Ecuaciones Diferenciales Ordinarias (2)

Yarko Niño C. y Paulo Herrera R.

Departamento de Ingeniería Civil, Universidad de Chile

Semestre Primavera 2011

Resumen clase pasada

Solución del problema:

$$y' = f(t, y) \qquad \text{con} \quad y(t_0) = y_0$$

Método de Euler
Despreciando términos de orden superior a h,

$$\hat{y}(t + h) \approx y(t) + hf(t, y)$$

En forma discreta,

$$\hat{y}_{k+1} = \hat{y}_k + h_k f(t_k, \hat{y}_k)$$

Método de Euler en Reversa

- Método de Euler es **explícito** porque sólo utiliza información en tiempo t_k para calcular estado en tiempo t_{k+1}.
- Desventaja es que tiene limitada estabilidad.

Qué pasa si $f(t, y)$ es evaluada en (t_{k+1}, y_{k+1})?

Método de Euler en Reversa (*Backward Euler*)

$$\hat{y}_{k+1} = \hat{y}_k + h_k f(t_{k+1}, \hat{y}_{k+1})$$

- Esto implica que se deben resolver ecuaciones algebraicas.
- Método es **implícito**.

Ejemplo: Método de Euler en Reversa

$$y' = -y^3; \qquad y(0) = 1; \qquad h = 0{,}5$$

$$y_1 = y_0 + hf(t_1, y_1) = 1 - 0{,}5y_1^3$$

$$\Rightarrow 0{,}5y_1^3 + y_1 - 1 = 0$$

entonces, es necesario encontrar raíces de una ecuación cúbica.

Newton-Raphson

$$g(x) = 0; \qquad x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}; \qquad x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}$$

Para seleccionar valor inicial x_0 se puede utilizar valor a t_k ($\hat{y}_k$) o utilizar Euler para calcular una aproximación de $\hat{y}_{k+1}$.

Método de Euler en Reversa (cont.)

Por qué pagar un costo extra por evaluar $f(t_{k+1}, y_{k+1})$?

Considerando $y' = \lambda y$,

$$\hat{y}_{k+1} = \hat{y}_k + h\lambda\hat{y}_{k+1} \quad \Rightarrow (1 - h\lambda)\,\hat{y}_{k+1} = \hat{y}_k$$

Iteración,

$$\hat{y}_k = \left(\frac{1}{1 - h\lambda}\right)^k \hat{y}_0$$

Entonces, solución $\hat{y}_k$ es estable si,

$$\left|\frac{1}{1 - h\lambda}\right| \leq 1$$

Si $\lambda < 0$ se cumple para cualquier $h > 0 \Rightarrow$ método es **incondicionalmente estable**.

Método de Euler en Reversa (cont.)

Iteración,

$$\hat{y}_k = \left(\frac{1}{1-h\lambda}\right)^k \hat{y}_0$$

Expandiendo el factor de desarrollo,

$$\frac{1}{1-h\lambda} = 1 + h\lambda + (h\lambda)^2 + \dots$$

pero la verdadera solución,

$$e^{h\lambda} = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \dots$$

Entonces, **método de Euler en reversa es de primer orden**. Como método es incondicionalmente estable, único criterio para seleccionar h_k es la exactitud de la solución.

Método del Trapezoide

Euler explícito e implícito son aproximaciones de primer orden, pero qué pasa si los mezclamos?

$$\hat{y}_{k+1} = \hat{y}_k + h_k \left[f(t_k, \hat{y}_k) + f(t_{k+1}, \hat{y}_{k+1}) \right] / 2$$

Tomando el caso canónico $y' = \lambda y$,

$$\hat{y}_{k+1} = y_k + h \left(\lambda \hat{y}_k + \lambda \hat{y}_{k+1} \right) / 2$$

es decir,

$$\hat{y}_k = \left(\frac{1 + h\lambda/2}{1 - h\lambda/2} \right)^k y_0$$

Entonces la solución es estable si,

$$\left| \frac{1 + h\lambda/2}{1 - h\lambda/2} \right| < 1$$

esto se cumple para todo $h > 0$ si $\lambda < 0 \rightarrow$ método **es incondicionalmente estable**.

Método del Trapezoide (cont.)

Expandiendo el factor de desarrollo,

$$\frac{1 + h\lambda/2}{1 - h\lambda/2} = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{4} + \dots$$

versus,

$$e^{h\lambda} = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \dots$$

Entonces, **método es de segundo orden**.

Comentarios:

- **En general**, métodos implícitos tienen un rango de estabilidad mayor que el de métodos explícitos, **pero**
- Implícito $\neq$ incondicionalmente estable.

Método de resolución para EDOs

Método de Euler importante desde un punto de vista conceptual, pero *inútil* en la práctica (al menos, los libros lo dicen!).

Existen varios otros métodos que, aunque más sofisticados, de alguna u otra forma están basados en el método de Euler.

Tipos de métodos:

- Series de Taylor
- Runge-Kutta
- *Multistep* (predictor-corrector)
- Extrapolación
- *Multivalue*

Series de Taylor

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t) + \dots$$

Despreciando términos de orden superior a h^2,

$$\hat{y}_{k+1} = y_k + h_k y'_k + \frac{h_k^2}{2}y''_k$$

Cómo calculamos $y''(t)$?

$$y''(t) = f_t(t,y) + f_y(t,y)y' = f_t(t,y) + f_y(t,y)f(t,y)$$

Expresiones para métodos de orden más alto son más complicadas.
Alternativa, es usar diferenciación numérica.

Runge-Kutta

Runge-Kutta métodos son similares a aproximaciones basadas en Series de Taylor pero aproximan las derivadas usando diferencias finitas calculadas con valores de f evaluados en puntos intermedios entre t_k y t_{k+1}.

Por ejemplo, el método de Runge-Kutta de segundo orden puede derivarse comenzando por,

$$y''(t) = f_t(t, y) + f_y(t, y)y'$$

Por otra parte, la serie de Taylor en dos variables de f es,

$$f(t + h, y + hf) = f(t, y) + hf_t(t, y) + hf_y(t, y)f(t, y) + O(h^2)$$

entonces,

$$f_t(t, y) + f_y(t, y)f(t, y) = \frac{f(t + h, y + hf) - f(t, y)}{h} + O(h)$$

Runge-Kutta (cont.)

Sustituyendo la aproximación para la segunda derivada, obtenemos

$$\begin{aligned}\hat{y}_{k+1} &= \hat{y}_k + h_k f(t_k, \hat{y}_k) + \frac{h_k^2}{2} \frac{f(t_k + h_k, \hat{y}_k + h_k f(t_k, \hat{y}_k)) - f(t_k, \hat{y}_k)}{h_k} \\ &= \hat{y}_k + \frac{h_k}{2} \left[f(t_k, \hat{y}_k) + f(t_k + h_k, \hat{y}_k + h_k f(t_k, \hat{y}_k)) \right]\end{aligned}$$

Definiendo,

$$k_1 = f(t_k, \hat{y}_k) \qquad k_2 = f(t_k + h_k, \hat{y}_k + h_k k_1)$$

Obtenemos el método de Heun,

$$\hat{y}_{k+1} = \hat{y}_k + \frac{h_k}{2}(k_1 + k_2)$$

Runge-Kutta (cont.)

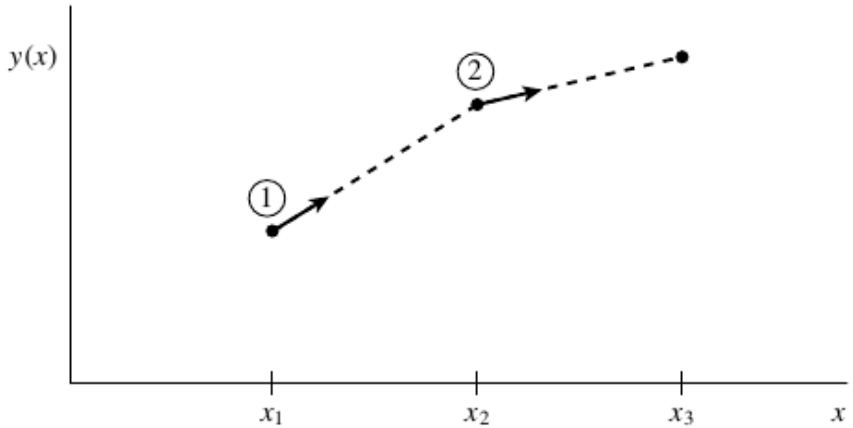


Figure 16.1.1. Euler's method. In this simplest (and least accurate) method for integrating an ODE, the derivative at the starting point of each interval is extrapolated to find the next function value. The method has first-order accuracy.

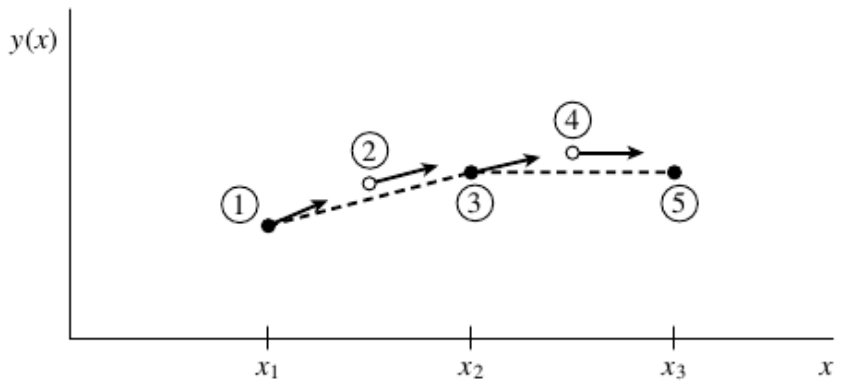


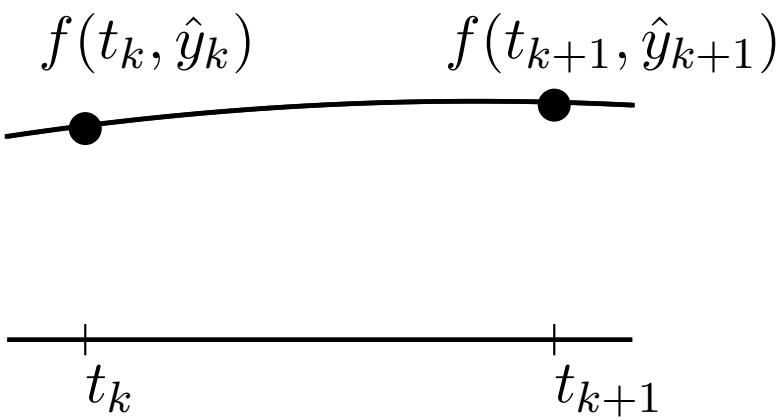
Figure 16.1.2. Midpoint method. Second-order accuracy is obtained by using the initial derivative at each step to find a point halfway across the interval, then using the midpoint derivative across the full width of the interval. In the figure, filled dots represent final function values, while open dots represent function values that are discarded once their derivatives have been calculated and used.

Runge-Kutta (cont.)

Por Teorema Fundamental del Cálculo, la solución de la EDO $y' = f(t, y)$ con $y(t = t_0) = y_0$, se puede escribir como,

$$y(t) = y_0 + \int_{t_0}^{t} f(t, y(\tau))\, \mathrm{d}\tau$$

Las aproximaciones del tipo Runge-Kutta pueden ser intepretados como métodos de integración basados en cuadraturas.



Método de Heun calcula la integral bajo la curva usando un trapezoide, pero aproximando

$$\hat{y}_{k+1} = \hat{y}_k + h_k k_1$$

Runge-Kutta (cont.)

Es posible aplicar la misma metodología para obtener métodos de orden más alto.

Por ejemplo, Runge-Kutta de orden 4,

$$\hat{y}_{k+1} = \hat{y}_k + \frac{h_k}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

donde,

$$\begin{aligned} k_1 &= f(t_k, \hat{y}_k) \\ k_2 &= f(t_k + h_k/2, y_k + h_k k_1/2) \\ k_3 &= f(t_k + h_k/2, y_k + h_k k_2/2) \\ k_4 &= f(t_k + h_k, y_k + h_k k_3) \end{aligned}$$

Runge-Kutta (cont.)

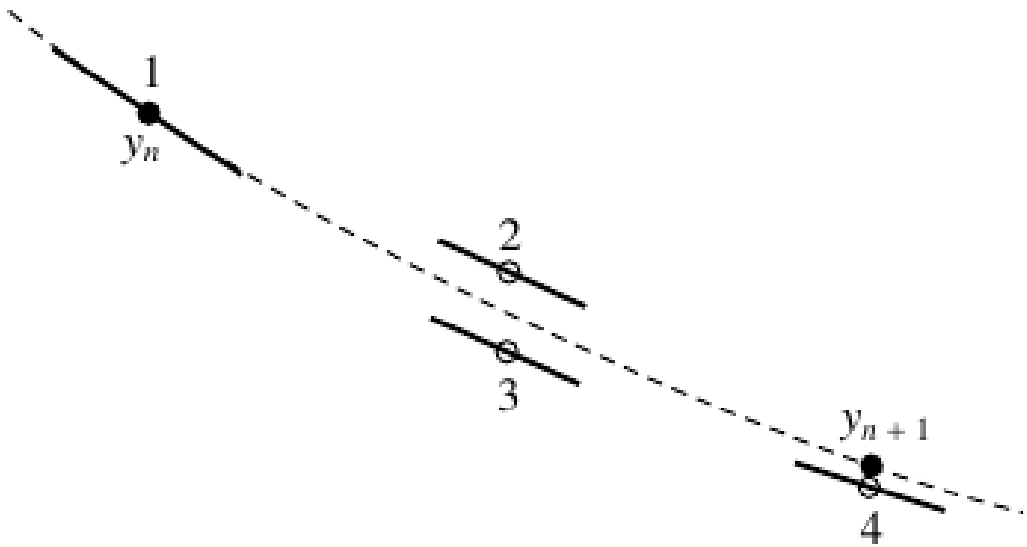


Figure 16.1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)

Control automático de discretización

Aplicar aproximación de Runge-Kutta de orden 4 para calcular solución usando h y $2h$,

$$\hat{y}(x+2h) = \hat{y}_1 + (2h)^5 C + O(h_6)$$
$$\hat{y}(x+2h) = \hat{y}_2 + 2(h)^5 C + O(h_6)$$

donde $\hat{y}_1$ aproximación en un paso $2h$ e $\hat{y}_2$ aproximación en dos pasos h. Definiendo $\Delta \equiv \hat{y}_2 - \hat{y}_1$, tenemos una estimación del error de truncación de la aproximación $\hat{y}_1$.

Si $\Delta > tol$ reducir h, si $\Delta < tol$ aumentar h.

Este es el principio para implementar algoritmos con **control automático de error**.

Control automático de discretización (cont.)

Comentarios,

- En general, el mismo procedimiento de refinamiento de la grilla numérica se puede aplicar para verificar convergencia de la solución con **cualquier método numérico**.

- La estimación del error de truncación es útil para **medir la calidad de la solución numérica**, sin embargo,

- **requiere cálculos adicionales**, lo que motivo el desarrollo de otros métodos que utilizan las mismas evaluaciones de f para calcular aproximaciones de distinto orden (por ej. Runge-Kutta-Fehlberg).

Control automático de discretización (cont.)

Teníamos,

$$\hat{y}(x+2h) = \hat{y}_1 + (2h)^5 C + O(h_6)$$
$$\hat{y}(x+2h) = \hat{y}_2 + 2(h)^5 C + O(h_6)$$

Combinando las expresiones para pasos h y $2h$ obtenemos,

$$\hat{y}(x+2h) = \hat{y}_2 + \frac{\Delta}{15} + O(h^6)$$

Esta última expresión es de orden 5, pero no tenemos una estimación de su error, por lo cual no es muy útil en la práctica.

Métodos de extrapolación

- Calcular solución con distintos pasos h_i de tal forma de obtener una serie de aproximaciones $Y_i = Y(h_i)$.
- Interpolar para obtener una aproximación continua, $Y = Y(h)$.
- Calcular la solución $y_{k+1} = Y(0)$.
- Estos métodos entregan soluciones **muy precisas**, pero a un **alto costo**.
- Se utilizan en problemas que **requieren soluciones de muy buena calidad**.

Métodos de extrapolación (cont.)

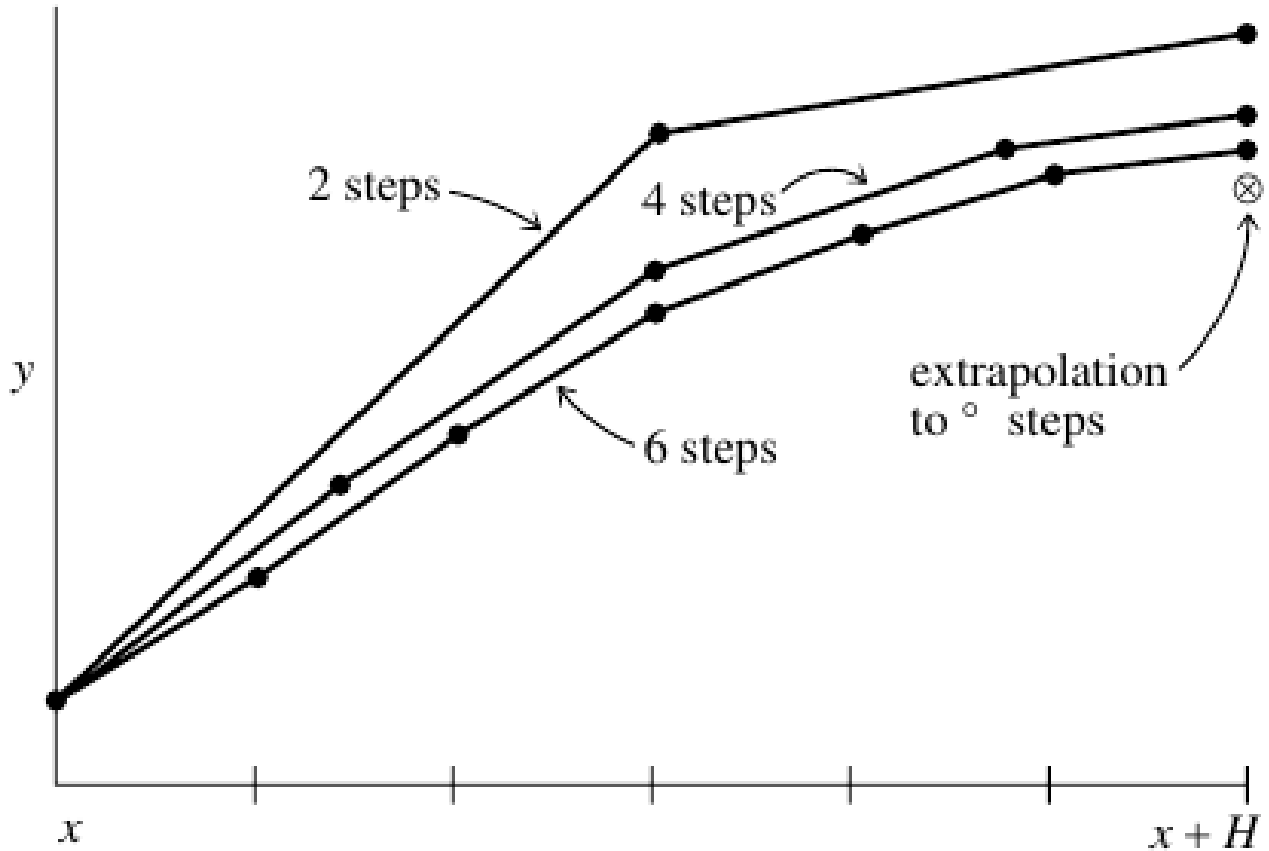


Figure 16.4.1. Richardson extrapolation as used in the Bulirsch-Stoer method. A large interval H is spanned by different sequences of finer and finer substeps. Their results are extrapolated to an answer that is supposed to correspond to infinitely fine substeps. In the Bulirsch-Stoer method, the integrations are done by the modified midpoint method, and the extrapolation technique is rational function or polynomial extrapolation.

Métodos *multisteps*

Usan información de varios niveles anteriores para estimar la solución en el próximo nivel (por eso se les llama **métodos con memoria**).

Aproximación calculada como,

$$\hat{y}_{k+1} = \sum_{i=1}^{m} \alpha_i \hat{y}_{k+1-i} + h \sum_{i=1}^{m} \beta_i f(t_{k+1-i}, \hat{y}_{k+1-i})$$

Método explícito si $\beta_0 = 0$, pero implícito si $\beta_0 \neq 0$.

Métodos **predictor-corrector** son ejemplos de este tipo de métodos (mirar apuntes del curso).

Métodos para Problemas de Valor Inicial

Métodos más comunes corresponden a tres familias,

- *Runge-Kutta*: propagar la información de la solución en un intervalo aplicando varias aproximaciones tipo Euler para sustituir en series de Taylor de algún orden.

- *Extrapolación*: suponer que es posible extrapolar aproximaciones al valor que **tendrían** si el paso fuera infinitamente pequeño.

- *Predictor-corrector*: almacenar las soluciones de niveles previos para extrapolar (o **predecir**)la solución al siguiente nivel. La solución extrapolada puede ser **corregida** usando la infomación de la derivada evaluada en el nuevo nivel.

Métodos para Problemas de Valor Inicial (cont.)

- Existen **muchos métodos derivados** de esas tres ideas básicas. También hay otros.
- Cada uno de esos métodos tiene ventajas y desventajas. En general, se debe considerar balance entre **exactitud vs esfuerzo**.
- Métodos de Runge-Kutta son los más comunes y fáciles de implementar.
- Existen varias bibliotecas científicas que implementan estos algoritmos $\Rightarrow$ **no hay necesidad de reinventar la rueda!**.
- Sin embargo, es bueno saber las principales propiedades de cada método para seleccionar el algoritmo de resolución óptimo.

- Scilab incluye una función **ode** que implementa varios de estos métodos. Por defecto, la rutina trata de seleccionar el **algoritmo óptimo** para cada problema.

Problemas de Valores de Borde

Cuando una EDO debe satisfacer condiciones de borde en más de un punto, el problema se denomina **Problema de Valores de Borde** (*Boundary Value Problem*).

Formalmente, el problema es definido por el conjunto de N EDOs de primer orden,

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = g_i(x, y_1, y_2, \ldots, y_N) \quad i = 1, 2, \ldots, N$$

sujetas a las condiciones de borde en x_1 y x_2,

$$B_{1j}(x_1, y_1, y_2, \ldots, y_N) = 0 \quad j = 1, \ldots, n_1$$

$$B_{2k}(x_2, y_1, y_2, \ldots, y_N) = 0 \quad k = 1, \ldots, n_2$$

con $n_2 = N - n_1$.

Problemas de Valores de Borde (cont.)

Diferencia con problemas de valor inicial es que en este caso la solución no es determinada sólo por condición inicial $\Rightarrow$ **no es posible comenzar a avanzar desde uno de los bordes para calcular una solución única**.

Estrategia de solución se basa en **algoritmos iterativos** para aproximar posibles soluciones a una solución única que satisface todas las condiciones de borde.

Esto significa que la solución de este tipo de problemas **requiere más esfuerzo computacional** que la solución de problemas de valor inicial.

Problemas de Valores de Borde (cont.)

Métodos de resolución

- *Shooting Method*
- *Collocation Method*
- Diferencias Finitas
- Elementos Finitos

Shooting Method

Estrategia general: *Disparar* una solución desde un punto para tratar de ajustar la solución en otro punto.
Existen dos estrategias:

1. Avanzar (*disparar*) desde el punto inicial utiliza un algoritmo para resolver problemas de valor inicial. En el punto inicial sólo se satisfacen n_1 condiciones de borde, por lo que existen $N - n_1$ pará metros que pueden ser ajustados para alcanzar el objetivo en el otro punto con condiciones de borde conocidas.

2. Avanzar desde dos puntos extremos para alcanzar un punto intermedio. Ajustar parámetros libres de tal forma que la solución sea continua en el punto de encuentro.

Shooting Method (cont.)

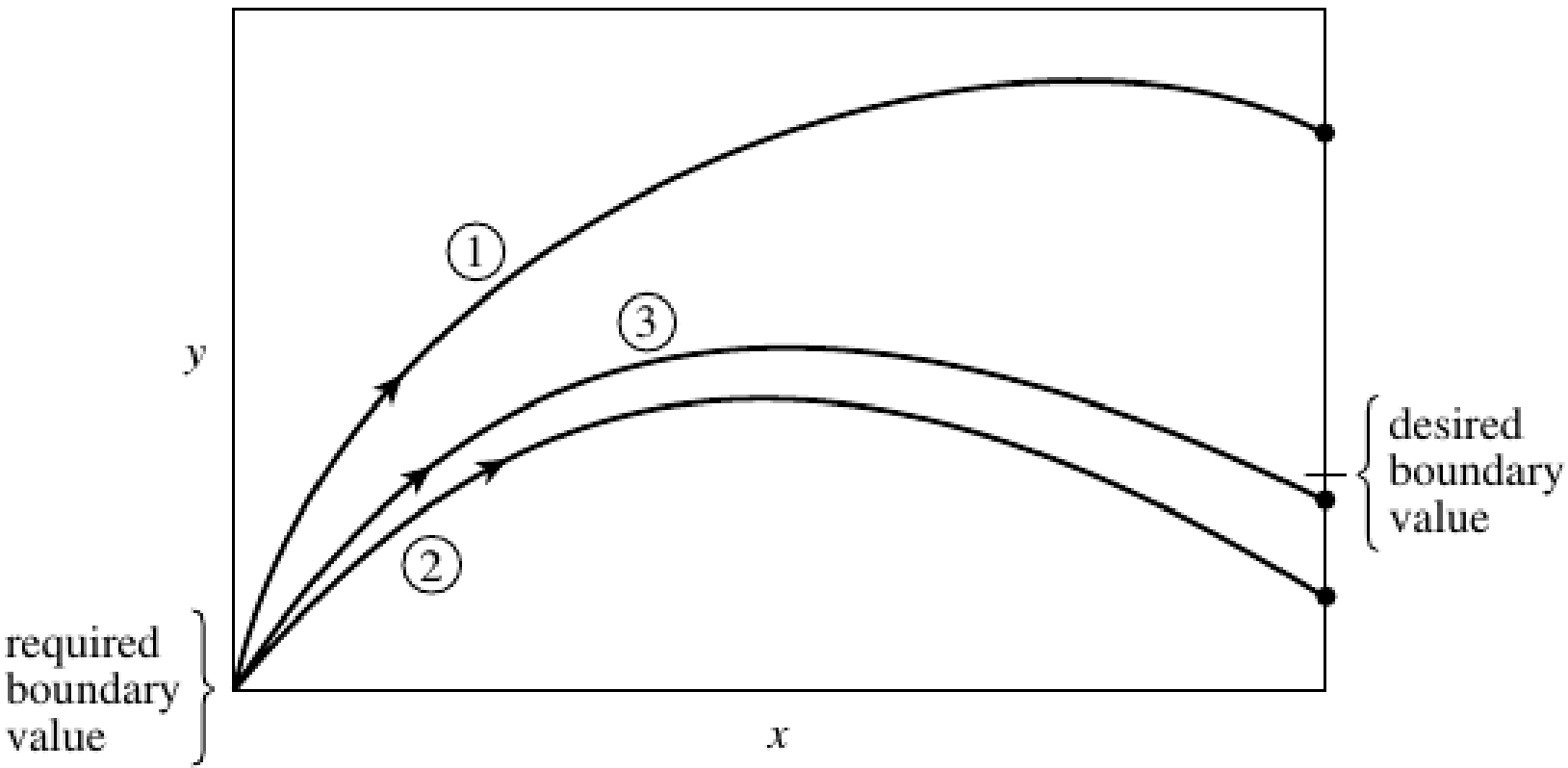


Figure 17.0.1. Shooting method (schematic). Trial integrations that satisfy the boundary condition at one endpoint are "launched." The discrepancies from the desired boundary condition at the other endpoint are used to adjust the starting conditions, until boundary conditions at both endpoints are ultimately satisfied.

Shooting Method: Estrategia 1

1. En el punto inicial x_1 se requiere especificar N valores iniciales de y_i, pero sólo hay n_1 condiciones de borde.
2. Entones, podemos especificar los valores iniciales como funciones de un vector de parámetros $\mathbf{V}$ de tal forma que los valores iniciales en x_1 son,

$$y_i(x_1) = y_i(x_1; V_1, \ldots, V_{n_2}) \quad i = 1, \ldots, N$$

3. Ahora integramos la EDO para obtener un conjunto de soluciones $\mathbf{y}(x_2)$.
4. En x_2 la solución puede no satisfacer las n_2 condiciones de borde, por lo que definimos un vector de discrepancia F. En general,

$$F_k = B_{2k}(x_2, \mathbf{y})$$

5. Ahora el problema es encontrar un vector $\mathbf{V}^*$ tal que $\mathbf{F} = 0 \Rightarrow$ Newton-Raphson.

Shooting Method: Estrategia 1 (cont.)

Newton-Raphson:

1. Encontrar incremento en la solución,

$$\mathbf{J} \cdot \delta\mathbf{V} = -\mathbf{F}, \qquad J_{ij} = \frac{\partial F_i}{\partial V_j}$$

2. Jacobiano es evaluado en forma numérica,

$$\frac{\partial F_i}{\partial V_j} \approx \frac{F_i(V_1, \ldots, V_j + \Delta V_j, \ldots) - F_i(V_1, \ldots, V_j, \ldots)}{\Delta V_j}$$

3. Actualizar solución,

$$\mathbf{V}^{\text{new}} = \mathbf{V}^{\text{old}} + \delta\mathbf{V}$$