实验16 常微分方程初值问题的数值解法

参考答案

1. 利用Euler方法求初值问题的程序：function E=eulers(f,a,b,ya,h)（见代码文件）

取步长分别为*h* = 0.5和*h* = 0.25求得的序列分别为

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 |
| *y* | 1 | 0.75 | 0.6875 | 0.765625 | 0.94921875 | 1.2119140625 | 1.533935546875 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 0 | 0.25 | 0.5 | 0.75 | 1 | 1.25 | 1.5 |
| *y* | 1 | 0.875 | 0.796875 | 0.759765625 | 0.758544921875 | 0.788726806640625 | 0.846385955810547 |
| *t* | 1.75 | 2 | 2.25 | 2.5 | 2.75 | 3 |  |
| *y* | 0.928087711334229 | 1.03082674741745 | 1.15197340399027 | 1.28922672849149 | 1.44057338743005 | 1.60425171400129 |  |

易见，该问题的精确解为*y*=3e-*t*/2-2+*t*，故*y*(3)=1.66939048044529。所以*t* =3处的最终全局误差为*E*(*y*(3), 0.5)= 0.135454933570289, *E*(*y*(3), 0.25)= 0.0651387664439962。得到的步进结果图如下。可见，对同一种方法，采用较小的步长误差比较小。



1. 改进的Euler方法求初值问题的程序：function IME=impeuler(f,a,b,ya,h)（见代码文件）

取步长分别为*h* = 0.5和*h* = 0.25求得的序列分别为

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 |
| *y* | 1 | 0.84375 | 0.8310546875 | 0.930511474609375 | 1.11758708953857 | 1.37311491370201 | 1.6821210263297 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 0 | 0.25 | 0.5 | 0.75 | 1 | 1.25 | 1.5 |
| *y* | 1 | 0.8984375 | 0.83807373046875 | 0.814080715179443 | 0.822196256369352 | 0.858657632576069 | 0.920143066258561 |
| *t* | 1.75 | 2 | 2.25 | 2.5 | 2.75 | 3 |  |
| *y* | 1.00372005068139 | 1.10679973224216 | 1.22709663862003 | 1.36259312628175 | 1.51150799429561 | 1.67226877621409 |  |

*t* =3处的最终全局误差为*E*(*y*(3), 0.5)= -0.0127305458844067，

*E*(*y*(3), 0.25)= -0.00287829576879939。得到的步进结果图如下。可见，改进的Euler方法比原始Euler方法误差明显变小，原因是改进的Euler方法的最终全局误差阶为*O*(*h*3)，而原始的Euler方法的误差阶为*O*(*h*2)。



1. 经典RK4方法求初值问题的程序：function RK4=rk4m(f,a,b,ya,h)（见代码文件）

取步长分别为*h* = 0.5和*h* = 0.25求得的序列分别为

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 |
| *y* | 1 | 0.83642578125 | 0.819628477096558 | 0.917142295395024 | 1.10368259822025 | 1.35955749226626 | 1.66943076179916 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *t* | 0 | 0.25 | 0.5 | 0.75 | 1 | 1.25 | 1.5 |
| *y* | 1 | 0.897491455078125 | 0.836403668237229 | 0.81186958242375 | 0.819594033650794 | 0.855786551933871 | 0.917102058308097 |
| *t* | 1.75 | 2 | 2.25 | 2.5 | 2.75 | 3 |  |
| *y* | 1.00058853011477 | 1.10364081576586 | 1.22395987640518 | 1.35951681679056 | 1.50852114264965 | 1.66939274788701 |  |

*t* =3处的最终全局误差为*E*(*y*(3), 0.5)= -4.02813538697977×10-5，

*E*(*y*(3), 0.25)= -2.26744172548976×10-6。得到的步进结果图如下。可见，经典4阶Runge-Lutta方法比Euler方法误差明显变小，原因是RK4方法的最终全局误差阶为*O*(*h*4)，比上两题的Euler方法的误差阶更高，但RK4方法每一步需要求4个函数值。



1. 使用Matlab内置的ode23和ode45函数计算得到的步进结果图如下。对于ode23函数，整个区间上的最大误差出现在*t*=2.26处，误差为*Err*=0.000163137202267238；对于ode45函数，整个区间上的最大误差出现在*t*=0.15处，误差为*Err*=1.35671083256739×10-7。实际上，这两个函数是不同阶的自适应步长的Runge-Kutta方法：ode23使用的是显式的2阶-3阶Runge-Kutta方法；而ode45则使用了显式的4阶-5阶Runge-Kutta方法。在每一步中，使用两个不同的求近似解的方法，并比较计算结果：如果两个结果的差小于给定误差限，则接受该近似；如果两个结果的差大于给定误差限，则减小步长；如果答案超过了要求的有效数字位数，则增加步长。

