Bench report : 25-Feb-2013 à 11h13m11s

Monday 25th 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

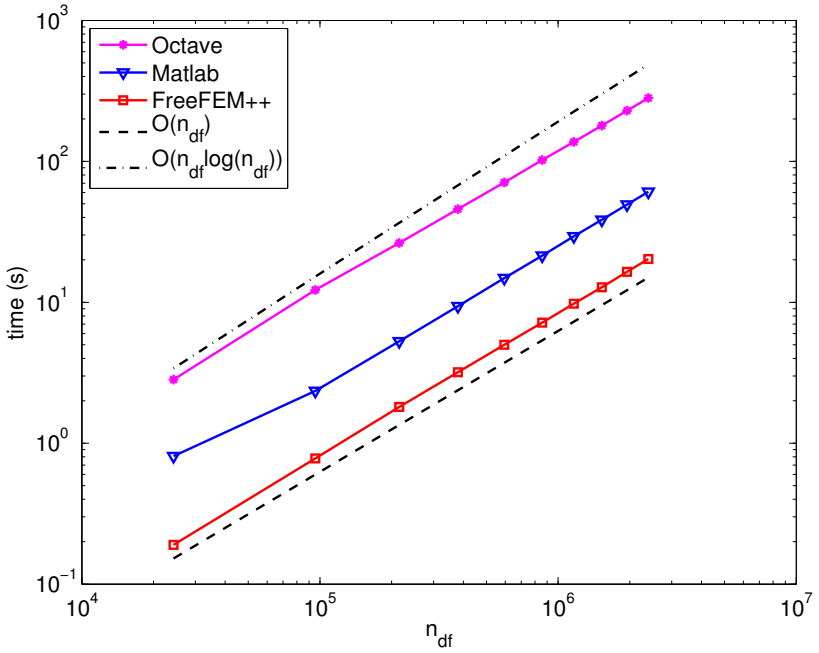

Figure 1: Comparison of Matlab/Octave function `MassVFAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
12139	24278	2.830 (s) x 1.00	0.809 (s) x 3.50	0.190 (s) x 14.90
47834	95668	12.236 (s) x 1.00	2.352 (s) x 5.20	0.780 (s) x 15.69
107625	215250	26.345 (s) x 1.00	5.282 (s) x 4.99	1.810 (s) x 14.56
189794	379588	45.858 (s) x 1.00	9.368 (s) x 4.90	3.190 (s) x 14.38
297290	594580	70.847 (s) x 1.00	14.837 (s) x 4.78	4.990 (s) x 14.20
428549	857098	102.184 (s) x 1.00	21.415 (s) x 4.77	7.170 (s) x 14.25
581973	1163946	137.583 (s) x 1.00	29.408 (s) x 4.68	9.770 (s) x 14.08
762704	1525408	179.508 (s) x 1.00	38.466 (s) x 4.67	12.790 (s) x 14.03
973281	1946562	228.726 (s) x 1.00	49.391 (s) x 4.63	16.450 (s) x 13.90
1197591	2395182	281.311 (s) x 1.00	60.811 (s) x 4.63	20.280 (s) x 13.87

Table 1: Computational cost of the `MassVF` matrix assembly versus n_q/n_{df}, with the `OptV1` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV1` Octave version.

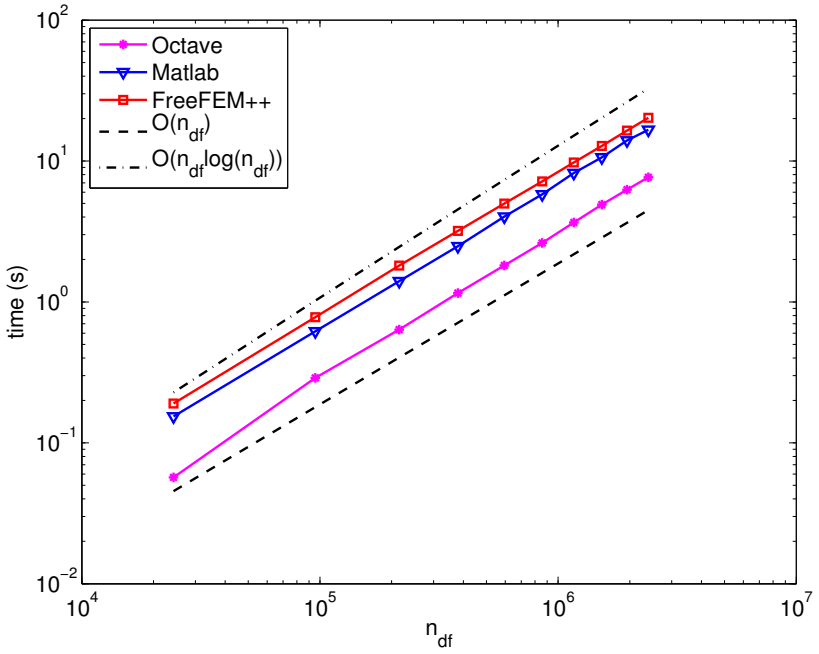

Figure 2: Comparison of Matlab/Octave function `MassVFAssemblingOptV2` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
12139	24278	0.057 (s) x 1.00	0.154 (s) x 0.37	0.190 (s) x 0.30
47834	95668	0.289 (s) x 1.00	0.619 (s) x 0.47	0.780 (s) x 0.37
107625	215250	0.637 (s) x 1.00	1.403 (s) x 0.45	1.810 (s) x 0.35
189794	379588	1.155 (s) x 1.00	2.486 (s) x 0.46	3.190 (s) x 0.36
297290	594580	1.815 (s) x 1.00	4.028 (s) x 0.45	4.990 (s) x 0.36
428549	857098	2.624 (s) x 1.00	5.796 (s) x 0.45	7.170 (s) x 0.37
581973	1163946	3.658 (s) x 1.00	8.215 (s) x 0.45	9.770 (s) x 0.37
762704	1525408	4.901 (s) x 1.00	10.612 (s) x 0.46	12.790 (s) x 0.38
973281	1946562	6.255 (s) x 1.00	13.984 (s) x 0.45	16.450 (s) x 0.38
1197591	2395182	7.667 (s) x 1.00	16.668 (s) x 0.46	20.280 (s) x 0.38

Table 2: Computational cost of the `MassVF` matrix assembly versus n_q/n_{df}, with the `OptV2` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV2` Octave version.

2.2 Function : StiffElasAssembling

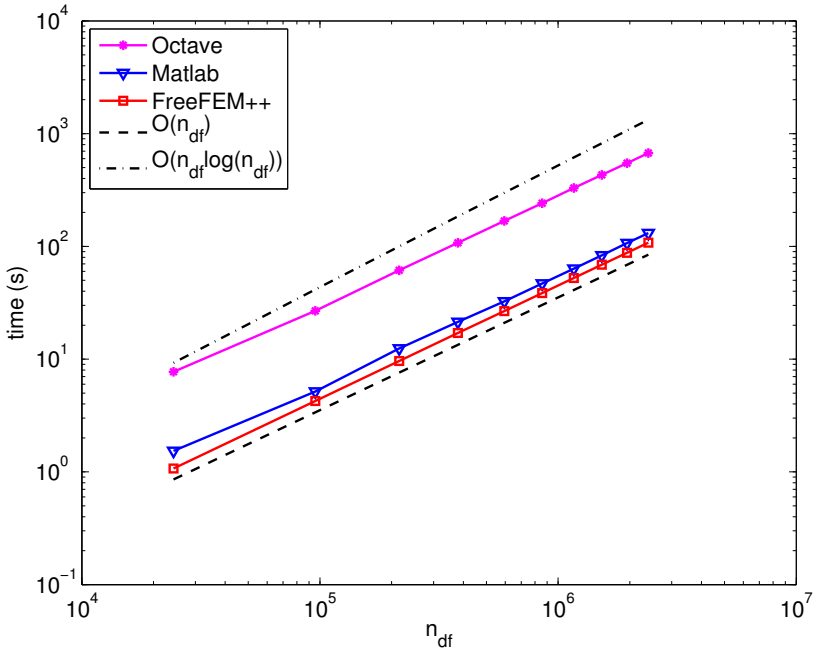

Figure 3: Comparison of Matlab/Octave function `StiffElasAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
12139	24278	7.723 (s) x 1.00	1.536 (s) x 5.03	1.070 (s) x 7.22
47834	95668	26.829 (s) x 1.00	5.155 (s) x 5.20	4.250 (s) x 6.31
107625	215250	61.425 (s) x 1.00	12.454 (s) x 4.93	9.640 (s) x 6.37
189794	379588	107.590 (s) x 1.00	21.406 (s) x 5.03	17.040 (s) x 6.31
297290	594580	168.267 (s) x 1.00	32.544 (s) x 5.17	26.700 (s) x 6.30
428549	857098	241.932 (s) x 1.00	46.937 (s) x 5.15	38.520 (s) x 6.28
581973	1163946	329.547 (s) x 1.00	63.424 (s) x 5.20	52.350 (s) x 6.30
762704	1525408	431.005 (s) x 1.00	83.280 (s) x 5.18	68.740 (s) x 6.27
973281	1946562	546.691 (s) x 1.00	106.990 (s) x 5.11	87.740 (s) x 6.23
1197591	2395182	673.828 (s) x 1.00	131.406 (s) x 5.13	107.840 (s) x 6.25

Table 3: Computational cost of the `StiffElas` matrix assembly versus n_q/n_{df}, with the `OptV1` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV1` Octave version.

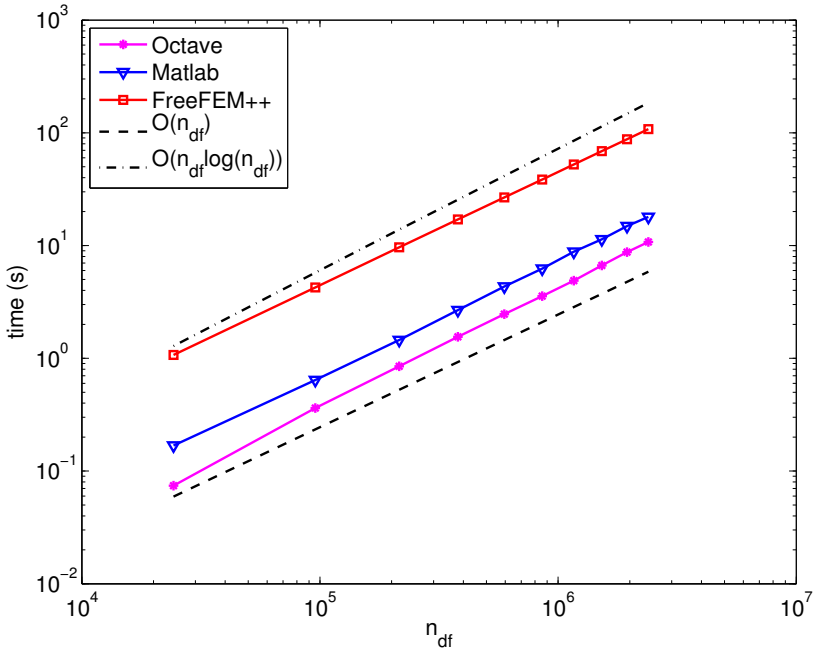

Figure 4: Comparison of Matlab/Octave function `StiffElasAssemblingOptV2` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
12139	24278	0.074 (s) x 1.00	0.169 (s) x 0.44	1.070 (s) x 0.07
47834	95668	0.362 (s) x 1.00	0.641 (s) x 0.56	4.250 (s) x 0.09
107625	215250	0.850 (s) x 1.00	1.451 (s) x 0.59	9.640 (s) x 0.09
189794	379588	1.549 (s) x 1.00	2.673 (s) x 0.58	17.040 (s) x 0.09
297290	594580	2.464 (s) x 1.00	4.337 (s) x 0.57	26.700 (s) x 0.09
428549	857098	3.561 (s) x 1.00	6.246 (s) x 0.57	38.520 (s) x 0.09
581973	1163946	4.887 (s) x 1.00	8.814 (s) x 0.55	52.350 (s) x 0.09
762704	1525408	6.657 (s) x 1.00	11.369 (s) x 0.59	68.740 (s) x 0.10
973281	1946562	8.741 (s) x 1.00	14.878 (s) x 0.59	87.740 (s) x 0.10
1197591	2395182	10.761 (s) x 1.00	17.945 (s) x 0.60	107.840 (s) x 0.10

Table 4: Computational cost of the `StiffElas` matrix assembly versus n_q/n_{df}, with the `OptV2` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV2` Octave version.

3 Mesh : disque4-1

3.1 Function : MassVFAssembling

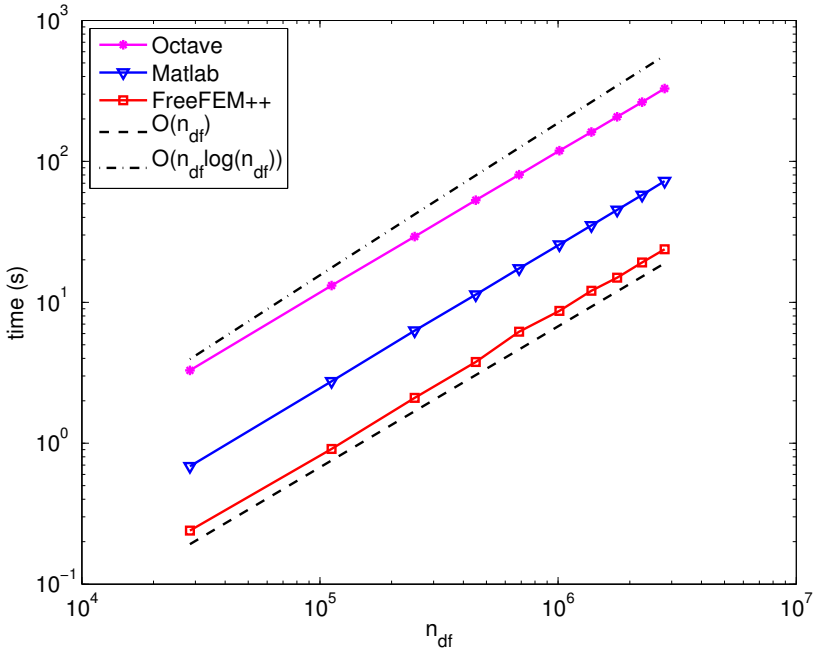

Figure 5: Comparison of Matlab/Octave function `MassVFAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
14222	28444	3.290 (s) x 1.00	0.686 (s) x 4.79	0.240 (s) x 13.71
55919	111838	13.152 (s) x 1.00	2.749 (s) x 4.78	0.910 (s) x 14.45
125010	250020	29.260 (s) x 1.00	6.291 (s) x 4.65	2.100 (s) x 13.93
225547	451094	53.063 (s) x 1.00	11.325 (s) x 4.69	3.770 (s) x 14.08
343082	686164	80.457 (s) x 1.00	17.376 (s) x 4.63	6.180 (s) x 13.02
506706	1013412	118.925 (s) x 1.00	25.641 (s) x 4.64	8.680 (s) x 13.70
689716	1379432	161.689 (s) x 1.00	35.115 (s) x 4.60	12.080 (s) x 13.38
885521	1771042	206.866 (s) x 1.00	45.232 (s) x 4.57	14.950 (s) x 13.84
1127090	2254180	263.296 (s) x 1.00	57.762 (s) x 4.56	19.180 (s) x 13.73
1401129	2802258	328.621 (s) x 1.00	72.359 (s) x 4.54	23.790 (s) x 13.81

Table 5: Computational cost of the `MassVF` matrix assembly versus n_q/n_{df}, with the `OptV1` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV1` Octave version.

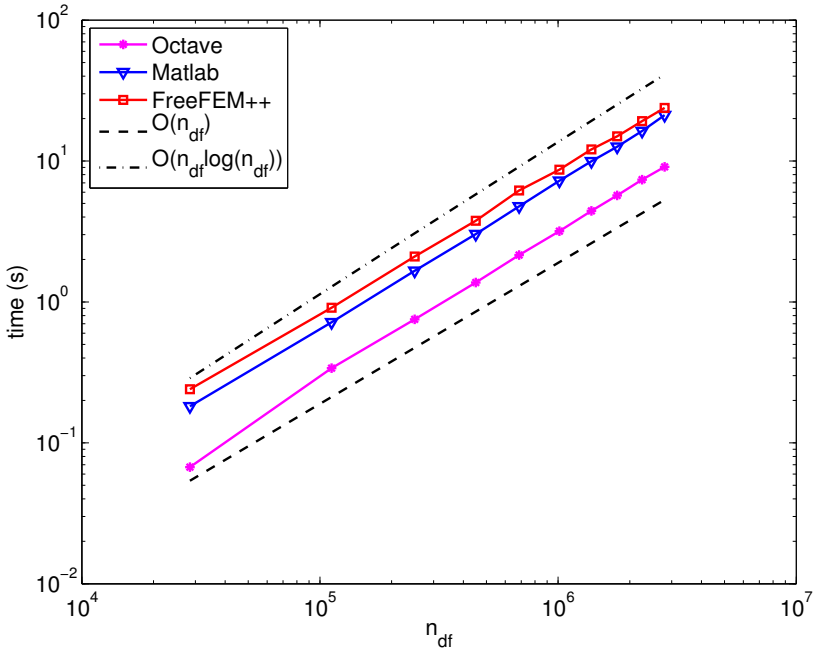

Figure 6: Comparison of Matlab/Octave function `MassVFAssemblingOptV2` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
14222	28444	0.067 (s) x 1.00	0.182 (s) x 0.37	0.240 (s) x 0.28
55919	111838	0.339 (s) x 1.00	0.717 (s) x 0.47	0.910 (s) x 0.37
125010	250020	0.753 (s) x 1.00	1.662 (s) x 0.45	2.100 (s) x 0.36
225547	451094	1.372 (s) x 1.00	3.031 (s) x 0.45	3.770 (s) x 0.36
343082	686164	2.149 (s) x 1.00	4.768 (s) x 0.45	6.180 (s) x 0.35
506706	1013412	3.178 (s) x 1.00	7.242 (s) x 0.44	8.680 (s) x 0.37
689716	1379432	4.428 (s) x 1.00	9.957 (s) x 0.44	12.080 (s) x 0.37
885521	1771042	5.700 (s) x 1.00	12.623 (s) x 0.45	14.950 (s) x 0.38
1127090	2254180	7.375 (s) x 1.00	16.328 (s) x 0.45	19.180 (s) x 0.38
1401129	2802258	9.084 (s) x 1.00	21.169 (s) x 0.43	23.790 (s) x 0.38

Table 6: Computational cost of the `MassVF` matrix assembly versus n_q/n_{df}, with the `OptV2` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV2` Octave version.

3.2 Function : StiffElasAssembling

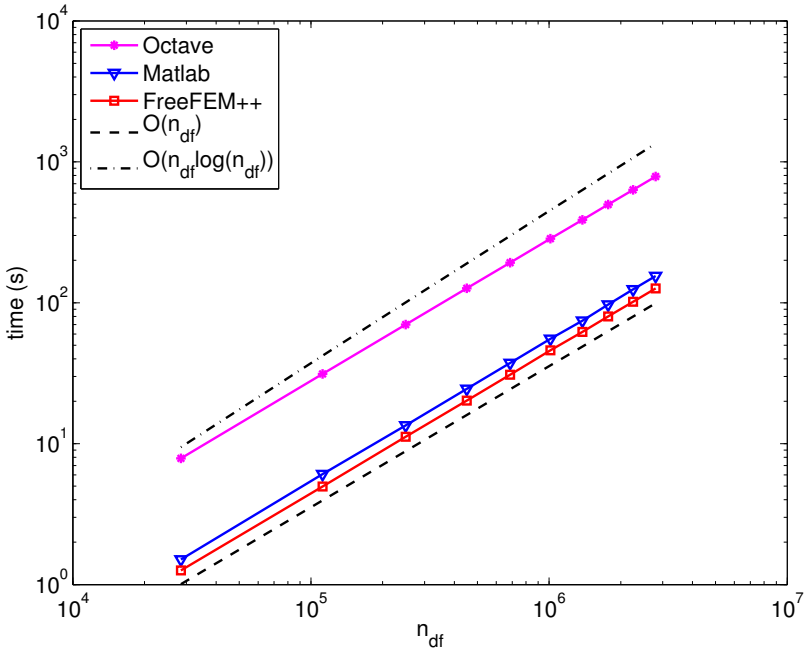

Figure 7: Comparison of Matlab/Octave function `StiffElasAssemblingOptV1` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
14222	28444	7.873 (s) x 1.00	1.509 (s) x 5.22	1.260 (s) x 6.25
55919	111838	31.230 (s) x 1.00	6.084 (s) x 5.13	4.970 (s) x 6.28
125010	250020	70.025 (s) x 1.00	13.490 (s) x 5.19	11.190 (s) x 6.26
225547	451094	126.593 (s) x 1.00	24.509 (s) x 5.17	20.230 (s) x 6.26
343082	686164	192.608 (s) x 1.00	37.392 (s) x 5.15	30.840 (s) x 6.25
506706	1013412	284.692 (s) x 1.00	55.489 (s) x 5.13	45.930 (s) x 6.20
689716	1379432	387.570 (s) x 1.00	74.665 (s) x 5.19	62.170 (s) x 6.23
885521	1771042	497.374 (s) x 1.00	97.201 (s) x 5.12	79.910 (s) x 6.22
1127090	2254180	633.324 (s) x 1.00	124.398 (s) x 5.09	101.730 (s) x 6.23
1401129	2802258	786.855 (s) x 1.00	154.827 (s) x 5.08	126.470 (s) x 6.22

Table 7: Computational cost of the `StiffElas` matrix assembly versus n_q/n_{df}, with the `OptV1` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV1` Octave version.

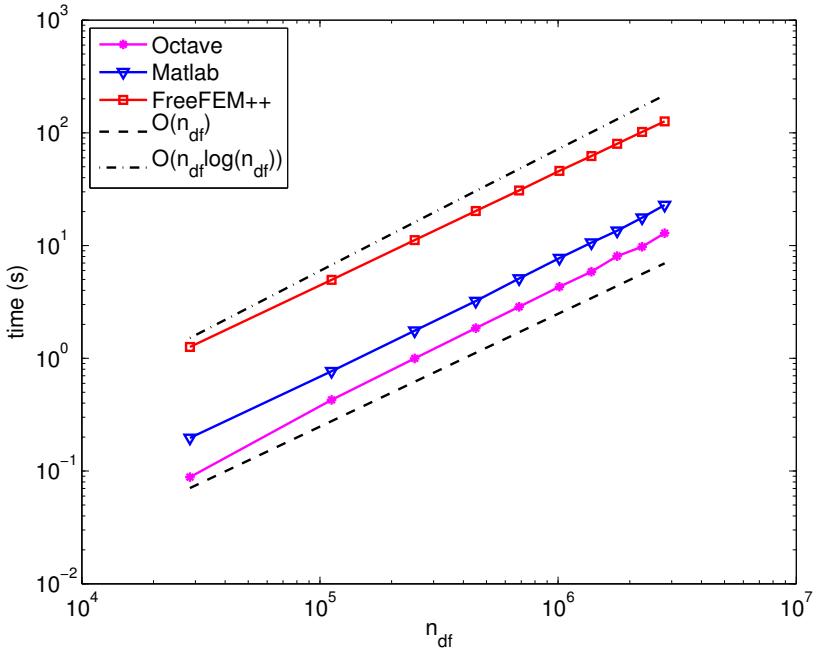

Figure 8: Comparison of Matlab/Octave function `StiffElasAssemblingOptV2` and FreeFEM++

n_q	n_{df}	Octave (3.6.3)	Matlab (R2012b)	FreeFEM++ (3.20)
14222	28444	0.088 (s) x 1.00	0.197 (s) x 0.45	1.260 (s) x 0.07
55919	111838	0.428 (s) x 1.00	0.769 (s) x 0.56	4.970 (s) x 0.09
125010	250020	0.997 (s) x 1.00	1.757 (s) x 0.57	11.190 (s) x 0.09
225547	451094	1.849 (s) x 1.00	3.221 (s) x 0.57	20.230 (s) x 0.09
343082	686164	2.862 (s) x 1.00	5.102 (s) x 0.56	30.840 (s) x 0.09
506706	1013412	4.304 (s) x 1.00	7.728 (s) x 0.56	45.930 (s) x 0.09
689716	1379432	5.865 (s) x 1.00	10.619 (s) x 0.55	62.170 (s) x 0.09
885521	1771042	8.059 (s) x 1.00	13.541 (s) x 0.60	79.910 (s) x 0.10
1127090	2254180	9.764 (s) x 1.00	17.656 (s) x 0.55	101.730 (s) x 0.10
1401129	2802258	12.893 (s) x 1.00	22.862 (s) x 0.56	126.470 (s) x 0.10

Table 8: Computational cost of the `StiffElas` matrix assembly versus n_q/n_{df}, with the `OptV2` Matlab/Octave codes (columns 3,4) and with FreeFEM++ (column 5) : time in seconds (top value) and speedup (bottom value). The speedup reference is `OptV2` Octave version.