PERF013 – Performances d'un calcul modal en parallèle

Résumé :

L'objectif de ce cas-test est de mesurer les performances en mode parallèle d'un calcul modal standard. Pour l'instant, seules les étapes de construction des matrices et, surtout, de résolution des systèmes linéaires associées sont traitées en parallèle. C'est cette dernière étape qui dimensionne les coûts en temps et en mémoire RAM du calcul modal. Elle est ici parallélisée *via* le solveur linéaire MUMPS [U2.08.06]. Les gains en temps et en mémoire sont intéressants et comparables aux résultats présentés par les autres codes généralistes en mécanique des structures.

Pour améliorer ces performances, il faudra aussi paralléliser le solveur modal englobant qui pilote les résolutions de systèmes linéaires.

Le calcul modal utilisé reprend la modélisation C du cas-test *perf003*: plaque carrée maillée en éléments de coques, éléments finis linéaires, recherche de 10 modes propres *via* l'opérateur CALC_MODES et la méthode de Sorensen.

1 Problème de référence

Voir le cas-test perf003c.

2 Solution de référence

Voir le cas-test perf003c.

3 Modélisation A

3.1 Caractéristiques de la modélisation A

Nombre de processeur : 1

On utilise la modélisation C du cas-test perf0013: plaque carrée maillée en éléments de coques, éléments finis linéaires, recherche de 10 modes propres *via* l'opérateur CALC_MODES et la méthode de Sorensen.

Caractéristiques du maillage : 167 281 NOEUD, 1 632 SEG2 et 166 464 QUAD4.
Nombre de degrés de liberté : 1 013 490.

Coté solveur linéaire, on utilise le produit externe MUMPS en cherchant à privilégier les consommations en temps (au détriment *a priori* de celles en mémoire). On utilise donc MUMPS en mode In-Core (GESTION_MEMOIRE='IN_CORE').

Un point important est qu'il faut ici imposer un renuméroteur basique à MUMPS (RENUM='QAMD' par exemple) plutôt que de lui laisser choisir un renuméroteur plus sophistiqué (RENUM='AUTO' par défaut). Car sinon la phase d'analyse de MUMPS, qui est purement séquentielle, devient exagérément grande et le parallélisme n'apporte plus de gain en terme de temps consommé, seulement en terme de mémoire (cf. remarque de la doc [U2.08.03] associée au mot-clé RENUM).

3.2 Résultats

Grandeur	Référence
FREQ n°12	993.5

4 Modélisation B

4.1 Caractéristiques de la modélisation B

Identique à la modélisation A mais le calcul est effectué ici sur 4 processeurs.

4.2 Résultats

Grandeur	Référence
FREQ n°12	993.5

5 Modélisation C

5.1 Caractéristiques de la modélisation C

Identique à la modélisation A mais avec un maillage raffiné uniformément *via* le logiciel HOMARD (un SEG2 est coupé en 2 et un QUAD4 en 4). Calcul sur 1 processeur.

Caractéristiques du maillage: 667 489 NOEUD, 3 264 SEG2 et 665 856 QUAD4.
Nombre de degrés de liberté : 4 024 530.

5.2 Résultats

Grandeur	Référence
FREQ n°12	993.5

6 Modélisation D

6.1 Caractéristiques de la modélisation D

Identique à la modélisation C mais le calcul est effectué ici sur 4 processeurs.

6.2 Résultats

Grandeur	Référence
FREQ n°12	993.5

7 Modélisation E

7.1 Caractéristiques de la modélisation E

Identique à la modélisation C mais le calcul est effectué ici sur 16 processeurs.

7.2 Résultats

Grandeur	Référence
FREQ n°12	993.5

8 Modélisation F

8.1 Caractéristiques de la modélisation F

Identique à la modélisation C mais le calcul est effectué ici en cherchant à privilégier les consommations en mémoire RAM (au détriment *a priori* de celles en temps). On utilise donc MUMPS en mode Out-Of-Core (GESTION_MEMOIRE='OUT_OF_CORE').

Le calcul est effectué sur 1 processeur.

8.2 Résultats

Grandeur	Référence
FREQ n°12	993.5

9 Synthèse des résultats

Machine	Modélisation (nbre de procs)	Mémoire RAM (en Mo)				Temps ELAPSED (seulement CALC_MODES) (en s)				
		Allouée	Utilisée par...			Solveur linéaire MUMPS				Total opérateur
Linux 64 bits «aster4» V10,01			MUMPS	Aster (JEVEUX)	VmPeak	Analyse	Factorisation	Solve	Total Solveur	Total opérateur
Idem	A (1)	2500	1811	1920	3910	10	182	31	223	254
Idem	B (4)	1500	676	1411	2219	14	63	17	94	124
Idem	C (1)	11000	8780	7590	16227	43	1469	147	1658	1786
Idem	D (4)	8000	3700	7590	10209	51	520	76	648	784
Idem	E (16)	8000	1270	7590	8887	59	367	140	456	576
Idem	F (1)	3000	1524	2998	6676	230	894	502	1625	1779

Sur les aspects purement solveur linéaire/parallélisme, on constate au vu de ces résultats:

- En séquentiel, avec le paramétrage par défaut (« priorité temps ») la consommation mémoire du calcul est dictée par celle de MUMPS. Ce n'est plus vrai lorsqu'on parallélise ne serait-ce que sur 4 processeurs ou si on active les options du solveur linéaire limitant les consommations mémoires (« priorité mémoire RAM », modélisation F). D'ailleurs ces dernières ne grèvent pas toujours les performances totales en temps elapsed, certaines pertes (les descentes-remontées de l'étape de « solve » ralenties par l'OOC) étant parfois compensées par des gains spectaculaires sur la factorisation (comme ici entre C et F *via* le changement de renuméroteur).
- Les speed-ups théoriques de la parallélisation sur 4 procs de la modélisation A sont de 2.7 en temps et de 4.0 en RAM. Avec la modélisation B on trouve les speed-ups effectifs, respectivement, 2.0 et 2.7. Les speed-ups théoriques de la parallélisation sur 4/16 procs de la modélisation C sont de 3.0/6.4 en temps et de 4.0/16.0 en RAM. Avec les modélisations D et E, on trouve les speed-ups effectifs, respectivement, 2.3/3.1 et 2.4/6.9. Ces accélérations sont correctes et conforment à celles observées avec les autres codes généralistes en mécanique des structures.

Donc, le fait de paramétrer finement les aspects solveur linéaire et/ou de paralléliser un calcul modal standard peut permettre des gains significatifs (jusqu'à 80%) en temps elapsed et en mémoire RAM.

Toutefois, ces gains ont tendance à plafonner à partir d'une dizaine de processeurs. Pour améliorer ces performances, il faudra étendre le « périmètre parallèle » du code et donc paralléliser aussi le solveur modal englobant (qui pilote les résolutions de systèmes linéaires).