Part I: Galerkin discretization

We consider a general integral equation

$$\mathcal{G}u + \lambda < u, \cdot >= f \tag{1}$$

in a Hilbert space H, where

$\mathcal{G}$ is an integral Operator mapping H into its dual space H',
f $\in$ H' is an element of the dual space,
$\lambda \in \mathbb{R}$ is some parameter and
u $\in$ H is the solution we are looking for.

The variational counterpart of the equation above is

$$a(u, v) + \lambda m(u, v) = f(v) \tag{2}$$

for all $v \in H$, where

$$a(u, v) :=< \mathcal{G}u, v >_{H' \times H} \; and \quad m(u, v) :=< u, v >_{H \times H} \; .$$

In typical situations, the bilinear form $a(\cdot, \cdot)$ representing the integral operator can be written as

$$a(u, v) = \int\limits_{\Omega} v(x) \int\limits_{\Omega} g(x, y)u(y)dydx \tag{3}$$

for a kernel function $g(\cdot, \cdot)$ and a domain or manifold Ω.

The equation (2) is discretized by choosing an n-dimensional subspace H_n of H and considering the problem of finding a function $u_n \in H_n$ s.t.

$$a(u_n, v_n) + \lambda m(u_n, v_n) = f(v_n)$$

holds $\forall v_n \in H_n$.

For any basis $(\varphi_i)_{i\in\mathcal{I}}$ of H_n, this is equivalent to

$$a(u_n, \varphi_i) + \lambda m(u_n, \varphi_i) = f(\varphi_i)$$

$\forall i \in \mathcal{I}$.

Since the solution u_n is an element of H_n, there is a coefficient vector $(x_i)_{i\in\mathcal{I}}$ satisfying

$$u_n = \sum_{j\in\mathcal{I}} x_j \varphi_j,$$

s.t. the coefficients satisfy the equation

$$\sum_{j\in\mathcal{I}} x_j a(\varphi_j, \varphi_i) + \lambda \sum_{j\in\mathcal{I}} x_j m(\varphi_j, \varphi_i) = f(\varphi_i)$$

for all $i \in \mathcal{I}$.

This is a system of linear equations and can be written in matrix form

$$Gx + \lambda Mx = b$$

by introducing matrices $G, M \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ and a vector $b \in \mathbb{R}^{\mathcal{I}}$ with

$$G_{ij} := a(\varphi_j, \varphi_i), \tag{4}$$

$$M_{ij} := m(\varphi_j, \varphi_i), \tag{5}$$

$$b_i := f(\varphi_i). \tag{6}$$

If we use standard finite element basis functions $(\varphi_i)_{i\in\mathcal{I}}$, the matrix M will be sparse, but G will be densely populated, since typical kernel functions have global support.

Storing G directly doesn't lead to efficient algorithms. Therefore we approximate G by a matrix that can be treated efficiently, by replacing the kernel function $k(\cdot, \cdot)$ by local degenerate approximations, and this leads to a hierarchical matrix.

Recall: Degenerate approximation

Last week we've already talked about degenerate approximation, but as I can imagine that a few people in here may not remember it very vividly, I'll quickly explain what it is about.

The idea is to approximate the kernel function $g(\cdot,\cdot)$ by using interpolation instead of Taylor expansion, and thus avoiding the need of being able to evaluate the derivatives of g efficiently.

Idea:

Let

$(x_\nu)_{\nu\in K}$ be a family of interpolation points in $\mathbb{R}^d$

$(\mathcal{L}_\nu)_{\nu\in K}$ be the Lagrange functions with

$$\mathcal{L}_\nu(x_\mu) = \delta_{\nu,\mu}$$

for $\nu, \mu \in K$.

We define

$$\tilde{g}(x,y) := \sum_{\nu\in K} g(x_\nu, y)\mathcal{L}_\nu(x).$$

As mentioned before, we don't need any derivative of g for that approximation.

We can now replace the matrix G with $\tilde{G}$ defined by

$$\tilde{G}_{ij} := \int\limits_\Omega \varphi_i(x) \int\limits_\Omega \tilde{g}(x,y)\varphi_j(y)dydx = (AB^T)_{ij},$$

where

$$A_{i\nu} := \int\limits_\Omega \varphi_i(x)\mathcal{L}_\nu(x)dx$$

and

$$B_{j\nu} := \int\limits_\Omega \varphi_j(y)g(x_\nu, y)dy.$$

Part II: Boundary Element Method in 2D

We consider a closed curve in 2-dimensional space, given as an array vertex of n points. We use piecewise constant basis functions and choose the characterizing point to be the middle of the corresponding interval. We will now solve integral equations on this curve.

We are interested in a boundary integral problem, i.e. the set Ω is a submanifold. Here Ω is a a curve. Since we'll be talking about integral equations, I will recall the meaning of an integral on a curve:

1 Curve integrals

Let $\gamma : [0,1] \to \mathbb{R}^2$ be injective in [0,1[, $\gamma \in C^1, \gamma' \in C^0$. We write $\Gamma := \gamma([0,1])$. Let $u \in C^0(\Gamma)$. We introduce a partition $0 = x_0 < x_1 < ... < x_n = 1$ of [0,1] and consider the sum

$$I_x := \sum_{i=1}^{n} u(\gamma(x_i)) \|\gamma(x_i) - \gamma(x_{i-1})\|.$$

Lemma (Curve integral) : *Let $\epsilon \in \mathbb{R}_{>0}$. There is a $\delta \in \mathbb{R}_{>0}$ s.t. $\forall$ partitions $0 = x_0 < x_1 < ... < x_n = 1$ with $x_i - x_{i-1} < \delta(i \in 1, ..., n)$ we have*

$$\|I_x - \int_0^1 u(\gamma(y))\|\gamma'(y)\|dy\| \leq \epsilon.$$

Proof: elementar analysis

Definition: We define the *curve integral* : let $(\gamma_i)_{i=1}^{m}$ be a tuple of injective functions in $C^1([0,1], \mathbb{R}^2)$. For all $i \in 1, ..., m$, we set $\Gamma_i :=$

$\gamma_i([0,1])$. The *curve integral* over the piecewise differentiable curve $\Gamma := \cup_{i=1}^m \Gamma_i$ is given by

$$\int_\Gamma u(x)dx := \sum_{i=1}^m \int_0^1 u(\gamma_i(y))\|\gamma_i'(y)\|dy.$$

2 Single layer potential

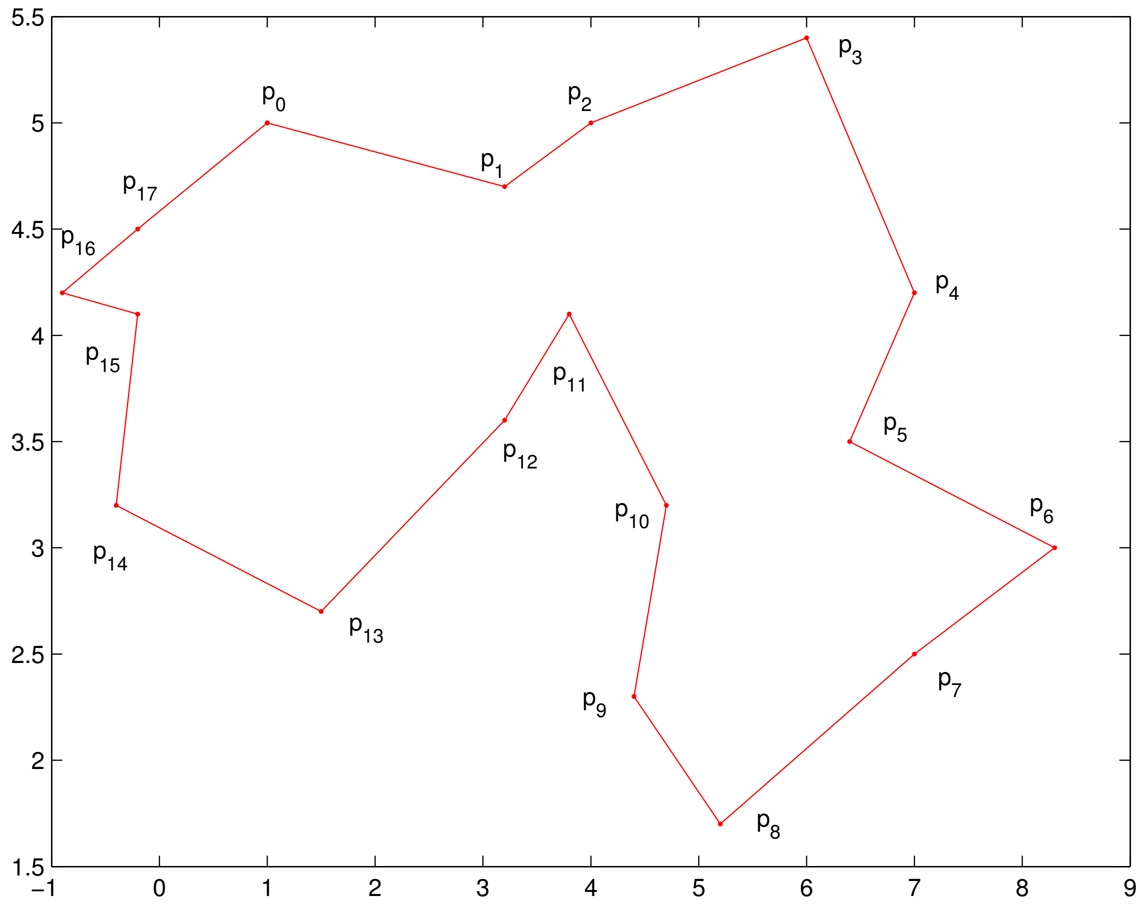
We fix n points $p_0, ..., p_{n-1} \in \mathbb{R}^2$, set $p_n := p_0$ and define the affine parametrizations

$$\gamma_i : [0,1] \to \mathbb{R}^2, \quad y \mapsto p_{i-1}(1-y) + p_i y,$$

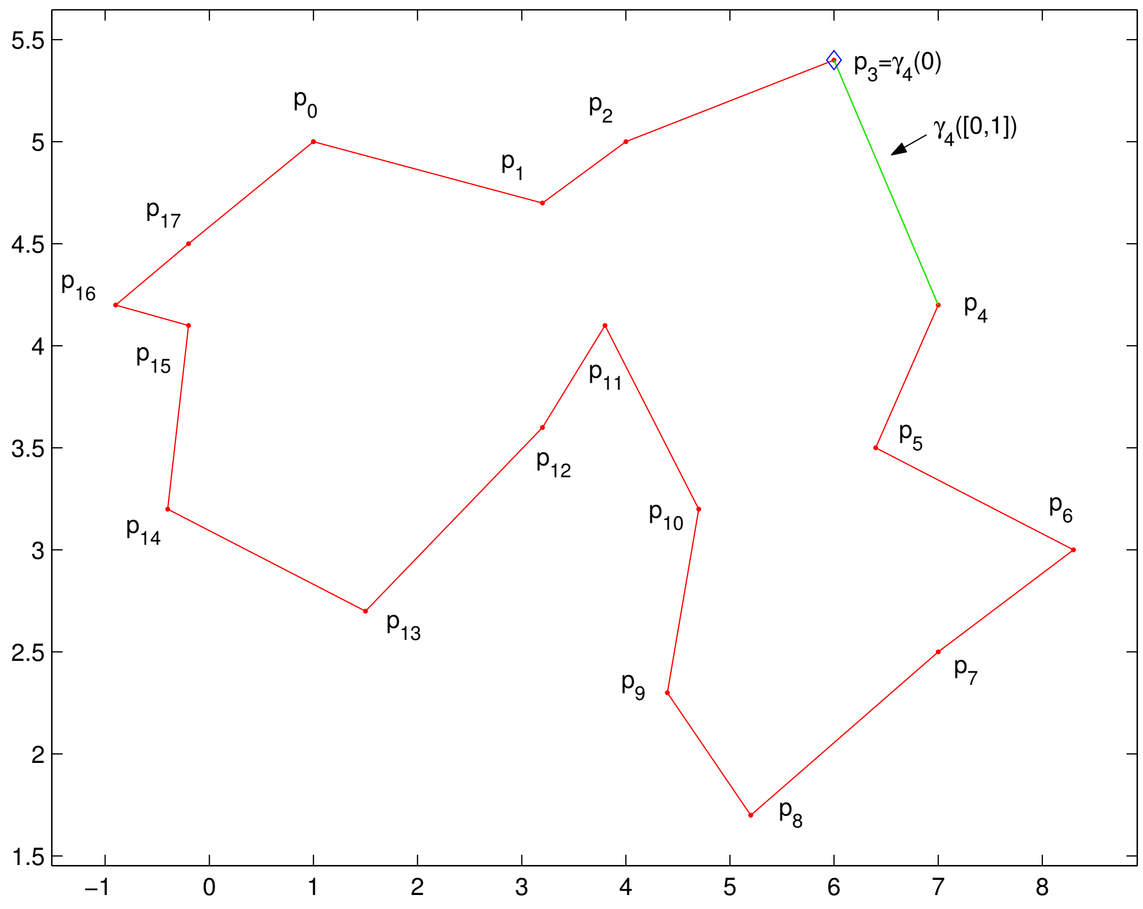
for $i \in 1, ..., n$. As long as $p_i \neq p_j$ holds $\forall i, j \in 0, ..., n-1$ with $i \neq j$, this defines a polygonal curve $\Gamma := \cup_{i=1}^m \gamma_i([0,1])$.

Example:

We consider the following points:

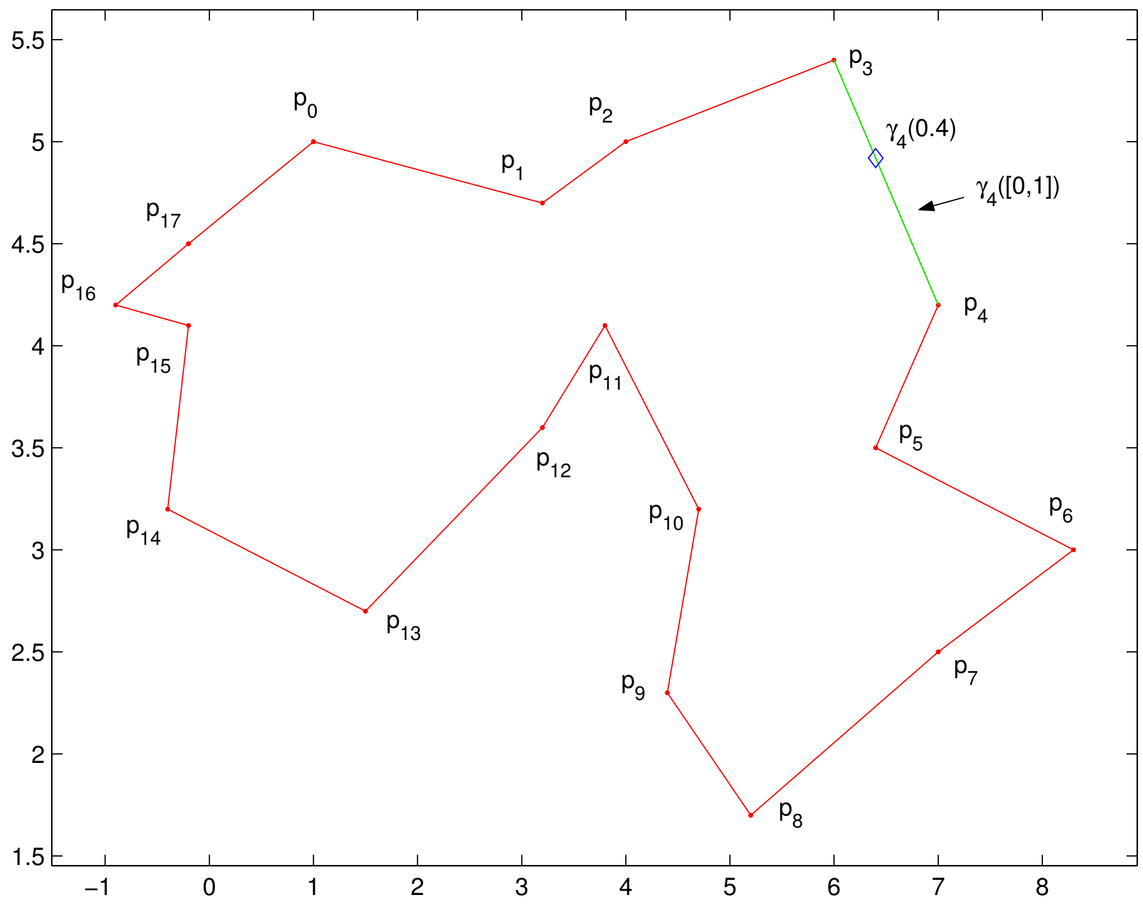


We observe γ_4 and its evaluation in t=0:



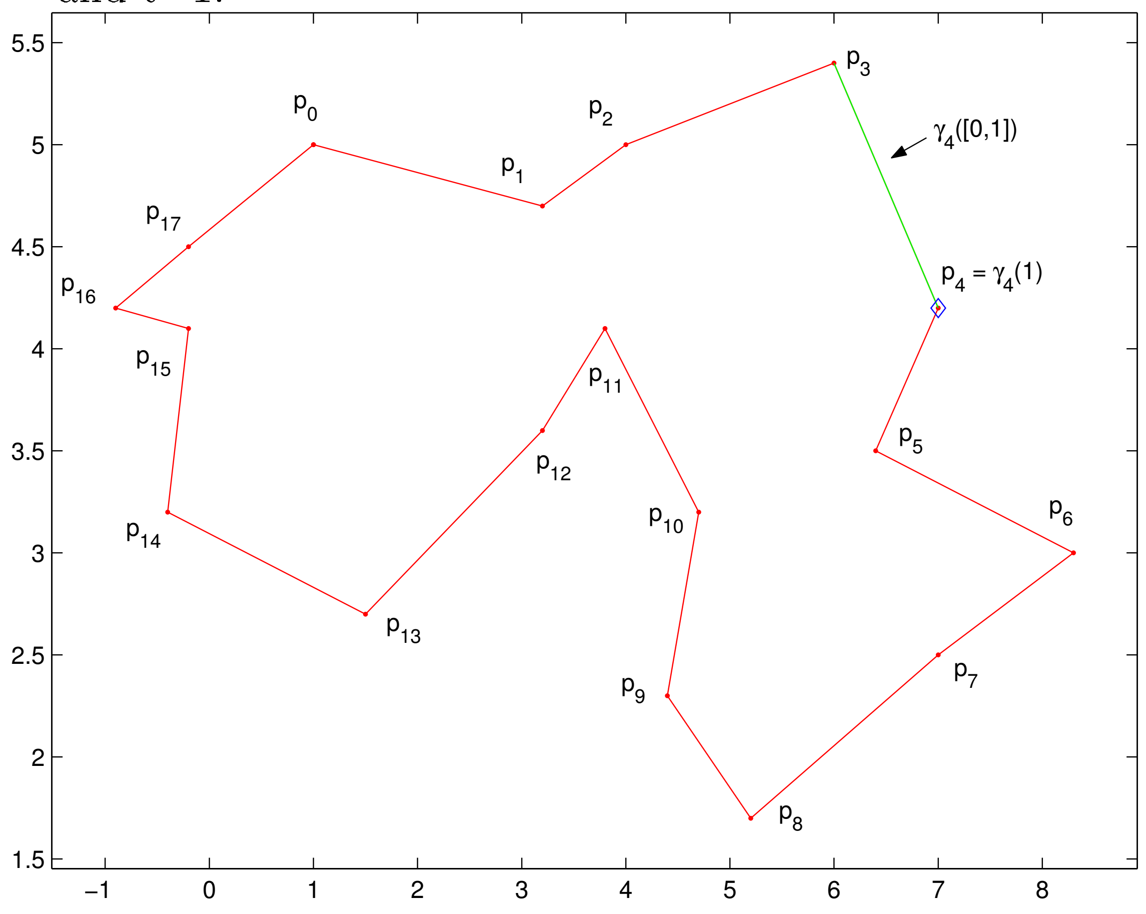
p_0
p_1
p_2
$p_3=\gamma_4(0)$
$\gamma_4([0,1])$
p_4
p_5
p_6
p_7
p_8
p_9
p_{10}
p_{11}
p_{12}
p_{13}
p_{14}
p_{15}
p_{16}
p_{17}

t=0.4:



p_0
p_1
p_2
p_3
p_4
p_5
p_6
p_7
p_8
p_9
p_10
p_11
p_12
p_13
p_14
p_15
p_16
p_17
γ_4(0.4)
γ_4([0,1])

and t=1:



On the curve Γ we now define the *single layer potential* operator

$$\mathcal{G}_{slp}[u](x) := \int_\Gamma \log(\|x-y\|)u(y)dy$$

and the corresponding bilinear form

$$a_{slp}(u,y) := \int_\Gamma v(x) \int_\Gamma \log(\|x-y\|)u(y)dydx.$$

The kernel $\log(\|x-y\|)$ is not asymptotically smooth here, since it has a singularity in $x=y$.

What do we do?

We discretize $a_{slp}(\cdot,\cdot)$ using piecewise $c^{\underline{st}}$ functions $(\varphi_i)_{i=1}^n$ defined through

$$\varphi_i \circ \gamma_j \equiv \delta_{ij}$$

for $i, j \in \mathcal{I} := 1, ..., n$. The coeff. of the corresponding matrix are given by

$$\begin{aligned} G_{ij} &= a_{slp}(\varphi_i, \varphi_j) = \int_{\Gamma} \varphi_i(x) \int_{\Gamma} \log(\|x-y\|)\varphi_j(y) dy dx \\ &= \|p_i - p_{i-1}\| \|p_j - p_{j-1}\| \int_0^1 \int_0^1 \log(\|\gamma_i(x) - \gamma_j(y)\|) dy dx. \end{aligned}$$

From $p_i \neq p_j$ it follows that this matrix is full. As long as $\gamma_i([0,1])$ doesn't intersect $\gamma_j([0,1]) \forall i, j \in$ 1,...,n, we don't have singularities and can replace the kernel by degenerate approximations.

3 Implementation

We'll now focus on computing the entries of the hierarchical matrix, in particular the low-rank blocks, so we suppose that we have a function that initializes the full matrices and now consider the treatment of the low-rank blocks.

They correspond to admissible pairs (t,s) of clusters and require the evaluation of a degenerate approximation of the kernel function. Let's assume that diam(Q^t)≤diam(Q^s), it follows

$$\tilde{(g)}(x,y) = \sum_{\nu \in K} \log(\|x_\nu^t - y\|)(L)_\nu^t(x)$$

and we compute

$$\begin{aligned} A_{i\nu}^{t,s} &= \int_{\Gamma} \varphi_i(x)(L)_\nu^t(x) dx = \|p_i - p_{i-1}\| \int_0^1 (L)_\nu^t(\gamma_i(x)) dx, \\ B_{j\nu}^{t,s} &= \int_{\Gamma} \varphi_j(x) \log(\|x_\nu^t - y\|) dy = \|p_j - p_{j-1}\| \int_0^1 \log(\|x_\nu^t - \gamma_j(y)\|) dy. \end{aligned}$$

γ_i is affine, so $A^{t,s}_{i\nu}$ are polynomials of degree m. We can thus use an exact quadrature rule for its evaluation.

In order to do that, we need:

- An array `vertex` of dimension `n` containing the coordinates of the points $(p_i)_{i=0}^{n-1}$.
- Arrays `xq` and `wq` of dim. `q` containing the pts and weights of a suitable quadrature rule.
- An array `l` of dim. `p` containing the transformed interpolation pts.

Recall(Gauss-quadrature):
The idea is to approximate an integral as

$$\int_\Omega f(\xi)d\xi \approx \sum_{K\in\mathcal{M}} |K| \sum_{l=1}^{P_K} w_l^K f(\pi_l^K).$$

The w_l^K are called local weights and the points π_l^K local nodes. The Gauss-quadrature is exact for polynomials of degree up to 2P - 1 using only P nodes.
The quadrature is numerically stable if all the weights are positive.

Example(Gauss-quadrature, 1-dimensional):
We consider the integral $\int_{-1}^{1} f(x)dx$ and want to approximate it with a Gauss-quadrature of degree P=2:

$$\int_{-1}^{1} f(\xi)d\xi = w_1 f(x_1) + w_2 f(x_2)$$

This approximation has to be exact for polynomials $\in \mathcal{P}_3$, so we choose successively $f(x) = x^0, x^1, x^2, x^3$. We now have 4 equations to solve:

$$\begin{aligned}
\int\limits_{-1}^{1} 1dx &= 2 = w_1 + w_2 \\
\int\limits_{-1}^{1} xdx &= 0 = w_1x_1 + w_2x_2 \\
\int\limits_{-1}^{1} x^2dx &= \frac{2}{3} = w_1x_1^2 + w_2x_2^2 \\
\int\limits_{-1}^{1} x^3dx &= 0 = w_1x_1^3 + w_2x_2^3.
\end{aligned}$$

We replace $w_2 = -w_1\frac{x_1}{x_2}$ from the second equation in the last and get

$$w_1x_1^3 - w_1\frac{x_1}{x_2}x_2^3 = 0 \Rightarrow x_1^2 = x_2^2$$

And thus $x_1 = -x_2$, since we want $x_1 \neq x_2$.

Further $0 = w_1 - w_2 \Rightarrow w_1 = w_2$
$2 = w_1 + w_2 \Rightarrow 2w_1 = 2 \Rightarrow w_1 = w_2 = 1$
$\frac{2}{3} = (w_1 + w_2)x_1^2 = 2x_1^2 \Rightarrow x_1^2 = \frac{1}{3}$
And so we have

$$\begin{aligned}
w_1 &= w_2 = 1 \\
x_1 &= -\frac{1}{\sqrt{3}} \\
x_2 &= \frac{1}{\sqrt{3}}
\end{aligned}$$

The evaluation of $B^{t,s}$ requires us to integrate the kernel function for points x_ν^t on intervals given by p_{i-1} and p_i. Here, it can be done analytically. In more general cases, we can use the same quadrature rule as for polynomials, but the result will no longer necessarily be exact.

Example(affine γ in $\mathbb{R}^2$): *Let $\gamma : [0,1] \to \mathbb{R}^2$ and $c \in \mathbb{R}^2$ given by*

$$\gamma(t) := \begin{pmatrix} s_x + td_x \\ s_y + td_y \end{pmatrix} \textit{and} \quad c := \begin{pmatrix} c_x \\ c_y \end{pmatrix}.$$

We assume that $c \notin \gamma([0,1])$. We want to compute the value of

$$b = \int_0^1 \log(\|\gamma(t) - c\|_2)\|\gamma'(t)\|_2 dt.$$

With Gauss-quadrature we obtain b=1.9755 for

$$\gamma(t) := \begin{pmatrix} 1+t \\ 1+2t \end{pmatrix} \textit{and} \quad c := \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

and if we change γ to

$$\gamma(t) := \begin{pmatrix} 1+2t \\ 1+4t \end{pmatrix}$$

(and thus doubling its"speed") we obtain b=5.4231.

The `supermatrix` structure can be initialized by a simple recursion: If the `supermatrix` contains an `rkmatrix`, we compare the diameters of the clusters involved and use the procedure described above to initialize the fields `a` and `b` of the `rkmatrix`.

If the `supermatrix` contains a `fullmatrix`, we evaluate singular integrals and fill its field `e`.

Otherwise, we proceed recursively with the subblocks that are given by the array `s`.