第41卷第6期 计 算 力 学 学 报 Vol. 41, No. 6
2024年12月 Chinese Journal of Computational Mechanics December 2024

DOI: 10.7511/jslx20230406003

非线性结构动力方程的自适应步长数值算法

王海波*, 王鸿燊, 纪海潮

(中南大学 土木工程学院,长沙 410075)

摘 要:基于Runge-Kutta法实现对时间步长的自适应选择,研究提高非线性结构动力方程的计算精度。利用Runge-Kutta公式的局部截断误差,得出误差估计值 ξ_{n+1},根据 ξ_{n+1} 的大小自适应调节时间步长的大小,为算法提供一个判断语句,其能使算法流程图更加多样性。将该思想应用于经典Runge-Kutta算法和精细Runge-Kutta算法中,得到自适应步长的经典Runge-Kutta算法和精细Runge-Kutta算法,使算法的时间步长依赖于给定的每步误差限值,提高计算精度,数值算例论证了本文方法的有效性。

关键词:非线性动力方程;自适应步长;精细积分法;Runge-Kutta法

中图分类号:O322 **文献标志码**:A **文章编号**:1007-4708(2024)06-1045-08

1 引 言

1994年,钟万勰[1]放弃了传统数值积分法中的差分格式,效仿哈密顿体系对偶变量的引入,提出了精细积分法,为非线性系统的时程分析提供了新的思路方法;赵秋玲[2]运用精细积分法将非线性系统线性化求解;张森文等[3]将辛普生积分法引入精细积分法中,得出求解系统响应的状态方程直接积分法;裘春航等[4]用 j 次多项式来近似方程的非线性部分,借助分段积分法求解;汪梦甫等[5]将高斯积分方法与精细积分法中的指数矩阵运算技巧相结合,建立精细积分法的更新形式及计算过程;陈伯望等[6]将精细积分法与预估-校正 Adams-Bashforth-Moulton 多步法相结合,该方法需要对表头进行计算;李炜华等[7]提出了求解非线性结构动力方程的辛时间子域法,该方法不必对状态矩阵求逆,无需计算高阶导数,具有较高计算精度和稳定性;王海波等[8]结合广义精细积分法和预估-校正法,提出用于非线性动力分析的广义精细积分法;王海波等[9]结合泰勒级数和广义精细积分法,避免对状态方程求逆,提出一种对线性动力分析的通用积分格式;王永等[10]将精确积分法和微分求积法结合起来,在计算过程中对未知值进行估计,并提出了一种高效的精确积分单步法,避免了状态矩阵的求逆;梅树立等[11]利用龙贝格积分法的自适应性解决时间步长的选择问题,提出了结构非线性动力方程的自适应精细积分算法;李健等[12]基于积分区间逐次半分的思想实现了任意时间步长的自适应求积;王海波等[13]基于 adams 公式对时间步长进行自适应调整,通过合理设置误差限,提高计算精度和适用性。

本文基于Runge-Kutta法实现对时间步长的自适应,使得时间步长依赖于给定的误差限值。利用经典Runge-Kutta算法和精细Runge-Kutta算法,得到自适应步长的经典Runge-Kutta算法和精细Runge-Kutta算法。算例证明了该方法的有效性。

2 误差估计表达式

一阶微分方程

$$\dot{y}=f(x,y) \tag{1}$$

以经典 R-K 法(四阶 Runge-Kutta 法)为例

$$\begin{cases} K_1=f(x_n,y_n) \\ K_2=f(x_n+h/2,y_n+(h/2)K_1) \\ K_3=f(x_n+h/2,y_n+(h/2)K_2) \\ K_4=f(x_n+h,y_n+hK_3) \\ y_{n+1}=y_n+(h/6)(K_1+2K_2+2K_3+K_4) \end{cases} \tag{2}$$

收稿日期:2023-04-06; **修改稿收到日期**:2023-05-28.

基金项目:国家自然科学基金(50908230)资助项目.

作者简介:王海波*(1974-),男,博士,副教授(E-mail:haibarg@163.com).

引用本文:王海波,王鸿燊,纪海潮.非线性结构动力方程的自适应步长数值算法[J].计算力学学报,2024,**41**(6):1045-1052.

WANG Hai-bo, WANG Hong-shen, JI Hai-chao. Adaptive step numerical algorithm for nonlinear structural dynamic equations [J]. *Chinese Journal of Computational Mechanics*, 2024, **41**(6): 1045-1052.

式(2)的局部截断误差约为 $O(h^5)$，以时间节点 x_n 为起点，先取时间步长为 h，得到关于 $y_{n+1}^{(h)}$（上标 h 表示以 h 为时间步长计算得到的 y_{n+1}）的局部误差表达式，即

$$y(x_{n+1})-y_{n+1}^{(h)}\approx ch^5 \tag{3}$$

式中 $y(x_{n+1})$ 表示真实解，c 为常数。接着取时间步长为 $h/2$，从时间节点 x_n 出发，计算两次到时间节点 x_{n+1}，得到关于 $y_{n+1}^{(h/2)}$（上标 $h/2$ 表示以 $h/2$ 为时间步长计算得到的 y_{n+1}）的局部误差表达式，即

$$y(x_{n+1})-y_{n+1}^{(h/2)}\approx 2c\left(\frac{h}{2}\right)^5 \tag{4}$$

式(3)与式(4)相除得

$$\frac{y(x_{n+1})-y_{n+1}^{(h/2)}}{y(x_{n+1})-y_{n+1}^{(h)}}\approx\frac{1}{16} \tag{5}$$

$$y(x_{n+1})-y_{n+1}^{(h/2)}\approx\frac{1}{15}(y_{n+1}^{(h/2)}-y_{n+1}^{(h)}) \tag{6}$$

误差估计表达式为

$$\xi'(x_{n+1})=\frac{1}{15}(y_{n+1}^{(h/2)}-y_{n+1}^{(h)}) \tag{7}$$

$$\xi(x_{n+1})=\max[\mathrm{abs}(\xi'(x_{n+1}))] \tag{8}$$

$$\overline{y(x_{n+1})}=y_{n+1}^{(h/2)}+\xi'(x_{n+1}) \tag{9}$$

$\xi(x_{n+1})$ 为每一时间步长内误差的估计值，式(9)为此时间步长对 $y(x_{n+1})$ 的预估值。$\xi(x_{n+1})<a$ 为算法程序结构提供了一个判断语句。给定误差限值 a，当 $\xi(x_{n+1})>a$ 时，将时间步长减半，直到满足 $\xi(x_{n+1})<a$。

稳定性判断。当求解方法为单步法，如果式(1)中的 $f(x,y)$ 满足关于 y 的 Lipschitz 条件，方法的局部截断误差 $\varepsilon_n=O(h^{p+1})$，初始值精确。则方法的整体截断误差为 $e_n=O(h^p)$，且当 $p\geqslant 1$ 时，方法收敛。因此经典 R-K 算法稳定，同时通过局部截断误差去判断整体截断误差，本文采用 R-K 法的整体截断误差均为 $O(h^4)$。

3 算法流程

算法流程基于 R-K 算法进行编写，如图 1 所示。当前计算步的计算步长不影响下一计算步的计算步长。将变步长算法的初始时间步长 Δt^* 设置为与定步长算法一样，便于与定步长算法进行对比分析。

Δt^* 设为算法流程(图 1)设置的初始时间步长，其是自适应步长精细积分算法计算问题时时间步长的上限值，所有的时间步长都在其基础上进行缩小，故能为自适应步长精细积分法提供一个最小精度的保证。

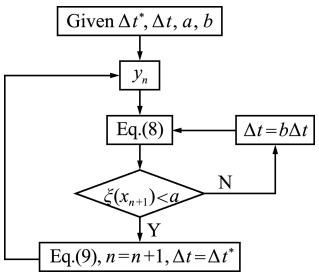


图 1 算法流程
Fig. 1 Date flow diagram

a 是人为设定的误差限值，在给定误差限值的时候，要注意每一计算步误差的大小与采用 R-K 公式的精度相关，对于四阶 R-K 公式，其局部截断误差的大小是时间步长的 5 次方。根据实际问题允许的误差范围，结合采用的 R-K 公式的精度及希望的时间步长的范围，综合考虑。

b 是控制计算步长缩小的参数，在一个计算步长内，函数的阶数越高，该计算步长的误差就越大，就需要缩短计算步长，以降低计算步长内函数的阶数。当一个计算步内的函数阶数越高，b 值越大越能提高计算效率，但当 b 值过大，计算量会增大。一般 b 值取 0.5。

4 经典 R-K 自适应步长算法

4.1 经典 R-K 法

非线性结构动力方程为

$$\dot{v}=Hv+f(v,t) \tag{10}$$

经典 R-K 法公式为

$$\begin{cases}\boldsymbol{K}_1=\boldsymbol{H}\boldsymbol{v}_n+\boldsymbol{f}(\boldsymbol{v}_n,t_n)\\ \boldsymbol{K}_2=\boldsymbol{H}(\boldsymbol{v}_n+(\tau/2)\boldsymbol{K}_1)+\boldsymbol{f}(\boldsymbol{v}_n+(\tau/2)\boldsymbol{K}_1,t_n+\tau/2)\\ \boldsymbol{K}_3=\boldsymbol{H}(\boldsymbol{v}_n+(\tau/2)\boldsymbol{K}_2)+\boldsymbol{f}(\boldsymbol{v}_n+(\tau/2)\boldsymbol{K}_2,t_n+\tau/2)\\ \boldsymbol{K}_4=\boldsymbol{H}(\boldsymbol{v}_n+\tau\boldsymbol{K}_3)+\boldsymbol{f}(\boldsymbol{v}_n+\tau\boldsymbol{K}_3,t_{n+1})\\ \boldsymbol{v}_{n+1}=\boldsymbol{v}_n+(\tau/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\end{cases} \tag{11}$$

4.2 经典 R-K 自适应步长算法

经典 R-K 自适应步长算法详细步骤为

$$\Delta t=\Delta t^* \tag{12}$$

式中 Δt^* 为给定的初始时间步长。

从 t_n 出发，以 Δt 为时间步长，计算 1 次，得到 $v_{n+1}^{(\Delta t)}$

$$\begin{cases}\boldsymbol{K}_1=\boldsymbol{H}\boldsymbol{v}_n+\boldsymbol{f}(\boldsymbol{v}_n,t_n)\\ \boldsymbol{K}_2=\boldsymbol{H}(\boldsymbol{v}_n+(\Delta t/2)\boldsymbol{K}_1)+\boldsymbol{f}(\boldsymbol{v}_n+(\Delta t/2)\boldsymbol{K}_1,t_n+\Delta t/2)\\ \boldsymbol{K}_3=\boldsymbol{H}(\boldsymbol{v}_n+(\Delta t/2)\boldsymbol{K}_2)+\boldsymbol{f}(\boldsymbol{v}_n+(\Delta t/2)\boldsymbol{K}_2,t_n+\Delta t/2)\\ \boldsymbol{K}_4=\boldsymbol{H}(\boldsymbol{v}_n+\Delta t\boldsymbol{K}_3)+\boldsymbol{f}(\boldsymbol{v}_n+\Delta t\boldsymbol{K}_3,t_{n+1})\\ \boldsymbol{v}_{n+1}=\boldsymbol{v}_n+\Delta t/6(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\end{cases} \tag{13}$$

从 t_n 出发，以 $\Delta t/2$ 为时间步长，计算 2 次，得到 $v_{n+1}^{(\Delta t/2)}$

$$\begin{cases}\tau'=\Delta t/2\\ \boldsymbol{K}_1=\boldsymbol{H}\boldsymbol{v}_n+\boldsymbol{f}(\boldsymbol{v}_n,t_n)\\ \boldsymbol{K}_2=\boldsymbol{H}(\boldsymbol{v}_n+(\tau'/2)\boldsymbol{K}_1)+\boldsymbol{f}(\boldsymbol{v}_n+(\tau'/2)\boldsymbol{K}_1,t_n+\tau'/2)\\ \boldsymbol{K}_3=\boldsymbol{H}(\boldsymbol{v}_n+(\tau'/2)\boldsymbol{K}_2)+\boldsymbol{f}(\boldsymbol{v}_n+(\tau'/2)\boldsymbol{K}_2,t_n+\tau'/2)\\ \boldsymbol{K}_4=\boldsymbol{H}(\boldsymbol{v}_n+(\tau'\boldsymbol{K}_3)+\boldsymbol{f}(\boldsymbol{v}_n+\tau'\boldsymbol{K}_3,t_n+\tau')\\ \boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}=\boldsymbol{v}_n+(\tau'/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\\ \boldsymbol{K}_1=\boldsymbol{H}\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)},t_n+\tau')\\ \boldsymbol{K}_2=\boldsymbol{H}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+(\tau'/2)\boldsymbol{K}_1)+\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+(\tau'/2)\boldsymbol{K}_1,\\ \qquad t_n+3\tau'/2)\\ \boldsymbol{K}_3=\boldsymbol{H}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+(\tau'/2)\boldsymbol{K}_2)+\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+(\tau'/2)\boldsymbol{K}_2,\\ \qquad t_n+3\tau'/2)\\ \boldsymbol{K}_4=\boldsymbol{H}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+\tau'\boldsymbol{K}_3)+\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+\tau'\boldsymbol{K}_3,t_{n+1})\\ \boldsymbol{v}_{n+1}^{(\Delta t/2)}=\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+(\tau'/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\end{cases}\tag{14}$$

根据式(7,8)

$$\xi_{n+1}=\max\left\{abs\left[(1/15)(\boldsymbol{v}_{n+1}^{(\Delta t/2)}-\boldsymbol{v}_{n+1}^{(\Delta t)})\right]\right\}\tag{15}$$

当 $\xi_{n+1}\leqslant a$ 时，执行式(16)，计算下一计算步，初始时间步长为 Δt^*。

$$\boldsymbol{v}_{n+1}=\boldsymbol{v}_{n+1}^{(\Delta t/2)}+(1/15)(\boldsymbol{v}_{n+1}^{(\Delta t/2)}-\boldsymbol{v}_{n+1}^{(\Delta t)})\tag{16}$$

当 $\xi_{n+1}>a$ 时，执行式(17)，返回式(13,14)，重新计算该计算步。

$$\Delta t=b\Delta t\tag{17}$$

5　精细 R-K 自适应步长精细积分法

5.1　精细 R-K 法

精细 R-K 算法[14]公式为

$$\begin{cases}\boldsymbol{K}_1=e^{\boldsymbol{H}\Delta t}\boldsymbol{f}(\boldsymbol{v}_n,t_n)\\ \boldsymbol{K}_2=e^{\boldsymbol{H}\Delta t/2}\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_1/2,t_n+\Delta t/2)\\ \boldsymbol{K}_3=e^{\boldsymbol{H}\Delta t/2}\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_2/2,t_n+\Delta t/2)\\ \boldsymbol{K}_4=\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_3,t_{n+1})\\ \boldsymbol{v}_{n+1}=e^{\boldsymbol{H}\Delta t}\boldsymbol{v}_n+(\Delta t/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\end{cases}\tag{18}$$

5.2　精细 R-K 自适应步长精细积分法

精细 R-K 自适应步长算法详细步骤为

$$\Delta t=\Delta t^*\tag{19}$$

式中 Δt^* 为给定的初始时间步长。与经典 R-K 自适应步长精细积分法步骤相同

$$\begin{cases}\boldsymbol{K}_1=e^{\boldsymbol{H}\Delta t}\boldsymbol{f}(\boldsymbol{v}_n,t_n)\\ \boldsymbol{K}_2=e^{\boldsymbol{H}\Delta t/2}\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_1/2,t_n+\Delta t/2)\\ \boldsymbol{K}_3=e^{\boldsymbol{H}\Delta t/2}\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_2/2,t_n+\Delta t/2)\\ \boldsymbol{K}_4=\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_3,t_{n+1})\\ \boldsymbol{v}_{n+1}^{(\Delta t)}=e^{\boldsymbol{H}\Delta t}\boldsymbol{v}_n+(\Delta t/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\end{cases}\tag{20}$$

$$\begin{cases}\tau'=\Delta t/2\\ \boldsymbol{K}_1=e^{\boldsymbol{H}\tau'}\boldsymbol{f}(\boldsymbol{v}_n,t_n)\\ \boldsymbol{K}_2=e^{\boldsymbol{H}\tau'/2}\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_1/2,t_n+\tau'/2)\\ \boldsymbol{K}_3=e^{\boldsymbol{H}\tau'/2}\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_2/2,t_n+\tau'/2)\\ \boldsymbol{K}_4=\boldsymbol{f}(\boldsymbol{v}_n+\boldsymbol{K}_3,t_n+\tau')\\ \boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}=e^{\boldsymbol{H}\tau'}\boldsymbol{v}_n+(\tau'/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\\ \boldsymbol{K}_1=e^{\boldsymbol{H}\tau'}\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)},t_n+\tau')\\ \boldsymbol{K}_2=e^{\boldsymbol{H}\tau'/2}\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+\boldsymbol{K}_1/2,t_n+3\tau'/2)\\ \boldsymbol{K}_3=e^{\boldsymbol{H}\tau'/2}\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+\boldsymbol{K}_2/2,t_n+3\tau'/2)\\ \boldsymbol{K}_4=\boldsymbol{f}(\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+\boldsymbol{K}_3,t_{n+1})\\ \boldsymbol{v}_{n+1}^{(\Delta t/2)}=\boldsymbol{v}_{n+\tau'}^{(\Delta t/2)}+(\tau'/6)(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4)\end{cases}\tag{21}$$

执行式(15)

当 $\xi_{n+1}\leqslant a$ 时，执行式(16)，计算下一计算步，初始时间步长为 Δt^*。

当 $\xi_{n+1}>a$ 时，执行式(17)，返回式(20,21)，重新计算该计算步。

6　自适应步长算法和定步长算法数值算例对比分析

算例 1　刚度软化问题[15]。已知非线性结构动力方程 $\ddot{x}+4\pi^2\times5^{-t}x=0$，初始条件为 $x_0=0.1$，$\dot{x}_0=0$，设 $x=v_1$，$\dot{x}=v_2$，相应的一阶微分方程组为

$$\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\end{Bmatrix}=\begin{bmatrix}0&1\\1&0\end{bmatrix}\begin{Bmatrix}v_1\\ v_2\end{Bmatrix}+\begin{Bmatrix}0\\(-1-4\pi^2\times5^{-t})v_1\end{Bmatrix}$$

采用经典 R-K 法、经典 R-K 自适应步长算法、精细 R-K 法和精细 R-K 自适应步长算法进行求解。经典 R-K 法和经典 R-K 自适应步长算法对比结果如图 2 所示，各时间段内包含时间节点的数量列入表 1。精细 R-K 法和精细 R-K 自适应步长算法对比结果如图 3 所示，各时间段内包含时间节点的数量列入表 2。经典 R-K 法 $\Delta t=0.1$ s；经典 R-K 自适应步长算法 $\Delta t^*=0.1$ s，$a=1.0\times10^{-7}$ mm，$b=0.5$；精细 R-K 法 $\Delta t=0.01$ s；精细 R-K 自适应步长算法 $\Delta t^*=0.01$ s，$a=1.0\times10^{-7}$ mm，$b=0.5$。

表 1　各时间段包含时间节点的个数

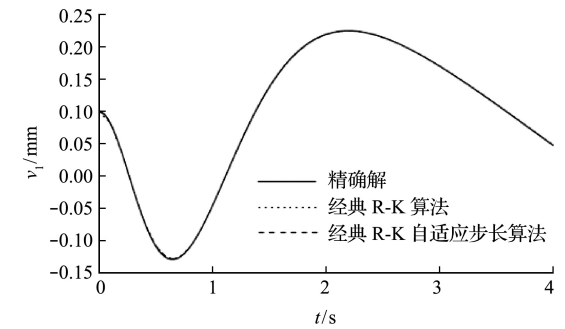
Tab.1　Number of time nodes in each time period

时间段	0 s～1 s	1 s～2 s	2 s～3 s	3 s～4 s
经典 R-K 算法	10	10	10	10
经典 R-K 自适应步长算法	19	11	10	10

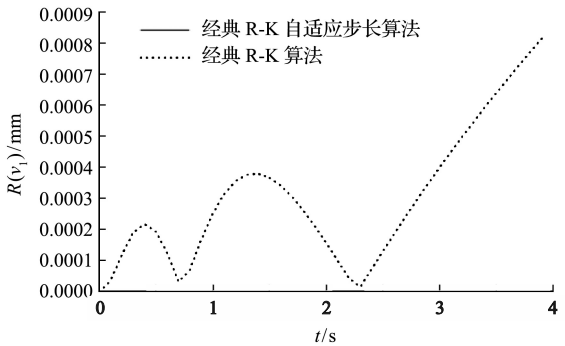
表 2　各时间段包含时间节点的个数

Tab.2　Number of time nodes in each time period

时间段	0 s～1 s	1 s～2 s	2 s～3 s	3 s～4 s
精细 R-K 算法	100	100	100	100
精细 R-K 自适应步长算法	151	100	100	100



(a) 位移时程曲线
(a) Displacement



(b) 绝对误差
(b) Displacement error

(注:图 2(b)中,经典 R-K 自适应步长算法的绝对误差时程曲线和横坐标轴重合。)

图 2 以经典 R-K 算法为基础的对比结果

Fig. 2 Comparison results based on the classical R-K algorithm

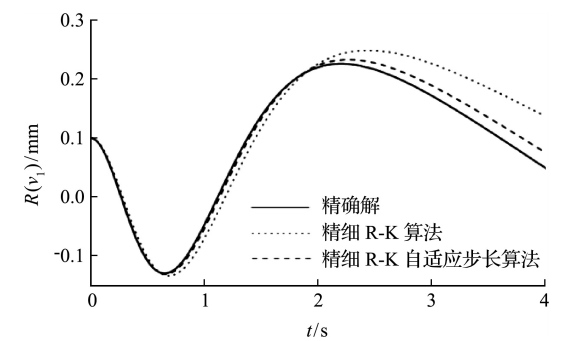


图 3 以精细 R-K 算法为基础的对比结果

Fig. 3 Comparison results based on the precise R-K algorithm

根据图 2(或图 3)可知,由经典 R-K 算法和经典 R-K 自适应步长算法(或精细 R-K 算法和精细 R-K 自适应步长算法)对比可知,自适应步长算法的精度高于定步长算法。原因可以由表 1(或表 2)得到,在 0 s～1 s 的时间段内,自适应步长算法比定步长算法多出 9(51)个时间节点。说明自适应步长的精细积分法能够缩小计算步误差较大时间段内的时间步长,从而提高该时间段内的计算精度,进一步提高整体的计算精度。

由表 1(或表 2)可知,在 1 s～4 s 的时间段内,定步长算法和自适应步长算法的时间步长一样的,仅在 0 s～1 s 的时间段内,自适应步长算法的时间步长小于定步长算法的时间步长。所以初始时间段内计算步的误差会影响后续计算步的精度,对于初始时间段内计算步误差较大的问题,更能够体现出自适应步长算法的优越性。

算例 2 求解非线性方程(二阶) $x\ddot{x}+\dot{x}^2=0$,该方程有解析解。其解析解为 $x=\sqrt{7.2t+0.09}$。$x=v_1$,$\dot{x}=v_2$,初值条件为 $x_0=0.3$,$\dot{x}_0=12$。相应的一阶微分方程组为

$$\dot{\boldsymbol{V}}(t)=\boldsymbol{H}(t)\boldsymbol{V}(t)+\boldsymbol{f}(t)$$

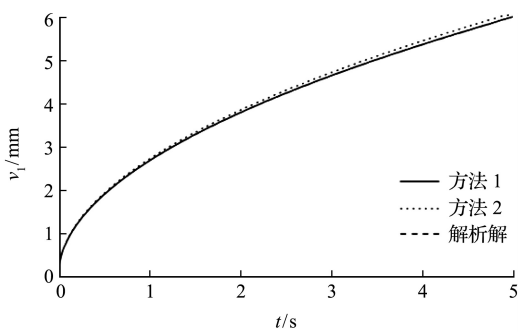
$$\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\end{Bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -v_2v_1^{-1}\end{bmatrix}\begin{Bmatrix}v_1\\ v_2\end{Bmatrix}+\begin{Bmatrix}0\\ 0\end{Bmatrix}$$

本文对应的状态方程形式为

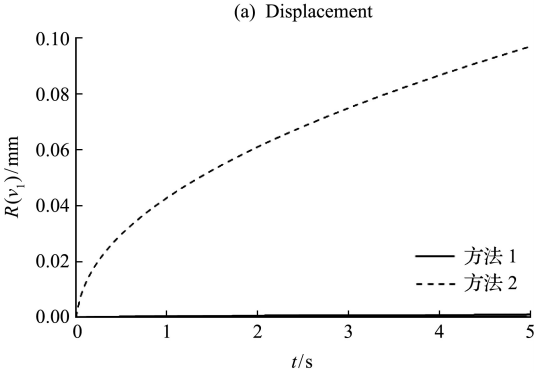
$$\dot{\boldsymbol{V}}(t)=\boldsymbol{H}\boldsymbol{V}(t)+\boldsymbol{f}(\boldsymbol{V},t)$$

$$\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\end{Bmatrix}=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{Bmatrix}v_1\\ v_2\end{Bmatrix}+\begin{Bmatrix}0\\ -v_2^2v_1^{-1}-v_1\end{Bmatrix}$$

采用经典 R-K 和经典 R-K 自适应步长算法进行求解。经典 R-K 法和经典 R-K 自适应步长算法对比结果如图 4 所示,各时间段内包含时间节点的数量列入表 3。不同误差限值 a 条件下,经典 R-K 自适应步长算法计算过程中,最大绝对误差列入表 4。不同时间步长条件下,经典 R-K 算法的位移时程曲线如图 5 所示。



(a) 位移时程曲线
(a) Displacement



(b) 绝对误差
(b) Displacement error

图 4 $\Delta t=\Delta t^*=0.01$ s 时两种方法的对比结果

Fig. 4 Comparison of the two methods ($\Delta t=\Delta t^*=0.01$ s)

方法1，经典R-K自适应步长算法；方法2，经典R-K算法。图4和表3中方法1的参数为 $\Delta t^*=0.01$ s，$a=1.0\times10^{-7}$ mm。

表3　图4中两种方法各时间段包含时间节点的数量

Tab. 3　Number of time nodes in each time period of the two methods in Fig. 4

时间段	0 s～1 s	1 s～2 s	2 s～3 s	3 s～4 s	4 s～5 s
方法1	117	100	100	100	100
方法2	101	100	100	100	100

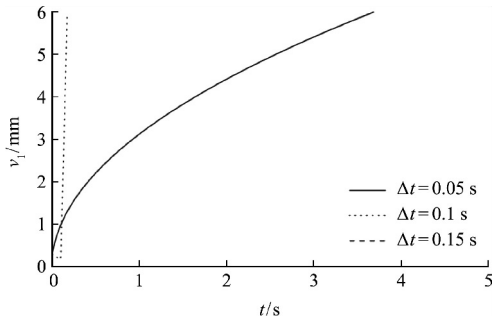


图5　不同时间步长条件下采用方法2求解的位移时程曲线

Fig. 5　Displacement curves solved by method 2 under different time steps

表4　$\Delta t^*=0.01$ s，不同 a 值时方法1在5 s内包含时间节点的个数及最大误差

Tab. 4　Number of time nodes and the maximum error included in the method 1 within 5 s with different a values

	$a=1.0\times10^{-5}$	$a=1.0\times10^{-6}$	$a=1.0\times10^{-7}$	$a=1.0\times10^{-8}$	$a=1.0\times10^{-9}$
节点个数	503	507	518	544	605
MAX($R(x_1)$)	0.007347	0.002495	0.000953	0.000336	0.000113

由表3可知，经典R-K自适应步长算法在时间段0 s～1 s内的时间节点个数为117个，比相应时间段经典R-K算法的时间节点数量多16个，但从图4可以看出，经典R-K自适应步长算法的精度比经典R-K算法高一阶。在0 s～1 s时间段内，计算步的误差较大，故经典R-K自适应步长算法通过减小该时间段内的时间步长，提高该时间段内的精度，从而提高整体的计算精度。

由表4可知，a 值设置越小，计算的时间节点个数越多，故计算的精度越高。对于该问题，时间节点的个数仅增加16个，但对精度显著提高。

图5表明了经典R-K算法稳定性的情况，当时间步长 $\Delta t=0.1$ s 和 $\Delta t=0.15$ s 时，该算法是不稳定的。

算例3　刚度硬化问题[15]。已知非线性结构动力 $\ddot{x}+4\pi^2/9\times5^{3/2}tx=0$，初始条件为 $x_0=0.1$，$\dot{x}_0=0$。设 $x=v_1$，$\dot{x}=v_2$，一阶微分方程组为

$$\dot{\boldsymbol{V}}(t)=\boldsymbol{H}(t)\boldsymbol{V}(t)+\boldsymbol{f}(t)$$

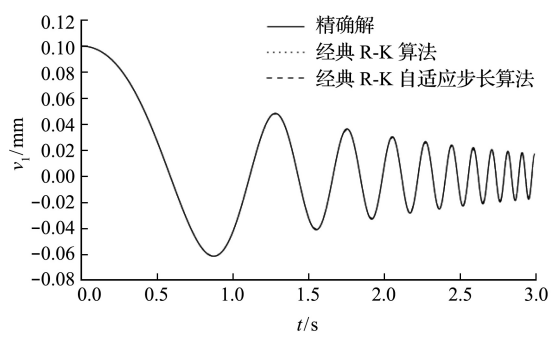
$$\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\end{Bmatrix}=\begin{Bmatrix}0 & 1\\ -4\pi^2/9\times(5^{3/2}t) & 0\end{Bmatrix}\begin{Bmatrix}v_1\\ v_2\end{Bmatrix}+\begin{Bmatrix}0\\ 0\end{Bmatrix}$$

本文对应的状态方程形式为

$$\dot{\boldsymbol{V}}(t)=\boldsymbol{H}\boldsymbol{V}(t)+\boldsymbol{f}(\boldsymbol{V},t)$$

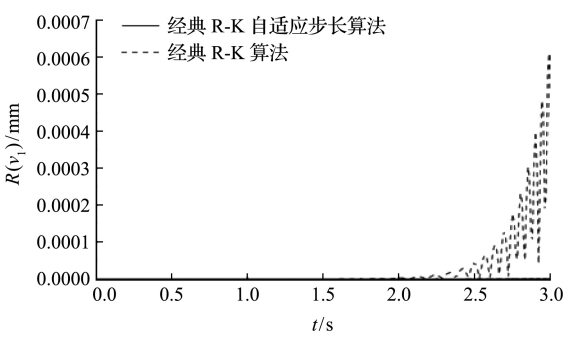
$$\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\end{Bmatrix}=\begin{Bmatrix}0 & 1\\ 1 & 0\end{Bmatrix}\begin{Bmatrix}v_1\\ v_2\end{Bmatrix}+\begin{Bmatrix}0\\ -1-\dfrac{4\pi^2}{9}\cdot5^{3/2}t)v_i\end{Bmatrix}$$

采用经典R-K法、经典R-K自适应步长算法、精细R-K法和精细R-K自适应步长算法进行求解。经典R-K法和经典R-K自适应步长算法对比结果如图6所示，各时间段内包含时间节点的数量列入表5。精细R-K法和精细R-K自适应步长算法对比结果如图7所示，各时间段内包含时间节点的数量列入表6。图6中经典R-K自适应步长算法的参数为 $a=1.0\times10^{-7}$ mm，$b=0.5$ 和 $\Delta t^*=0.01$s；图7中精细R-K自适应步长算法的参数为 $a=1.0\times10^{-7}$ mm，$b=0.5$ 和 $\Delta t^*=0.0005$ s。



(a) 位移时程曲线

(a) Displacement



(b) 绝对误差

(b) Displacement error

图6　$\Delta t=\Delta t^*=0.01$ s时两种算法的对比结果

Fig. 6　Comparison of the two methods ($\Delta t=\Delta t^*=0.01$ s)

图6(b)中经典R-K自适应步长精细积分法的误差曲线和横坐标轴几乎重合。

从图6(或图7)可以看出,自适应步长精细积分法的精度高于相应定步长精细积分法的精度,原因由表5(或表6)可知,本算例在2.5 s～3 s时间段内计算步误差较大,所以自适应步长精细积分法通过缩短该时间段内的时间步长,提高整体的计算精度。

7 误差限值的讨论

将误差估计表达式应用于经典Runge-Kutta法中,但不进行时间步长的调整,对算例进行求解,得出不同时间步长条件下,各时间段内所有计算步误差的误差估计平均值。

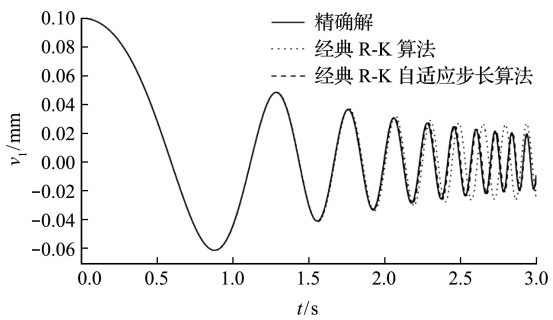


图7 $\Delta t=\Delta t^*=0.0005$ s时两种算法的对比结果

Fig. 7 Comparison of the two algorithms ($\Delta t=\Delta t^*=0.0005$ s)

表5 图6两种算法各时间段包含时间节点的个数

Tab. 5 Number of time nodes in each time period of the two algorithms in Fig. 6

时间段	0 s～0.5 s	0.5 s～1.0 s	1.0 s～1.5 s	1.5 s～2.0 s	2.0 s～2.5 s	2.5 s～3.0 s
经典R-K算法	50	50	50	50	50	50
经典R-K自适应步长算法	50	50	50	50	50	95

表6 图7两种算法各时间段包含时间节点的个数

Tab. 6 Number of time nodes in each time period of the two algorithms in Fig. 7

时间段	0 s～0.5 s	0.5 s～1.0 s	1.0 s～1.5 s	1.5 s～2.0 s	2.0 s～2.5 s	2.5 s～3.0 s
精细R-K算法	1000	1000	1000	1000	1000	1000
精细R-K自适应步长算法	1000	1000	1000	1000	1000	1666

如果不考虑求解具体问题对计算步误差的影响,那么当时间步长分别为$\Delta t=0.1$ s,$\Delta t=0.01$ s和$\Delta t=0.001$ s对应的计算步误差的限值为1×10^{-4},1×10^{-8}和1×10^{-12}。由表7可知,对于该问题,每一计算步的误差都满足误差限值的要求。

如果设置误差限值时不考虑求解具体问题的影响,那么当时间步长分别为$\Delta t=0.1$ s,$\Delta t=0.01$ s和$\Delta t=0.001$ s对应的误差限值为1×10^{-4},1×10^{-8}和1×10^{-12}。由表8可知,0 s～1 s时间段内的计算步误差不满足误差限值的要求,在采用自适应步长精细积分法进行求解时,会缩短时间步长,满足误差限值的要求,提高整体的计算精度。

如果设置误差限值时不考虑求解具体问题的影响,那么当时间步长分别为$\Delta t=0.01$ s和$\Delta t=0.001$ s对应的误差限值为1×10^{-8}和1×10^{-12}。由表9可知,2.5 s～3.0 s时间段内的计算步误差不满足误差限值的要求,在采用自适应步长精

表7 算例1不同时间步长条件下各个时间段内所有计算步误差估计的平均值

Tab. 7 Average of the error in each time period in example 1

时间段	0 s～1 s	1 s～2 s	2 s～3 s	3 s～4 s
$\Delta t=0.1$ s	9×10^{-7}	4×10^{-8}	2×10^{-9}	1×10^{-10}
$\Delta t=0.01$ s	1×10^{-11}	2×10^{-13}	1×10^{-14}	1×10^{-15}
$\Delta t=0.001$ s	1×10^{-16}	2×10^{-18}	5×10^{-19}	3×10^{-19}

表8 算例2不同时间步长条件下各个时间段内所有计算步误差估计的平均值

Tab. 8 Average of the error in each time period in example 2

时间段	0 s～1 s	1 s～2 s	2 s～3 s	3 s～4 s	4 s～5 s
$\Delta t=0.1$ s	5×10^{-3}	5×10^{-8}	3×10^{-8}	2×10^{-8}	1×10^{-8}
$\Delta t=0.01$ s	6×10^{-7}	2×10^{-12}	2×10^{-12}	1×10^{-12}	1×10^{-12}
$\Delta t=0.001$ s	5×10^{-11}	2×10^{-16}	2×10^{-16}	1×10^{-16}	1×10^{-16}

表9 算例3不同时间步长条件下各个时间段内所有计算步误差估计的平均值

Tab. 9 Average of the error in each time period in example 3

时间段	0 s～0.5 s	0.5 s～1.0 s	1.0 s～1.5 s	1.5 s～2.0 s	2.0 s～2.5 s	2.5 s～3.0 s
$\Delta t=0.01$ s	4×10^{-13}	9×10^{-12}	1×10^{-10}	2×10^{-9}	3×10^{-8}	5×10^{-7}
$\Delta t=0.001$ s	5×10^{-18}	9×10^{-17}	1×10^{-15}	2×10^{-14}	3×10^{-13}	5×10^{-12}

细积分法进行求解时，会缩短时间步长，满足误差限值的要求，提高整体的计算精度。

综上所述，在设置误差限值 a 时，可以不考虑求解具体问题的影响，只考虑采用的Runge-Kutta公式阶数和时间步长的大小。当由于求解具体问题的影响导致某些时间段内的计算步误差不满足误差限值 a 的要求时，自适应步长精细积分法就会缩短时间步长，从而满足误差限值的要求，进而提高整体的计算精度。

8　结　论

本文基于Runge-Kutta公式实现了对时间步长进行自适应调整，将其应用于经典R-K法和精细R-K法中，得到经典R-K自适应步长算法和精细R-K自适应步长算法。主要内容和结论如下。

(1) 经典R-K自适应步长精细积分法和精细R-K自适应步长算法的初始时间步长 Δt^* 是算法求解问题过程中时间步长能达到的最大值，故为整体计算精度提供一个最低的精度保障。

(2) 在计算步误差较大时间段，自适应步长精细积分法能够缩短该时间段内的时间步长，从而提高该时间段的计算精度，进一步提高整体的计算精度。对于初始时间段内计算步误差较大的问题，该时间段内产生的较大计算误差对后续计算步精度的影响是巨大的，所以对于初始时间段内计算步误差较大的问题，更加能够体现出自适应步长精细积分法的优越性。

(3) 基于Runge-Kutta公式得到实现自适应步长的方式，将该方式应用于经典R-K算法和精细R-K算法中，得到经典R-K自适应步长算法和精细R-K自适应步长算法能够解决刚度硬化问题和刚度软化问题，具有广泛的适用性。

参考文献(References)：

[1] 钟万勰. 结构动力方程的精细时程积分法[J]. 大连理工大学学报，1994，**34**(2)：131-136. (ZHONG Wan-xie. On precise time-integration method for structural dynamics[J]. *Journal of Dalian University of Technology*，1994，**34**(2)：131-136. (in Chinese))

[2] 赵秋玲. 非线性动力学方程的精细积分法[J]. 力学与实践，1998，**20**(6)：24-26. (ZHAO Qiu-ling. An accurate integration method for solving nonlinear dynamic problems[J]. *Mechanics in Engineering*，1998，**20**(6)：24-26. (in Chinese))

[3] 张森文，曹开彬. 计算结构动力响应的状态方程直接积分法[J]. 计算力学学报，2000，**17**(1)：94-97，118. (ZHANG Sen-wen，CAO Kai-bin. Direct integration of state equation method for dynamic response of structure[J]. *Chinese Journal of Computational Mechanics*，2000，**17**(1)：94-97，118. (in Chinese))

[4] 裘春航，吕和祥，钟万勰. 求解非线性动力学方程的分段直接积分法[J]. 力学学报，2002，**34**(3)：369-378. (QIU Chun-hang，LÜ He-xiang，ZHONG Wan-xie. On segmented-direct-integration method for nonlinear dynamics equations[J]. *Chinese Journal of Theoretical and Applied Mechanics*，2002，**34**(3)：369-378. (in Chinese))

[5] 汪梦甫，周锡元. 结构动力方程的高斯精细时程积分法[J]. 工程力学，2004，**21**(4)：13-16. (WANG Meng-fu，ZHOU Xi-yuan. Gauss precise time-integration of structural dynamic analysis[J]. *Engineering Mechanics*，2004，**21**(4)：13-16. (in Chinese))

[6] 陈伯望，王海波. 结构非线性动力分析的精细积分多步法[J]. 工程力学，2009，**26**(5)：41-46. (CHEN Bo-wang，WANG Hai-bo. Precise integral multi-step method for nonlinear dynamic analysis of structures[J]. *Engineering Mechanics*，2009，**26**(5)：41-46. (in Chinese))

[7] 李炜华，王　堉，罗　恩. 求解非线性结构动力方程的预估校正-辛时间子域法.[J]. 计算力学学报，2014，**31**(4)：453-458. (LI Wei-hua，WANG Yu，LUO En. Predictor-corrector symplectic time-subdomain algorithm for nonlinear dynamicequations[J]. *Chinese Journal of Computational Mechanics*，2014，**31**(4)：453-458. (in Chinese))

[8] 王海波，何崇检. 非线性动力分析的广义精细积分法[J]. 振动与冲击，2018，**37**(21)：220-226. (WANG Hai-bo，HE Chong-jian. Generalized precise time domain integration method for nonlinear dynamic analysis[J]. *Journal of Vibration and Shock*，2018，**37**(21)：220-226. (in Chinese))

[9] 王海波，何崇检，贾耀威. 线性动力分析的一种通用积分格式[J]. 振动与冲击，2019，**38**(10)：43-48. (WANG Hai-bo，HE Chong-jian，JIA Yao-wei. General integration scheme for linear dynamic analysis[J]. *Journal of Vibration and Shock*，2019，**38**(10)：43-48. (in Chinese))

[10] 王　永，马　骏，李靖翔，等. 非齐次动力学方程的一种精细积分单步方法[J]. 计算力学学报，2020，**37**(2)：212-217. (WANG Yong，MA Jun，LI Jing-xiang，et al. A precise integration single-step method for nonhomogeneous dynamic equations[J]. *Chinese Journal of Computational Mechanics*，2020，**37**(2)：212-217. (in Chinese))

[11] 梅树立，张森文，徐加初，等. 非线性动力学方程的自适应精细积分[J]. 暨南大学学报(自然科学版)，

2005,**26**(3):19-323.(MEI Shu-li,ZHANG Sen-wen,XU Jia-chu,et al. Adaptive precise integration algorithm for nonlinear dynamical equation[J]. *Journal of Jinan University*(Natural Science),2005,**26**(3):319-323.(in Chinese))

[12] 李 健,高广军,张 洁.离散精细时程积分的自适应求积算法研究[J].应用力学学报,2014,**31**(4):502-505.(LI Jian,GAO Guang-jun,ZHANG Jie. Research of adaptive quadrature algorithm for discrete precise time-integration method[J]. *Chinese Journal of Applied Mechanics*,2014,**31**(4):502-505.(in Chinese))

[13] 王海波,纪海潮.非线性动力方程精细积分法的自适应步长研究[J].计算力学学报,2021,**38**(3):371-376.(WANG Hai-bo,JI Hai-chao. Study on adaptive step size of precise intrgration method for nonlinear dynamic equations[J]. *Chinese Journal of Computational Mechanics*,2021,**38**(3):371-376.(in Chinese))

[14] Zhang S Y,Deng Z C,Li W C. A precise Runge-Kutta integration and its application for solving nonlinear dynamical systems[J]. *Applied Mathematics and Computation*,2007,**184**(2):496-502.

[15] 崔雪娜,王焕定.刚度解析表达时高阶单步法的研究[J].地震工程与工程振动,2006,**26**(4):63-67.(CUI Xue-na,WANG Huan-ding. Research on a high order single step method for analytic expression of stiffness[J]. *Earthquake Engineering and Engineering Dynamics*,2006,**26**(4):63-67.(in Chinese))

Adaptive step numerical algorithm for nonlinear structural dynamic equations

WANG Hai-bo*, WANG Hong-shen, JI Hai-chao

(School of Civil Engineering,Central South University,Changsha 410075,China)

Abstract: In this paper, we examine the enhancement of computational accuracy of nonlinear structural dynamic equations by using adaptive selection of the time step based on Runge-Kutta method. The local truncation error of Runge-Kutta formula is used to obtain the error estimate value ξ_{n+1}, and the size of the time step is adaptively adjusted according to the sizes of ξ_{n+1}, providing ξ_{n+1} judgment statement for the algorithm, which can make the flow chart of the algorithm more diverse. This idea is applied to the classical Runge-Kutta algorithm and the fine Runge-Kutta algorithm, and the adaptive step sizes of the classical Runge-Kutta algorithm and the fine Runge-Kutta algorithm are obtained, so that the time step size of the algorithm is dependent on the given error limit of each step to improve the calculation accuracy. Numerical examples demonstrate the validity of the proposed ideas.

Key words: nonlinear dynamic equation; adaptive step; precise integration; Runge-Kutta