Bench report : 01-Feb-2013 à 19h19m27s

Friday 1st February, 2013

Contents

1 Introduction

Soient $\Omega \subset \mathbb{R}^2$ et $\boldsymbol{u}$, $\boldsymbol{v}$ deux champs de vecteurs (Vector Fields) On suppose que $\boldsymbol{u} = (u_1, u_2) \in \left(H^1(\Omega)\right)^2 = H^1(\Omega) \times H^1(\Omega)$ et $\boldsymbol{v} = (v_1, v_2) \in \left(H^1(\Omega)\right)^2 = H^1(\Omega) \times H^1(\Omega)$.

1.1 Matrice de masse en champs de vecteurs

On définie l'application bilinéaire $\mathcal{L}$ par $\mathcal{L}(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$.

La matrice de masse en champs de vecteurs est la matrice obtenue lors de la discrétisation par la méthode des éléments finis P_1-Lagrange de l'intégrale

$$\int_\Omega \mathcal{L}(\boldsymbol{u}, \boldsymbol{v})(\mathrm{q}) d\,\mathrm{q}\,.$$

1.2 Matrice de rigidité élastique en déformation plane

On note

$$\underline{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{pmatrix} \quad \text{et} \quad \underline{\boldsymbol{\epsilon}} = \begin{pmatrix} \epsilon_{11} \\ \epsilon_{22} \\ 2\epsilon_{12} \end{pmatrix}$$

respectivement le tenseur des contraintes vectorisé et le tenseur des déformations vectorisé en déformation plane.

On note $\mathcal{D}$ l'opérateur différentiel qui relie le déplacement aux déformations :

$$\underline{\boldsymbol{\epsilon}}(\boldsymbol{u}) = \mathcal{D}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla(\boldsymbol{u}) + \nabla^t(\boldsymbol{u})\right) \tag{1}$$

Ceci donne en notation vectorielle, après réduction au plan,

$$\mathcal{D} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix}. \tag{2}$$

De plus, la loi de comportement s'écrit

$$\underline{\boldsymbol{\sigma}} = \mathbb{C}\underline{\boldsymbol{\epsilon}} = \begin{pmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{pmatrix} \underline{\boldsymbol{\epsilon}} \tag{3}$$

On définie l'application bilinéaire $\mathcal{L}$ par $\mathcal{L}(\boldsymbol{u}, \boldsymbol{v}) = \underline{\boldsymbol{\epsilon}}^t(\boldsymbol{u})\underline{\boldsymbol{\sigma}}(\boldsymbol{v}) = \underline{\boldsymbol{\epsilon}}^t(\boldsymbol{u})\mathbb{C}\underline{\boldsymbol{\epsilon}}(\boldsymbol{u})$.

La matrice de rigidité élastique est la matrice obtenue lors de la discrétisation par la méthode des éléments finis P_1-Lagrange de l'intégrale

$$\int_\Omega \mathcal{L}(\boldsymbol{u}, \boldsymbol{v})(\mathrm{q}) d\,\mathrm{q}\,.$$

2 Mesh : carre4-1

2.1 Function : MassVFAssembling

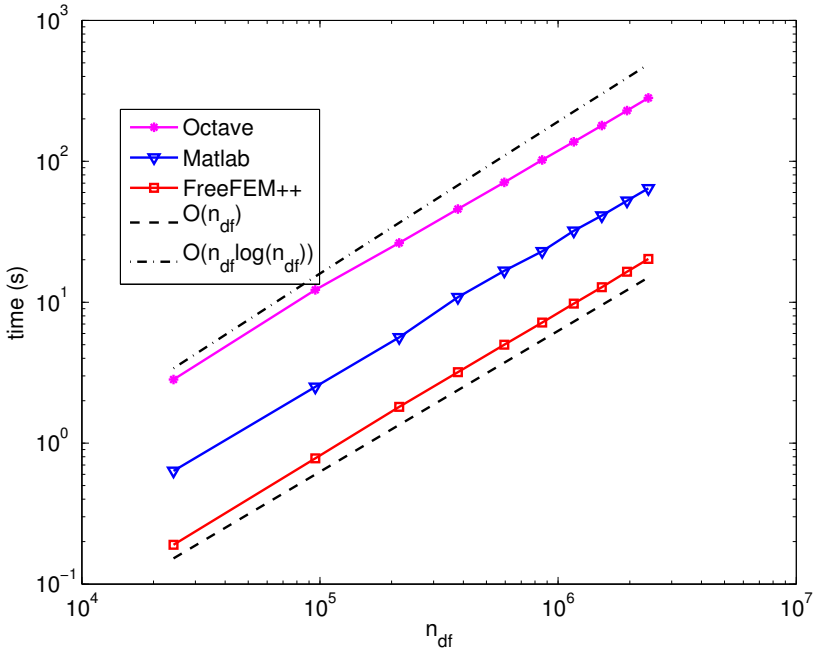

Figure 1: Comparison of Matlab/Octave function `MassVFAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
12139	24278	2.830 (s) x 1.00	0.635 (s) x 4.46	0.190 (s) x 14.90
47834	95668	12.236 (s) x 1.00	2.509 (s) x 4.88	0.780 (s) x 15.69
107625	215250	26.345 (s) x 1.00	5.635 (s) x 4.68	1.810 (s) x 14.56
189794	379588	45.858 (s) x 1.00	10.851 (s) x 4.23	3.190 (s) x 14.38
297290	594580	70.847 (s) x 1.00	16.718 (s) x 4.24	4.990 (s) x 14.20
428549	857098	102.184 (s) x 1.00	22.988 (s) x 4.45	7.170 (s) x 14.25
581973	1163946	137.583 (s) x 1.00	32.148 (s) x 4.28	9.770 (s) x 14.08
762704	1525408	179.508 (s) x 1.00	41.175 (s) x 4.36	12.790 (s) x 14.03
973281	1946562	228.726 (s) x 1.00	52.489 (s) x 4.36	16.450 (s) x 13.90
1197591	2395182	281.311 (s) x 1.00	64.238 (s) x 4.38	20.280 (s) x 13.87

Table 1: Computational cost of the `MassVF` matrix assembly versus n_q, with the OptV2 Matlab/Octave codes ($1^{st}/2^{nd}$ columns) and with FreeFEM++ (3^{rd} column) on unit square meshes : time in seconds (top value) and speedup (bottom value). The speedup reference is OptV2 Octave version.

2.2 Function : StiffElasAssembling

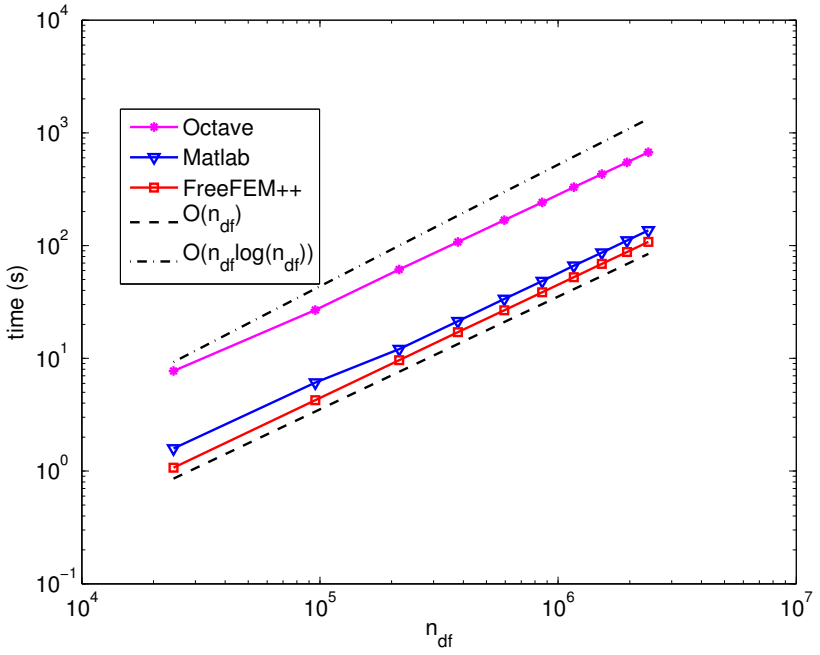

Figure 2: Comparison of Matlab/Octave function `StiffElasAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
12139	24278	7.723 (s) x 1.00	1.585 (s) x 4.87	1.070 (s) x 7.22
47834	95668	26.829 (s) x 1.00	6.090 (s) x 4.41	4.250 (s) x 6.31
107625	215250	61.425 (s) x 1.00	12.078 (s) x 5.09	9.640 (s) x 6.37
189794	379588	107.590 (s) x 1.00	21.328 (s) x 5.04	17.040 (s) x 6.31
297290	594580	168.267 (s) x 1.00	33.599 (s) x 5.01	26.700 (s) x 6.30
428549	857098	241.932 (s) x 1.00	48.493 (s) x 4.99	38.520 (s) x 6.28
581973	1163946	329.547 (s) x 1.00	66.320 (s) x 4.97	52.350 (s) x 6.30
762704	1525408	431.005 (s) x 1.00	86.542 (s) x 4.98	68.740 (s) x 6.27
973281	1946562	546.691 (s) x 1.00	110.882 (s) x 4.93	87.740 (s) x 6.23
1197591	2395182	673.828 (s) x 1.00	136.381 (s) x 4.94	107.840 (s) x 6.25

Table 2: Computational cost of the `StiffElas` matrix assembly versus n_q, with the OptV2 Matlab/Octave codes ($1^{st}/2^{nd}$ columns) and with FreeFEM++ (3^{rd} column) on unit square meshes : time in seconds (top value) and speedup (bottom value). The speedup reference is OptV2 Octave version.

3 Mesh : disque4-1

3.1 Function : MassVFAssembling

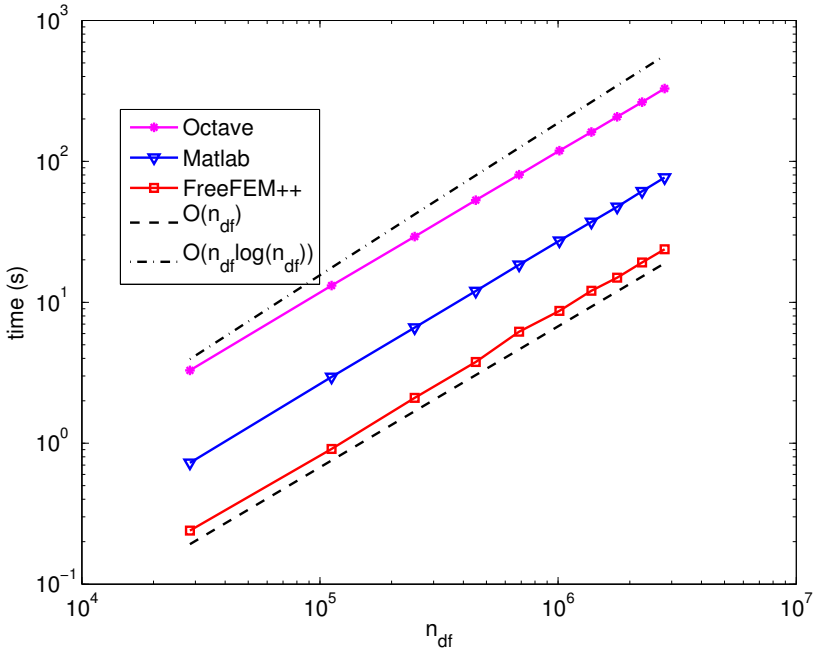

Figure 3: Comparison of Matlab/Octave function `MassVFAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
14222	28444	3.290 (s) x 1.00	0.726 (s) x 4.53	0.240 (s) x 13.71
55919	111838	13.152 (s) x 1.00	2.953 (s) x 4.45	0.910 (s) x 14.45
125010	250020	29.260 (s) x 1.00	6.619 (s) x 4.42	2.100 (s) x 13.93
225547	451094	53.063 (s) x 1.00	12.021 (s) x 4.41	3.770 (s) x 14.08
343082	686164	80.457 (s) x 1.00	18.450 (s) x 4.36	6.180 (s) x 13.02
506706	1013412	118.925 (s) x 1.00	27.367 (s) x 4.35	8.680 (s) x 13.70
689716	1379432	161.689 (s) x 1.00	37.169 (s) x 4.35	12.080 (s) x 13.38
885521	1771042	206.866 (s) x 1.00	47.592 (s) x 4.35	14.950 (s) x 13.84
1127090	2254180	263.296 (s) x 1.00	61.400 (s) x 4.29	19.180 (s) x 13.73
1401129	2802258	328.621 (s) x 1.00	76.753 (s) x 4.28	23.790 (s) x 13.81

Table 3: Computational cost of the `MassVF` matrix assembly versus n_q, with the OptV2 Matlab/Octave codes ($1^{st}/2^{nd}$ columns) and with FreeFEM++ (3^{rd} column) on unit disk meshes : time in seconds (top value) and speedup (bottom value). The speedup reference is OptV2 Octave version.

3.2 Function : StiffElasAssembling

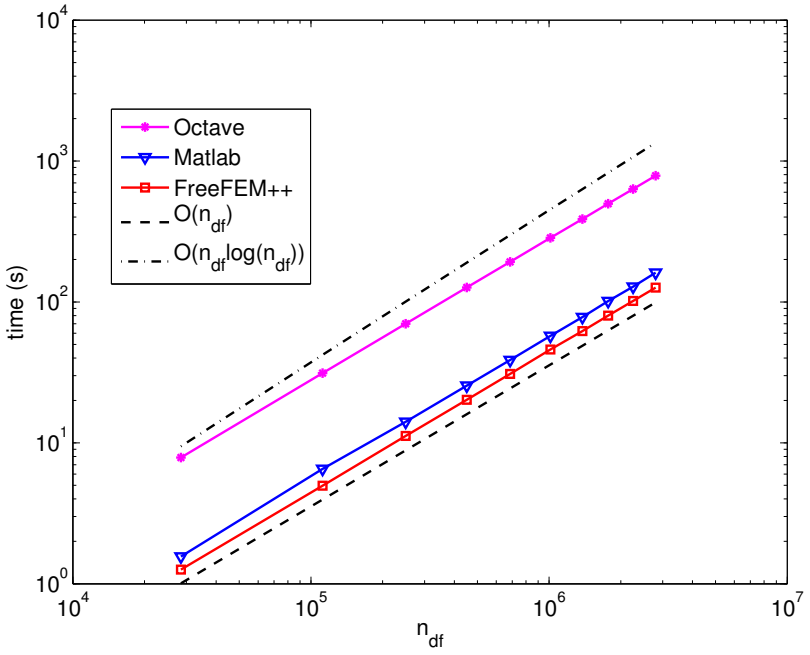

Figure 4: Comparison of Matlab/Octave function `StiffElasAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
14222	28444	7.873 (s) x 1.00	1.561 (s) x 5.04	1.260 (s) x 6.25
55919	111838	31.230 (s) x 1.00	6.543 (s) x 4.77	4.970 (s) x 6.28
125010	250020	70.025 (s) x 1.00	14.103 (s) x 4.97	11.190 (s) x 6.26
225547	451094	126.593 (s) x 1.00	25.483 (s) x 4.97	20.230 (s) x 6.26
343082	686164	192.608 (s) x 1.00	38.733 (s) x 4.97	30.840 (s) x 6.25
506706	1013412	284.692 (s) x 1.00	57.464 (s) x 4.95	45.930 (s) x 6.20
689716	1379432	387.570 (s) x 1.00	78.147 (s) x 4.96	62.170 (s) x 6.23
885521	1771042	497.374 (s) x 1.00	101.388 (s) x 4.91	79.910 (s) x 6.22
1127090	2254180	633.324 (s) x 1.00	128.207 (s) x 4.94	101.730 (s) x 6.23
1401129	2802258	786.855 (s) x 1.00	161.331 (s) x 4.88	126.470 (s) x 6.22

Table 4: Computational cost of the `StiffElas` matrix assembly versus n_q, with the OptV2 Matlab/Octave codes ($1^{st}/2^{nd}$ columns) and with FreeFEM++ (3^{rd} column) on unit disk meshes : time in seconds (top value) and speedup (bottom value). The speedup reference is OptV2 Octave version.