

Some computational results for $\mathbf{P}_1 - P_1$ and $\mathbf{P}_2 - P_1$ Trace FEM for the surface Stokes problem

Alexander Zhiliakov*

July 15, 2019

Contents

1 Inf-sup stability: generalized pressure Schur complement eigenvalues	1
1.1 Bilinear forms and matrices	1
1.2 Solution description	2
1.3 Numerical results: dependency of the spectrum on the mesh size	3
1.4 Numerical results: sensitivity of the spectrum to levelset shifts	6
2 Convergence results	8
2.1 Manufactured solution	8
2.2 $P_2 - P_1$ Trace FEM	9
2.2.1 Inconsistent penalty formulation	9
2.2.2 Consistent penalty formulation	10
3 Notes on DROPS implementation	11
3.1 Notations	11
3.2 Approximation of integrands involving n_Γ	11
3.3 Quadrature rules for \int_Γ and $\int_{\Omega_\Gamma^h}$	13
3.4 Approximation of the shape operator \mathbf{H}	14
3.5 Summary on the matrix assembly	15
3.6 Using exact normals in integrands of $\int_{\Gamma_{h/m}^{2 \rightarrow 1}}$ (updated summary)	15
3.7 Quadrature rules for the error computation	18

1 Inf-sup stability: generalized pressure Schur complement eigenvalues

1.1 Bilinear forms and matrices

We set n_A to be the number of velocity d.o.f. and n_S to be the number of pressure d.o.f. Vector stiffness, divergence, pressure mass, normal stabilization, and full stabilization matrices resulting from Trace FEM dis-

*Department of Mathematics, University of Houston, Houston, Texas 77204 (alex@math.uh.edu).

cretization of the surface Stokes problem [2] are defined via

$$\begin{aligned}
\langle \mathbf{A} \vec{\mathbf{u}}, \vec{\mathbf{v}} \rangle &\approx \int_{\Gamma} (E_s(\mathbf{u}) : E_s(\mathbf{v}) + \mathbf{u} \cdot \mathbf{v} + \tau (\mathbf{u} \cdot \mathbf{n}) (\mathbf{v} \cdot \mathbf{n})) \, ds + \rho_u \int_{\Omega_h^{\Gamma}} \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{n}} \, dx, \quad \mathbf{A} \in \mathbb{R}^{n_A \times n_A}, \\
\langle \mathbf{B} \vec{\mathbf{u}}, \vec{\mathbf{q}} \rangle &\approx - \int_{\Gamma} q \operatorname{div}_{\Gamma} \mathbf{u} \, ds, \quad \mathbf{B} \in \mathbb{R}^{n_S \times n_A}, \\
\langle \mathbf{M}_0 \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &\approx \int_{\Gamma} p q \, ds, \quad \mathbf{M}_0 \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_n \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &\approx \rho_p \int_{\Omega_h^{\Gamma}} \frac{\partial p}{\partial \mathbf{n}} \frac{\partial q}{\partial \mathbf{n}} \, dx, \quad \mathbf{C}_n \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_{\text{full}} \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &\approx \rho_p \int_{\Omega_h^{\Gamma}} \nabla p \cdot \nabla q \, dx, \quad \mathbf{C}_{\text{full}} \in \mathbb{R}^{n_S \times n_S},
\end{aligned} \tag{1}$$

respectively. We use notations as in [2], in particular, Ω_h^{Γ} is the domain consisting of tetrahedra cut by Γ . Here $\vec{\mathbf{u}}$ denotes a vector of d.o.f. corresponding to a FE interpolant \mathbf{u} (analogously for $\vec{\mathbf{p}}$ and p). See (23) and (24) for the computational details. Mesh-dependent parameters are set as

$$\tau = h^{-2}, \quad \rho_u = \rho_p = h, \tag{2}$$

and h is the typical mesh size for tetrahedra from Ω_h^{Γ} . Γ is chosen either as the unit sphere or torus, $\Gamma = \Gamma_{\text{sph}}$ or $\Gamma = \Gamma_{\text{tor}}$ (see Figure 1).

We also define matrices

$$\mathbf{C}_0 := \mathbf{0}, \quad \mathbf{M}_n := \mathbf{M}_0 + \mathbf{C}_n, \quad \mathbf{M}_{\text{full}} := \mathbf{M}_0 + \mathbf{C}_{\text{full}}. \tag{3}$$

We are interested in (generalized) extreme eigenvalues of the pressure Schur complement matrices

$$\mathbf{S}_0 := \mathbf{B} \mathbf{A}^{-1} \mathbf{B}^T, \quad \mathbf{S}_n := \mathbf{S}_0 + \mathbf{C}_n, \quad \mathbf{S}_{\text{full}} := \mathbf{S}_0 + \mathbf{C}_{\text{full}}, \tag{4}$$

i.e. in solving

$$\mathbf{S}_{\star} \mathbf{x} = \lambda \mathbf{M}_{\star} \mathbf{x}, \tag{5}$$

where “ \star ” stands for “0,” “ n ,” or “full.” We denote by $0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_{n_S} = O(1)$ the spectrum of (5).

1.2 Solution description

Computing \mathbf{A}^{-1} in (4) becomes troublesome already for $h = 5.21 \times 10^{-2}$ ($n_A = 32736$ for $\mathbf{u} \in \mathbf{P}_1$ FE space): although \mathbf{A} is sparse, \mathbf{A}^{-1} is dense and consumes 8.5+ GB in double-precision arithmetic. A quick research showed that **Mathematica** has no built-in matrix-free eigenvalue routines. Intel MKL’s FEAST algorithm for computing (generalized) eigenvalues in an interval is suitable for matrix-free implementations; however, it requires some expensive operations to be implemented (e.g. matrix-matrix multiplications $\mathbf{Y} \leftarrow \mathbf{S}_{\star} \mathbf{X}$, $\mathbf{Y} \leftarrow \mathbf{M}_{\star} \mathbf{X}$ and approximating the action of inverses in the form $\mathbf{y} \leftarrow (\sigma \mathbf{M}_{\star} - \mathbf{S}_{\star})^{-1} \mathbf{x}$).

Taking this into account, instead of (5) we consider a perturbed¹ problem

$$\underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & -\mathbf{C}_{\star} \end{bmatrix}}_{\mathcal{A}_{\star} :=} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mu \underbrace{\begin{bmatrix} \epsilon \mathbf{A} & \\ & \mathbf{M}_{\star} \end{bmatrix}}_{\mathcal{M}_{\star}^{\epsilon} :=} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{6}$$

with $0 < \epsilon \ll 1$. For \mathcal{A}_0 and $\mathcal{M}_{\star}^{\epsilon}$ we have

$$\mu = -\lambda + o(1) \quad \text{or} \quad \epsilon^{-1} + \lambda + o(1), \quad \epsilon \rightarrow 0. \tag{7}$$

This makes it easy to pick only “correct” eigenvalues. To ease the computation further we replace the $(1, 1)$ -block of $\mathcal{M}_\star^\epsilon$ with $\epsilon \mathbf{I}$.

To make sure that results are consistent we solve (6) for $\epsilon = 10^{-5}$ and $\epsilon = 10^{-6}$; for the coarse mesh levels we also check that the dense solver for (5) and the iterative one for (6) give solutions that coincide.

1.3 Numerical results: dependency of the spectrum on the mesh size

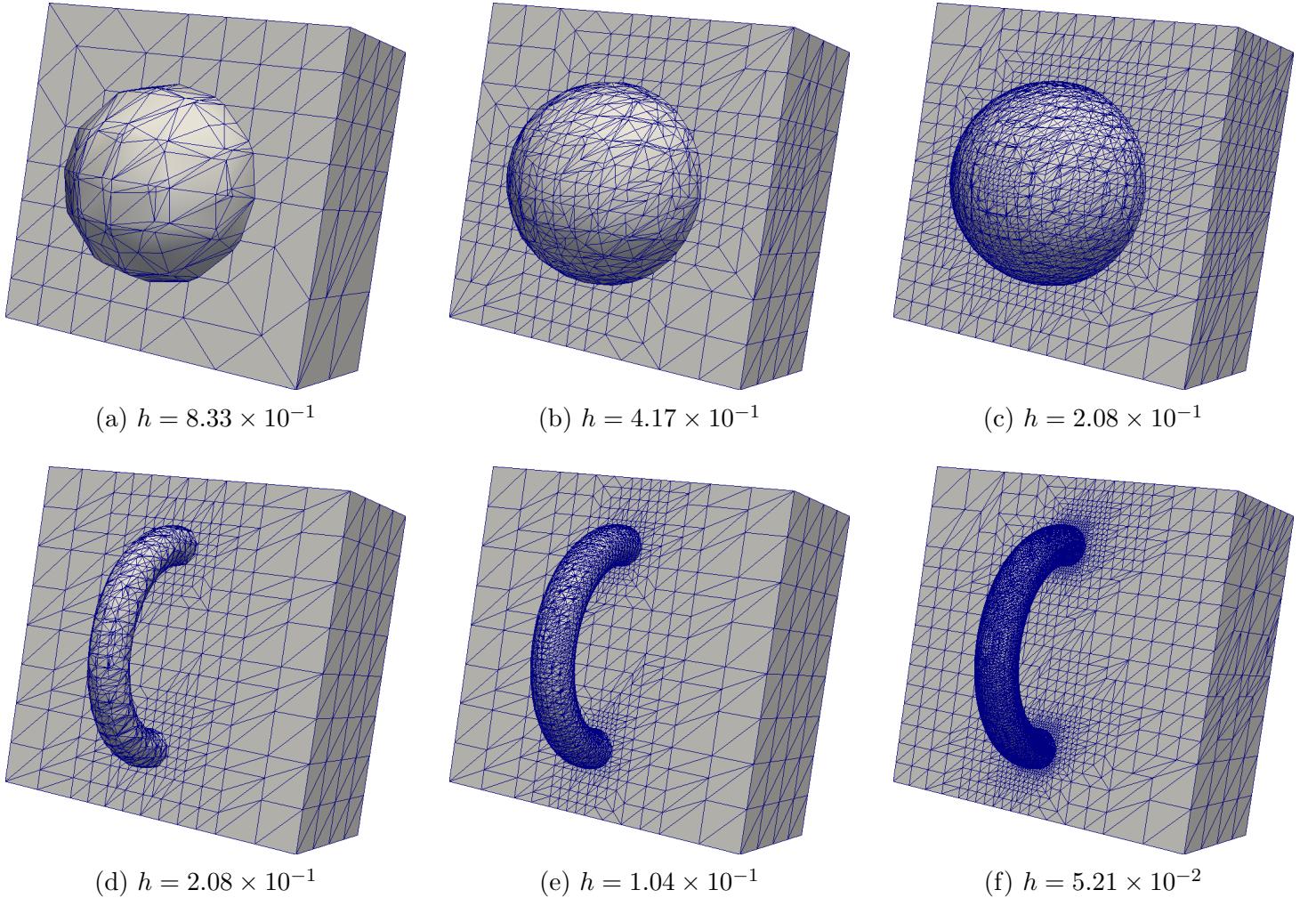


Figure 1: First three mesh levels for Γ_{sph} (top) and Γ_{tor} (bottom)

¹The majority of generalized eigenvalue solvers require left-hand-side matrix to be Hermitian and right-hand-side matrix to be Hermitian **positive definite**; that’s why we need to introduce $\epsilon > 0$.

Table 1: Spectrum of (5) for $\mathbf{P}_1 - P_1$

 (a) $\Gamma = \Gamma_{\text{sph}}$

h	$n_{\mathbf{A}}$	$n_{\mathbf{S}}$	\mathbf{S}_0		\mathbf{S}_n		\mathbf{S}_{full}	
			λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$
8.33×10^{-1}	153	51	1.32×10^{-2}	1.42	7.48×10^{-1}	1.13	9.58×10^{-1}	1.06
4.17×10^{-1}	570	190	5.12×10^{-3}	1.04	5.77×10^{-1}	1.	8.54×10^{-1}	1.
2.08×10^{-1}	1992	664	4.4×10^{-3}	7.93×10^{-1}	3.87×10^{-1}	1.	6.71×10^{-1}	1.
1.04×10^{-1}	8292	2764	2.01×10^{-3}	7.79×10^{-1}	2.19×10^{-1}	1.	5.82×10^{-1}	1.
5.21×10^{-2}	32736	10912	6.04×10^{-5}	9.81×10^{-1}	1.17×10^{-1}	1.	5.37×10^{-1}	1.
2.6×10^{-2}	131592	43864	3.53×10^{-5}	8.67×10^{-1}	5.72×10^{-2}	1.	5.16×10^{-1}	1.
1.3×10^{-2}	525864	175288	2.16×10^{-6}	7.34×10^{-1}	2.84×10^{-2}	1.	5.04×10^{-1}	1.

 (b) $\Gamma = \Gamma_{\text{tor}}$

h	$n_{\mathbf{A}}$	$n_{\mathbf{S}}$	\mathbf{S}_0		\mathbf{S}_n		\mathbf{S}_{full}	
			λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$
2.08×10^{-1}	972	324	5.04×10^{-2}	4.93	2.84×10^{-1}	1.35	3.64×10^{-1}	1.19
1.04×10^{-1}	4740	1580	2.99×10^{-3}	3.83	1.58×10^{-1}	1.02	3.35×10^{-1}	1.01
5.21×10^{-2}	19704	6568	1.11×10^{-3}	5.45	7.73×10^{-2}	1.01	3.25×10^{-1}	1.
2.6×10^{-2}	80808	26936	1.2×10^{-4}	5.42	3.07×10^{-2}	1.01	3.21×10^{-1}	1.
1.3×10^{-2}	327036	109012	1.77×10^{-5}	5.23	1.18×10^{-2}	1.01	3.16×10^{-1}	1.

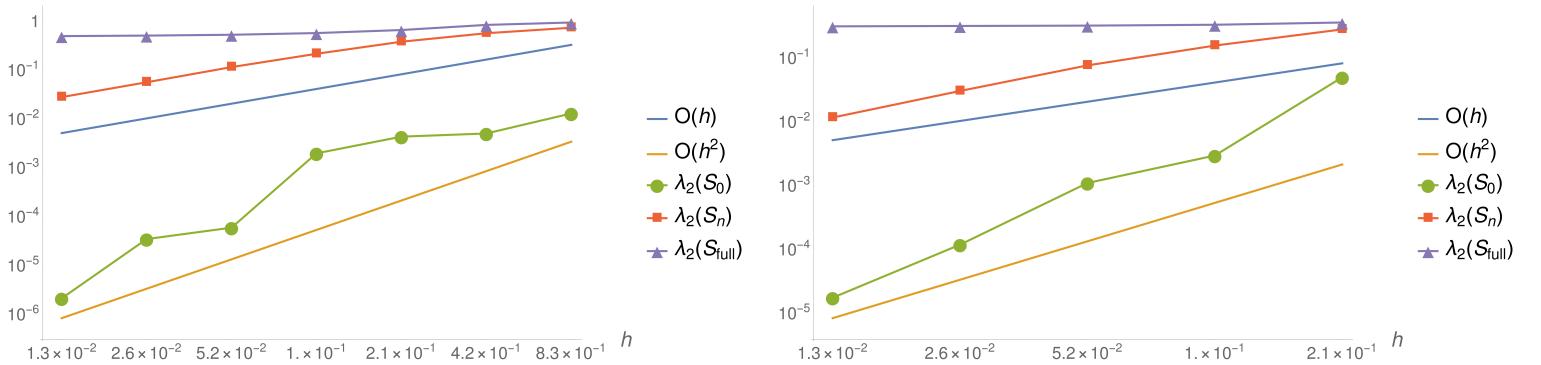

 Figure 2: Log-log plot of λ_2 for Tables 1a (left) and 1b (right)

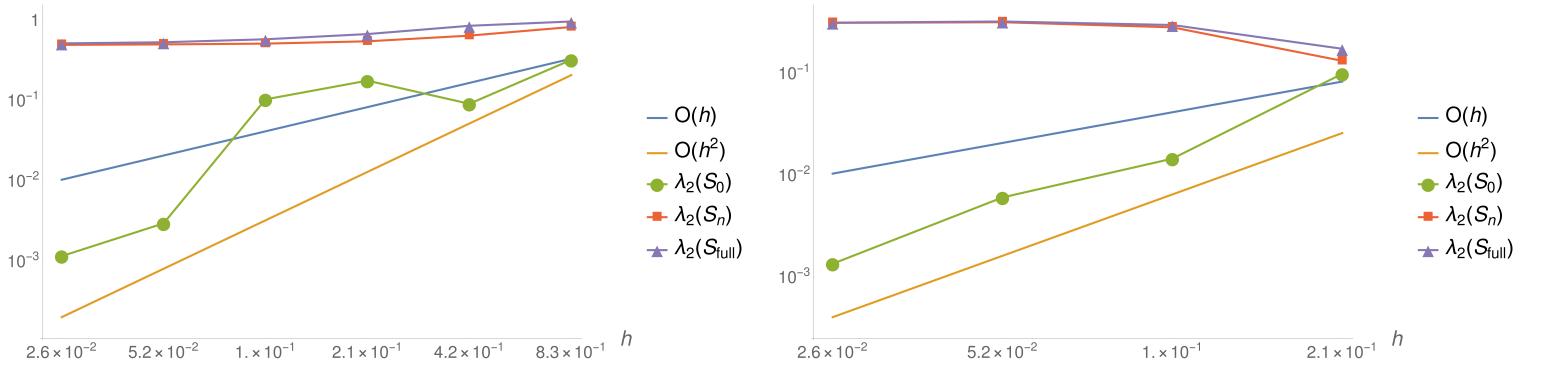
Table 2: Spectrum of (5) for $\mathbf{P}_2 - P_1$

 (a) $\Gamma = \Gamma_{\text{sph}}$

h	$n_{\mathbf{A}}$	$n_{\mathbf{S}}$	\mathbf{S}_0		\mathbf{S}_n		\mathbf{S}_{full}	
			λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$
8.33×10^{-1}	789	51	3.22×10^{-1}	1.73	8.27×10^{-1}	1.17	9.68×10^{-1}	1.07
4.17×10^{-1}	3240	190	9.17×10^{-2}	1.08	6.45×10^{-1}	1.	8.56×10^{-1}	1.
2.08×10^{-1}	11718	664	1.78×10^{-1}	8.31×10^{-1}	5.49×10^{-1}	1.	6.75×10^{-1}	1.
1.04×10^{-1}	48762	2764	1.04×10^{-1}	8.35×10^{-1}	5.14×10^{-1}	1.	5.82×10^{-1}	1.
5.21×10^{-2}	193014	10912	2.99×10^{-3}	9.89×10^{-1}	5.02×10^{-1}	1.	5.34×10^{-1}	1.
2.6×10^{-2}	775998	43864	1.17×10^{-3}	7.9×10^{-1}	4.96×10^{-1}	1.	5.17×10^{-1}	1.

 (b) $\Gamma = \Gamma_{\text{tor}}$

h	$n_{\mathbf{A}}$	$n_{\mathbf{S}}$	\mathbf{S}_0		\mathbf{S}_n		\mathbf{S}_{full}	
			λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$	λ_2	$\lambda_{n_{\mathbf{S}}}$
2.08×10^{-1}	5184	324	9.92×10^{-2}	3.89	1.33×10^{-1}	1.37	1.75×10^{-1}	1.19
1.04×10^{-1}	27906	1580	1.46×10^{-2}	4.35	2.84×10^{-1}	1.04	2.99×10^{-1}	1.02
5.21×10^{-2}	116568	6568	6.08×10^{-3}	4.85	3.19×10^{-1}	1.01	3.24×10^{-1}	1.01
2.6×10^{-2}	477660	26936	1.36×10^{-3}	4.92	3.14×10^{-1}	1.01	3.16×10^{-1}	1.


 Figure 3: Log-log plot of λ_2 for Tables 2a (left) and 2b (right)

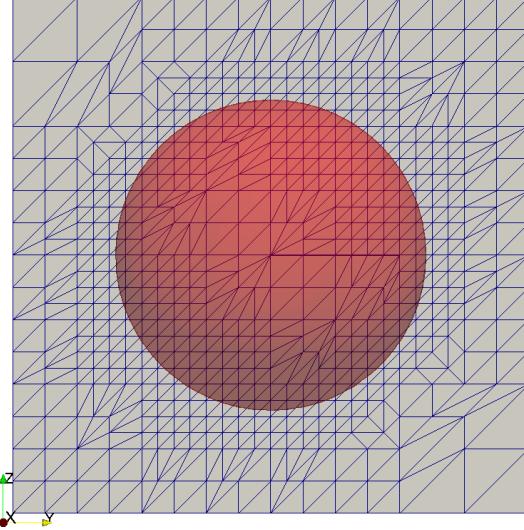
1.4 Numerical results: sensitivity of the spectrum to levelset shifts

In this section we investigate the sensitivity of the spectrum to levelset shifts

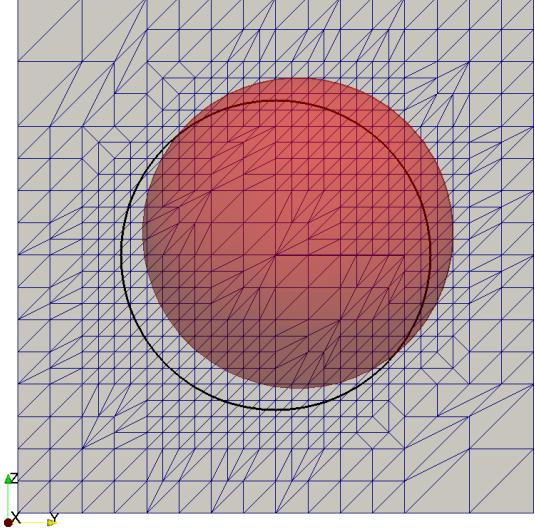
$$\Gamma \mapsto \Gamma + \alpha \mathbf{s}, \quad (8)$$

for some $\alpha \in \mathbb{R}$ and $\mathbf{s} \in \mathbb{R}^3$, $\|\mathbf{s}\| = 1$.

We construct the bulk mesh Ω_h^Γ and then perform the assembly of matrices (1) using the shifted levelset (8). That is, the refinement of Ω_h^Γ is performed using Γ , not $\Gamma + \alpha \mathbf{s}$, and $\Omega_h^{\Gamma+\alpha\mathbf{s}}$ is never constructed. We choose $\alpha \in [0, h]$ to guarantee the appearance of “small cuts” in Ω_h^Γ .



(a) Γ_{sph}



(b) $\Gamma_{\text{sph}} + \alpha \mathbf{s}$

Figure 4: The unit sphere (left) and the shifted unit sphere (right). Here $\mathbf{s} = (0, 1, 1)^T / \sqrt{2}$, $\alpha = 0.2$, and $h = 2.08 \times 10^{-1}$. The bulk mesh Ω_h^Γ is computed for Γ_{sph} and then used for $\Gamma_{\text{sph}} + \alpha \mathbf{s}$

Table 3: Spectrum of (5) for perturbed levelset $\Gamma_{\text{sph}} + \alpha \mathbf{s}$. Here $\mathbf{s} = (1, 1, 1)^T / \sqrt{3}$, $h = 1.04 \times 10^{-1}$

(a) $\mathbf{P}_1 - P_1$

Surface	\mathbf{S}_0		\mathbf{S}_n		\mathbf{S}_{full}	
	λ_2	λ_{n_s}	λ_2	λ_{n_s}	λ_2	λ_{n_s}
Γ_{sph}	2.006×10^{-3}	7.79×10^{-1}	2.19×10^{-1}	1.	5.818×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.1 h \mathbf{s}$	4.832×10^{-4}	8.01×10^{-1}	2.195×10^{-1}	1.	5.818×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.3 h \mathbf{s}$	7.278×10^{-4}	8.17×10^{-1}	2.203×10^{-1}	1.	5.818×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.5 h \mathbf{s}$	3.121×10^{-4}	8.67×10^{-1}	2.221×10^{-1}	1.	5.82×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.7 h \mathbf{s}$	1.438×10^{-3}	1.51	2.254×10^{-1}	1.	5.82×10^{-1}	1.
$\Gamma_{\text{sph}} + h \mathbf{s}$	1.79×10^{-3}	2.07	2.332×10^{-1}	1.	5.827×10^{-1}	1.

(b) $\mathbf{P}_2 - P_1$

Surface	\mathbf{S}_0		\mathbf{S}_n		\mathbf{S}_{full}	
	λ_2	λ_{n_s}	λ_2	λ_{n_s}	λ_2	λ_{n_s}
Γ_{sph}	1.041×10^{-1}	8.35×10^{-1}	5.138×10^{-1}	1.	5.841×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.1 h \mathbf{s}$	1.705×10^{-3}	8.58×10^{-1}	5.138×10^{-1}	1.	5.841×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.3 h \mathbf{s}$	2.293×10^{-3}	8.81×10^{-1}	5.137×10^{-1}	1.	5.841×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.5 h \mathbf{s}$	5.63×10^{-3}	9.35×10^{-1}	5.138×10^{-1}	1.	5.844×10^{-1}	1.
$\Gamma_{\text{sph}} + 0.7 h \mathbf{s}$	8.188×10^{-3}	1.77	5.138×10^{-1}	1.	5.843×10^{-1}	1.
$\Gamma_{\text{sph}} + h \mathbf{s}$	1.93×10^{-2}	2.21	5.142×10^{-1}	1.	5.852×10^{-1}	1.

2 Convergence results

2.1 Manufactured solution

We solve model problem from [2, p. 20], $\Gamma = \Gamma_{\text{sph}}^2$. We set

$$\tilde{\mathbf{u}}(x, y, z) := (-z^2, y, x)^T, \quad \tilde{p}(x, y, z) := xy^2 + z, \quad \phi(\mathbf{x}) := \|\mathbf{x}\|^2 - 1. \quad (9)$$

The exact solution on the unit sphere is chosen as

$$\mathbf{u} := \mathbf{P} \tilde{\mathbf{u}}^e, \quad p := \tilde{p}^e. \quad (10)$$

Thus we have $\int_{\Gamma} p d\mathbf{x} = 0$, $p \equiv p^e$, $\mathbf{u} \equiv \mathbf{u}^e$ in $\mathcal{O}(\Gamma)$, and \mathbf{u} is a tangential field. Note that for our choice of ϕ in (9) we have

$$\mathbf{n}_{\Gamma_h^2} = \mathbf{n}_{\Gamma}^e \text{ in } \mathcal{O}(\Gamma), \quad \Gamma_{h/m}^{2 \rightarrow 1} = \Gamma_{h/m}^1, \quad \mathbf{n}_{\Gamma_{h/m}^1} = \mathbf{n}_{\Gamma_{h/m}^1} \text{ on } \Gamma, \quad (11)$$

and

$$\mathbf{n}_{\Gamma_{h/m}^1} \rightarrow \mathbf{n}_{\Gamma}, \quad \Gamma_{h/m}^1 \rightarrow \Gamma \quad (12)$$

as one increases m even for fixed h .

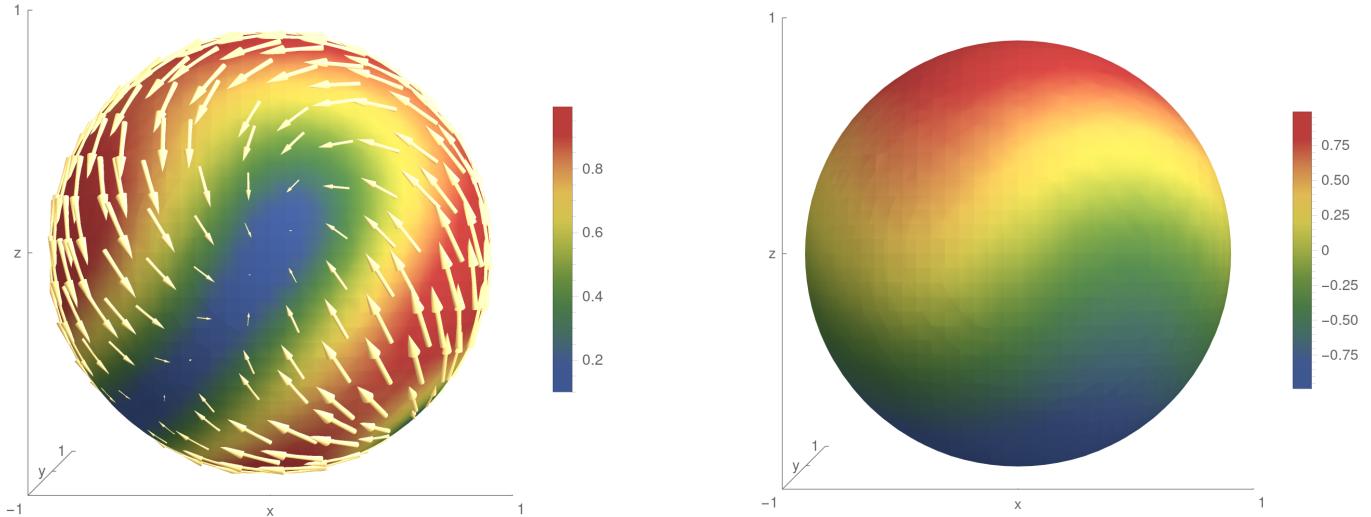


Figure 5: Exact velocity solution (Left) and pressure solution (Right) as in (10)

We have

$$\mathbf{n}_{\Gamma_{\text{sph}}}(\mathbf{x}) = \|\mathbf{x}\|^{-1} \mathbf{x} = \mathbf{n}_{\Gamma_{\text{sph}}}^e(\mathbf{x}), \quad (13)$$

$$\mathbf{P}_{\Gamma_{\text{sph}}}(\mathbf{x}) = \|\mathbf{x}\|^{-2} \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -xy & x^2 + z^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{pmatrix} = \mathbf{P}_{\Gamma_{\text{sph}}}^e(\mathbf{x}), \quad (14)$$

$$\mathbf{H}_{\Gamma_{\text{sph}}}(\mathbf{x}) = \|\mathbf{x}\|^{-3} \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -xy & x^2 + z^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{pmatrix} \neq \mathbf{H}_{\Gamma_{\text{sph}}}^e(\mathbf{x}) = \mathbf{P}_{\Gamma_{\text{sph}}}(\mathbf{x}). \quad (15)$$

We consider two choices for virtual refinement: $m \propto h^{-1/2}$ and $m \propto h^{-1}$. The first choice assures h^3 -accurate approximation of Γ and $h^{3/2}$ accurate approximation of the normal vector, whereas the second choice assures h^4 - and h^2 -approximations. We refer to Figure 6.

²In [2] they use $\mathbf{u} := \mathbf{P} \tilde{\mathbf{u}}$, i.e. $\mathbf{u} \neq \mathbf{u}^e$. I prefer $\mathbf{u} \equiv \mathbf{u}^e$ as in [1].

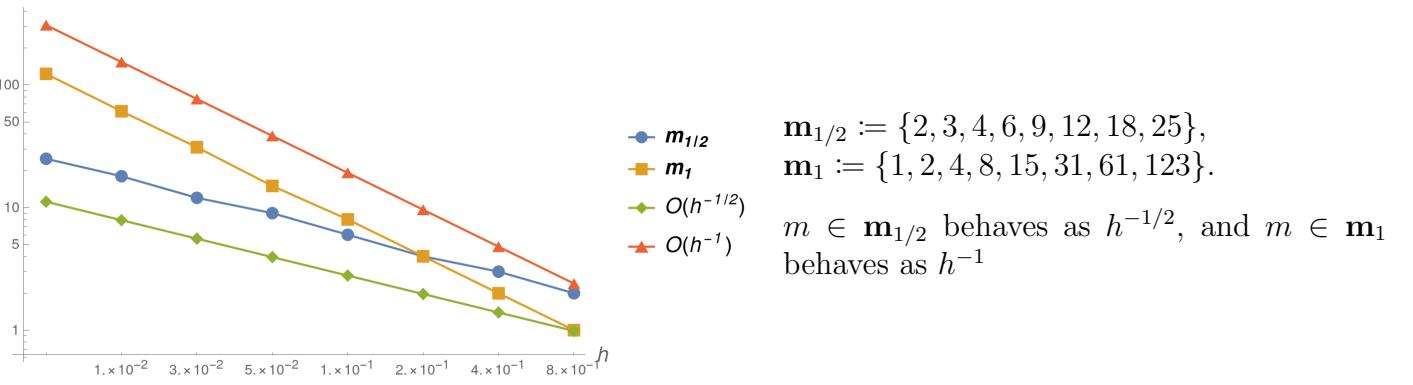


Figure 6: Virtual refinement parameter m for $\Gamma_{h/m}^{2 \rightarrow 1}$

Table 4: Errors for normals and shape operator. Please see (11) and (12)

$m \in \mathbf{m}_{1/2}$ as in Figure 6					
h	$\ \mathbf{n}_\Gamma^e - \mathbf{n}_{\Gamma_h^2}\ _{\mathbb{L}^2(\Gamma_{h/m}^1)}$	$\ \mathbf{n}_\Gamma^e - \mathbf{n}_{\Gamma_{h/m}^1}\ _{\mathbb{L}^2(\Gamma_{h/m}^1)}$	Order	$\ \mathbf{H}_\Gamma^e - \mathbf{H}_{\Gamma_h^2}\ _{\mathbb{L}^2(\Gamma_{h/m}^1)}$	Order
8.3×10^{-1}	8.1×10^{-16}	6.4×10^{-1}		2.3×10^{-1}	
4.2×10^{-1}	1.1×10^{-15}	$2. \times 10^{-1}$	1.7	2.5×10^{-2}	3.2
2.1×10^{-1}	2.3×10^{-15}	7.6×10^{-2}	1.4	3.5×10^{-3}	2.8
$1. \times 10^{-1}$	5.2×10^{-15}	2.5×10^{-2}	1.6	3.9×10^{-4}	3.2
5.2×10^{-2}	9.3×10^{-15}	8.4×10^{-3}	1.6	4.3×10^{-5}	3.2

2.2 $P_2 - P_1$ Trace FEM

Next we compare inconsistent and consistent Trace FEM penalty formulations.

2.2.1 Inconsistent penalty formulation

We use the normal stabilization matrix \mathbf{C}_n . We stick to the approach (23), so with (11) we have

$$\begin{aligned}
\langle \mathbf{A} \vec{\mathbf{u}}, \vec{\mathbf{v}} \rangle &= \int_{\Gamma_{h/2}^1}^5 \left(2 E_{s,\Gamma}(\mathbf{u}) : E_{s,\Gamma}(\mathbf{v}) + \mathbf{u} \cdot \mathbf{v} + \tau (\mathbf{u} \cdot \mathbf{n}_\Gamma) (\mathbf{v} \cdot \mathbf{n}_\Gamma) \right) ds \\
&\quad + \rho_u \int_{\Omega_h^\Gamma}^5 \frac{\partial \mathbf{u}}{\partial \mathbf{n}_\Gamma} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{n}_\Gamma} d\mathbf{x}, \quad \mathbf{A} \in \mathbb{R}^{n_A \times n_A}, \\
\langle \mathbf{B} \vec{\mathbf{u}}, \vec{\mathbf{q}} \rangle &= \int_{\Gamma_{h/2}^1}^5 \nabla_\Gamma q \cdot \mathbf{u} ds, \quad \mathbf{B} \in \mathbb{R}^{n_S \times n_A}, \\
\langle \mathbf{M}_0 \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \int_{\Gamma_{h/2}^1}^5 p q ds, \quad \mathbf{M}_0 \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_n \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \rho_p \int_{\Omega_h^\Gamma}^5 \frac{\partial p}{\partial \mathbf{n}_\Gamma} \frac{\partial q}{\partial \mathbf{n}_\Gamma} d\mathbf{x}, \quad \mathbf{C}_n \in \mathbb{R}^{n_S \times n_S}.
\end{aligned} \tag{16}$$

Table 5: Convergence results. $\tau = h^{-2}$, $\rho_u = h$, $\rho_p = h$. Matrices are assembled as in (16)

$m \in \mathbf{m}_{1/2}$ as in Figure 6						
h	$\ \mathbf{u} - \mathbf{u}_h\ _{\mathbb{H}^1}$	Order	$\ \mathbf{u} - \mathbf{u}_h\ _{\mathbb{L}^2}$	Order	$\ p - p_h\ _{\mathbb{L}^2}$	Order
8.3×10^{-1}	3.		1.8		2.1	
4.2×10^{-1}	1.8	7.4×10^{-1}	9.1×10^{-1}	9.7×10^{-1}	1.7	2.7×10^{-1}
2.1×10^{-1}	$7. \times 10^{-1}$	1.4	3.4×10^{-1}	1.4	6.9×10^{-1}	1.3
$1. \times 10^{-1}$	$2. \times 10^{-1}$	1.8	9.9×10^{-2}	1.8	$2. \times 10^{-1}$	1.8
5.2×10^{-2}	5.2×10^{-2}	1.9	2.6×10^{-2}	1.9	5.2×10^{-2}	1.9

h	$\ \mathbf{u}_h \cdot \mathbf{n}\ _{\mathbb{L}^2}$	Order	Outer iterations	Residual norm
8.33×10^{-1}	1.8		24	6.2×10^{-9}
4.17×10^{-1}	9.2×10^{-1}	9.4×10^{-1}	31	5.4×10^{-9}
2.08×10^{-1}	3.5×10^{-1}	1.4	30	9.8×10^{-9}
1.04×10^{-1}	9.9×10^{-2}	1.8	27	7.8×10^{-9}
5.21×10^{-2}	2.6×10^{-2}	1.9	26	8.3×10^{-9}

For statistics: using 64 CPUs, computation of the meshlevel 3 ($h = 2.08 \times 10^{-1}$) takes ~ 1 minute, meshlevel 4 takes ~ 7 minutes, meshlevel 5 takes ~ 50 minutes, meshlevel 6 takes 4.8 hours, and meshlevel 7 takes ~ 21.3 hours.

2.2.2 Consistent penalty formulation

We consider the same formulation as in (16), but with the first term in the definition of \mathbf{A} changed as

$$\begin{aligned} & \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 2(E_{s, \Gamma_{h/m}^{2 \rightarrow 1}}(\mathbf{u}) - (\mathbf{u} \cdot \mathbf{n}_{\Gamma_h^2}) \mathbf{H}_{\Gamma_h^2}) : (E_{s, \Gamma_{h/m}^{2 \rightarrow 1}}(\mathbf{v}) - (\mathbf{v} \cdot \mathbf{n}_{\Gamma_h^2}) \mathbf{H}_{\Gamma_h^2}) \, ds \stackrel{(11)}{=} \\ & \int_{\Gamma_{h/m}^1}^5 2(E_{s, \Gamma}(\mathbf{u}) - (\mathbf{u} \cdot \mathbf{n}_{\Gamma}) \mathbf{H}_{\Gamma}) : (E_{s, \Gamma}(\mathbf{v}) - (\mathbf{v} \cdot \mathbf{n}_{\Gamma}) \mathbf{H}_{\Gamma}) \, ds. \end{aligned} \quad (17)$$

Shape operator $\mathbf{H}_{\Gamma_h^2}$ is computed as explained in Section 3.4.

Table 6: Convergence results. $\tau = h^{-2}$, $\rho_u = h^{-1}$, $\rho_p = h$. Matrices are assembled as in (16)–(17)

$m \in \mathbf{m}_{1/2}$ as in Figure 6						
h	$\ \mathbf{u} - \mathbf{u}_h\ _{\mathbb{H}^1}$	Order	$\ \mathbf{u} - \mathbf{u}_h\ _{\mathbb{L}^2}$	Order	$\ p - p_h\ _{\mathbb{L}^2}$	Order
8.3×10^{-1}	1.2		4.8×10^{-1}		4.2×10^{-1}	
4.2×10^{-1}	3.7×10^{-1}	1.8	6.1×10^{-2}	3.	1.1×10^{-1}	2.
2.1×10^{-1}	9.2×10^{-2}	2.	5.8×10^{-3}	3.4	2.5×10^{-2}	2.1
$1. \times 10^{-1}$	2.2×10^{-2}	2.1	5.6×10^{-4}	3.4	6.3×10^{-3}	2.
5.2×10^{-2}	5.4×10^{-3}	2.	5.2×10^{-5}	3.4	1.7×10^{-3}	1.8

h	$\ \mathbf{u}_h \cdot \mathbf{n}\ _{\mathbb{L}^2}$	Order	Outer iterations	Residual norm
8.33×10^{-1}	3.4×10^{-1}		26	7.3×10^{-9}
4.17×10^{-1}	5.3×10^{-2}	2.7	33	4.8×10^{-9}
2.08×10^{-1}	4.9×10^{-3}	3.4	31	$6. \times 10^{-9}$
1.04×10^{-1}	$5. \times 10^{-4}$	3.3	27	8.3×10^{-9}
5.21×10^{-2}	4.9×10^{-5}	3.4	26	8.6×10^{-9}

For statistics: using 80 CPUs, computation of the meshlevel 7 ($h = 1.3 \times 10^{-2}$) takes ~ 74.3 hours with $m = 64$ (computation also involves errors for normals and the shape operator).

3 Notes on DROPS implementation

3.1 Notations

We denote by $P_h^n \subset \bar{P}_h^n$ spaces of continuous and discontinuous nodal P_n interpolants defined on Ω_Γ^h , respectively. For a function f , $I_h^n(f) \in P_h^n$ is the corresponding interpolant; we will use the notation f_h^n to emphasize that $f_h^n \in P_h^n$ and f_h^n approximates f in some sense, but $I_h^n(f) \neq f_h^n$.

We set

$$\Gamma_h^n := \{\mathbf{x} \in \mathbb{R}^3 : (I_h^n(\phi))(\mathbf{x}) = 0\}, \quad (18)$$

$$\mathbf{n}_{\Gamma_h^n} = \frac{\nabla I_h^n(\phi)}{\|\nabla I_h^n(\phi)\|} \notin \bar{P}_h^m \text{ for any } m \text{ if } n > 1. \quad (19)$$

Note that Γ_h^n is a continuous piecewise P_n surface in Ω_Γ^h , and $\Gamma_h^n \neq I_h^n(\Gamma)$. The unit normal $\mathbf{n}_{\Gamma_h^n}$ is not a rational function; it is continuous in $T \in \Omega_\Gamma^h$ and discontinuous on faces. We also define

$$\Gamma_{h/m}^{2 \rightarrow 1} := \{\mathbf{x} \in \mathbb{R}^3 : (I_{h/m}^1(I_h^2(\phi)))(\mathbf{x}) = 0\}. \quad (20)$$

Note that $I_{h/2}^1(I_h^2(\phi)) = I_{h/2}^1(\phi)$ (since in order to build both $I_{h/2}^1$ and I_h^2 the same values of ϕ are used), and $I_{h/m}^1(I_h^2(\phi)) \neq I_{h/m}^1(\phi)$ for $m > 2$. Thus we have $\Gamma_{h/2}^{2 \rightarrow 1} = \Gamma_{h/2}^1$, and $\Gamma_{h/m}^{2 \rightarrow 1} \neq \Gamma_{h/m}^1$ for $m > 2$.

3.2 Approximation of integrands involving \mathbf{n}_Γ

We start with description of the continuous levelset ϕ of $\Gamma = \{\mathbf{x} \in \mathbb{R}^3 : \phi(\mathbf{x}) = 0\}$. It is stored in `levelset_fun` variable. For example, for the unit sphere we have:

```
// surfnavierstokes_funcs.h
DROPS::Point3DCL sphere_2_shift(0.);
double sphere_2 (const DROPS::Point3DCL& p, double) {
    return pow(p[0] - sphere_2_shift[0], 2.) +
        pow(p[1] - sphere_2_shift[1], 2.) +
        pow(p[2] - sphere_2_shift[2], 2.) - 1.;
}

// surfnavierstokes.cpp
instat_scalar_fun_ptr levelset_fun;
// ...
levelset_fun = &sphere_2;
```

Continuous piecewise P_2 interpolant $I_h^2(\phi)$ of ϕ is built on Ω_Γ^h via iterating over vertices and edges of Ω_Γ^h . It is stored in `lset` object:

```
// levelset.cpp
void LevelsetP2ContCL::Init( instat_scalar_fun_ptr phi0, double t) {
    const Uint lvl= Phi.GetLevel(),
    idx= Phi.RowIdx->GetIdx();
    for (auto it = MG_.GetTriangVertexBegin(lvl), end = MG_.GetTriangVertexEnd(lvl); it !=
```

```

    if (it->Unknowns.Exist(idx))
        Phi.Data[it->Unknowns(idx)] = phi0( it->GetCoord(), t);
}
for (auto it = MG_.GetTriangEdgeBegin(lvl), end = MG_.GetTriangEdgeEnd(lvl); it != end;
    ++it) {
    if (it->Unknowns.Exist(idx))
        Phi.Data[it->Unknowns(idx)] = phi0( GetBaryCenter( *it), t);
}
}

// surfnavierstokes.cpp
DROPS::LevelsetP2CL& lset(*DROPS::LevelsetP2CL::Create(mg, lsBnd, sf));
// ...
lset.Init(levelset_fun);

```

In order to assemble matrices in (1) for e.g. $\mathbf{P}_1 - P_1$ elements, one calls

```
| SetupNavierStokesIF_P1P1(mg, &A, /* ... */ lset.Phi, /* ... */);
```

(Interestingly enough, this function does not get `lset` object that represents the interpolant; it gets only `lset.Phi`, which is the object of type `VecDescCL`. `lset.Phi` is essentially just a vector of values of ϕ at interpolation points (i.e. vertices and edges' centroids of Ω_Γ^h). That is, the assembling function above has no idea what `lset.Phi` actually represents: one may interpret it as an element of P_h^2 or e.g. $P_{h/2}^1$. Who knows?..)

No interpolation is built explicitly for $\mathbf{n}_{\Gamma_h^2}$ in (19); it is implicitly represented via `qnormal` data field:

```
// ifacetransp.cpp
class LocalStokesCL {
    // ...
    GridFunctionCL<Point3DCL> qnormal;
    // ...
}
```

`qnormal` object is essentially a set of values of type `Point3DCL` which are obtained by mapping a (vector valued) function to suitable quadrature nodes. This is how it is constructed:

```
// ifacetransp.cpp
void LocalStokesCL::Get_Normals(const LocalP2CL & ls, LocalP1CL<Point3DCL> & Normals) {
    for(int i=0; i<10 ; ++i)
        Normals+=ls[i]*P2Grad[i];
}
// ...
void LocalStokesCL::calcIntegrands(const SMatrixCL<3,3> & T, const LocalP2CL & ls, const
    TetraCL & tet) {
    // ...
    LocalP1CL<Point3DCL> Normals;
    Get_Normals(ls, Normals);
    resize_and_evaluate_on_Vertexes(Normals, q2Ddomain, qnormal);
    for(Uint i=0; i<qnormal.size(); ++i)
        qnormal[i] = qnormal[i]/qnormal[i].norm();
    // ...
}
```

First $\nabla I_h^2(\phi|_T)$ is built (locally for a tetrahedron $T \in \Omega_\Gamma^h$ represented by `tet`) and saved to `Normals` object. `ls[i]` gives the value of $\phi|_T$ at i th node (vertices and edges' centroids—there are 10 of them for tetrahedra), and `P2Grad[i]` represents the gradient of quadratic basis function which itself is linear. (Actually, it is sufficient to have $4 < 10$ linear functions to represent $\nabla I_h^2(\phi|_T)$, but this is how it is implemented here.) Finally, `qnormal` object is built via evaluating `Normals` at quadrature nodes and normalization.

Objects `qnormal` for surface integrals and `q3Dnormal` for volume integrals are used in approximation of $\mathbf{P} =$

$\mathbf{I} - \mathbf{n}\mathbf{n}^T$, normal derivatives, and taking-normal-components in (1). `q3Dnormal` is constructed as `qnormal` but for quadrature points of tetrahedrons, not triangles.

For one, $\mathbf{P}\nabla f_h^2$, $f_h^2 := P_2$ basis function on Ω_Γ^h , is approximated via `qsurfP2grad` object:

```
// ifacettransp.cpp
void LocalStokesCL::calcIntegrands(/* ... */) {
    // ...
    for(int j=0; j<10 ;++j) {
        resize_and_evaluate_on_vertexes(P2Grad[j], q2Ddomain, qsurfP2grad[j]);
        qsurfP2grad[j] -= dot(qsurfP2grad[j], qnormal)*qnormal;
    }
    // ...
}
```

The term $\int_{\Omega_h^\Gamma} \frac{\partial p}{\partial \mathbf{n}} \frac{\partial q}{\partial \mathbf{n}} d\mathbf{x}$ in (1) is computed as

```
// ifacettransp.cpp
void LocalStokesCL::setupA_P1_stab(double A_P1_stab[4][4], double absdet) {
    for (int i=0; i<4; ++i)
        for (int j=0; j<4; ++j)
            A_P1_stab[i][j] = quad(dot(q3Dnormal, q3DP1Grad[i])*dot(q3Dnormal, q3DP1Grad[j]),
                                    absdet, q3Ddomain, AllTetraC);
}
```

3.3 Quadrature rules for \int_Γ and $\int_{\Omega_\Gamma^h}$

All the 3D integrals in (1) are computed via iteration over $T \in \Omega_\Gamma^h$ without any virtual refinements. `q3Ddomain` object represents the set of quadrature nodes and weights:

```
// ifacettransp.cpp
void LocalStokesCL::calc3DIntegrands(/* ... */) {
    make_SimpleQuadDomain<Quad5DataCL> (q3Ddomain, AllTetraC);
    // ...
}
```

It is used e.g. in `setupA_P1_stab` above. 15 nodes and weights are used, and the quadrature is exact for functions in \bar{P}_h^5 .

All the surface integrals are also computed via iteration over $T \in \Omega_\Gamma^h$, but using $\Gamma_{h/2}^1$. One extra “virtual” refinement is achieved via setting

```
// ifacettransp.cpp
LocalStokesCL(bool fullGradient)
: lat(PrincipalLatticeCL::instance(2))
, /* ... */ { /* ... */ }
```

`PrincipalLatticeCL::instance(2)` means that each edge of the tetrahedron $T \in \Omega_\Gamma^h$ is split into 2 edges, and T is split into 8 smaller tetrahedrons. Changing 2 to 4 will give us $\Gamma_{h/4}^{2 \rightarrow 1}$ from (20) and so forth. `q2Ddomain` object represents the set of quadrature nodes and weights:

```
// ifacettransp.cpp
void LocalStokesCL::calcIntegrands(/* ... */) {
    // ...
    evaluate_on_vertexes( ls, lat, Addr( ls_loc));
    spatch.make_patch<MergeCutPolicyCL>( lat, ls_loc);
    make_CompositeQuad5Domain2D ( q2Ddomain, spatch, tet);
    // ...
}
```

Each linear subsurface in $T \in \Omega_\Gamma^h$ has 7 quadrature nodes and weights, and the quadrature rule is again exact for functions in \bar{P}_h^5 .

`spatch` represents a set of triangles that form $\Gamma_{h/2}^1$ inside T . That is, in order to approximate zeros of ϕ , $I_{h/2}^1(I_h^2(\phi)) = I_{h/2}^1(\phi)$ is used:

```
// subtriangulation.h
// ...
const double edge_bary1_cut= ls0/(ls0 - ls1); // the root of the level set function on
// the edge
// ...
```

Here $l(x) := ls0(1-x) + ls1x$ is a linear function defined on the master edge $[0, 1]$. Indeed, its root is $x = ls0/(ls0 - ls1)$.

3.4 Approximation of the shape operator \mathbf{H}

We have $\mathbf{H}_\Gamma := \nabla_\Gamma \mathbf{n}_\Gamma := \mathbf{P}_\Gamma \nabla \mathbf{n}_\Gamma^e \mathbf{P}_\Gamma$, $\mathbf{H}_\Gamma : \mathcal{O}(\Gamma) \rightarrow \mathbb{R}^3$. Note that $\mathbf{n}_\Gamma = \nabla\phi/\|\nabla\phi\|$ is defined in $\mathcal{O}(\Gamma)$, so $\nabla \mathbf{n}_\Gamma$ makes sense and

$$\nabla \mathbf{n}_\Gamma = \left(\mathbf{I} - \frac{\nabla\phi \nabla\phi^T}{\|\nabla\phi\|^2} \right) \frac{\nabla^2\phi}{\|\nabla\phi\|} = \mathbf{P}_\Gamma \frac{\nabla^2\phi}{\|\nabla\phi\|}.$$

If $\mathbf{n}_\Gamma = \mathbf{n}_\Gamma^e$, one gets

$$\mathbf{H}_\Gamma = \mathbf{P}_\Gamma \frac{\nabla^2\phi}{\|\nabla\phi\|} \mathbf{P}_\Gamma. \quad (21)$$

Thus we define $\mathbf{H}_{\Gamma_h^2}$ to be as in (21) but with ϕ replaced with $I_h^2(\phi)$:

$$\mathbf{H}_{\Gamma_h^2} := \mathbf{P}_{\Gamma_h^2} \frac{\nabla^2 I_h^2(\phi)}{\|\nabla I_h^2(\phi)\|} \mathbf{P}_{\Gamma_h^2}. \quad (22)$$

Indeed, computation of $\mathbf{H}_{\Gamma_h^2}$ requires Hessians of shape functions, but luckily this routine was implemented in DROPS.

Depending on the choice of ϕ , we may or may not have $\mathbf{n}_\Gamma = \mathbf{n}_\Gamma^e$. Note that the choice $\phi = d$ is sufficient for this, but not necessary. Consider this choices of ϕ for Γ_{sph} :

1. $\phi_1(\mathbf{x}) = \|\mathbf{x}\| - 1 = d(\mathbf{x})$, $\nabla\phi_1/\|\nabla\phi_1\| = \mathbf{n}_\Gamma^e$,
2. $\phi_2(\mathbf{x}) = \|\mathbf{x}\|^2 - 1 \in P^2$, $\nabla\phi_2/\|\nabla\phi_2\| = \nabla\phi_1/\|\nabla\phi_1\| = \mathbf{n}_\Gamma^e$,
3. $\phi_3(\mathbf{x}) = e^{\phi_2(\mathbf{x})} x^2 + y^2 + z^2 - 1$, $\nabla\phi_3/\|\nabla\phi_3\| \neq \mathbf{n}_\Gamma^e$, i.e. $\nabla\phi_3/\|\nabla\phi_3\| = \mathbf{n}_\Gamma$ only on Γ_{sph} .

As for the case 2: note that if ϕ is piecewise quadratic in Ω_h^Γ and defines a normal that is equal to its extension, then $\mathbf{H}_{\Gamma_h^2} = \mathbf{H}_\Gamma$, i.e. the approximation is **exact**.

For the approach (24), there is also an option to approximate \mathbf{H} as $\mathbf{P}_{\Gamma_{h/m}^{2 \rightarrow 1}} \frac{\nabla^2 I_h^2(\phi)}{\|\nabla I_h^2(\phi)\|} \mathbf{P}_{\Gamma_{h/m}^{2 \rightarrow 1}}$ since we build $\mathbf{P}_{\Gamma_{h/m}^{2 \rightarrow 1}}$ anyway. I chose to use (22) for both (23) and (24).

3.5 Summary on the matrix assembly

The matrices in (1) are assembled as

$$\begin{aligned}
\langle \mathbf{A} \vec{\mathbf{u}}, \vec{\mathbf{v}} \rangle &= \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 (E_{s, \Gamma_h^2}(\mathbf{u}) : E_{s, \Gamma_h^2}(\mathbf{v}) + \mathbf{u} \cdot \mathbf{v} + \tau (\mathbf{u} \cdot \mathbf{n}_{\Gamma_h^2}) (\mathbf{v} \cdot \mathbf{n}_{\Gamma_h^2})) \, ds \\
&\quad + \rho_u \int_{\Omega_h^\Gamma} \frac{\partial \mathbf{u}}{\partial \mathbf{n}_{\Gamma_h^2}} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{n}_{\Gamma_h^2}} \, d\mathbf{x}, \quad \mathbf{A} \in \mathbb{R}^{n_A \times n_A}, \\
\langle \mathbf{B} \vec{\mathbf{u}}, \vec{\mathbf{q}} \rangle &= - \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 q \operatorname{div}_{\Gamma_h^2} \mathbf{u} \, ds, \quad \mathbf{B} \in \mathbb{R}^{n_S \times n_A}, \\
\langle \mathbf{M}_0 \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 p q \, ds, \quad \mathbf{M}_0 \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_n \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \rho_p \int_{\Omega_h^\Gamma} \frac{\partial p}{\partial \mathbf{n}_{\Gamma_h^2}} \frac{\partial q}{\partial \mathbf{n}_{\Gamma_h^2}} \, d\mathbf{x}, \quad \mathbf{C}_n \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_{\text{full}} \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \rho_p \int_{\Omega_h^\Gamma} \nabla p \cdot \nabla q \, d\mathbf{x}, \quad \mathbf{C}_{\text{full}} \in \mathbb{R}^{n_S \times n_S},
\end{aligned} \tag{23}$$

Comments:

- $\int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 \cdot \, ds$ denotes a composite quadrature rule that is exact for $\bar{P}_h^5(\Gamma_{h/m}^{2 \rightarrow 1})$, i.e. this quadrature is exact for piecewise polynomials up to degree 5 on each triangular patch $\gamma \in \Gamma_{h/m}^{2 \rightarrow 1}$,
- $\int_{\Omega_h^\Gamma}^5 \cdot \, d\mathbf{x}$ denotes a composite quadrature rule that is exact for $\bar{P}_h^5(\Omega_h^\Gamma)$, i.e. this quadrature is exact for piecewise polynomials up to degree 5 on each tetrahedron $T \in \Omega_h^\Gamma$,
- E_{s, Γ_h^2} and $\operatorname{div}_{\Gamma_h^2}$ are defined as their continuous analogues with \mathbf{n}_Γ in \mathbf{P} replaced with $\mathbf{n}_{\Gamma_h^2}$,
- It is always the case that integrands use $\mathbf{n}_{\Gamma_h^2} \neq \mathbf{n}_{\Gamma_{h/m}^{2 \rightarrow 1}}$, and the actual domain of integration is $\Gamma_{h/m}^{2 \rightarrow 1} \neq \Gamma_h^2$,
- $\mathbf{n}_{\Gamma_h^2}$ is defined in (19) and it is not a polynomial even locally, thus quadrature rules are never exact (although for $\mathbf{P}_2 - P_1$ shape functions alone these quadratures are exact).

3.6 Using exact normals in integrands of $\int_{\Gamma_{h/m}^{2 \rightarrow 1}}$ (updated summary)

It was quite easy to update (23) such that the exact normals w.r.t. piecewise linear surface domain of integration are used. `spatch` (section 3.3) has a member function that gives **physical** normals to its triangles straightforward. Thus implementation of updated quadratures boiled down to constructing `GridFunction` object (described in section 3.2) out of these physical normals. Details of the implementation can be found in commit [dacc440](#).

Comments:

- It is also easy to extend this approach s.t. $\Gamma_{h/m}^1 \neq \Gamma_{h/m}^{2 \rightarrow 1}$ is used (please see section 3.1 and then figures 7 and 8). There is no difference between $\Gamma_{h/2}^{2 \rightarrow 1}$ and $\Gamma_{h/2}^1$. For $m > 2$, there is no difference between $\Gamma_{h/m}^{2 \rightarrow 1}$ and $\Gamma_{h/m}^1$ if $\phi \in P^2$. Right now $\Gamma_{h/m}^{2 \rightarrow 1}$ is implemented.
- It is **not** that easy to use the exact normal w.r.t. piecewise linear surface domain of integration in volume integrals. Our guess is that it shall not be a problem (we can leave $\mathbf{n}_{\Gamma_h^2}$ there),
- It is **not** easy to build $I_h^n(\phi)$ for $n > 2$. It is **not** implemented in DROPS as for now.

As for now, the matrices in (1) can also be assembled as

$$\begin{aligned}
\langle \mathbf{A} \vec{\mathbf{u}}, \vec{\mathbf{v}} \rangle &= \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 \left(E_{s, \Gamma_{h/m}^{2 \rightarrow 1}}(\mathbf{u}) : E_{s, \Gamma_{h/m}^{2 \rightarrow 1}}(\mathbf{v}) + \mathbf{u} \cdot \mathbf{v} + \tau (\mathbf{u} \cdot \mathbf{n}_{\Gamma_h^2}) (\mathbf{v} \cdot \mathbf{n}_{\Gamma_h^2}) \right) ds \\
&\quad + \rho_u \int_{\Omega_h^\Gamma} \frac{\partial \mathbf{u}}{\partial \mathbf{n}_{\Gamma_h^2}} \cdot \frac{\partial \mathbf{v}}{\partial \mathbf{n}_{\Gamma_h^2}} d\mathbf{x}, \quad \mathbf{A} \in \mathbb{R}^{n_A \times n_A}, \\
\langle \mathbf{B} \vec{\mathbf{u}}, \vec{\mathbf{q}} \rangle &= - \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 q \operatorname{div}_{\Gamma_{h/m}^{2 \rightarrow 1}} \mathbf{u} ds, \quad \mathbf{B} \in \mathbb{R}^{n_S \times n_A}, \\
\langle \mathbf{M}_0 \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \int_{\Gamma_{h/m}^{2 \rightarrow 1}}^5 p q ds, \quad \mathbf{M}_0 \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_n \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \rho_p \int_{\Omega_h^\Gamma} \frac{\partial p}{\partial \mathbf{n}_{\Gamma_h^2}} \frac{\partial q}{\partial \mathbf{n}_{\Gamma_h^2}} d\mathbf{x}, \quad \mathbf{C}_n \in \mathbb{R}^{n_S \times n_S}, \\
\langle \mathbf{C}_{\text{full}} \vec{\mathbf{p}}, \vec{\mathbf{q}} \rangle &= \rho_p \int_{\Omega_h^\Gamma} \nabla p \cdot \nabla q d\mathbf{x}, \quad \mathbf{C}_{\text{full}} \in \mathbb{R}^{n_S \times n_S},
\end{aligned} \tag{24}$$

Notations are as in (23). Note that the “ τ -term” uses $\mathbf{n}_{\Gamma_h^2}$. In order to switch between (23) and (24), one modifies JSON input file:

```
// No_Bnd_Condition.json
"Levelset": {
// ...
"NumberOfVirtualSubEdges" : 2,
"UseExactNormals" : "yes",
// ...
}
```

Here 2 corresponds to $m = 2$, and “yes” corresponds to (24).

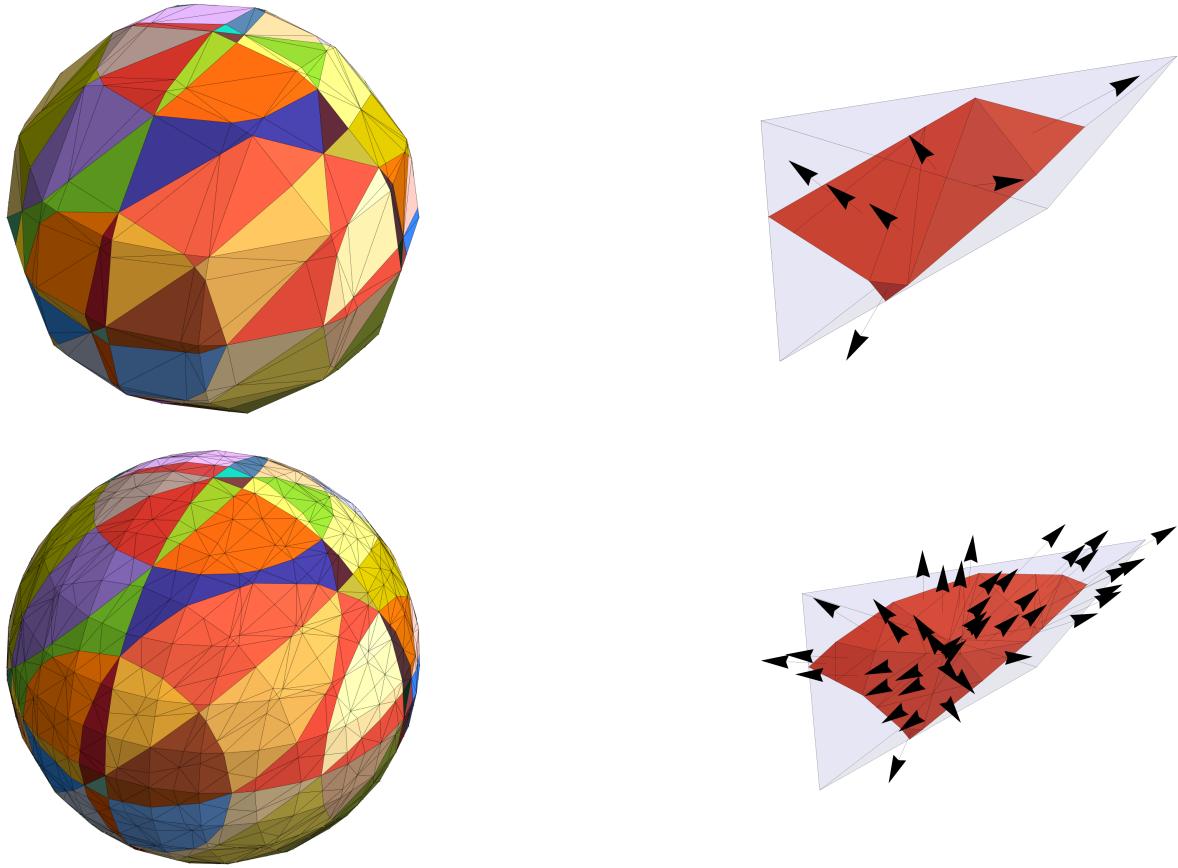


Figure 7: $\Gamma = \Gamma_{\text{sph}}$, $\phi(\mathbf{x}) = \|\mathbf{x}\|^2 - 1$, $h = 8.33 \times 10^{-1}$. Top-left: $\Gamma_{h/2}^{2 \rightarrow 1} = \Gamma_{h/2}^1$ (different color corresponds to a different spatch $\gamma \in \Gamma_{h/2}^{2 \rightarrow 1}$ as described in section 3.3). Top-right: a patch $\gamma \in \Gamma_{h/2}^{2 \rightarrow 1}$ and its normals. Bottom-left and bottom-right: same for $\Gamma_{h/4}^{2 \rightarrow 1} = \Gamma_{h/4}^1$. **Note that since $\phi \in P^2$, we have that $\Gamma_{h/m}^{2 \rightarrow 1} = \Gamma_{h/m}^1 \rightarrow \Gamma$ as $m \rightarrow \infty$ for fixed h**

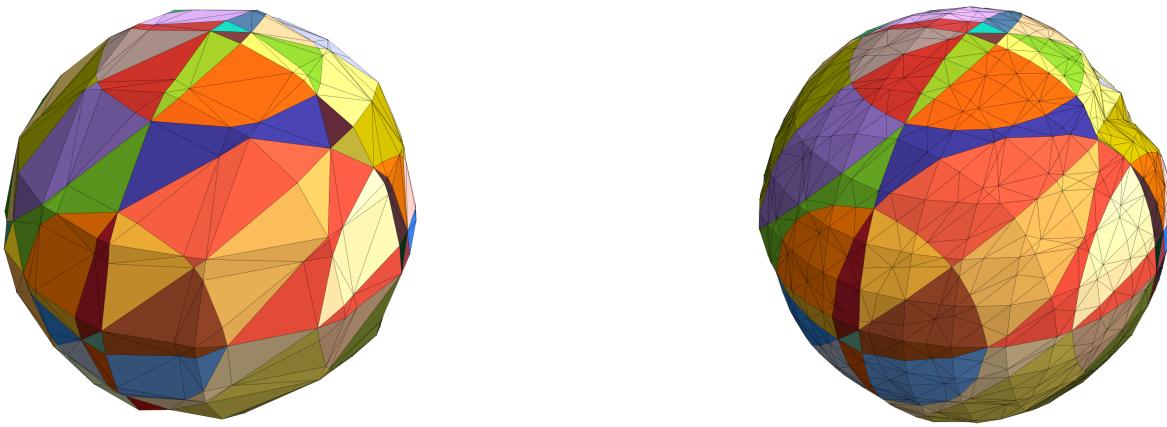


Figure 8: $\Gamma = \Gamma_{\text{sph}}$, $\phi(\mathbf{x}) = \|\mathbf{x}\|^{1/2} - 1$, $h = 8.33 \times 10^{-1}$. Left: $\Gamma_{h/2}^{2 \rightarrow 1} = \Gamma_{h/2}^1$ (different color corresponds to a different spatch $\gamma \in \Gamma_{h/2}^{2 \rightarrow 1}$ as described in section 3.3). Right: same for $\Gamma_{h/4}^{2 \rightarrow 1} \neq \Gamma_{h/4}^1$. **Note that since $\phi \notin \bar{P}_h^2$, we have that $\Gamma_{h/m}^{2 \rightarrow 1} \neq \Gamma_{h/m}^1$ for $m > 2$, and $\Gamma_{h/m}^{2 \rightarrow 1} \rightarrow \Gamma_h^2 \neq \Gamma$ as $m \rightarrow \infty$ for fixed h**

3.7 Quadrature rules for the error computation

When we first tried to test convergence for (24) in section 2, we noticed that the $\mathbb{H}^1(\Gamma)$ -error of the velocity decays much slower than expected, whereas its $\mathbb{L}^2(\Gamma)$ -error behaves as expected. Note that $\mathbb{H}^1(\Gamma)$ -error (for e.g. $\mathbf{P}_2 - P_1$ FE) can be computed as $\langle \mathbf{w}, \mathbf{A}_s \mathbf{w} \rangle^{1/2}$, $\mathbf{w} :=$ vector of d.o.f. corresponding to \mathbf{P}_h^2 interpolant $I_h^2(\mathbf{u}^e) - \mathbf{u}_h$, $\mathbf{A}_s :=$ matrix corresponding to the first term of \mathbf{A} in (24). Thus the errors are approximated as

$$\begin{aligned}\|\mathbf{u} - \mathbf{u}_h\|_{\mathbb{H}^1(\Gamma)} &= \|I_h^k(\mathbf{u}^e) - \mathbf{u}_h\|_{\mathbb{H}^1(\Gamma_{h/m}^{2 \rightarrow 1})} + O(h^k), \\ \|\mathbf{u} - \mathbf{u}_h\|_{\mathbb{L}^2(\Gamma)} &= \|I_h^k(\mathbf{u}^e) - \mathbf{u}_h\|_{\mathbb{L}^2(\Gamma_{h/m}^{2 \rightarrow 1})} + O(h^{k+1}), \\ \|p - p_h\|_{\mathbb{L}^2(\Gamma)} &= \|I_h^1(p^e) - p_h\|_{\mathbb{L}^2(\Gamma_{h/m}^{2 \rightarrow 1})} + O(h^2)\end{aligned}\quad (25)$$

for $m > 1$. Here $k = 1$ for $\mathbf{P}_1 - P_1$ FEM and $k = 2$ for $\mathbf{P}_2 - P_1$. For consistent penalty approach matrix \mathbf{A}_s is computed as in (??).

Interestingly enough, DROPS implementation did not use the assembled matrices to compute errors (and normals that are *different* from the ones in \mathbf{A}_s were used). We corrected it in commit [68443b0](#):

```
// surfnavierstokes.cpp
// ...
VectorCL vSolMinusV = vSol.Data - v.Data, pSolMinusP = pSol.Data - p.Data;
auto velL2          = sqrt(dot(v.Data, M.Data * v.Data));
auto velNormalL2    = sqrt(dot(v.Data, S.Data * v.Data));
auto velH1err        = sqrt(dot(vSolMinusV, A.Data * vSolMinusV));
auto velL2err        = sqrt(dot(vSolMinusV, M.Data * vSolMinusV));
auto preL2           = sqrt(dot(p.Data, Schur.Data * p.Data));
auto preL2err        = sqrt(dot(pSolMinusP, Schur.Data * pSolMinusP));
// ...
```

References

- [1] T. Jankuhn and A. Reusken. Trace Finite Element Methods for Surface Vector-Laplace Equations. *arXiv e-prints*, April 2019.
- [2] M. Olshanskii, A. Quaini, A. Reusken, and V. Yushutin. A finite element method for the surface stokes problem. *SIAM Journal on Scientific Computing*, 40(4):A2492–A2518, 2018.