速度 $\dot{\boldsymbol{q}}_0$、广义加速度 $\ddot{\boldsymbol{q}}_0$ 和拉格朗日乘子 $\boldsymbol{\lambda}_0$;

计算步骤:

1. 冗余约束分析:对 $\boldsymbol{C}_q(\boldsymbol{q}_0, 0)$ 进行选主元 LU 分解,剔除三角形矩阵中对角线元过小的行对应的约束;

2. 以 $\boldsymbol{q}_0$ 为初值 $\boldsymbol{q}$,迭代求解方程(7),然后用求出来 $\boldsymbol{q}$ 的更新 $\boldsymbol{q}_0$;

3. 求解方程(9),然后用求出的 $\dot{\boldsymbol{q}}$ 更新 $\dot{\boldsymbol{q}}_0$;

4. 求解方程(10),得到初始加速度 $\ddot{\boldsymbol{q}}_0$ 和拉格朗日乘子 $\boldsymbol{\lambda}_0$.

注 1 初始值修正对 DAE 的积分非常关键,但有时也会带来问题.例如对两个固定连接的刚体,如果修改其中一个刚体的广义坐标,但不同时修改另外一个刚体的相应坐标,用以上算法得出的结果与实际情形的关系不可预料.因此,在初始条件求解中非常关键的一步是利用先验知识,如果确定不需要修改某些变量,一定要相应把所要修改变量在权重矩阵中对应的对角元分量放大为 10^6 或更高,才能有效保证结果的可靠性.

算法 2 积分器中的拟牛顿迭代算法

输入:近似 Jacobian 矩阵 $\hat{\mathbf{J}}$ 的 LU 分解;松弛因子 υ;初始值估计 $\overset{\circ}{\boldsymbol{x}}$;NeedJac 指标以及误差限

输出:$\boldsymbol{x}$ 的更新量 $\boldsymbol{\delta}$, 全部或部分分量的迭代收敛速度 σ 及收敛性判定结果

计算过程:设置最大迭代次数 MaxIter ← 25

令修正量的初值 $\boldsymbol{\delta} \leftarrow 0$, 置当前 Jacobian 矩阵在迭代中的使用次数 $nj \leftarrow 0$,进入以下循环

FORIter ← 1 to MaxIter

1. **IF** NeedJac,则重新生成 $\hat{\mathbf{J}} = \dfrac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}(\overset{\circ}{\boldsymbol{x}} + \boldsymbol{\delta})$ 进行 LU 分解,且置 $nj \leftarrow 0$ 和 $\upsilon \leftarrow 0$;

2. 生成非线性方程组 $\boldsymbol{F}(\overset{\circ}{\boldsymbol{x}} + \boldsymbol{\delta})$,并计算 $\Delta\boldsymbol{\delta} \leftarrow -\upsilon\hat{\mathbf{J}}^{-1}\boldsymbol{F}(\overset{\circ}{\boldsymbol{x}} + \boldsymbol{\delta})$;

3. 更新 $nj \leftarrow nj + 1$ 及 $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \Delta\boldsymbol{\delta}$;

4. **IF** $\|\Delta\boldsymbol{\delta}\| < 10^{-4}$,迭代收敛,跳出循环;

5. 进行收敛性检测

IF $nj = 1$,设置 $\sigma \leftarrow 0$ 以及 NeedJac ← **FALSE**;

ELSEIF $\|\Delta\boldsymbol{\delta}\| > 0.9 \cdot OldNorm$,设置 NeedJac ← **TRUE**;

ELSE

a) 计算 $\sigma \leftarrow \max(0.5\sigma, \ /\!/\ \Delta\boldsymbol{\delta}\ /\!/\ /OldNorm)$,以及 $err \leftarrow \dfrac{\sigma}{1-\sigma}\ /\!/\ \Delta\boldsymbol{\delta}\ /\!/$;

b) **IF** $err < 0.001$,迭代收敛,跳出循环;

c) **IF** $nj \geqslant 5$, NeedJac ← **TRUE**;

d) **IF** $\mathrm{err} \cdot \sigma^{5-nj} > 0.5$, NeedJac ← **TRUE**;

6. 更新 $OldNorm \leftarrow /\!/ \ \Delta\boldsymbol{\delta} \ /\!/$.

END

收敛性判定：

IF Iter = MaxIter 并且 $err > 0.33$，判断迭代不收敛；否则判定迭代收敛；

注2 本迭代子程序只是用来在积分器的每一时间步中求解非线性代数方程组.一般来说，预测值已经相当精确，只需要很少几步迭代即可收敛，而且不同时间步的 Jacobian 可以复用，以最大可能地节省计算量；但在少数情形，可能遇到不收敛的问题，因此需要重新生成 Jacobian 进行计算；在极少数情形，迭代可能很久也不能收敛，通常意味着遇到了不连续性问题，需要特殊处理，判定为本次迭代失败即可.

算法 3 自适应步长广义 $\boldsymbol{\alpha}$ 方法：

输入：动力学方程组(1)，初始估计的 $\boldsymbol{q}_0$ 和 $\dot{\boldsymbol{q}}_0$，计算时间区间 $[t_0, t_e]$，积分参数 ρ 以及误差限

输出：一系列自适应的仿真时间节点 $t_0 < t_1 < \cdots < t_N = t_e$，和每个时间节点 t_n 对应的状态量 $\boldsymbol{q}_n$、$\dot{\boldsymbol{q}}_n$、$\ddot{\boldsymbol{q}}_n$ 和 $\boldsymbol{\lambda}_n$，其中每一步计算的误差满足误差条件要求.

计算过程：

1. 用算法 1 求解初值问题，得出合适的初值 $\boldsymbol{q}_0$、$\dot{\boldsymbol{q}}_0$、$\ddot{\boldsymbol{q}}_0$ 和 $\boldsymbol{\lambda}_0$，并取 $\boldsymbol{a}_0 \leftarrow \ddot{\boldsymbol{q}}_0$；

2. 利用公式(18)和(27)计算出参数 α_m、α_f、β、γ 和 η；

3. 设置 $n \leftarrow 0$，NeedJac $\leftarrow$ **TRUE**，done $\leftarrow$ **FALSE** 以及初始步长 $h \leftarrow 10^{-4} \times (t_e - t_n)$；

4. **IF** $t_n + h > t_e$，$h \leftarrow t_e - t_n$，done $\leftarrow$ **True**；

5. 设置步长失败标记 StepFail $\leftarrow$ **FALSE**，然后运行以下计算内循环直至本步计算成功.

a) 用方程(30)和(31)算出预测值 $\boldsymbol{q}_{n+1}^{(0)}$、$\dot{\boldsymbol{q}}_{n+1}^{(0)}$、$\ddot{\boldsymbol{q}}_{n+1}^{(0)}$、$\boldsymbol{a}_{n+1}^{(0)}$ 和 $\boldsymbol{\lambda}_{n+1}^{(0)}$；并将得到的表达式(29)和(31)代入离散动力学方程组(32)；

b) **IF** NeedJac = **TRUE**，用预测值生成 Jacobian 矩阵(33)并进行 LU 分解；

c) 用算法 2 结合方程(32)求解未知量 $(\boldsymbol{x}^T, \boldsymbol{z}^T)^T$；

d) 如果算法 2 迭代失败，进行不连续性检测，并重启积分器；

e) 更新 $\boldsymbol{q}_{n+1} \leftarrow \boldsymbol{q}_{n+1}^{(0)} + \beta h^2 \boldsymbol{x}$、$\dot{\boldsymbol{q}}_{n+1} \leftarrow \dot{\boldsymbol{q}}_{n+1}^{(0)} + \gamma h \boldsymbol{x}$、$\ddot{\boldsymbol{q}}_{n+1} \leftarrow \ddot{\boldsymbol{q}}_{n+1}^{(0)} + \eta \boldsymbol{x}$、$\boldsymbol{a}_{n+1} \leftarrow \boldsymbol{a}_{n+1}^{(0)} + \boldsymbol{x}$ 以及 $\boldsymbol{\lambda}_{n+1} \leftarrow \boldsymbol{\lambda}_{n+1}^{(0)} + \boldsymbol{z}^T$；

f) 用公式(34)计算出 Ξ，并更新 $h_{new} \leftarrow \min\left(\frac{0.9}{\Xi}, 2\right) \cdot h$；

g) Jacobian 复用性判定

IF $\sigma \frac{h_{new}}{h} + \left|\frac{h_{new}}{h} - 1\right| \leqslant \frac{1}{3}$，设置松弛因子 $\upsilon \leftarrow \frac{2h_{new}}{h_{new} + h}$，以及 NeedJac $\leftarrow$ **FALSE**；

ELSE 设置 Newton 迭代松弛因子 $\upsilon \leftarrow 1$，以及 NeedJac = **TRUE**

h) 进行内循环收敛性判断，即判断条件 $\Xi \leqslant 1$ 是否满足？

如果满足，存储更新 $t_{n+1} \leftarrow t_n + h$、$\boldsymbol{q}_{n+1}$、$\dot{\boldsymbol{q}}_{n+1}$、$\ddot{\boldsymbol{q}}_{n+1}$ 和 $\boldsymbol{\lambda}_{n+1}$，并跳出内循环；

如果不满足

i. 判断本步迭代中是否首次失败 18652331899

IF StepFail = TRUE，$h \leftarrow h/(2\Xi)$，NeedJac = **TRUE**；

ELSE 设置 StepFail $\leftarrow$ **TRUE**，$h \leftarrow h_{new}$ 及 done $\leftarrow$ **FALSE**

ii. 返回(a)，重启内循环

6 仿真中止性判断

IF done，仿真结束，停止；

更新 $n \leftarrow n + 1$ 和 $h \leftarrow h_{new}$，返回 4 继续计算.

注3 不连续性检测程序比较复杂，此处暂忽略.

注4 测试表明，为了达到同样的积分精度，广义 α 方法需要比 BDF 方法更严格的局部误差限.实践中

发现,广义α方法取相对误差限10^{-5}的整体仿真效果类似于BDF方法取相对误差限10^{-3}的结果.

算法4 自适应步长和阶数的Index-3变步长公式的BDF积分器:

输入:动力学方程组(1),初始估计的$\boldsymbol{q}_0$和$\dot{\boldsymbol{q}}_0$,计算时间区间$[t_0, t_e]$以及局部误差限

输出:一系列自适应的仿真时间节点$t_0 < t_1 < \cdots < t_N = t_e$,以及对应时间节点的状态变量$\boldsymbol{q}_n$、$\dot{\boldsymbol{q}}_n$、$\ddot{\boldsymbol{q}}_n$和$\boldsymbol{\lambda}_n$,其中每一步计算的误差满足误差条件要求

计算过程:

1. 用算法1求解初值问题,得出合适的初值$\boldsymbol{q}_0$、$\dot{\boldsymbol{q}}_0$、$\ddot{\boldsymbol{q}}_0$和$\boldsymbol{\lambda}_0$;

2. 设置$n \leftarrow 0$、$k \leftarrow 1$、done $\leftarrow$ **FALSE**以及初始步长$h \leftarrow 10^{-5} \times (t_e - t_n)$;

3. 设置初始差分矩阵$\boldsymbol{Q}_n^k \leftarrow h\dot{\boldsymbol{q}}_0$、$\boldsymbol{V}_n^k \leftarrow h\ddot{\boldsymbol{q}}_0$以及时间向量$\boldsymbol{\tau}_{k+1}^n \leftarrow (h, 2h)$;

4. **IF**$t_n + h > t_e$, $h \leftarrow t_e - t_n$, done $\leftarrow$ **True**;

5 设置步长失败标记StepFail $\leftarrow$ **FALSE**,然后运行以下计算内循环直至本步计算成功

a) 利用公式(66)计算$\boldsymbol{\tau}_{k+2}^{n+1}$,进而用公式(58)(64)和(67)生成$\boldsymbol{\omega}_k$、$\boldsymbol{c}_k$和$c_e^k$,同时由(71)得出$\hat{h}$;

b) 从公式(73)中得出$\boldsymbol{q}_{n+1}$、$\boldsymbol{v}_{n+1}$、$\dot{\boldsymbol{q}}_{n+1}$和$\dot{\boldsymbol{v}}_{n+1}$的初始估计,然后将公式(74)、(75)和(77)中的表达式代入离散格式方程组(78);

c) 判断条件$(n > 0)$和$(\sigma \dfrac{\hat{h}}{\hat{h}_{old}} + \left|\dfrac{\hat{h}}{\hat{h}_{old}} - 1\right| \leqslant \dfrac{1}{3})$是否同时满足

如果满足,设置松弛因子$\upsilon \leftarrow \dfrac{2\hat{h}}{\hat{h} + \hat{h}_{old}}$,以及NeedJac $\leftarrow$ **FALSE**;

如果不满足,设置松弛因子$\upsilon \leftarrow 1$以及NeedJac=**TRUE**;

d) 用算法2迭代求出方程组(78),得到更新量$\boldsymbol{\delta}$和$\boldsymbol{\lambda}$,进而利用公式(76)计算出$\boldsymbol{\varepsilon}$,并通过求解(80)计算出投影分量$\hat{\boldsymbol{\varepsilon}}$;同时用$\boldsymbol{q}_{n+1}$的更新量$\boldsymbol{\delta}$和$\boldsymbol{v}_{n+1}$的投影更新量$\hat{\boldsymbol{\varepsilon}}$进行迭代收敛性判断;

e) 如果算法2迭代失败,进行不连续性检测,并重启积分器;

f) 从$\nabla^{k+1}\boldsymbol{v}_{n+1}$(即$\boldsymbol{\varepsilon}$)、$\nabla^{k+2}\boldsymbol{v}_{n+1}$(即$\boldsymbol{\varepsilon} - \nabla^{k+1}\boldsymbol{v}_n$)和$\nabla^k\boldsymbol{v}_{n+1}$(即$\boldsymbol{\varepsilon} + \nabla^k\boldsymbol{v}_n$)出发,用投影修正公式(80)计算出对应投影分量$\hat{\nabla}^{k+1}\boldsymbol{v}_{n+1}$、$\hat{\nabla}^{k+2}\boldsymbol{v}_{n+1}$和$\hat{\nabla}^k\boldsymbol{v}_{n+1}$;接着对$j = k-1, k, k+1$计算出$e^j = \left\|\left(\nabla^{j+1}\boldsymbol{q}_{n+1};\ \hat{\nabla}^{j+1}\boldsymbol{v}_{n+1}\right)\right\|$,进而计算方程(85)中的$h_j$,并限定$h_j \quad \min(h_j, h \cdot f_j)$;

g) 设置incr $\leftarrow \left(e^k \leqslant e^{k-1}\right)\&\&(e^{k+1} \leqslant e^k)$以及decr $\leftarrow$ **FALSE**;若$k \geqslant 3$,令decr $\leftarrow \left(e^k > s_k e^{k-1}\right)$;

h) 判断后验误差是否满足误差条件要求,即$c_e^k e^k \leqslant 1$

如果满足,存储更新$t_{n+1} \leftarrow t_n + h$、$\boldsymbol{q}_{n+1}$、$\dot{\boldsymbol{q}}_{n+1}$、$\ddot{\boldsymbol{q}}_{n+1}$和$\boldsymbol{\lambda}_{n+1}$,进行下一步计算准备;

如果不满足,设置done $\leftarrow$ **FALSE**;更新$\hat{h}_{old} \leftarrow \hat{h}$;并且

i. 判断是否在本步迭代中首次失败

IF StepFail, $h \leftarrow h_k/2$; **IF** decr,需要降阶,即取$k \leftarrow k-1$及$h \leftarrow h_{k-1}/2$;

ELSE 设置$h \leftarrow h_k$; **IF** decr,需要降阶计算,即取$k \leftarrow k-1$及$h \leftarrow h_{k-1}$;

ii. 设置StepFail $\leftarrow$ **TRUE**;返回(a),重启内循环

i) IF $n = 0$,更新$\boldsymbol{Q}_{n+1}^{k+1}$和$\boldsymbol{V}_{n+1}^{k+1}$,并令$n \leftarrow n+1$,不变阶不变步长,直接进入下一步的起始步骤4;

j) 利用公式(59)递归更新$\boldsymbol{Q}_{n+1}^{k+2}$和$\boldsymbol{V}_{n+1}^{k+2}$,并从$k-1$阶到$k+1$阶的后验误差,选择下一步的阶数$k_{new}$和步长$h_{new}$如下(因此本算法每步至多升、降1阶):

i. 暂设定$k_{new} \leftarrow \arg\max(h_k, h_{k-1}, h_{k+1})$和$h_{new} \leftarrow h_{knew}$,并更新$\hat{h}_{old} \leftarrow \hat{h}$;

ii. **IF** incr &&$k < 5$,下一步可能要升阶,即若$h_{new} < h_{k+1}$,则设置$k_{new} \leftarrow k+1$及$h_{new} \leftarrow h_{k+1}$

iii. **IF** decr,下一步需要降阶,即令$k_{new} \leftarrow k-1$和$h_{new} \leftarrow h_{k-1}$,跳到6;

6 仿真中止性判断

IF done,仿真结束,停止;

更新 $n \leftarrow n+1$、$k \leftarrow k_{new}$ 和 $h \leftarrow h_{new}$，并返回4开始下一时间步的计算.

注5 BDF的速度误差修正以及选阶选步长算法有很多种不同的选择.本算法中采用的策略是:用投影滤波方法进行速度误差修正;结合GEAR的DIFSUB程序中的策略与Skelboe和DASSL的高阶稳定性判据来进行阶数和步长的自适应选择.其优点是程序设计简单,缺点是失败率略偏高(10%左右).为了进一步提高该积分器的仿真效率,可以同样从这三个方面来改进:速度误差的修正算法、结合高阶稳定性判据的选阶和选步长算法、以及不连续性检测和处理策略.