UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Mechanical Design 444
System Simulation Notes

Numerical Methods

Danie Els

Dept of Mech & Mechatron Eng
University of Stellenbosch

2017

Version 1.0

Contents

1. Root finding

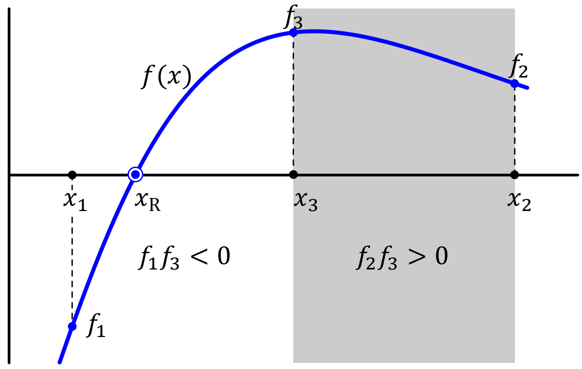

Figure 1: Bisection method for zero finding

1.1. Bisection Method

Assume $f(x)$ is a continuous real valued function and there are two real numbers x_1 and x_2 such that $f(x_1)f(x_2) < 0$. Then $f(x)$ has at least one root $f(x_R) = 0$, with $x_1 < x_R < x_2$.

The bisection procedure starts with an interval $[x_1, x_2]$ that brackets a root. The interval is halved and the halve where $f(x)$ changes sign is kept. The process is repeated unit the interval shrinks to required accuracy for the root. The benefit of the bisection method is that it must succeed. If the interval happens to contain two or more roots, bisection will find one of them. If the interval contains no roots and merely straddles a singularity, it will converge on the singularity.

Algorithm 1 Bisection method for zero finding

```
[x_1 ; x_2]                          // Set search bracket
f_1 ← f(x_1)
f_2 ← f(x_2)
for i ← 1 to i_max do
    x_3 ← (x_1 + x_2)/2              // Midpoint
    f_3 ← f(x_3)

    if f_1 f_3 < 0 then              // Bracket root in [x_1, x_3]
        x_2 ← x_3
        f_2 ← f_3
    else                             // Bracket root in [x_3, x_2]
        x_1 ← x_3
        f_1 ← f_3
    end if
    if |x_2 − x_1| ≤ ε then          // Check for root
        return x_3
    end if

end for
return Error: i_max reached
```

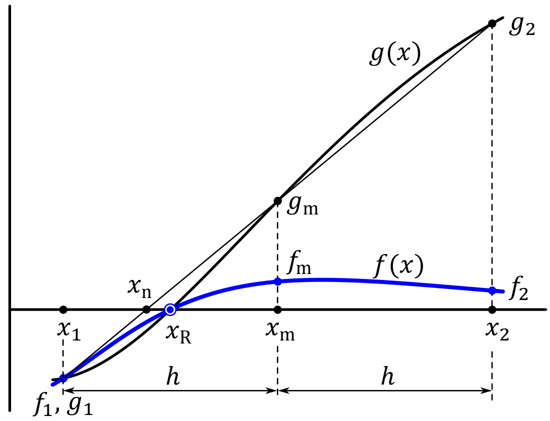

Figure 2: Ridders' method for zero finding

1.2. Ridders' method for zero finding

The method of Ridders (1979) is a modification of Regula Falsi method. Assume that the root is bracketed in $[x_1, x_2]$ or $f(x_1)f(x_2) < 0$. Define the midpoint

$$x_{\mathrm{m}} = \tfrac{1}{2}(x_1 + x_2) \tag{1}$$

Define the function

$$g(x) = f(x)\ \mathrm{e}^{(x-x_1)Q} \tag{2}$$

where the constant Q is determined by requiring that the points $\big(x_1, g(x_1)\big)$, $\big(x_{\mathrm{m}}, g(x_{\mathrm{m}})\big)$ and $\big(x_2, g(x_2)\big)$ lies on a straight line or $g(x_{\mathrm{m}}) = [g(x_1) + g(x_2)]/2$, resulting in

$$f(x_{\mathrm{m}})\,\mathrm{e}^{hQ} = \tfrac{1}{2}\left(f(x_1) + f(x_2)\,\mathrm{e}^{2hQ}\right) \tag{3}$$

where $h = (x_2 - x_1)/2$. This equation is quadratic in e^{hQ}, with

$$\mathrm{e}^{hQ} = \frac{f(x_{\mathrm{m}}) \pm \sqrt{f(x_{\mathrm{m}})^2 - f(x_1)\,f(x_2)}}{f(x_2)} \tag{4}$$

Linear interpolation now yield a new guess for the root, x_{n}

$$x_{\mathrm{n}} = x_{\mathrm{m}} - g(x_{\mathrm{m}})\,\frac{x_{\mathrm{m}} - x_1}{g(x_{\mathrm{m}}) - g(x_1)} = x_{\mathrm{m}} - f(x_{\mathrm{m}})\,\mathrm{e}^{hQ}\,\frac{x_{\mathrm{m}} - x_1}{f(x_{\mathrm{m}})\,\mathrm{e}^{hQ} - f(x_1)} \tag{5}$$

Substitute e^{hQ} from equation (4) then after some algebra the new guess for the root is

$$x_{\mathrm{n}} = x_{\mathrm{m}} + (x_{\mathrm{m}} - x_1)\frac{\operatorname{sgn}\left[f(x_{\mathrm{m}}) - f(x_1)\right]f(x_{\mathrm{m}})}{\sqrt{f(x_{\mathrm{m}})^2 - f(x_1)\,f(x_2)}} \tag{6}$$

Equation (6) has some nice properties. First x_{n} is always in the bracket $[x_1, x_2]$. Secondly, the convergence of successive applications of (6) is quadratic.

The next step is to test for convergence, $|f(x_{\mathrm{n}})| \le \varepsilon$, else bracket the root again by establishing if it is in $[x_{\mathrm{m}}, x_{\mathrm{n}}]$ or $[x_1, x_{\mathrm{n}}]$ or $[x_{\mathrm{n}}, x_2]$ and repeat the process (See algorithm 2).

Algorithm 2 Ridders' method for zero finding

$[x_1\,; x_2]$ *// Set search bracket*
$f_1 \leftarrow f(x_1)$
$f_2 \leftarrow f(x_2)$
for $i \leftarrow 1$ **to** $i_{\max}$ **do**
  $x_\mathrm{m} \leftarrow (x_1 + x_2)/2$ *// Midpoint*
  $f_\mathrm{m} \leftarrow f(x_\mathrm{m})$
  $s \leftarrow \sqrt{f_\mathrm{m}^2 - f_1 f_2}$

  if $s = 0.0$ **then**
    return *Error: division by zero*
  end if

  $x_\mathrm{n} \leftarrow x_\mathrm{m} + (x_\mathrm{m} - x_1)\,\mathrm{sgn}\left(f_\mathrm{m} - f_1\right) f_\mathrm{m}/s$ *// New value*
  $f_\mathrm{n} \leftarrow f(x_\mathrm{n})$

  if $|f_\mathrm{n}| \leq \varepsilon$ **then** *// Check for root*
    return x_n
  end if

  if $f_\mathrm{m} f_\mathrm{n} < 0$ **then** *// Bracket root in* $[x_m, x_n]$
    $x_1, f_1 \leftarrow x_\mathrm{m}, f_\mathrm{m}$
    $x_2, f_2 \leftarrow x_\mathrm{n}, f_\mathrm{n}$
  else if $f_1 f_\mathrm{n} < 0$ **then** *// Bracket root in* $[x_1, x_n]$
    $x_2, f_2 \leftarrow x_\mathrm{n}, f_\mathrm{n}$
  else *// Bracket root in* $[x_n, x_2]$
    $x_1, f_1 \leftarrow x_\mathrm{n}, f_\mathrm{n}$
  end if

end for
return *Error:* $i_{\max}$ *reached*

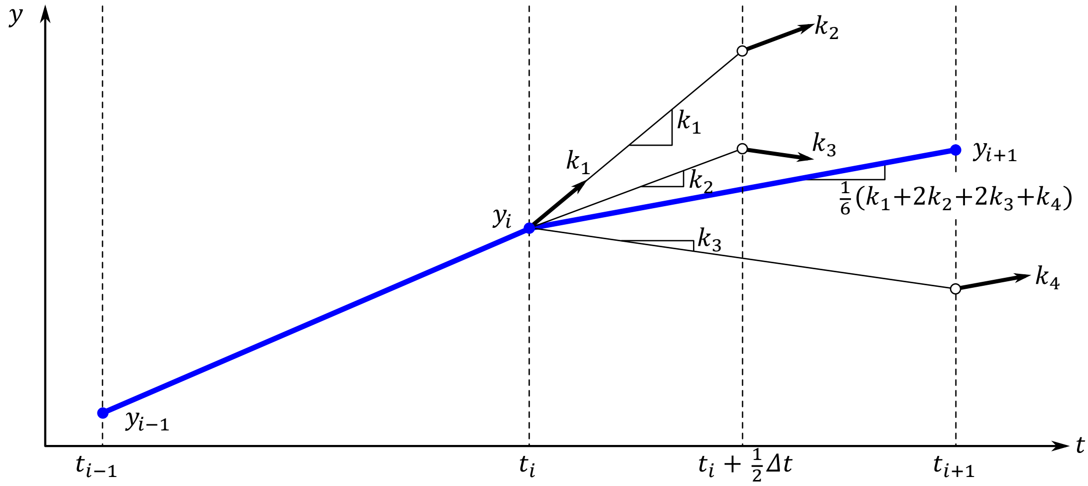

Figure 3: Runge-Kutta 4th order method

2. Runge-Kutta 4th Order Algorithm

Consider the ordinary differential equation (ODE) given as the initial value problem

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = F\big(t, y(t)\big), \qquad y(t_0) = y_0, \tag{7}$$

The ODE can be solved over an increment Δt with

$$y(t+\Delta t) = y(t) + \int_t^{t+\Delta t} F\big(\tau, y(\tau)\big)\,\mathrm{d}\tau \tag{8}$$

For numerical integration with an explicit Runge-Kutta method, equation (8) can be approximated with a quadrature. Let $y_i \approx y(t_i)$ and $y_{i+1} \approx y(t_i + \Delta t)$ then

$$y_{i+1} = y_i + \Delta t \sum_{n=1}^{N} \omega_n\, F\big(t_i+\alpha_n\Delta t,\ y(t_i+\alpha_n\Delta t)\big) \tag{9}$$

with N the order of the method, ω_n the weights and α_n the position of the nodes. Determination of the coefficients ω_i and α_i is rather complicated.

4th Order method: The classical 4th order Runge-Kutta method (without proof) is given by

$$y_{i+1} = y_i + \frac{\Delta t}{6}\,(k_1 + 2k_2 + 2k_3 + k_4) \tag{10a}$$

$$t_{i+1} = t_i + \Delta t \tag{10b}$$

with coefficients

$$k_1 = F\big(t_i,\qquad y_i\big) \tag{11a}$$

$$k_2 = F\big(t_i + \tfrac{1}{2}\Delta t,\quad y_i + \tfrac{1}{2}\Delta t\,k_1\big) \tag{11b}$$

$$k_3 = F\big(t_i + \tfrac{1}{2}\Delta t,\quad y_i + \tfrac{1}{2}\Delta t\,k_2\big) \tag{11c}$$

$$k_4 = F\big(t_i + \Delta t,\quad y_i + \Delta t\,k_3\big) \tag{11d}$$

Algorithm 3 Fourth order Runge-Kutta

$F(t, y(t)) \leftarrow \dfrac{\mathrm{d}y(t)}{\mathrm{d}t}, \qquad t_a \leq t \leq t_b$ // *Set differential equation function*
$t_0 \leftarrow t_a$ // *Set initial time* $t = t_a$
$y_0 \leftarrow y(t_a)$ // *Set initial function value at* $t = t_a$
$\Delta t \leftarrow (t_b - t_a)/N$ // *Set time step*
$i \leftarrow 0$
while $t_i \leq t_b$ **do**
  $k_1 \leftarrow F\left(t_i, \quad y_i\right)$ //
  $k_2 \leftarrow F\left(t_i + \frac{1}{2}\Delta t, \quad y_i + \frac{1}{2}\Delta t\, k_1\right)$ // *Coefficients*
  $k_3 \leftarrow F\left(t_i + \frac{1}{2}\Delta t, \quad y_i + \frac{1}{2}\Delta t\, k_2\right)$ //
  $k_4 \leftarrow F\left(t_i + \Delta t, \quad y_i + \Delta t\, k_3\right)$ //
  $y_{i+1} \leftarrow y_i + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ // *Update* $y(t)$
  $t_{i+1} \leftarrow t_i + \Delta t$ // *Update t*
  $i \leftarrow i + 1$
end while

References

Ridders, C.F.J. (1979). A new algorithm for computing a single root of a real continuous function. *IEEE Transactions on Circuits and Systems*, vol. 26, no. 11, pp. 979–980.