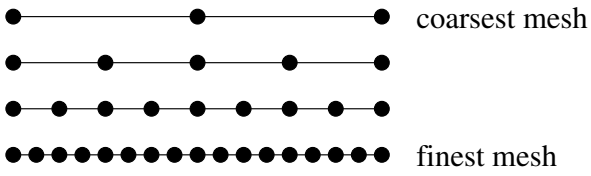
In this tutorial we consider the MDS (multilevel diagonal scaling) preconditioner. Let $\{\mathcal{T}_\ell\}_{\ell=1}^L$ be nested meshes of the interval $(0, 1)$. We start with a fixed mesh $\mathcal{T}_1$ (with a few elements) and construct the other meshes by uniform refinement as shown below.



For each $\ell = 1, \dots, L$ we define

$$V_\ell := \{v \in H^1(0, 1) : v_{|T} \in \mathcal{P}^1 \quad \forall T \in \mathcal{T}_\ell\} \;=\; \operatorname{span}\{\varphi_{\ell,i}\}_{i=0}^{n_\ell}$$

with the nodal basis functions $\{\varphi_{\ell,i}\}_{i=0}^{n_\ell}$ and n_ℓ being the number of elements of $\mathcal{T}_\ell$.

44 Consider two consecutive meshes $\mathcal{T}_\ell$ and $\mathcal{T}_{\ell+1}$ and the corresponding finite element spaces $V_\ell \subset V_{\ell+1}$. Recall that for every $w_\ell \in V_\ell$ there exist a vector $\underline{w}_\ell = [w_{\ell,i}]_{i=0}^{n_\ell}$ such that

$$w_\ell(x) \;=\; \sum_{i=0}^{n_\ell} w_{\ell,i}\, \varphi_{\ell,i}(x).$$

Let $w_\ell \in V_\ell$ be fixed. Since $w_\ell \in V_{\ell+1}$ as well, there exists a vector $\underline{w}_{\ell+1} = [w_{\ell+1,i}]_{i=0}^{n_{\ell+1}}$ such that

$$w_\ell(x) \;=\; \sum_{i=0}^{n_{\ell+1}} w_{\ell+1,i}\, \varphi_{\ell+1,i}(x).$$

Write the coefficients $w_{\ell+1,i}$ in terms of $w_{\ell,i}$. Find a matrix $I_\ell^{\ell+1} \in \mathbb{R}^{n_{\ell+1}+1 \times n_\ell+1}$ such that

$$\underline{w}_{\ell+1} \;=\; I_\ell^{\ell+1} \underline{w}_\ell\,.$$

45 Let $R \in V^*$ a bounded linear functional. Let $\ell < L$ be fixed and define the residual vector $\underline{r}_{\ell+1} = [r_{\ell+1,i}]_{i=0}^{n_{\ell+1}}$ by $r_{\ell+1,i} := \langle R, \varphi_{\ell+1,i}\rangle$. Then,

$$\langle R, v_{\ell+1}\rangle \;=\; \sum_{i=0}^{n_{\ell+1}} r_{\ell+1,i}\, v_{\ell+1,i} \;=\; (\underline{r}_{\ell+1}, \underline{v}_{\ell+1})_{\ell^2}$$

for all $v_{\ell+1} \in V_{\ell+1}$ with basis representation $\underline{v}_{\ell+1}$.
As above, define $\underline{r}_\ell = [r_{\ell,i}]_{i=0}^{n_\ell}$ with $r_{\ell,i} := \langle R, \varphi_{\ell,i}\rangle$.

Write $\underline{r}_\ell$ in terms of $\underline{r}_{\ell+1}$ such that

$$\langle R, v_\ell\rangle \;=\; (\underline{r}_\ell, \underline{v}_\ell)_{\ell^2}$$

for all $v_\ell \in V_\ell$ with basis representation $\underline{v}_\ell$. Show that

$$\underline{r}_\ell \;=\; I_{\ell+1}^{\ell}\, \underline{r}_{\ell+1} \qquad \text{with} \quad I_{\ell+1}^{\ell} \;=\; (I_\ell^{\ell+1})^\top.$$

46 In the lecture, we have treated the subspace correction equation

$$a(\varphi_{\ell,i}, \varphi_{\ell,i}) w_{\ell,i} \;=\; \langle F, \varphi_{\ell,i}\rangle - a(u_L^{(k)}, \varphi_{\ell_i}) \;=:\; \langle R^{(k)}, \varphi_{\ell,i}\rangle.$$

We define $\underline{r}_\ell^{(k)} = [\langle R^{(k)}, \varphi_{\ell,i}\rangle]_{i=0}^{n_\ell}$. For any arbitrary level $\ell < L$, write $\underline{r}_\ell^{(k)}$ in terms of $\underline{f}_L - K_L\, \underline{u}_L^{(k)}$, where $\underline{f}_L$ denotes the load vector on level L and K_L the stiffness matrix. *Hint:* Use Exercise **45**.

Programming

47 Write a function

```
void RefineUniform (const Mesh& coarseMesh, Mesh& fineMesh);
```

that computes the refined mesh $\mathcal{T}_{\ell+1}$=`fineMesh` from a given mesh $\mathcal{T}_\ell$=`coarseMesh` as shown above.

48 (a) Write a function

```
void Restrict (const Vector& fineRes, Vector& coarseRes);
```

that computes the coarse residual `coarseRes`=$\underline{r}_\ell = I_{\ell+1}^{\ell}\,\underline{r}_{\ell+1}$ from the fine residual `fineRes`=$\underline{r}_{\ell+1}$.
Hint: Use the entries of $I_{\ell+1}^{\ell}$ from above, but set $r_{\ell,0} = 0$ (due to the incorporated Dirichlet condition).

(b) Write a function

```
void Prolongate (const Vector& coarseVec, Vector& fineVec);
```

that computes `fineVec`=$\underline{v}_{\ell+1} = I_{\ell}^{\ell+1}\underline{v}_\ell$ from `coarseVec`=$\underline{v}_\ell$.
Hint: Use the entries of $I_{\ell}^{\ell+1}$ from above, but set $v_{\ell,0} = 0$ (due to the incorporated Dirichlet condition).

49 Consider `mds.hh` from the website and implement the class routine `ApplyCL`, i.e.

apply the Jacobi preconditioner to get a correction `w` from `r`
if `level > 0`
  restrict `r` to a coarse residual `rc`
  call `ApplyCL(level-1, rc, wc)` (recursively) to get a coarse correction `wc`
  prolongate `wc` to a fine correction `wf`
  add `wf` to `w`

Your code does *not* have to compile!

```
class MDSPreconditioner
{
public:
  // constructor: initialize with (total) number of levels
  MDSPreconditioner (int numLevels);

  // destructor
  ~MDSPreconditioner ();

  // set the diagonal for Jacobi preconditioner for a certain level
  void InitDiagonal (int level, const SMatrix& K);

  // apply the preconditioner C^{-1}
  inline
  void Solve (const Vector& r, Vector& w) const
  {
    ApplyCL (numLevels_-1, r, w);
  }

private:

  // recursive helper routine
  void ApplyCL (int level, const Vector& r, Vector& w) const;

  //--- members

  int numLevels_;                // number of levels
  std::vector<Vector> jacobi_;   // diagonal for Jacobi preconditioner on each level
};
```