1.4. Integrationsbasierte Formen

Typische Linearform: $b(v) = \int_\Omega f(x)\, v(x)\, \lambda^d(dx)$

Lebesgue-Maß

numerisch: $b_h(v) = \int \underbrace{f(x)\, v(x)}_{g(x)}\, Q(dx)$

Punktmaßapproximation von $\mathbb{1}_\Omega \lambda^d$

$$\Pi = \sum_{i=1}^{q} w_i\, \delta_{p_i}$$

$$= \sum_{i=1}^{q} w_i\, g(p_i)$$

Konstruktion von Gewichten w_i und Punkten p_i:

T Triangulierung mit einer Referenzzelle:

$$\int_\Omega g(x)\, \lambda^d(dx) = \sum_{i=1}^{M} \int_{T_i} g(x)\, \lambda^d(dx) = \sum_{i=1}^{M} \int_{T_{ref}} g(F_i(s))\, |\det F_i'|\, \lambda^d(ds)$$

$$\approx \sum_{i=1}^{M} \int g(F_i(s))\, |\det F_i'|\, \Pi_{ref}(ds)$$

Punktmaßapproximation von $\mathbb{1}_{T_{ref}} \lambda^d$

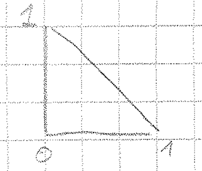
$$= \sum_{i=1}^{M} \sum_{k=1}^{q_{ref}} \underbrace{w_k^{ref}\, |\det F_i'|}_{\text{Gewicht}}\, g(\underbrace{F_i(p_k^{ref})}_{\text{Punkt}})$$

Gute Quadraturformeln auf Referenzgebieten.

$T_{ref} =$    Gauß-Legendre Π^1_{GL} (hoher Exaktheitsgrad wählbar)

$T_{ref} = (0,1)^d$ Produkte $\underbrace{\Pi^1_{GL} \otimes \cdots \otimes \Pi^1_{GL}}_{d \text{ Faktoren}}$

$(x,y) = H(t,x) = (x, xt)$

$T_{ref} =$ 

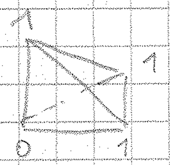
$$\int_{T_{ref}} g(s)\, \lambda^2(ds) = \int_0^1 \int_0^x g(x,y)\, dy\, dx = \int_0^1 \int_0^1 g(x, xt)\, x\, dt\, dx$$

$$\approx \int_0^1 f(x)\, x\, dx \qquad f(x) := \int g(x, xt)\, \Pi^1_{GL}(dt)$$

$$\approx \int f(x)\, \Pi^1_{GJ}(dx)$$

Gauß-Jacobi Approximation vom Maß $x \mathbb{1}_{[0,1]}(x)\, \lambda^1(dx)$

Maßapproximation $H\left(\Pi^1_{GJ} \otimes \Pi^1_{GL}\right)$

analoge Idee für $T_{ref} =$ 

Umsetzung durch Klasse pointMeasure:

Verwaltet Punkte p, Gewichte w

Methoden: mtimes (Produktmaß)

imageOf (Bildmaß)

↳ Transformation

Eingabe: F affine Transformation

$$\left.\begin{array}{l} w_j \rightsquigarrow |\det F'|\, w_j \\ p_j \rightsquigarrow F(p_j) \end{array}\right\} \text{Ausgabe: pointMeasure}$$

Konkrete Punktmaße werden durch <u>abgeleitete Klassen</u> umgesetzt

Beispiel 1D Mittelpunktregel auf $[0,1]$

"erbt von"

```
classdef midpoint1D < pointMeasure

    properties (SetAccess = private)
        dom % polytope
    end

    methods

        function Q = midpoint1D

            Q.dom = cube(1);
            % Zugriff auf SetAccess = protected Eigenschaften von pointMeasure
            Q.dim = 1;
            Q.q = 1;
            Q.w = 1;
            Q.p = 0.5;
        end
    end
end
```

zusätzliche Eigenschaften der speziellen Klasse

zusätzliche Methoden

• alle Methoden aus pointMeasure stehen automatisch in midpoint1D zur Verfügung

Vorgabe: T affin erzeugte Triangulierung

Q Punktmaß auf T_{ref}

$\mathcal{B}$ Basis eines Teilraums von $V(T, M)$

Aufbau von b_h:

$c = \text{zeros}(1, \mathcal{B}.\text{dim})$

for $k = 1 : M$

Fähigkeiten der Triangulierung:

- Elementanzahl M bekannt geben
- k-te affine Abbildung bereitstellen (mit $\det F_k'$)

$Q_k = Q.\text{Trans}(F_k)$;

$\text{fval} = f(Q_k.p)$;

for $l = I_k$ von $\mathcal{B}$

$c(l) = c(l) + Q_k.w * (\text{fval} .* \psi_l(Q_k.p))$;

end

end

$b_h = \text{linearForm}(\mathcal{B}, c)$;

ähnlich bei Bilinearformen, z.B.

$$a(u, v) = \int_\Omega \alpha(x)\, u(x)\, v(x)\, \lambda^d(dx)$$

$$a_h(u, v) = \sum_{\ell=1}^{q} \sum_{k=1}^{M} |\det F_k'|\, w_j\, (\alpha u v)(F_k(p_j))$$

Unterschied beim Abspeichern: nutze sparse-Matrix Befehl

$$A = \text{sparse}(I, J, V, d, d)$$

Ergebnis $\quad A_{ij} = \sum_{\substack{I(k)=i \\ J(k)=j}} V(k) \qquad 1 \le i, j \le d$

```
I = []; J = []; V = [];
for k = 1 : M

    Q = Q.LTrans(F_k)
    αval = α(Q_k.p);
    for l = I_k
        for m = I_k
            I = [I; l]
            J = [J; m]
            V = [V; Q_k.w * (αval .* ψ_l(Q_k.p) .* ψ_m(Q_k.p)) ];
        end
    end
end

a_h = bilinearForm(B, sparse(I, J, V, B.dim, B.dim));
```