Numerical Analysis

More on Newton-Cotes Integration Formula

목 차

1 Another Form of Newton Interpolation

2 Generalization of Newton-Cotes Interpolation Formula

3 Computer Model for General Newton-Cotes Integration

4 Program Structure and Application Test

Another Form of Newton Interpolation

Newton's Interpolating Polynomial (1)

(1) Given Data $\{(x_0,y_0),(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_j,y_j),\cdots,(x_n,y_n)\}=\{(x_j,y_j)\}_{j=0}^{j=n}$

(2) Naïve Newton Interpolation Polynomial

$$\begin{aligned} f(x;\mathbf{a}) &= a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+a_3(x-x_0)(x-x_1)(x-x_2) \\ &\quad +\cdots+a_n(x-x_0)(x-x_1)(x-x_2)\cdots(x-x_{n-2})(x-x_{n-1}) \end{aligned}$$

(3) Expression using the Non-Dimensional Independent Variables

$$\tau=\frac{x-x_0}{x_n-x_0}\in[0,1]$$

$$x=x_0+(x_n-x_0)\tau=x_0+\Delta x\tau \quad \leftarrow \Delta x=(x_n-x_0)$$

$$x_j=x_0+(x_n-x_0)\tau_j \quad (j=0,1,2,\cdots,n)$$

(4) Newton Interpolation Polynomial using the Non-Dimensional Independent Variables

$$\begin{aligned} f(\tau;\mathbf{a}) &= a_0+a_1(\tau-\tau_0)+a_2(\tau-\tau_0)(\tau-\tau_1)+a_3(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2) \\ &\quad +\cdots+a_n(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2)\cdots(\tau-\tau_{n-2})(\tau-\tau_{n-1}) \end{aligned}$$

Another Form of Newton Interpolation

Newton's Interpolating Polynomial (2)

(4) Newton Interpolation Polynomial using the Non-Dimensional Independent Variables

$$\begin{aligned} f(\tau;\mathbf{a}) = {} & a_0 + a_1(\tau-\tau_0) + a_2(\tau-\tau_0)(\tau-\tau_1) + a_3(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2) \\ & + \cdots + a_n(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2)\cdots(\tau-\tau_{n-2})(\tau-\tau_{n-1}) \end{aligned}$$

(5) Computing Coefficients

$$a_0 = y_0$$

$$a_1 = \frac{y_1 - a_0}{(\tau_1-\tau_0)}$$

$$a_2 = \frac{y_2 - a_0 - a_1(\tau_2-\tau_0)}{(\tau_2-\tau_0)(\tau_2-\tau_1)}$$

$$a_3 = \frac{y_3 - a_0 - a_1(\tau_3-\tau_0) - a_2(\tau_3-\tau_0)(\tau_3-\tau_1)}{(\tau_3-\tau_0)(\tau_3-\tau_1)(\tau_3-\tau_2)}$$

$$\vdots$$

$$a_n = \frac{y_3 - a_0 - a_1(\tau_n-\tau_0) - a_2(\tau_n-\tau_0)(\tau_n-\tau_1) - \cdots - a_{n-1}(\tau_n-\tau_0)(\tau_n-\tau_1)\cdots(\tau_n-\tau_{n-2})}{(\tau_n-\tau_0)(\tau_n-\tau_1)(\tau_n-\tau_2)\cdots(\tau_n-\tau_{n-2})(\tau_n-\tau_{n-1})}$$

Another Form of Newton Interpolation

Newton’s Interpolating Polynomial (3)

(6) Expression of Newton’s Interpolating Polynomial using polynomial functions

$$\begin{aligned} f(\tau;\mathbf{a}) &= a_0 + a_1(\tau-\tau_0) + a_2(\tau-\tau_0)(\tau-\tau_1) + a_3(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2) \\ &\quad + \cdots + a_n(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2)\cdots(\tau-\tau_{n-2})(\tau-\tau_{n-1}) \\ &= a_0 + a_1 g_0(\tau) + a_2 g_1(\tau) + a_3 g_2(\tau) + \cdots + a_n g_{n-1}(\tau) \end{aligned}$$

Where

$$g_k(\tau) = (\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2)\cdots(\tau-\tau_k)$$

Then, the coefficients can be expressed as

$$a_0 = y_0$$

$$a_1 = \frac{y_1 - a_0}{g_0(\tau_1)}$$

$$a_2 = \frac{y_2 - a_0 - a_1 g_0(\tau_2)}{g_1(\tau_2)}$$

$$a_3 = \frac{y_3 - a_0 - a_1 g_0(\tau_3) - a_2 g_1(\tau_3)}{g_2(\tau_3)}$$

$$\vdots$$

$$a_n = \frac{y_3 - a_0 - a_1 g_0(\tau_n) - a_2 g_1(\tau_n) - \cdots - a_{n-1} g_{n-2}(\tau_n)}{g_{n-1}(\tau_n)}$$

Another Form of Newton Interpolation

Recursive Algorithm Computing Polynomials (1)

(1) General Expression for Recursive Algorithm

$$
\begin{aligned}
g_k(\tau) &= (\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2)\cdots(\tau-\tau_k) = \tau^{k+1} + \alpha_k\tau^k + \alpha_{k-1}\tau^{k-1} + \cdots + \alpha_1\tau + \alpha_0 \\
g_{k+1}(\tau) &= \tau^{k+2} + \beta_{k+1}\tau^{k+1} + \beta_k\tau^k + \beta_{k-1}\tau^{k-1} + \cdots + \beta_1\tau + \beta_0 \\
&= g_k(\tau)(\tau-\tau_{k+1}) \\
&= (\tau^{k+1} + \alpha_k\tau^k + \alpha_{k-1}\tau^{k-1} + \cdots + \alpha_1\tau + \alpha_0)(\tau-\tau_{k+1}) \\
&= \tau^{k+2} + (\alpha_k - \tau_{k+1})\tau^{k+1} + (\alpha_{k-1} - \alpha_k\tau_{k+1})\tau^k + \cdots + (\alpha_1 - \alpha_2\tau_{k+1})\tau^2 + (\alpha_0 - \alpha_1\tau_{k+1})\tau - \alpha_0\tau_{k+1}
\end{aligned}
$$

(2) Recursive formula

$$
\boxed{
\begin{aligned}
\beta_0 &= -\alpha_0\tau_{k+1} \\
\beta_j &= \alpha_{j-1} - \alpha_j\tau_{k+1} \qquad (j = 1, 2, 3, \cdots, k) \\
\beta_{k+1} &= \alpha_k - \tau_{k+1}
\end{aligned}
}
$$

Another Form of Newton Interpolation

Recursive Algorithm Computing Polynomials (2): Examples

(1) Three-point Formula with equal spacing $(\tau_0,\ \tau_1,\ \tau_2) = (0,\ 0.5,\ 1) \quad \leftarrow \{(x_0, y_0), (x_1, y_1), (x_2, y_2)\}$

$$\begin{aligned} f(\tau;\mathbf{a}) &= a_0 + a_1(\tau-\tau_0) + a_2(\tau-\tau_0)(\tau-\tau_1) \\ &= a_0 + a_1 g_0(\tau) + a_2 g_1(\tau) \end{aligned}$$

$$\boxed{\begin{aligned} \beta_0 &= -\alpha_0\tau_{k+1} \\ \beta_j &= \alpha_{j-1} - \alpha_j\tau_{k+1} \quad (j = 1, 2, 3, \cdots, k) \\ \beta_{k+1} &= \alpha_k - \tau_{k+1} \end{aligned}}$$

$$g_0(\tau) = \tau \rightarrow \begin{pmatrix} \alpha_0 = 0, \quad \tau_1 = 0.5 \\ \beta_0 = -\alpha_0\tau_1 = 0 \\ \beta_1 = \alpha_0 - \alpha_1\tau_1 = -0.5 \end{pmatrix}$$

$$g_1(\tau) = \tau^2 + \beta_1\tau + \beta_0 = \tau^2 - 0.5\tau$$

$$\boxed{\begin{aligned} g_0(\tau) &= \tau \\ g_1(\tau) &= \tau^2 - 0.5\tau \end{aligned}} \qquad \boxed{\begin{aligned} g_0(\tau_1) &= 0.5 \\ g_0(\tau_2) &= 1.0 \\ g_1(\tau_2) &= 0.5 \end{aligned}}$$

$$\boxed{\begin{aligned} a_0 &= y_0 \\ a_1 &= \frac{y_1 - a_0}{g_0(\tau_1)} = 2(y_1 - y_0) \\ a_2 &= \frac{y_2 - a_0 - a_1 g_0(\tau_2)}{g_1(\tau_2)} \\ &= 2(y_2 - y_0 - 2y_1 + 2y_0) \\ &= 2(y_2 - 2y_1 + y_0) \end{aligned}}$$

$$\boxed{\begin{aligned} f(\tau;\mathbf{a}) &= a_0 + a_1 g_0(\tau) + a_2 g_1(\tau) \\ &= y_0 + 2(y_1 - y_0)\tau + 2(y_2 - 2y_1 + y_0)(\tau^2 - 0.5\tau) \end{aligned}}$$

Another Form of Newton Interpolation

Recursive Algorithm Computing Polynomials (3): Examples

(2) Four-point Formula with equal spacing $(\tau_0,\ \tau_1,\ \tau_2,\ \tau_3)=\left(0,\ \frac{1}{3},\frac{2}{3},\ 1\right)\ \leftarrow\{(x_0,y_0),\cdots,(x_3,y_3)\}$

$$\begin{aligned} f(\tau;\mathbf{a}) &= a_0+a_1(\tau-\tau_0)+a_2(\tau-\tau_0)(\tau-\tau_1)+a_3(\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2) \\ &= a_0+a_1g_0(\tau)+a_2g_1(\tau)+a_3g_2(\tau) \end{aligned}$$

$$\boxed{\begin{aligned} \beta_0 &= -\alpha_0\tau_{k+1} \\ \beta_j &= \alpha_{j-1}-\alpha_j\tau_{k+1} \\ \beta_{k+1} &= \alpha_k-\tau_{k+1} \end{aligned}}$$

$$g_0(\tau)=\tau$$

$$g_1(\tau)=\tau^2-\frac{1}{3}\tau \leftarrow \beta_0=0,\quad \beta_1=-\frac{1}{3}$$

$$g_2(\tau)=\tau^3-\tau^2+\frac{2}{9}\tau \leftarrow \beta_0=0,\quad \beta_1=\frac{2}{9},\quad \beta_2=-1$$

$$g_0(\tau_1)=\frac{1}{3},\quad g_0(\tau_2)=\frac{2}{3},\quad g_0(\tau_3)=1$$

$$g_1(\tau_2)=\frac{2}{9},\quad g_1(\tau_3)=\frac{2}{3},\quad g_2(\tau_3)=\frac{2}{9}$$

$$\boxed{\begin{aligned} a_0 &= y_0 \\ a_1 &= \frac{y_1-a_0}{g_0(\tau_1)} \\ a_2 &= \frac{y_2-a_0-a_1g_0(\tau_2)}{g_1(\tau_2)} \\ a_3 &= \frac{y_3-a_0-a_1g_0(\tau_3)-a_2g_1(\tau_3)}{g_2(\tau_3)} \end{aligned}}$$

$$\begin{aligned} a_0 &= y_0 \\ a_1 &= 3(y_1-y_0) \\ a_2 &= \frac{9}{2}(y_2-y_0-2y_1+2y_0)=\frac{9}{2}(y_2-2y_1+y_0) \\ a_3 &= \frac{9}{2}(y_3-y_0-3y_1+3y_0-3y_2+6y_1-3y_0)=\frac{9}{2}(y_3-3y_2+3y_1-y_0) \end{aligned}$$

목차

1. Another Form of Newton Interpolation
2. Generalization of Newton-Cotes Interpolation Formula
3. Computer Model for General Newton-Cotes Integration
4. Program Structure and Application Test

Generalization of Newton-Cotes Interpolation Formula

Generalization of Integration Formula(1) : Derivation

(1) Given Data $\{(x_0,y_0),(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_j,y_j),\cdots,(x_n,y_n)\}=\{(x_j,y_j)\}_{j=0}^{j=n}$

(2) Another Form of Newton Interpolation

$$f(\tau;\mathbf{a})=a_0+a_1g_0(\tau)+a_2g_1(\tau)+\cdots+a_ng_{n-1}(\tau)=a_0+\sum_{k=1}^{k=n}a_kg_{k-1}(\tau)\leftarrow\tau=\frac{x-x_0}{x_n-x_0}\in[0,1]$$

Where $$g_{k-1}(\tau)\ =\tau^k+\beta_{k-1}\tau^{k-1}+\beta_{k-2}\tau^{k-2}+\cdots+\beta_1\tau+\beta_0=\sum_{m=0}^{m=k-1}\beta_m\tau^m+\tau^k$$

(3) Generalized Newton-Cotes Interpolation Formula

$$I(x_0,x_n)=\int_{x_0}^{x_n}f(\tau;\mathbf{a})dx=\int_{x_0}^{x_n}\left(a_0+\sum_{k=1}^{k=n}a_kg_{k-1}(\tau)\right)dx\leftarrow dx=(x_n-x_0)d\tau$$

$$=(x_n-x_0)a_0+(x_n-x_0)\sum_{k=1}^{k=n}a_k\left(\int_0^1 g_{k-1}(\tau)d\tau\right)$$

$$=(x_n-x_0)\left\{a_0+\sum_{k=1}^{k=n}a_k\left(\sum_{m=0}^{m=k-1}\frac{\beta_m}{m+1}+\frac{1}{k+1}\right)\right\}$$

$$\boxed{I(x_0,x_n)=\int_{x_0}^{x_n}y(x)dx\approx(x_n-x_0)\left\{a_0+\sum_{k=1}^{k=n}a_k\left(\sum_{m=0}^{m=k-1}\frac{\beta_m}{m+1}+\frac{1}{k+1}\right)\right\}}$$

Generalization of Integration Formula(2) : Computational Procedure

(1) Given Data $\{(x_0,y_0),(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_j,y_j),\cdots,(x_n,y_n)\}=\{(x_j,y_j)\}_{j=0}^{j=n}$

(2) Build Basis Functions for Newton Interpolation Formula using

$$\begin{aligned} g_k(\tau) &= (\tau-\tau_0)(\tau-\tau_1)(\tau-\tau_2)\cdots(\tau-\tau_k) \\ &= \tau^{k+1}+\alpha_k\tau^k+\alpha_{k-1}\tau^{k-1}+\cdots+\alpha_1\tau+\alpha_0 \\ g_{k+1}(\tau) &= \tau^{k+2}+\beta_{k+1}\tau^{k+1}+\beta_k\tau^k+\beta_{k-1}\tau^{k-1}+\cdots+\beta_1\tau+\beta_0 \end{aligned}$$

$$\boxed{\begin{aligned} \beta_0 &= -\alpha_0\tau_{k+1} \\ \beta_j &= \alpha_{j-1}-\alpha_j\tau_{k+1} \\ \beta_{k+1} &= \alpha_k-\tau_{k+1} \end{aligned}}$$

$$\tau=\frac{x-x_0}{x_n-x_0}\in[0,1]$$

(3) Apply Integration Formula

Let's identify function in the coefficients

$$g_{k-1}(\tau) \;=\tau^k+\beta_{k-1}^{(k-1)}\tau^{k-1}+\beta_{k-2}^{(k-1)}\tau^{k-2}+\cdots+\beta_1^{(k-1)}\tau+\beta_0^{(k-1)}=\sum_{m=0}^{m=k-1}\beta_m^{(k-1)}\tau^m+\tau^k$$

$$\boxed{I(x_0,x_n)=\int_{x_0}^{x_n}y(x)dx\approx(x_n-x_0)\left\{a_0+\sum_{k=1}^{k=n}a_k\left(\sum_{m=0}^{m=k-1}\frac{\beta_m^{(k-1)}}{m+1}+\frac{1}{k+1}\right)\right\}}$$

Generalization of Newton-Cotes Interpolation Formula

Example Application (1) : Three-point Formula at Nodes with the equal spacing

Basis Functions for Newton Interpolation Formula

$$\boxed{\begin{aligned} g_0(\tau) &= \tau \\ g_1(\tau) &= \tau^2 - 0.5\tau \end{aligned}}$$

$$\beta_0^{(0)} = 0$$

$$\beta_0^{(1)} = 0, \quad \beta_1^{(1)} = -0.5$$

Function Coefficients

$$a_0 = y_0$$

$$a_1 = 2(y_1 - y_0)$$

$$a_2 = 2(y_2 - 2y_1 + y_0)$$

Step Size

$$h = \frac{1}{2}(x_2 - x_0) \rightarrow x_2 - x_0 = 2h$$

Apply Integration Formula

Simpson's 1/3-rule

$$\boxed{\begin{aligned} I(x_0, x_2) &= \int_{x_0}^{x_2} y(x)dx \approx (x_2 - x_0)\left\{a_0 + \sum_{k=1}^{k=2} a_k \left(\sum_{m=0}^{m=k-1} \frac{\beta_m^{(k-1)}}{m+1} + \frac{1}{k+1}\right)\right\} \\ &= 2h\left\{y_0 + 2(y_1 - y_0)\left(\frac{1}{2}\right) + 2(y_2 - 2y_1 + y_0)\left(\frac{-1}{4} + \frac{1}{3}\right)\right\} \\ &= 2h\left\{y_0 + y_1 - y_0 + \frac{1}{6}(y_2 - 2y_1 + y_0)\right\} = \frac{h}{3}(y_0 + 4y_1 + y_2) \end{aligned}}$$

Generalization of Newton-Cotes Interpolation Formula

Example Application (2) : Four-point Formula at Nodes with the equal spacing

Basis Functions for Newton Interpolation Formula

$$g_0(\tau) = \tau \qquad \beta_0^{(0)} = 0$$

$$g_1(\tau) = \tau^2 - \frac{1}{3}\tau \qquad \beta_0^{(1)} = 0, \quad \beta_1^{(1)} = -1/3$$

$$g_2(\tau) = \tau^3 - \tau^2 + \frac{2}{9}\tau \qquad \beta_0^{(2)} = 0, \quad \beta_1^{(2)} = 2/9, \quad \beta_2^{(2)} = -1$$

Step Size $\quad h = \frac{1}{3}(x_3 - x_0) \rightarrow x_3 - x_0 = 3h$

Function Coefficients

$$a_0 = y_0$$

$$a_1 = 3(y_1 - y_0)$$

$$a_2 = \frac{9}{2}(y_2 - 2y_1 + y_0)$$

$$a_3 = \frac{9}{2}(y_3 - 3y_2 + 3y_1 - y_0)$$

Apply Integration Formula

Simpson's 8/3-rule

$$I \approx (x_3 - x_0)\left\{ a_0 + \sum_{k=1}^{k=3} a_k \left(\sum_{m=0}^{m=k-1} \frac{\beta_m^{(k-1)}}{m+1} + \frac{1}{k+1} \right) \right\}$$

$$= 3h\left\{ y_0 + 3(y_1 - y_0)\left(\frac{1}{2}\right) + \frac{9}{2}(y_2 - 2y_1 + y_0)\left(-\frac{1}{6} + \frac{1}{3}\right) + \frac{9}{2}(y_3 - 3y_2 + 3y_1 - y_0)\left(\frac{1}{9} - \frac{1}{3} + \frac{1}{4}\right) \right\}$$

$$= 3h\left\{ y_0 + \frac{3}{2}(y_1 - y_0) + \frac{3}{4}(y_2 - 2y_1 + y_0) + \frac{1}{8}(y_3 - 3y_2 + 3y_1 - y_0) \right\}$$

$$= \frac{3h}{8}\left\{ 8y_0 + 12(y_1 - y_0) + 6(y_2 - 2y_1 + y_0) + y_3 - 3y_2 + 3y_1 - y_0 \right\} = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3)$$

목 차

1 Another Form of Newton Interpolation

2 Generalization of Newton-Cotes Interpolation Formula

3 Computer Model for General Newton-Cotes Integration

4 Program Structure and Application Test

Computer Model for General Newton-Cotes Integration

Pseudocode (1)

1. Input Data: $n, \{x_j\}_{j=0}^{j=n}$ **with arbitrary node spacing**

2. Function Evaluation: $\left\{ \tau_j = \dfrac{x_j - x_0}{x_n - x_0},\ f(x_j) \right\}_{j=0}^{j=n}$

3. Compute Basis Function Coefficients: $k = 0, 1, \cdots, n-1$

$$\boxed{g_k(\tau) = \beta_{k+1}^{(k)}\tau^{k+1} + \beta_k^{(k)}\tau^k + \beta_{k-1}^{(k)}\tau^{k-1} + \cdots + \beta_1^{(k)}\tau + \beta_0^{(k)} = \sum_{m=0}^{m=k+1} \beta_m^{(k)}\tau^m}$$

$\beta_1^{(0)} = 1, \quad \beta_0^{(0)} = 0$ **Since** $g_0(\tau) = \tau \leftarrow \tau_0 = 0$

For $k = 1, 2, \cdots, n$

$$\beta_{k+1}^{(k)} = 1$$
$$\beta_k^{(k)} = \beta_{k-1}^{(k-1)} - \tau_k$$
$$\beta_j^{(k)} = \beta_{j-1}^{(k-1)} - \beta_j^{(k-1)}\tau_k \quad (j = 1, 2, \cdots, k-1)$$
$$\beta_0^{(k)} = -\beta_0^{(k-1)}\tau_k$$

or

$$\beta_0^{(k)} = -\beta_0^{(k-1)}\tau_k$$
$$\beta_j^{(k)} = \beta_{j-1}^{(k-1)} - \beta_j^{(k-1)}\tau_k \quad (j = 1, 2, \cdots, k-1)$$
$$\beta_k^{(k)} = \beta_{k-1}^{(k-1)} - \tau_k$$
$$\beta_{k+1}^{(k)} = 1$$

Computer Model for General Newton-Cotes Integration

Pseudocode (2)

4. Integrate the Basis Function

$$w_{k-1} = \int_0^1 g_{k-1}(\tau)d\tau \quad = \sum_{m=0}^{m=k} \beta_m^{(k-1)} \int_0^1 \tau^m d\tau = \sum_{m=0}^{m=k} \frac{\beta_m^{(k-1)}}{m+1}$$

5. Compute Coefficients for Newton's Interpolation: $\left\{a_j\right\}_{j=0}^{j=n-1}$

$$\boxed{f(\tau) = a_0 + a_1 g_0(\tau) + a_2 g_1(\tau) + a_3 g_2(\tau) + \cdots + a_n g_{n-1}(\tau) = a_0 + \sum_{j=1}^{j=n} a_j g_{j-1}(\tau)}$$

$$\boxed{\begin{aligned} & a_0 = y_0 \\ & a_j = \frac{y_j - y_0 - \sum_{k=1}^{j-1} a_k g_{k-1}(\tau_j)}{g_{j-1}(\tau_j)} \qquad (j = 1, 2, \cdots, n) \end{aligned}}$$

$$a_0 = y_0$$

$$a_1 = \frac{y_1 - a_0}{g_0(\tau_1)}$$

$$a_2 = \frac{y_2 - a_0 - a_1 g_0(\tau_2)}{g_1(\tau_2)}$$

$$a_3 = \frac{y_3 - a_0 - a_1 g_0(\tau_3) - a_2 g_1(\tau_3)}{g_2(\tau_3)}$$

$$\vdots$$

$$a_n = \frac{y_3 - a_0 - a_1 g_0(\tau_n) - a_2 g_1(\tau_n) - \cdots - a_{n-1} g_{n-2}(\tau_n)}{g_{n-1}(\tau_n)}$$

Computer Model for General Newton-Cotes Integration

Pseudocode (3)

5. Final Integration Formula

$$\boxed{\begin{aligned} f(\tau) &= a_0 + a_1 g_0(\tau) + a_2 g_1(\tau) + a_3 g_2(\tau) + \cdots + a_n g_{n-1}(\tau) \\ &= a_0 + \sum_{j=1}^{j=n} a_j g_{j-1}(\tau) \end{aligned}}$$

$$\begin{aligned} I &= \int_{x_0}^{x_n} y(x)dx \\ &\approx (x_n - x_0)\int_0^1 f(\tau)d\tau \\ &= (x_n - x_0)\int_0^1 \left\{ a_0 + \sum_{k=1}^{k=n} a_k g_{k-1}(\tau) \right\} \\ &= (x_n - x_0)\left\{ a_0 + \sum_{k=1}^{k=n} a_k \left(\int_0^1 g_{k-1}(\tau)d\tau \right) \right\} \\ &= (x_n - x_0)\left\{ a_0 + \sum_{k=1}^{k=n} a_k \left(\sum_{m=0}^{m=k} \frac{\beta_m^{(k-1)}}{m+1} \right) \right\} \end{aligned}$$

$$I = \int_{x_0}^{x_n} y(x)dx \approx (x_n - x_0)\left\{ a_0 + \sum_{k=1}^{k=n} a_k \left(\sum_{m=0}^{m=k} \frac{\beta_m^{(k-1)}}{m+1} \right) \right\} \approx (x_n - x_0)\left(a_0 + \sum_{k=1}^{k=n} a_k w_{k-1} \right)$$

w_{k-1}

목차

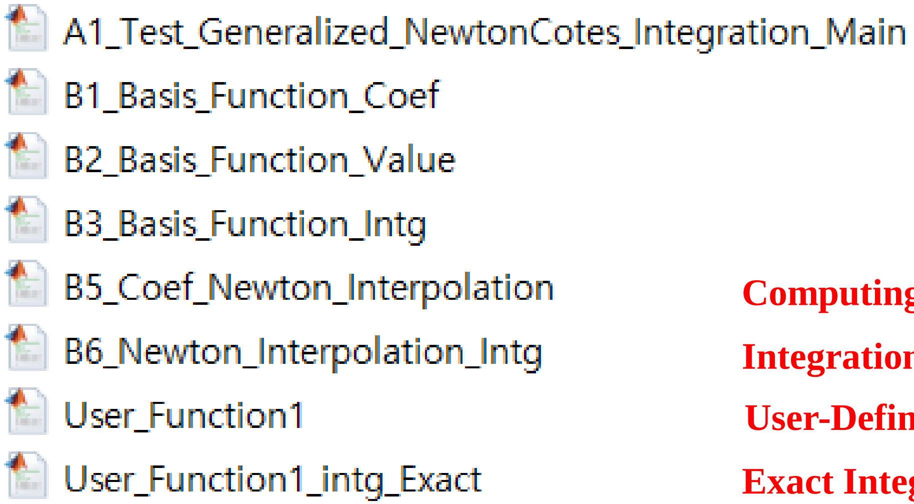
1 Another Form of Newton Interpolation

2 Generalization of Newton-Cotes Interpolation Formula

3 Computer Model for General Newton-Cotes Integration

4 Program Structure and Application Test

Program Structure (1): Overall Structure



A1_Test_Generalized_NewtonCotes_Integration_Main	Main Test Program
B1_Basis_Function_Coef	Computing Basis-Function Coefficients
B2_Basis_Function_Value	Computing Basis-Function Values
B3_Basis_Function_Intg	Integration of Basis-Function over Interval
B5_Coef_Newton_Interpolation	Computing Newton-Polynomial-Interpolation Coefficients
B6_Newton_Interpolation_Intg	Integration of Newton-Polynomial-Interpolation Function
User_Function1	User-Defined Test Function : exp(x)
User_Function1_intg_Exact	Exact Integration of User-Defined Test Function : exp(x)

Program Structure (2): A1_Test_Generalized_NewtonCotes_Integration_Main

```
%-----------------------------------------
%  Test Program for Generalized Newton-Cotes Integration Formula
%-----------------------------------------
%  (1) Number of Nodes and User Functions f(x) = exp(x)   0<= x <=2
%-----------------------------------------
    Node =    6 ;
    Xmin = -2.0 ;
    Xmax =  2.0 ;
%-----------------------------------------
%  (2) Uniform node generation and function evaluation
%-----------------------------------------
    DeLX = (Xmax-Xmin)    ;
    dX   = DeLX /(Node-1) ;
    for j= 1: Node
        X1 = Xmin + dX*(j-1)    ;
        F1 = User_Function1(X1) ;
%
        X(j) = X1 ;
        F(j) = F1 ;
        T(j) = (X1-Xmin)/DeLX ; % Tau
    end
%-----------------------------------------
```

Number of Nodes

Integration Interval

You can use Non-Uniform Nodes

User-Defined Test Function : exp(x)

Non-Dimensional Time Nodes (Tau)

Program Structure and Application Test

Program Structure (3): A1_Test_Generalized_NewtonCotes_Integration_Main

```
%-----------------------------------------
%  (3) Basis Function Coefficients
%-----------------------------------------
   Beta = B1_Basis_Function_Coef(Node,T) ;
%-----------------------------------------
%  (4) Integration of Basis Functions
%-----------------------------------------
   W = B3_Basis_Function_Intg(Node,Beta) ;
%-----------------------------------------
%  (5) Coefficients for Newton¡¯s Interpolation
%-----------------------------------------
   A = B5_Coef_Newton_Interpolation(Node,T,F,Beta) ;
%-----------------------------------------
%  (6) Integration of Newten-Interpolation Function
%-----------------------------------------
   Fint = B6_Newton_Interpolation_Intg(Node,A,W);
%-----------------------------------------
%  (6-1) Resut of Newton-Cotes Integration
%-----------------------------------------
   Fint_NC    = Fint*DeLX  ;
%-----------------------------------------
%  (6-2) Exact Integration
%-----------------------------------------
   Fint_exact = User_Function1_intg_Exact(Xmax) - User_Function1_intg_Exact(Xmin);
%-----------------------------------------
%  (6-3) Error in Newton-Cotes Integration
%-----------------------------------------
   Fint_err = Fint_exact - Fint_NC;
%-----------------------------------------
```

Program Structure and Application Test

Program Structure (4): A1_Test_Generalized_NewtonCotes_Integration_Main

```
%-----------------------------------------
%  (6-4) Display
%-----------------------------------------
X1=sprintf('  (1)  Resut of Newton-Cotes Integration:     Fint_NewtonCotes =
%d',Fint_NC);
X2=sprintf('  (2)  Exact                  Integration:     Fint_exact       =
%d',Fint_exact);
X3=sprintf('  (3)  Error in Newton-Cotes Integration:     Fint_Error       =
%d',Fint_err);
%
disp(X1)
disp(X2)
disp(X3)
%-----------------------------------------
%
%-----------------------------------------
```

Program Structure and Application Test

Application Tests with Different Number of Nodes

Problem Statement: $I = \int_0^3 f(x)dx \leftarrow f(x) = e^x$ $\qquad I_{exact} = e^3 - 1 = 19.0855369$

Node=2

```
(1)  Resut of Newton−Cotes Integration:    Fint_NewtonCotes    =   3.162831e+01
(2)  Exact                      Integration:    Fint_exact          =   1.908554e+01
(3)  Error in Newton−Cotes  Integration:    Fint_Error          =  −1.254277e+01
```

Node=4

```
(1)  Resut of Newton−Cotes Integration:    Fint_NewtonCotes    =   1.927783e+01
(2)  Exact                      Integration:    Fint_exact          =   1.908554e+01
(3)  Error in Newton−Cotes  Integration:    Fint_Error          =  −1.922946e−01
```

Node=6

```
(1)  Resut of Newton−Cotes Integration:    Fint_NewtonCotes    =   1.908865e+01
(2)  Exact                      Integration:    Fint_exact          =   1.908554e+01
(3)  Error in Newton−Cotes  Integration:    Fint_Error          =  −3.114738e−03
```

Node=8

```
(1)  Resut of Newton−Cotes Integration:    Fint_NewtonCotes    =   1.908557e+01
(2)  Exact                      Integration:    Fint_exact          =   1.908554e+01
(3)  Error in Newton−Cotes  Integration:    Fint_Error          =  −3.678323e−05
```

Node=10

```
(1)  Resut of Newton−Cotes Integration:    Fint_NewtonCotes    =   1.908554e+01
(2)  Exact                      Integration:    Fint_exact          =   1.908554e+01
(3)  Error in Newton−Cotes  Integration:    Fint_Error          =  −3.149889e−07
```

End of Lecture