Programming sheet 2.

Meeting week: **31. March – 4. April 2025**

Hierarchical interpolation

Let $D \subset \mathbb{R}^2$ be a bounded and connected domain with boundary ∂D and let $J \in \mathbb{N}$. Consider the sequence of nested quasi-uniform triangulations with points $X_0 \subset X_1 \subset \cdots \subset X_J \subset D$, where each triangle is dyadically divided into four triangles in each refinement step. Let $V_j := \operatorname{span}\{\phi_{j,k} : 1 \leq k \leq N_j\}$, $j = 1, \ldots, J$, denote the associated spaces of piecewise linear, nodal triangular functions such that $\phi_{j,k}(\mathbf{x}_{j,\ell}) = \delta_{k,\ell}$.

The *hierarchical interpolant* $f_J \in V_J$ for a given function $f \in C(D)$ is defined as

$$f(\mathbf{x}) \approx f_J(\mathbf{x}) := \sum_{j=0}^{J} \sum_{k \in Y_j} c_{j,k}\phi_{j,k}(\mathbf{x}), \quad c_{j,k} = f(\mathbf{x}_{j,k}) - \frac{f(\mathbf{x}_{j-1,a(k)}) + f(\mathbf{x}_{j-1,b(k)})}{2}, \tag{1}$$

where $Y_j := \{k : \mathbf{x}_{j,k} \notin X_{j-1}\}$ and each $\mathbf{x}_{j,k}$ is the midpoint of the edge from $\mathbf{x}_{j-1,a(k)}$ and $\mathbf{x}_{j-1,b(k)}$. At the initial level, we have $c_{0,k} = f(\mathbf{x}_{0,k})$, $k \in \{k : \mathbf{x}_{0,k} \in X_0\}$.

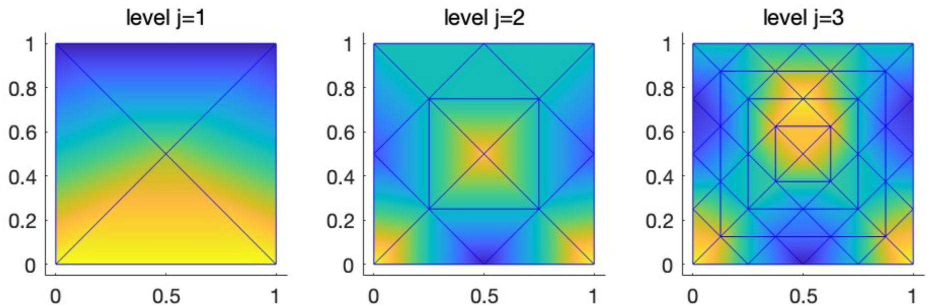


Figure 1: Nodal interpolant on first three levels.

Exercise 1. Implement a Matlab function

```
function cf = compute_multilevel_interpolation(ff, mgmesh, J),
```

which calculates the coefficients of the hierarchical interpolant (1). Store the coefficients in the cell array `c`, i.e., `c{j}(k)` $= c_{j,k+n_{j-1}}$ and n_{j-1} is the number of points on the previous level.

Exercise 2.

Plot the discretized L_∞-norm of contributions `norm(c{j}, inf)` on the each level up to $J = 8$ in a semi-log scale. Use `mesh = mgmesh_generation(J)` to generate a mesh and consider the function

```
ff = @(x) cos(2*pi*x(1))*cos(1.5*pi*x(2)).
```

You should observe a decay with rate 4^{-j}, compare the left plot in Figure 2.

Exercise 3.

To test your implementation, compare the hierarchical interpolant with the standard, nodal interpolant on the finest level. Consider the same function as in Exercise 2 and compute the hierarchical interpolant on level $J = 8$. Transform your multilevel representation to the single level representation by using the following code:

```
fj = c{1};
for j=2:J
    fj = [fj;c{j}];
    Tj = mgmesh.T{j-1};
    np1 = mgmesh.np(j-1);
    np2 = mgmesh.np(j);
    w = Tj*fj(1:np1);
    fj(np1+1:np2) = fj(np1+1:np2) + w(np1+1:np2);
end
```

Compare the transformed interpolant and the nodal interpolant obtained by

`fh = compute_nodal_interpolation_mg(ff, mgmesh, J).`

The discrete L_∞-norm of the error should be around zero.

Sprase grid interpolation

We shall next compute the *sparse interpolant* for a bivariate function $k \in C(D \times D)$, given by

$$k(\mathbf{x}, \mathbf{y}) \approx \widehat{k}_J(\mathbf{x}, \mathbf{y}) := \sum_{j_1+j_2 \le J} \sum_{k_1 \in Y_{j_1}} \sum_{k_2 \in Y_{j_2}} c_{(j_1,j_2),(k_1,k_2)} \phi_{j_1,k_2}(\mathbf{x}) \otimes \phi_{j_2,k_2}(\mathbf{y}) \tag{2}$$

Exercise 4. Implement a Matlab function

`function sck = compute_sparse_interpolation(kf, mgmesh, J),`

which calculates the coefficients of the hierarchical interpolant (1) a function in two variables. Store the coefficients in the cell array `sc`, i.e., `sc{j1,j2}(k1,k2)` $= c_{(j_1,j_2),(k_1+n_{j_1-1},k_2+n_{j_2-1})}$, where n_{j_1-1} and n_{j_2-1} is the number of elements on the previous level j_1 and j_2, respectively.

Compute the coefficients as `sck{j1,j2} = cj12(np11+1:np12, np21+1:np22)`, where

$$\texttt{cj12} = \begin{cases} \texttt{khj} & \texttt{if j1=1 and j2=1} \\ \texttt{khj - khj2*Tj2'} & \texttt{if j1=1} \\ \texttt{khj - Tj1*khj1} & \texttt{if j2=1} \\ \texttt{khj + Tj1*khj12*Tj2' - khj2*Tj2' - Tj1*khj1} & \texttt{else} \end{cases}$$

and

`khj = compute_bivariate_nodal_interpolation(kf,mgmesh,j1,j2),`

`khj1 = compute_bivariate_nodal_interpolation(kf,mgmesh,j1,j2-1),`

`khj2 = compute_bivariate_nodal_interpolation(kf,mgmesh,j1-1,j2),`

`khj12 = compute_bivariate_nodal_interpolation(kf,mgmesh,j1-1,j2-1),`

`Tj1 = mgmesh.T{j1-1},   Tj2 = mgmesh.T{j2-1},`

`np11 = mgmesh.np(j1-1),   np12 = mgmesh.np(j1),`

`np21 = mgmesh.np(j2-1),   np22 = mgmesh.np(j2).`

Obviously, `np11 = 0` if `j1 = 1` and `np21 = 0` if `j2 = 1`. The *sparse interpolant* (2) is obtained if one computes only those arrays `sck{j1,j2}` where `j1+j2 <= J`. The function

`khj = compute_bivariate_nodal_interpolation(kf,mgmesh,j1,j2)`

simply calculates the *nodal tensor product interpolant* on the levels `j1, j2`.

Exercise 5.

To test your implementation, compute the *hierarchical tensor product interpolant*, i.e., compute the arrays `sc{j1,j2}` for all `j1 <= J` and `j2 <= J`. Transform it to the *nodal tensor product interpolant* like in Exercise 2. This means, transform it first for one variable and then for the other variable. Realize this in the function

```
function kj = transform_hierarchical_to_single(sck, mgmesh, J).
```

Compute finally the L_∞-norm of its difference to the directly computed nodal tensor product interpolant. This error should be around zero. Test you code for the function

```
kf = @(x , y) exp(-((x(1)-y(1))^2+(x(2)-y(2))^2)).
```

The code should work relatively fast up to `J = 6` levels.

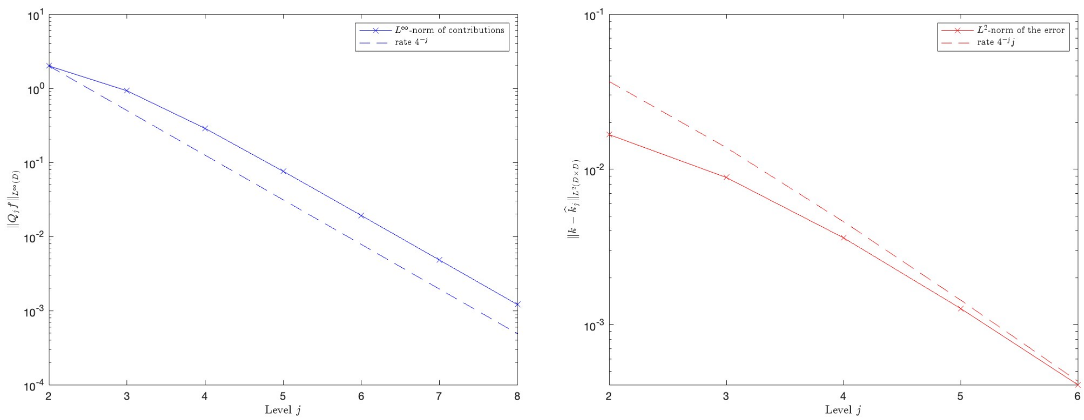
Exercise 6.

For the function `of` from the Exercise 5, plot the L^2-error $\|k_j - \widetilde{k}_j\|_{L^2(D\times D)}$ for $j = 2, \ldots, 6$, where k_j is the *nodal tensor product interpolant* and $\widetilde{k}_j$ is the *sparse interpolant* transformed to the nodal interpolant by using the function from Exercise 5. The evaluation of the L^2-error on level j requires the mass matrix on level j, which is computed by

```
M = mgsys_assemble(mgmesh).M{j}.
```

The L^2-error should decay like $4^{-j} j$, compare the right plot in Figure 2.

Hinweis. *To calculate $\|\cdot\|_{L^2(D\times D)}$ you do not need to compute tensor $M \otimes M$ directly, see Aufgabe 3 in the Übungsblatt 3.*



(a) (b)

Figure 2: (a): Norm of the contributions $\|Q_j f\|_{L^\infty(D)}$; (b): Error $\|k_j - \widetilde{k}_j\|_{L^2(D\times D)}$.