ZS 6

Freitag, 11. Januar 2019 15:45

6. Quadrature

Approximate an integral because we only have some data points, the integral is too hard or it's overall faster.

$$Q_n(f) = \sum_{j=1}^{n} w_j \cdot f(c_j)$$

error : $\left| \int_a^b f(t)dt - Q_n(f) \right|$

$$\int_a^b f(t)\,dt \approx \int_a^b I_T[f(t_1),\dots,f(t_n)]^T(t)\,dt = \sum_{j=1}^{n} f(t_j) \underbrace{\int_a^b (I_T(e_j))(t)\,dt}_{\text{weight } w_j}$$

The following methods differ in which nodes $(t_1, \dots, t_n)$ and which weights they choose or what form $I_T[f(t_1),\dots,f(t_n)]^T(t)$ has. (z.B. polynom)

Order : $\text{order}(Q_n) := \max\{m \in \mathbb{N}_0 : Q_n(p) = \int_a^b p(t)\,dt \ \forall p \in P_m\} + 1$

This means: We only have n Nodes as a budget. To which degree of an arbitrary polynom p is:

$$Q_n(f) = \sum_{j=1}^{n} w_j f(c_j) \text{ exactly } \int_a^b p(t)\,dt$$

where p is a random polynom.

Note: If we have Polynomial Quadrature Formulas (= based on polynomial interpolation with Lagrange Basis) then we know: $\text{order}(Q_n) \geq n$.
Because with n nodes we have an exact interpolation for a polynom of degree $n-1$. So for any polynom p of degree $n-1$ we can have the exact interpolation and hence Quadrature. The $+1$ makes it then to n.

Note: If in $\int_a^b f(t) \approx Q_n(f) = \int_a^b p_{n-1}(t)dt$ $f(t)$ is a polynom of degree $2n$ but we would have that $\text{order}(Q_n) \geq 2n+1$ then we could have:

$$\int_a^b f(t) = Q_n(f)$$

even though we only have n Nodes as a budget. Clearly if $f(t)$ is a polynom of degree $2n$ and we could have $2n+1$ Nodes as a budget (Q_{2n+1}) then of course it is the same no matter where we choose our Nodes. But with only n as a budget, we need to use them wisely and might fail nontheless to get that $=$ and not the $\approx$

Polynomial Quadrature Formulas (Lagrange Basis)

$$\int_a^b f(t)dt \approx Q_n(f) = \int_a^b p_{n-1}(t)\,dt = \int_a^b \sum_{j=0}^{n-1} f(t_j) L_j(t) = \sum_{j=1}^{n} f(t_{j-1}) \cdot \underbrace{\int_a^b L_{j-1}(t)\,dt}_{\text{weights}}$$

We always take Lagrange as a Basis because it is a cardinal Basis because only then we have that

$$p_{n-1}(t) = \sum_{j=0}^{n-1} \underline{f(t_j)} L_j(t)$$

For other Basis we would have a $c(t_j)$ and this wouldn't fullfill the Form of the Quadrature. $(Q_n(f) \neq \sum \underline{c(t_j)}\, T_j(t))$

Error : $\left| \int_a^b f(t)dt - Q_n(f) \right| \leq C \cdot q^{-r} |b-a|^{r+1} \cdot \| f^{(r)} \|_{L^\infty([a,b])}$ for $q \geq r$

$$\leq \frac{|b-a|^{q+1}}{q!} \cdot \| f^{(q)} \|_{L^\infty([a,b])} \quad \text{for } q < r$$

$q = \text{order}(Q_n)$, $r = f(t) \in C^r([a,b])$

$\Rightarrow f \in C^r$: algebraic

$\Rightarrow f \in C^\infty$: exponential

Newton-Cotes formulas

Note: Don't let the name trick you. You have $\int_a^b L_{j-1}(t)\,dt$ as weights.

We take equidistant nodes

Midpoint (n=1): $\int_a^b f(t)dt \approx Q(f) = (b-a) f(\tfrac{1}{2}(a+b))$

Trapezoidal (n=2): $\int_a^b f(t)dt \approx Q(f) = \frac{b-a}{2}(f(a) + f(b))$

Simpson (n=3): $\int_a^b f(t)dt \approx Q(f) = \frac{b-a}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b))$

All of these use equidistant Nodes and are therefore unstable
Therefore we want to choose our Nodes differently.
Astonishingly the best choice won't be the Chebyshev nodes but the Gaussian nodes.

Note: There exists Clenshaw-Curtis Quadrature that uses Chebyshev Nodes but it isn't a polynomial Quadrature Formula and is not as good as Gauss Quadrature.

Convergence : $f \in C^\infty$: algebraic or exponential

$f \in C^r$: algebraic

Gauss Quadrature

Optimal choice for polynomial Quadrature Formulas. To get the highest order (Q_n) we need to choose our Nodes wisely.
It turns out:

- To have $\text{order}(Q_n) \geq n$ you need Lagrange polynomials as weights
- To have $\text{order}(Q_n) = 2n$ we need the nodes to be the roots of the n-th Legendre polynomial.

$\Rightarrow$ We get $\text{order}(Q_n) = 2n$ by choosing Lagrange as weights and Legendre for the nodes.

Convergence : $f \in C^\infty$: exponential

$f \in C^r$: algebraic

Chebyshev Quadrature

We take Lagrange weights but Chebyshev nodes.

Convergence : $f \in C^\infty$: exponential

$f \in C^r$: algebraic

Composite Quadrature

Split Intervall in many Subintervalls and then apply QF on each of them.

Note: Gauss is always better.

You only choose it when you can't do Gauss hence not freely choose your Nodes or when the function is highly oscillatory.

Convergence : $f \in C^\infty$: algebraic $[O(n^{-q})]$ to the width

$f \in C^r$: algebraic $[O(n^{-\min\{r,q\}})]$ to the width