Initial-Value Problems for ODEs

Error Control & Runge-Kutta-Fehlberg Method

Numerical Analysis (9th Edition)

R L Burden & J D Faires

Beamer Presentation Slides
prepared by
John Carroll
Dublin City University

Outline

Outline

Outline

Outline

Outline

Introduction to Adaptive Methods

Rationale

Introduction to Adaptive Methods

Rationale

- We now turn to the appropriate use of varying step sizes when approximating the solution to an initial-value problem.

Introduction to Adaptive Methods

Rationale

- We now turn to the appropriate use of varying step sizes when approximating the solution to an initial-value problem.
- We will be concerned with the development of efficient methods that are not unduly disadvantaged by increased complication in their aplication.

Introduction to Adaptive Methods

Rationale

- We now turn to the appropriate use of varying step sizes when approximating the solution to an initial-value problem.
- We will be concerned with the development of efficient methods that are not unduly disadvantaged by increased complication in their aplication.
- The methods will incorporate in the step-size procedure an estimate of the truncation error that does not require the approximation of the higher derivatives of the function.

Introduction to Adaptive Methods

Rationale

- We now turn to the appropriate use of varying step sizes when approximating the solution to an initial-value problem.
- We will be concerned with the development of efficient methods that are not unduly disadvantaged by increased complication in their aplication.
- The methods will incorporate in the step-size procedure an estimate of the truncation error that does not require the approximation of the higher derivatives of the function.
- Such methods are called adaptive because they adapt the number and position of the nodes used in the approximation to ensure that the truncation error is kept within a specified bound.

Introduction to Adaptive Methods

One-Step Method: General Framework

Any one-step method for approximating the solution, $y(t)$, of the initial-value problem

$$y' = f(t, y), \quad \text{for } a \leq t \leq b, \quad \text{with } y(a) = \alpha$$

Introduction to Adaptive Methods

One-Step Method: General Framework

Any one-step method for approximating the solution, $y(t)$, of the initial-value problem

$$y' = f(t, y), \quad \text{for } a \leq t \leq b, \quad \text{with } y(a) = \alpha$$

can be expressed in the form

$$w_{i+1} = w_i + h_i \phi(t_i, w_i, h_i), \quad \text{for } i = 0, 1, \ldots, N-1$$

for some function ϕ.

Introduction to Adaptive Methods

$$w_{i+1} = w_i + h_i \phi(t_i, w_i, h_i), \quad \text{for } i = 0, 1, \ldots, N-1,$$

Desirable Properties

Introduction to Adaptive Methods

$$w_{i+1} = w_i + h_i \phi(t_i, w_i, h_i), \quad \text{for } i = 0, 1, \ldots, N-1,$$

Desirable Properties

- An ideal difference-equation method would have the property that, given a tolerance $\varepsilon > 0$, a minimal number of mesh points could be used to ensure that the global error, $|y(t_i) - w_i|$, did not exceed ε for any $i = 0, 1, \ldots, N$.

Introduction to Adaptive Methods

$$w_{i+1} = w_i + h_i \phi(t_i, w_i, h_i), \quad \text{for } i = 0, 1, \ldots, N-1,$$

Desirable Properties

- An ideal difference-equation method would have the property that, given a tolerance $\varepsilon > 0$, a minimal number of mesh points could be used to ensure that the global error, $|y(t_i) - w_i|$, did not exceed ε for any $i = 0, 1, \ldots, N$.
- Having a minimal number of mesh points and also controlling the global error of a difference method is, not surprisingly, inconsistent with the points being equally spaced in the interval.

Introduction to Adaptive Methods

$$w_{i+1} = w_i + h_i\phi(t_i, w_i, h_i), \quad \text{for } i = 0, 1, \ldots, N-1,$$

Desirable Properties

- An ideal difference-equation method would have the property that, given a tolerance $\varepsilon > 0$, a minimal number of mesh points could be used to ensure that the global error, $|y(t_i) - w_i|$, did not exceed ε for any $i = 0, 1, \ldots, N$.
- Having a minimal number of mesh points and also controlling the global error of a difference method is, not surprisingly, inconsistent with the points being equally spaced in the interval.
- We will examine techniques used to control the error of a difference-equation method in an efficient manner by the appropriate choice of mesh points.

Outline

Global Error .v. Local Truncation Error

Using the LTE

Global Error .v. Local Truncation Error

Using the LTE

- Although we cannot generally determine the global error of a method, it can be shown that there is a close connection between the local truncation error and the global error.

Global Error .v. Local Truncation Error

Using the LTE

- Although we cannot generally determine the global error of a method, it can be shown that there is a close connection between the local truncation error and the global error.
- By using methods of differing order, we can predict the local truncation error and, using this prediction, choose a step size that will keep it and the global error in check.

Local Truncation Error Estimation

Illustration

Suppose that we have two approximation techniques.

Local Truncation Error Estimation

Illustration

Suppose that we have two approximation techniques.

- The first is obtained from an nth-order Taylor method of the form

$$y(t_{i+1}) = y(t_i) + h\phi(t_i, y(t_i), h) + O(h^{n+1})$$

  and produces approximations w_{i+1} with local truncation error $\tau_{i+1}(h) = O(h^n)$.

Local Truncation Error Estimation

Illustration

Suppose that we have two approximation techniques.

- The first is obtained from an nth-order Taylor method of the form

$$y(t_{i+1}) = y(t_i) + h\phi(t_i, y(t_i), h) + O(h^{n+1})$$

  and produces approximations w_{i+1} with local truncation error $\tau_{i+1}(h) = O(h^n)$.

- The second method is similar but one order higher; it comes from an $(n+1)$st-order Taylor method of the form

$$y(t_{i+1}) = y(t_i) + h\tilde{\phi}(t_i, y(t_i), h) + O(h^{n+2})$$

  and produces approximations $\tilde{w}_{i+1}$ with local truncation error $\tilde{\tau}_{i+1}(h) = O(h^{n+1})$.

Local Truncation Error Estimation

Estimating $\tau_{i+1}(h)$

Local Truncation Error Estimation

Estimating $\tau_{i+1}(h)$

We first make the assumption that $w_i \approx y(t_i) \approx \tilde{w}_i$ and choose a fixed step size h to generate the approximations w_{i+1} and $\tilde{w}_{i+1}$ to $y(t_{i+1})$.

Local Truncation Error Estimation

Estimating $\tau_{i+1}(h)$

We first make the assumption that $w_i \approx y(t_i) \approx \tilde{w}_i$ and choose a fixed step size h to generate the approximations w_{i+1} and $\tilde{w}_{i+1}$ to $y(t_{i+1})$. Then

$$\tau_{i+1}(h) = \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i), h)$$

Local Truncation Error Estimation

Estimating $\tau_{i+1}(h)$

We first make the assumption that $w_i \approx y(t_i) \approx \tilde{w}_i$ and choose a fixed step size h to generate the approximations w_{i+1} and $\tilde{w}_{i+1}$ to $y(t_{i+1})$. Then

$$\begin{aligned}\tau_{i+1}(h) &= \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i), h) \\ &\approx \frac{y(t_{i+1}) - w_i}{h} - \phi(t_i, w_i, h)\end{aligned}$$

Local Truncation Error Estimation

Estimating $\tau_{i+1}(h)$

We first make the assumption that $w_i \approx y(t_i) \approx \tilde{w}_i$ and choose a fixed step size h to generate the approximations w_{i+1} and $\tilde{w}_{i+1}$ to $y(t_{i+1})$. Then

$$\begin{aligned} \tau_{i+1}(h) &= \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i), h) \\ &\approx \frac{y(t_{i+1}) - w_i}{h} - \phi(t_i, w_i, h) \\ &= \frac{y(t_{i+1}) - [w_i + h\phi(t_i, w_i, h)]}{h} \end{aligned}$$

Local Truncation Error Estimation

Estimating $\tau_{i+1}(h)$

We first make the assumption that $w_i \approx y(t_i) \approx \tilde{w}_i$ and choose a fixed step size h to generate the approximations w_{i+1} and $\tilde{w}_{i+1}$ to $y(t_{i+1})$. Then

$$\begin{aligned}
\tau_{i+1}(h) &= \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i), h) \\
&\approx \frac{y(t_{i+1}) - w_i}{h} - \phi(t_i, w_i, h) \\
&= \frac{y(t_{i+1}) - [w_i + h\phi(t_i, w_i, h)]}{h} \\
&= \frac{1}{h}(y(t_{i+1}) - w_{i+1})
\end{aligned}$$

Local Truncation Error Estimation

Estimating $\tilde{\tau}_{i+1}(h)$

Local Truncation Error Estimation

Estimating $\tilde{\tau}_{i+1}(h)$

In a similar manner, we have

$$\tilde{\tau}_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - \tilde{w}_{i+1})$$

Local Truncation Error Estimation

Estimating $\tilde{\tau}_{i+1}(h)$

In a similar manner, we have

$$\tilde{\tau}_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - \tilde{w}_{i+1})$$

As a consequence, we have

$$\tau_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - w_{i+1})$$

Local Truncation Error Estimation

Estimating $\tilde{\tau}_{i+1}(h)$

In a similar manner, we have

$$\tilde{\tau}_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - \tilde{w}_{i+1})$$

As a consequence, we have

$$\begin{aligned} \tau_{i+1}(h) &= \frac{1}{h}(y(t_{i+1}) - w_{i+1}) \\ &= \frac{1}{h}[(y(t_{i+1}) - \tilde{w}_{i+1}) + (\tilde{w}_{i+1} - w_{i+1})] \end{aligned}$$

Local Truncation Error Estimation

Estimating $\tilde{\tau}_{i+1}(h)$

In a similar manner, we have

$$\tilde{\tau}_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - \tilde{w}_{i+1})$$

As a consequence, we have

$$\begin{aligned}
\tau_{i+1}(h) &= \frac{1}{h}(y(t_{i+1}) - w_{i+1}) \\
&= \frac{1}{h}[(y(t_{i+1}) - \tilde{w}_{i+1}) + (\tilde{w}_{i+1} - w_{i+1})] \\
&= \tilde{\tau}_{i+1}(h) + \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})
\end{aligned}$$

Local Truncation Error Estimation

$$\tau_{i+1}(h) = \tilde{\tau}_{i+1}(h) + \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})$$

The LTE Approximation

Local Truncation Error Estimation

$$\tau_{i+1}(h) = \tilde{\tau}_{i+1}(h) + \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})$$

The LTE Approximation

Since $\tau_{i+1}(h)$ is $O(h^n)$ and $\tilde{\tau}_{i+1}(h)$ is $O(h^{n+1})$, the significant portion of $\tau_{i+1}(h)$ must come from

$$\frac{1}{h}\left(\tilde{w}_{i+1} - w_{i+1}\right).$$

Local Truncation Error Estimation

$$\tau_{i+1}(h) = \tilde{\tau}_{i+1}(h) + \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})$$

The LTE Approximation

Since $\tau_{i+1}(h)$ is $O(h^n)$ and $\tilde{\tau}_{i+1}(h)$ is $O(h^{n+1})$, the significant portion of $\tau_{i+1}(h)$ must come from

$$\frac{1}{h}\left(\tilde{w}_{i+1} - w_{i+1}\right).$$

This gives us an easily computed approximation for the local truncation error of the $O(h^n)$ method:

$$\tau_{i+1}(h) \approx \frac{1}{h}\left(\tilde{w}_{i+1} - w_{i+1}\right)$$

Outline

Local Truncation Error Control

$$\tau_{i+1}(h) \approx \frac{1}{h}\left(\tilde{w}_{i+1} - w_{i+1}\right)$$

Using the LTE Estimate

Local Truncation Error Control

$$\tau_{i+1}(h) \approx \frac{1}{h}\left(\tilde{w}_{i+1} - w_{i+1}\right)$$

Using the LTE Estimate

- The object, however, is not simply to estimate the local truncation error but to adjust the step size to keep it within a specified bound.

Local Truncation Error Control

$$\tau_{i+1}(h) \approx \frac{1}{h}\left(\tilde{w}_{i+1} - w_{i+1}\right)$$

Using the LTE Estimate

- The object, however, is not simply to estimate the local truncation error but to adjust the step size to keep it within a specified bound.
- To do this we now assume that since $\tau_{i+1}(h)$ is $O(h^n)$, a number K, independent of h, exists with

$$\tau_{i+1}(h) \approx Kh^n$$

Local Truncation Error Control

Adjusting the Step Size

Then, the local truncation error produced by applying the nth-order method with a new step size qh, can be estimated using the original approximations w_{i+1} and $\tilde{w}_{i+1}$:

$$\tau_{i+1}(qh) \approx K(qh)^n = q^n(Kh^n) \approx q^n \tau_{i+1}(h) \approx \frac{q^n}{h}(\tilde{w}_{i+1} - w_{i+1})$$

Local Truncation Error Control

Adjusting the Step Size

Then, the local truncation error produced by applying the nth-order method with a new step size qh, can be estimated using the original approximations w_{i+1} and $\tilde{w}_{i+1}$:

$$\tau_{i+1}(qh) \approx K(qh)^n = q^n(Kh^n) \approx q^n \tau_{i+1}(h) \approx \frac{q^n}{h}(\tilde{w}_{i+1} - w_{i+1})$$

To bound $\tau_{i+1}(qh)$ by ε, we choose q so that

$$\frac{q^n}{h}|\tilde{w}_{i+1} - w_{i+1}| \approx |\tau_{i+1}(qh)| \leq \varepsilon$$

Local Truncation Error Control

Adjusting the Step Size

Then, the local truncation error produced by applying the *n*th-order method with a new step size qh, can be estimated using the original approximations w_{i+1} and $\tilde{w}_{i+1}$:

$$\tau_{i+1}(qh) \approx K(qh)^n = q^n(Kh^n) \approx q^n \tau_{i+1}(h) \approx \frac{q^n}{h}(\tilde{w}_{i+1} - w_{i+1})$$

To bound $\tau_{i+1}(qh)$ by ε, we choose q so that

$$\frac{q^n}{h}|\tilde{w}_{i+1} - w_{i+1}| \approx |\tau_{i+1}(qh)| \leq \varepsilon$$

that is, so that

$$q \leq \left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|}\right)^{1/n}$$

Outline

Runge-Kutta-Fehlberg Method

Runge-Kutta-Fehlberg Method

One popular technique that uses the inequality

$$q \leq \left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/n}$$

for error control is the Runge-Kutta-Fehlberg method.

Runge-Kutta-Fehlberg Method

One popular technique that uses the inequality

$$q \leq \left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/n}$$

for error control is the Runge-Kutta-Fehlberg method. This technique uses a Runge-Kutta method with local truncation error of order 5,

$$\tilde{w}_{i+1} = w_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6$$

Runge-Kutta-Fehlberg Method

One popular technique that uses the inequality

$$q \leq \left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/n}$$

for error control is the Runge-Kutta-Fehlberg method. This technique uses a Runge-Kutta method with local truncation error of order 5,

$$\tilde{w}_{i+1} = w_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6$$

to estimate the local error in a Runge-Kutta method of order 4 given by

$$w_{i+1} = w_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5$$

Runge-Kutta-Fehlberg Method

Coefficient equations of RKF

$$
\begin{aligned}
k_1 &= hf(t_i, w_i) \\
k_2 &= hf\left(t_i + \frac{h}{4}, w_i + \frac{1}{4}k_1\right) \\
k_3 &= hf\left(t_i + \frac{3h}{8}, w_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \\
k_4 &= hf\left(t_i + \frac{12h}{13}, w_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\
k_5 &= hf\left(t_i + h, w_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right) \\
k_6 &= hf\left(t_i + \frac{h}{2}, w_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right)
\end{aligned}
$$

Runge-Kutta-Fehlberg Method

Computational Advantages

Runge-Kutta-Fehlberg Method

Computational Advantages

- An advantage to this method is that only 6 evaluations of f are required per step.

Runge-Kutta-Fehlberg Method

Computational Advantages

- An advantage to this method is that only 6 evaluations of f are required per step.
- Arbitrary Runge-Kutta methods of orders 4 and 5 used together require at least 4 evaluations of f for the 4th-order method and an additional 6 for the 5th-order method, for a total of at least 10 function evaluations.

Runge-Kutta-Fehlberg Method

Computational Advantages

- An advantage to this method is that only 6 evaluations of f are required per step.
- Arbitrary Runge-Kutta methods of orders 4 and 5 used together require at least 4 evaluations of f for the 4th-order method and an additional 6 for the 5th-order method, for a total of at least 10 function evaluations.
- So the Runge-Kutta-Fehlberg method has at least a 40% decrease in the number of function evaluations over the use of a pair of arbitrary 4th- and 5th-order methods.

Runge-Kutta-Fehlberg Method

Error-Control Theory

Runge-Kutta-Fehlberg Method

Error-Control Theory

- An initial value of h at the ith step is used to find the first values of w_{i+1} and $\tilde{w}_{i+1}, \ldots$

Runge-Kutta-Fehlberg Method

Error-Control Theory

- An initial value of h at the ith step is used to find the first values of w_{i+1} and $\tilde{w}_{i+1}, \ldots$
- which leads to the determination of q for that step, and then the calculations were repeated.

Runge-Kutta-Fehlberg Method

Error-Control Theory

- An initial value of h at the ith step is used to find the first values of w_{i+1} and $\tilde{w}_{i+1}, \ldots$
- which leads to the determination of q for that step, and then the calculations were repeated.
- This procedure requires twice the number of function evaluations per step as without the error control.

Runge-Kutta-Fehlberg Method

Error-Control Theory

- An initial value of h at the ith step is used to find the first values of w_{i+1} and $\tilde{w}_{i+1}, \ldots$
- which leads to the determination of q for that step, and then the calculations were repeated.
- This procedure requires twice the number of function evaluations per step as without the error control.
- In practice, the value of q to be used is chosen somewhat differently in order to make the increased function-evaluation cost worthwhile.

Runge-Kutta-Fehlberg Method

Use of q when determined at the i-th step

Runge-Kutta-Fehlberg Method

Use of q when determined at the i-th step

- When $q < 1$: we reject the initial choice of h at the ith step and repeat the calculations using qh, and

Runge-Kutta-Fehlberg Method

Use of q when determined at the i-th step

- When $q < 1$: we reject the initial choice of h at the ith step and repeat the calculations using qh, and
- When $q \geq 1$: we accept the computed value at the ith step using the step size h, but change the step size to qh for the $(i+1)$st step.

Runge-Kutta-Fehlberg Method

Use of q when determined at the i-th step

- When $q < 1$: we reject the initial choice of h at the ith step and repeat the calculations using qh, and
- When $q \geq 1$: we accept the computed value at the ith step using the step size h, but change the step size to qh for the $(i+1)$st step.

Because of the penalty in terms of function evaluations that must be paid if the steps are repeated, q tends to be chosen conservatively.

Runge-Kutta-Fehlberg Method

Use of q when determined at the i-th step

- When $q < 1$: we reject the initial choice of h at the ith step and repeat the calculations using qh, and
- When $q \geq 1$: we accept the computed value at the ith step using the step size h, but change the step size to qh for the $(i+1)$st step.

Because of the penalty in terms of function evaluations that must be paid if the steps are repeated, q tends to be chosen conservatively. In fact, for the Runge-Kutta-Fehlberg method with $n = 4$, a common choice is

$$q = \left(\frac{\varepsilon h}{2|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/4} = 0.84 \left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/4}$$

Runge-Kutta-Fehlberg Algorithm

To approximate the solution of the initial-value problem

$$y' = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha,$$

with local truncation error within a given tolerance:

Runge-Kutta-Fehlberg Algorithm

To approximate the solution of the initial-value problem

$$y' = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha,$$

with local truncation error within a given tolerance:

INPUT endpoints a, b; initial condition α; tolerance TOL; maximum step size $hmax$; minimum step size $hmin$.

Runge-Kutta-Fehlberg Algorithm

To approximate the solution of the initial-value problem

$$y' = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha,$$

with local truncation error within a given tolerance:

INPUT endpoints a, b; initial condition α; tolerance TOL; maximum step size $hmax$; minimum step size $hmin$.

OUTPUT t, w, h where w approximates $y(t)$ and the step size h was used, or a message that the minimum step size was exceeded.

Runge-Kutta-Fehlberg Algorithm

To approximate the solution of the initial-value problem

$$y' = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha,$$

with local truncation error within a given tolerance:

INPUT endpoints a, b; initial condition α; tolerance TOL; maximum step size $hmax$; minimum step size $hmin$.

OUTPUT t, w, h where w approximates $y(t)$ and the step size h was used, or a message that the minimum step size was exceeded.

Step 1 Set $t = a$; $w = \alpha$; $h = hmax$; $FLAG = 1$;
OUTPUT (t, w)

Runge-Kutta-Fehlberg Algorithm

To approximate the solution of the initial-value problem

$$y' = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha,$$

with local truncation error within a given tolerance:

INPUT endpoints a, b; initial condition α; tolerance TOL; maximum step size *hmax*; minimum step size *hmin*.

OUTPUT t, w, h where w approximates $y(t)$ and the step size h was used, or a message that the minimum step size was exceeded.

Step 1 Set $t = a$; $w = \alpha$; $h = hmax$; $FLAG = 1$;
OUTPUT (t, w)

Step 2 While ($FLAG = 1$) do Steps 3–11:

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3 $K_1 = hf(t, w)$

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3 $K_1 = hf\,(t, w)$

$$K_2 = hf\left(t + \frac{1}{4}h, w + \frac{1}{4}K_1\right)$$

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3 $K_1 = hf\left(t, w\right)$

$$K_2 = hf\left(t + \frac{1}{4}h, w + \frac{1}{4}K_1\right)$$

$$K_3 = hf\left(t + \frac{3}{8}h, w + \frac{3}{32}K_1 + \frac{9}{32}K_2\right)$$

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3

$$K_1 = hf\,(t, w)$$

$$K_2 = hf\left(t + \tfrac{1}{4}h, w + \tfrac{1}{4}K_1\right)$$

$$K_3 = hf\left(t + \tfrac{3}{8}h, w + \tfrac{3}{32}K_1 + \tfrac{9}{32}K_2\right)$$

$$K_4 = hf\left(t + \tfrac{12}{13}h, w + \tfrac{1932}{2197}K_1 - \tfrac{7200}{2197}K_2 + \tfrac{7296}{2197}K_3\right)$$

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3 $K_1 = hf(t, w)$

$$K_2 = hf\left(t + \tfrac{1}{4}h, w + \tfrac{1}{4}K_1\right)$$

$$K_3 = hf\left(t + \tfrac{3}{8}h, w + \tfrac{3}{32}K_1 + \tfrac{9}{32}K_2\right)$$

$$K_4 = hf\left(t + \tfrac{12}{13}h, w + \tfrac{1932}{2197}K_1 - \tfrac{7200}{2197}K_2 + \tfrac{7296}{2197}K_3\right)$$

$$K_5 = hf\left(t + h, w + \tfrac{439}{216}K_1 - 8K_2 + \tfrac{3680}{513}K_3 - \tfrac{845}{4104}K_4\right)$$

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3 $K_1 = hf\,(t, w)$

$$K_2 = hf\left(t + \tfrac{1}{4}h, w + \tfrac{1}{4}K_1\right)$$

$$K_3 = hf\left(t + \tfrac{3}{8}h, w + \tfrac{3}{32}K_1 + \tfrac{9}{32}K_2\right)$$

$$K_4 = hf\left(t + \tfrac{12}{13}h, w + \tfrac{1932}{2197}K_1 - \tfrac{7200}{2197}K_2 + \tfrac{7296}{2197}K_3\right)$$

$$K_5 = hf\left(t + h, w + \tfrac{439}{216}K_1 - 8K_2 + \tfrac{3680}{513}K_3 - \tfrac{845}{4104}K_4\right)$$

$$K_6 = hf\left(t + \tfrac{1}{2}h, w - \tfrac{8}{27}K_1 + 2K_2 - \tfrac{3544}{2565}K_3 + \tfrac{1859}{4104}K_4 - \tfrac{11}{40}K_5\right)$$

Runge-Kutta-Fehlberg Algorithm (Steps 3 & 4)

Step 3 $K_1 = hf\left(t, w\right)$

$$K_2 = hf\left(t + \tfrac{1}{4}h, w + \tfrac{1}{4}K_1\right)$$

$$K_3 = hf\left(t + \tfrac{3}{8}h, w + \tfrac{3}{32}K_1 + \tfrac{9}{32}K_2\right)$$

$$K_4 = hf\left(t + \tfrac{12}{13}h, w + \tfrac{1932}{2197}K_1 - \tfrac{7200}{2197}K_2 + \tfrac{7296}{2197}K_3\right)$$

$$K_5 = hf\left(t + h, w + \tfrac{439}{216}K_1 - 8K_2 + \tfrac{3680}{513}K_3 - \tfrac{845}{4104}K_4\right)$$

$$K_6 = hf\left(t + \tfrac{1}{2}h, w - \tfrac{8}{27}K_1 + 2K_2 - \tfrac{3544}{2565}K_3 + \tfrac{1859}{4104}K_4 - \tfrac{11}{40}K_5\right)$$

Step 4 Set $R = \frac{1}{h}\left|\frac{1}{360}K_1 - \frac{128}{4275}K_3 - \frac{2197}{75240}K_4 + \frac{1}{50}K_5 + \frac{2}{55}K_6\right|$

(*Note:* $R = \frac{1}{h}|\tilde{w}_{i+1} - w_{i+1}|$)

Runge-Kutta-Fehlberg Algorithm (Steps 5 to 9)

Step 5 If $R \leq TOL$ then do Steps 6 & 7:

Step 6 Set $t = t + h$; *(Approximation accepted)*

$$w = w + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5$$

Step 7 OUTPUT (t, w, h)

Runge-Kutta-Fehlberg Algorithm (Steps 5 to 9)

Step 5 If $R \leq TOL$ then do Steps 6 & 7:

Step 6 Set $t = t + h$; *(Approximation accepted)*

$$w = w + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5$$

Step 7 OUTPUT (t, w, h)

Step 8 Set $\delta = 0.84(TOL/R)^{1/4}$

Runge-Kutta-Fehlberg Algorithm (Steps 5 to 9)

Step 5 If $R \leq TOL$ then do Steps 6 & 7:

Step 6 Set $t = t + h$; *(Approximation accepted)*

$$w = w + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5$$

Step 7 OUTPUT (t, w, h)

Step 8 Set $\delta = 0.84(TOL/R)^{1/4}$

Step 9 If $\delta \leq 0.1$ then set $h = 0.1h$
else if $\delta \geq 4$ then set $h = 4h$
else set $h = \delta h$

Runge-Kutta-Fehlberg Algorithm (Steps 10 to 12)

Step 10 If $h > hmax$ then set $h = hmax$

Runge-Kutta-Fehlberg Algorithm (Steps 10 to 12)

Step 10 If $h > hmax$ then set $h = hmax$

Step 11 If $t \geq b$ then set FLAG $= 0$

else if $t + h > b$ then set $h = b - t$

else if $h < hmin$ then set FLAG $= 0$
OUTPUT (‘*minimum h exceeded*’)
(*Procedure completed unsuccessfully*)

Runge-Kutta-Fehlberg Algorithm (Steps 10 to 12)

Step 10 If $h > hmax$ then set $h = hmax$

Step 11 If $t \geq b$ then set FLAG $= 0$

else if $t + h > b$ then set $h = b - t$

else if $h < hmin$ then set FLAG $= 0$
OUTPUT (‘*minimum h exceeded*’)
(*Procedure completed unsuccessfully*)

Step 12 (*The procedure is complete*)

STOP

Application of the Runge-Kutta-Fehlberg Method

Example

Use the Runge-Kutta-Fehlberg method with a tolerance $TOL = 10^{-5}$, a maximum step size $hmax = 0.25$, and a minimum step size $hmin = 0.01$ to approximate the solution to the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \leq t \leq 2, \qquad y(0) = 0.5$$

and compare the results with the exact solution

$$y(t) = (t+1)^2 - 0.5e^t$$

Application of the Runge-Kutta-Fehlberg Method

Example

Use the Runge-Kutta-Fehlberg method with a tolerance $TOL = 10^{-5}$, a maximum step size $hmax = 0.25$, and a minimum step size $hmin = 0.01$ to approximate the solution to the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \leq t \leq 2, \qquad y(0) = 0.5$$

and compare the results with the exact solution

$$y(t) = (t+1)^2 - 0.5e^t$$

Note: We will work through the first step of the calculations and then apply the RKF Algorithm determine the remaining results.

Application of the Runge-Kutta-Fehlberg Method

Solution (1/8)

The initial condition gives $t_0 = 0$ and $w_0 = 0.5$.

Application of the Runge-Kutta-Fehlberg Method

Solution (1/8)

The initial condition gives $t_0 = 0$ and $w_0 = 0.5$. To determine w_1 using $h = 0.25$, the maximum allowable stepsize, we compute

$$k_1 = hf(t_0, w_0) = 0.25\left(0.5 - 0^2 + 1\right) = 0.375$$

Application of the Runge-Kutta-Fehlberg Method

Solution (1/8)

The initial condition gives $t_0 = 0$ and $w_0 = 0.5$. To determine w_1 using $h = 0.25$, the maximum allowable stepsize, we compute

$$\begin{aligned} k_1 &= hf(t_0, w_0) = 0.25\left(0.5 - 0^2 + 1\right) = 0.375 \\ k_2 &= hf\left(t_0 + \frac{1}{4}h, w_0 + \frac{1}{4}k_1\right) \\ &= 0.25\left(\frac{1}{4}0.25, 0.5 + \frac{1}{4}0.375\right) = 0.3974609 \end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (1/8)

The initial condition gives $t_0 = 0$ and $w_0 = 0.5$. To determine w_1 using $h = 0.25$, the maximum allowable stepsize, we compute

$$\begin{aligned}
k_1 &= hf(t_0, w_0) = 0.25\left(0.5 - 0^2 + 1\right) = 0.375 \\
k_2 &= hf\left(t_0 + \frac{1}{4}h, w_0 + \frac{1}{4}k_1\right) \\
&= 0.25\left(\frac{1}{4}0.25, 0.5 + \frac{1}{4}0.375\right) = 0.3974609 \\
k_3 &= hf\left(t_0 + \frac{3}{8}h, w_0 + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \\
&= 0.25\left(0.09375, 0.5 + \frac{3}{32}0.375 + \frac{9}{32}0.3974609\right) = 0.4095383
\end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (2/8)

$$\begin{aligned} k_4 &= hf\left(t_0 + \frac{12}{13}h, w_0 + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\ &= 0.25\left(0.2307692, 0.5 + \frac{1932}{2197}0.375 - \frac{7200}{2197}0.3974609\right. \\ &\qquad \left. + \frac{7296}{2197}0.4095383\right) = 0.4584971 \end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (2/8)

$$
\begin{aligned}
k_4 &= hf\left(t_0 + \frac{12}{13}h, w_0 + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\
&= 0.25\left(0.2307692, 0.5 + \frac{1932}{2197}0.375 - \frac{7200}{2197}0.3974609 \right. \\
&\qquad \left. + \frac{7296}{2197}0.4095383\right) = 0.4584971 \\
k_5 &= hf\left(t_0 + h, w_0 + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right) \\
&= 0.25\left(0.25, 0.5 + \frac{439}{216}0.375 - 8(0.3974609) \right. \\
&\qquad \left. + \frac{3680}{513}0.4095383 - \frac{845}{4104}0.4584971\right) = 0.4658452
\end{aligned}
$$

Application of the Runge-Kutta-Fehlberg Method

Solution (3/8)

$$\begin{aligned} k_6 &= hf\left(t_0 + \frac{1}{2}h, w_0 - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right) \\ &= 0.25\left(0.125, 0.5 - \frac{8}{27}0.375 + 2(0.3974609)\right. \\ &\qquad \left. - \frac{3544}{2565}0.4095383 + \frac{1859}{4104}0.4584971 - \frac{11}{40}0.4658452\right) \\ &= 0.4204789 \end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (4/8)

The two approximations to $y(0.25)$, namely $\tilde{w}_1$ and w_1, are then found to be:

$$\begin{aligned}
\tilde{w}_1 &= w_0 + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \\
&= 0.5 + \frac{16}{135}0.375 + \frac{6656}{12825}0.4095383 + \frac{28561}{56430}0.4584971 \\
&\qquad -\frac{9}{50}0.4658452 + \frac{2}{55}0.4204789 \\
&= 0.9204870
\end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (5/8)

and

$$\begin{aligned} w_1 &= w_0 + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \\ &= 0.5 + \frac{25}{216}0.375 + \frac{1408}{2565}0.4095383 + \frac{2197}{4104}0.4584971 \\ &\qquad -\frac{1}{5}0.4658452 \\ &= 0.9204886 \end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (5/8)

This also implies that

$$R = \frac{1}{0.25}\left|\frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6\right|$$

Application of the Runge-Kutta-Fehlberg Method

Solution (5/8)

This also implies that

$$\begin{aligned} R &= \frac{1}{0.25}\left|\frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6\right| \\ &= 4\left|\frac{1}{360}0.375 - \frac{128}{4275}0.4095383 - \frac{2197}{75240}0.4584971 \right. \\ &\qquad \left. + \frac{1}{50}0.4658452 + \frac{2}{55}0.4204789\right| \\ &= 0.00000621388 \end{aligned}$$

Application of the Runge-Kutta-Fehlberg Method

Solution (5/8)

This also implies that

$$\begin{aligned} R &= \frac{1}{0.25}\left|\frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6\right| \\ &= 4\left|\frac{1}{360}0.375 - \frac{128}{4275}0.4095383 - \frac{2197}{75240}0.4584971\right. \\ &\qquad \left. + \frac{1}{50}0.4658452 + \frac{2}{55}0.4204789\right| \\ &= 0.00000621388 \end{aligned}$$

and

$$q = 0.84\left(\frac{\varepsilon}{R}\right)^{1/4} = 0.84\left(\frac{0.00001}{0.00000621388}\right)^{1/4} = 0.9461033291$$

Application of the Runge-Kutta-Fehlberg Method

Solution (7/8)

Application of the Runge-Kutta-Fehlberg Method

Solution (7/8)

- Since $q < 1$ we can accept the approximation 0.9204886 for $y(0.25)$ ...

Application of the Runge-Kutta-Fehlberg Method

Solution (7/8)

- Since $q < 1$ we can accept the approximation 0.9204886 for $y(0.25)$ ...
- but we should adjust the step size for the next iteration to $h = 0.9461033291(0.25) \approx 0.2365258$.

Application of the Runge-Kutta-Fehlberg Method

Solution (7/8)

- Since $q < 1$ we can accept the approximation 0.9204886 for $y(0.25)$ ...
- but we should adjust the step size for the next iteration to $h = 0.9461033291(0.25) \approx 0.2365258$.
- However, only the leading 5 digits of this result would be expected to be accurate because R has only about 5 digits of accuracy.

Application of the Runge-Kutta-Fehlberg Method

Solution (7/8)

- Since $q < 1$ we can accept the approximation 0.9204886 for $y(0.25)\ldots$
- but we should adjust the step size for the next iteration to $h = 0.9461033291(0.25) \approx 0.2365258$.
- However, only the leading 5 digits of this result would be expected to be accurate because R has only about 5 digits of accuracy.
- Because we are effectively subtracting the nearly equal numbers w_i and $\tilde{w}_i$ when we compute R, there is a good likelihood of round-off error.

Application of the Runge-Kutta-Fehlberg Method

Solution (7/8)

- Since $q < 1$ we can accept the approximation 0.9204886 for $y(0.25) \ldots$
- but we should adjust the step size for the next iteration to $h = 0.9461033291(0.25) \approx 0.2365258$.
- However, only the leading 5 digits of this result would be expected to be accurate because R has only about 5 digits of accuracy.
- Because we are effectively subtracting the nearly equal numbers w_i and $\tilde{w}_i$ when we compute R, there is a good likelihood of round-off error.
- This is an additional reason for being conservative when computing q.

Application of the Runge-Kutta-Fehlberg Method (8/8)

		RKF-4				RKF-5	
t_i	$y_i = y(t_i)$	w_i	h_i	R_i	$\lvert y_i - w_i \rvert$	$\hat{w}_i$	$\lvert y_i - \hat{w}_i \rvert$
0	0.5	0.5			0.5		
0.2500000	0.9204873	0.9204886	0.2500000	6.2×10^{-6}	1.3×10^{-6}	0.9204870	2.424×10^{-7}
0.4865522	1.3964884	1.3964910	0.2365522	4.5×10^{-6}	2.6×10^{-6}	1.3964900	1.510×10^{-6}
0.7293332	1.9537446	1.9537488	0.2427810	4.3×10^{-6}	4.2×10^{-6}	1.9537477	3.136×10^{-6}
0.9793332	2.5864198	2.5864260	0.2500000	3.8×10^{-6}	6.2×10^{-6}	2.5864251	5.242×10^{-6}
1.2293332	3.2604520	3.2604605	0.2500000	2.4×10^{-6}	8.5×10^{-6}	3.2604599	7.895×10^{-6}
1.4793332	3.9520844	3.9520955	0.2500000	7×10^{-7}	1.11×10^{-5}	3.9520954	1.096×10^{-5}
1.7293332	4.6308127	4.6308268	0.2500000	1.5×10^{-6}	1.41×10^{-5}	4.6308272	1.446×10^{-5}
1.9793332	5.2574687	5.2574861	0.2500000	4.3×10^{-6}	1.73×10^{-5}	5.2574871	1.839×10^{-5}
2.0000000	5.3054720	5.3054896	0.0206668		1.77×10^{-5}	5.3054896	1.768×10^{-5}

For small values of t, the error in the 5th-order method is less than the error in the 4th-order method, but the error exceeds that of the 4th-order method when t increases.