



Simul'Elec and PASTIX interface specifications

Maher Alaya, Mathieu Faverge, Xavier Lacoste, Alexandre Péré-Laperne,
Jacques Péré-Laperne, Pierre Ramet, Theophile Terraz

► To cite this version:

Maher Alaya, Mathieu Faverge, Xavier Lacoste, Alexandre Péré-Laperne, Jacques Péré-Laperne, et al.. Simul'Elec and PASTIX interface specifications. [Rapport Technique] RT-0458, INRIA Bordeaux; AlgoTech; INRIA. 2015. hal-01142204

HAL Id: hal-01142204

<https://inria.hal.science/hal-01142204>

Submitted on 14 Apr 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Simul'Elec and PASTIX interface specifications

M. Alaya, M. Faverge, X. Lacoste, A. Pétré-laperne, J. Pétré-laperne,
P. Ramet, T. Terraz

**TECHNICAL
REPORT**

N° 458

April 2015

Project-Team HIEPACS

ISSN 0249-0803

ISBN INRIA/RT-458-FR+ENG



Simul'Elec and PASTIX interface specifications

M. Alaya*, M. Faverge[†], X. Lacoste[†], A. Péré-laperne*,
J. Péré-laperne*, P. Ramet[†], T. Terraz[†]

Project-Team HIEPACS

Technical Report n° 458 — April 2015 — 19 pages

Abstract: This document presents the interface specification of the electromagnetic modeling module Simul'Elec – from the CAO/DAO tool Pack'ElecBuilder – with the sparse linear system solving library PASTIX.

Key-words: Electromagnetic compatibility, Sparse matrices, sparse solver.

* Algo'Tech Informatique, 20 tue du pont des halles - Technopole Izarbel - Hôtel d'entreprises - 64210 BIDART
† INRIA, 351 cours de la Libération - F-33405 Talence

**RESEARCH CENTRE
BORDEAUX – SUD-OUEST**

200 avenue de la Vieille Tour
33405 Talence Cedex

Spécifications de l'interface entre Simul'Elec et PASTIX

Résumé : Ce document présente les spécification de l'interface entre le module de modélisation électromagnétique Simul'Elec, de l'outil de CAO/DAO de circuits électriques Pack'ElecBuilder, et la bibliothèque de résolution de système linéaire creux PASTIX.

Mots-clés : Compatibilité électromagnétique, Matrices creuses, solveur creux.

1 Introduction

Dans le cadre de l'initiative HPC-PME puis du projet Fortissimo, nous avons travaillé à l'intégration des solutions de résolutions de systèmes linéaires creux proposées par Inria Bordeaux - Sud-Ouest. Dans le cadre de nos travaux plusieurs solutions ont été envisagées que nous présenterons ici. Nous avons tout d'abords interfacée la bibliothèque avec le code en Delphi en ajoutant un fichier permettant l'interfaçage au code de Pack'ElecBuilder. Cela nous a permis de remplacer les appels au solveur maison d'Algo'Tech par des appels à notre solveurs. Ensuite, nous avons extrait l'ensemble de la boucle en fréquence de Pack'ElecBuilder pour développer un binaire responsable d'effectuer cette boucle de résolution.

2 Interfaçage de la bibliothèque avec Pack'ElecBuilder

L'interfaçage de PASTIX avec le code Delphi a été assez simple. Après résolution des quelques problèmes de compilation sous Windows, assez limités vu le peu de dépendances de la bibliothèque PASTIX, l'ajout d'un simple fichier Delphi (`S_pastix.pas`, cf. annexe A) au projet décrivant les fonctions utilisées de PASTIX et quelques constantes nous a permis de pouvoir effectuer des appels au solveur d'Inria.

La fonction principale de la bibliothèque PASTIX est décrite dans le Listing 1. Cette fonc-

```

1  procedure z_pastix(pastix_data: Pointer;
2                      mpi_comm: Integer;
3                      n: Integer;
4                      colptr: TArray<Integer>;
5                      rows: TArray<Integer>;
6                      values: TArray<Complexe>;
7                      perm: TArray<Integer>;
8                      invp: TArray<Integer>;
9                      rhs: TArray<Complexe>;
10                     nrhs: Integer;
11                     iparm: TArray<Integer>;
12                     dparm: TArray<Double>);
13                     cdecl; external 'libpastix';

```

Listing 1: Prototype de la fonction principale de PASTIX

tion permet d'effectuer l'ensemble des appels à la bibliothèque: initialisation, prétraitement, factorisation, résolution, raffinement, et libération des données internes.

Seule la version utilisant des complexes double précision est utilisée ici. La variable `pastix_data` correspondant à une adresse mémoire, elle est initialisée à `Null` par l'utilisateur puis PASTIX l'alloue et l'utilisateur doit la conserver afin de la transmettre au différents appels de PASTIX. Elle permet en effet de stocker les différentes données persistantes requise par la bibliothèque. Le communicateur MPI n'est pas utilisé ici, PASTIX est compilé sans support du MPI sous Windows, un entier de n'importe quelle valeur fera donc l'affaire.

L'entier `n` permet de donner la dimension de la matrice et les tableaux `colptr`, `rows` et `values` permettent de décrire la matrice au format CSC (Compress Sparse Column).

Les tableaux `perm` et `invp` décrivent la permutation et permutation inverse calculée par le partitionneur de graphe utilisé par PASTIX (Scotch en l'occurrence). L'utilisateur doit les allouer et PASTIX se chargera de les remplir. Dans une utilisation de base, l'utilisateur n'a pas besoin d'y toucher par la suite.

Le vecteur `rhs` contient le second membre de l'équation à résoudre. Il est possible de donner plusieurs second membres en les concaténant dans le vecteur et en indiquant leur nombre avec `nrhs` mais ce ne sera pas le cas ici.

Enfin, les vecteurs `iparm` et `dparm` permettent de contrôler la bibliothèque et d'obtenir des informations sur les résultats. Le premier contient des paramètres entier alors que le second contient des valeurs flottantes en double précision. Les champs de ces vecteurs sont remplis en utilisant les constantes `IPARM_ACCESS` et `DPARM_ACCESS` et peuvent prendre les valeurs décrites par les constantes de type `API_...` décrites par `S_pastix.pas`. Il sera intéressant de compléter et d'intégrer le fichier d'interfaçage vers le Delphi (ou Pascal) à la bibliothèque PASTIX pour permettre un interfaçage de la bibliothèque vers ces langages.

3 Utilisation d'un exécutable dédié au parcours de la boucle en fréquence

Afin de permettre l'utilisation à la fois locale et déportée sur un serveur de calcul, nous avons extrait la boucle de calcul en fréquence de Simul'Elec pour en faire un exécutable.

3.1 Description de l'algorithme

La boucle en fréquence de Simul'Elec effectue une résolution d'un système du type $(\frac{A_1}{j\omega} + A_2 + j\omega A_3)X = B$ avec ω variant avec la fréquence. Tous ces systèmes utilisent la même structure de matrice, seules les valeurs changent. Il sera donc possible de n'effectuer le prétraitement qui consiste en la renumérotation de la matrice et sa factorisation symbolique pour prédire sa structure finale une seule fois. De plus tous ces systèmes sont indépendants, nous pourront donc répartir les résolutions sur un ensemble de processeurs sans contrainte.

L'Algorithme 1 décrit la boucle de résolutions.

Algorithm 1: Boucle en fréquence

```

Data:  $A_1, A_2, A_3, X, f_{ini}, f_{fin}, n_{points}$ 
Result:  $S_0 \dots S_{n_{point}-1}$ 
 $A := A_1 + A_2 + A_3;$ 
( $A$ );
 $n_{freq} = \frac{f_{ini}-f_{fin}}{n_{proc}}$ ;
 $my\_f_{ini} = my\_rank \times n_{freq}$ ;
 $my\_f_{fin} = my\_f_{ini} + n_{freq} - 1$ ;
for  $f \in [my\_f_{ini}; my\_f_{fin}]$  do
     $w = 2\pi f$ ;
     $A = \frac{A_1}{j\omega} + A_2 + j\omega A_3$ ;
     $S_f = FactAndSolve(A, X)$ ;
end
```

3.1.1 Version itérative

Les résolutions des itérations proches traitant des systèmes très proches numériquement, nous avons estimé judicieux d'utiliser une méthode hybride directe itérative pour accélérer la résolution. En effet, la solution au temps $t + 1$ est très proche de la solution au temps t . Une méthode itérative utilisant comme point de départ la solution du pas précédent convergera donc rapidement. De plus, la matrice, et donc la matrice factorisée, est également proche numériquement d'une itération à l'autre. Nous pouvons alors utiliser la matrice factorisée d'une itération précédente comme préconditionneur pour accélérer la convergence de la méthode itérative. L'algorithme obtenu alors est décrit par l'Algorithme 2.

Il est possible de paramétriser la méthode itérative en utilisant le paramètre `IPARM_REF_MODE`:

- Avec la valeur `API_REF_FACT` PASTIX factorise la matrice du système à résoudre et l'utilise comme préconditionneur. Le problème est alors résolu en direct. Si la factorisation n'a pas donné lieu à du pivotage le raffinement ne devrait pas être nécessaire.
- Avec la valeur `API_REF_ONLY` PASTIX n'effectue pas de factorisation, il utilise un raffinement itératif sans préconditionneur.
- Avec la valeur `API_REF_PREC` PASTIX utilise une factorisation calculée lors d'un appel précédent pour préconditionner le raffinement itératif.

Lorsque l'on utilise `API_REF_ONLY` ou `API_REF_PREC`, les étapes de factorisation et de résolution de PASTIX sont utilisées pour fournir respectivement la matrice du système à résoudre et la solution de départ au lieu de leur fonction normale. Ce paramètre est contrôlé dans le binaire `algotech` via les paramètres `-i <precision>` pour activer le raffinement itératif et fournir la précision exigée, et `-w` pour activer le préconditionnement.

Il est également possible de changer l'algorithme de raffinement itératif désiré grâce au paramètre `IPARM_REFINEMENT` qui peut prendre pour valeur:

`IPARM_RAF_GMRES` : algorithme GMRES (Generalized minimal residual);

Algorithm 2: Boucle en fréquence avec méthode hybride direct/iterative

```

Data:  $A_1, A_2, A_3, X, f_{ini}, f_{fin}, n_{points}, precision, max\_iter$ 
Result:  $S_0 \dots S_{n_{point}-1}$ 
 $A := A_1 + A_2 + A_3;$ 
 $preprocess(A);$ 
 $n_{freq} = \frac{f_{ini}-f_{fin}}{n_{proc}};$ 
 $my\_f_{ini} = my\_rank \times n\_freq;$ 
 $my\_f_{fin} = my\_f_{ini} + n\_freq - 1;$ 
 $need\_fact = true;$ 
for  $f \in [my\_f_{ini}; my\_f_{fin}]$  do
     $\omega = 2\pi f;$ 
     $A = \frac{A_1}{j\omega} + A_2 + j\omega A_3;$ 
    if  $need\_fact$  then
         $S_f = FactAndSolve(A, X, precision);$ 
         $need\_fact = false;$ 
    else
         $(S_f, n_{iter}) = IterativeSolve(A, X);$ 
        if  $n_{iter} > max\_iter$  then
             $need\_fact = true;$ 
        end
    end
end

```

IPARM_RAF_GRAD : gradient conjugué (disponible seulement dans le cas symétrique);

IPARM_RAF_PIVOT : raffinement itératif simple;

IPARM_RAF_BICGSTAB : algorithme BiCGSTAB (Bi-gradient conjugué stabilisé).

L'option **-a <integer>** du binaire algotech permet de choisir l'algorithme itératif: 0 pour le GMRES, 1 pour le gradient conjugué, 2 pour un raffinement itératif simple, et 3 pour le BiCGSTAB.

En fonction de la précision désirée, le solveur itératif effectuera plus ou moins d'itérations et sera plus ou moins rapide. Une valeur permettant d'obtenir des résultats corrects est $2.5e^{-5}$ (Figure 1). Avec une erreur autorisée de $1e^{-2}$ (Figure 2) par contre, les courbes obtenues présentent un profil en marches en escalier.

3.2 Interface

Cette sous-section décrit les entrées et sorties du binaire **algotech** qui est appelé par Pack'ElecBuilder pour résoudre les systèmes linéaires induits par la boucle en fréquence que ce soit pour les calculs locaux ou distants.

3.2.1 Paramètres

Le binaire **algotech** qui fait partie des exemples d'utilisation de la bibliothèque PASTIX propose les options décrites dans le Listing 2.

Une utilisation classique est donnée par la commande suivante : `./algotech -y -t -s 500000000 -e 2000000000 -p 999 -k 0 -o listOutputIdx -u -v 2 -i 2.5e-3 - Constants.mat Capacitance.mat Inductance.mat Rhs.mat`. La matrice est en effet symétrique et elle est traitée en transposée puisque le code de simulation produit des CSR d'où le **-y** et le **-t**. La simulation parcourt des fréquences allant de 50 KHz à 2 GHz avec 999 points intermédiaires. L'option de regroupement des solution est désactivée (**-k 0**) elle permet de rassembler les solutions par paquets dans des archives compressées. A l'heure actuelle nous compressons seulement l'ensemble des solutions dans le script de lancement de la commande sur une machine de calcul distante. Cette option n'est pas utilisée en calcul local. Le **-o listInputIdx** indique la position d'un fichier contenant la liste des valeurs dont la simulation en sortie. Ce seront ces valeurs qui seront contenues dans les fichiers `sol-<i>.mat`, où *i* est le numéro de la solution, qui seront relues par Simul'Elec. Le fichier `listOutputIdx` est écrit en binaire et contient une liste d'entiers, chacun sur 4 octets. Tout d'abords le nombre d'indices,

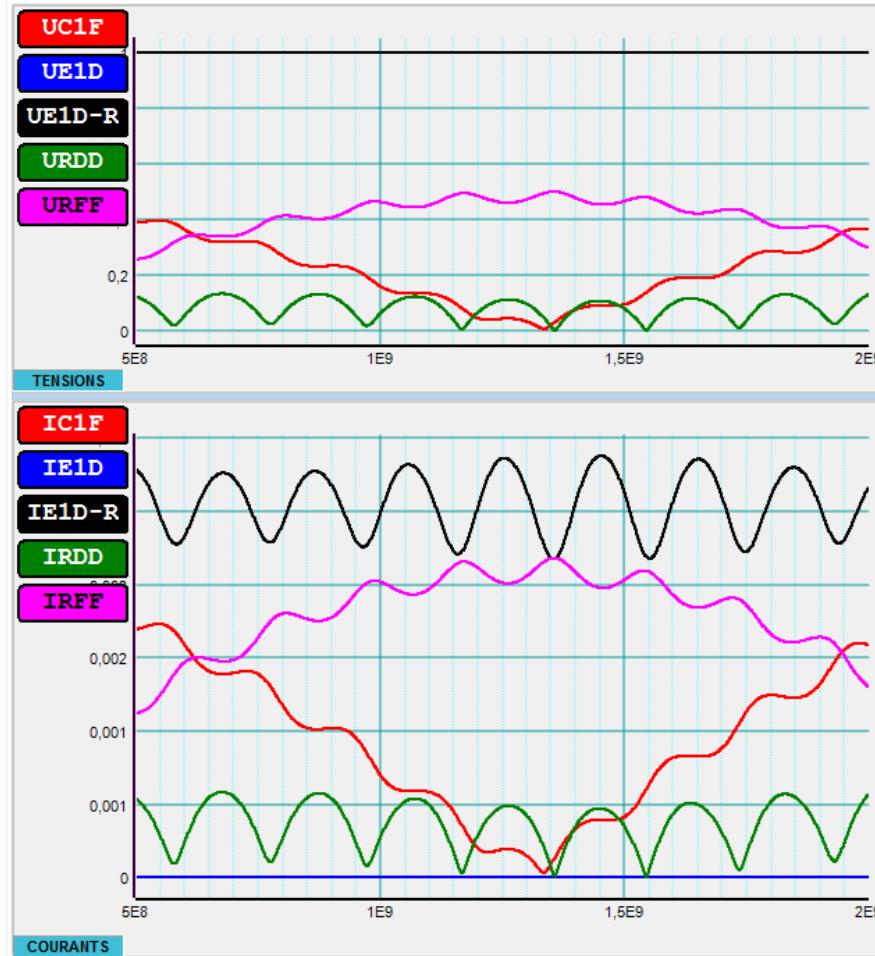


Figure 1: Affichage des résultats avec algorithme incomplet préconditionné utilisant un seuil de $2.5e^{-5}$

```

1 usage: ./algotech [options] — matrix1 matrix2 matrix3 rhs
  options : -y          — If matrix is symmetric
3      -x          — If matrix is hermitian
5      -c          — If check required
7      -t          — If transpose solve
9      -l          — Logarithmic step
11     -p <npoints>   — Number pf points
13     -k <packsize>  — number of solutions in a tar.gz pack
15     -s <double>     — first frequency of the loop
17     -e <double>     — last frequency of the loop
19     -v <verbosity>  — verbosity level
21     -r <threadNbr>  — Number of thread in PaStiX (IPARM_THREAD_NBR)
23     -o <filename>   — binary file containing idx required in outputs
25     -u             — gather all output in one file (only without MPI)
      -i <precision> — Activate iterative solve with given precision
      -m <maxiter>   — when -i is used, factorization is performed when
                        more than maxiter iteration are required ,
                        2*maxiter are authorized.
      -a <integer>   — when -i is used, choose iterative method:
                        0 = GMRES,
                        1 = conjugate gradient ,
                        2 = simple iterative refinement ,
                        3 = bi-conjugate gradient stabilized
      -w             — when -i is used, the iterative refinement
                        is performed with a preconditioner
      -h             — Display this help

```

Listing 2: Option du binaire algotech

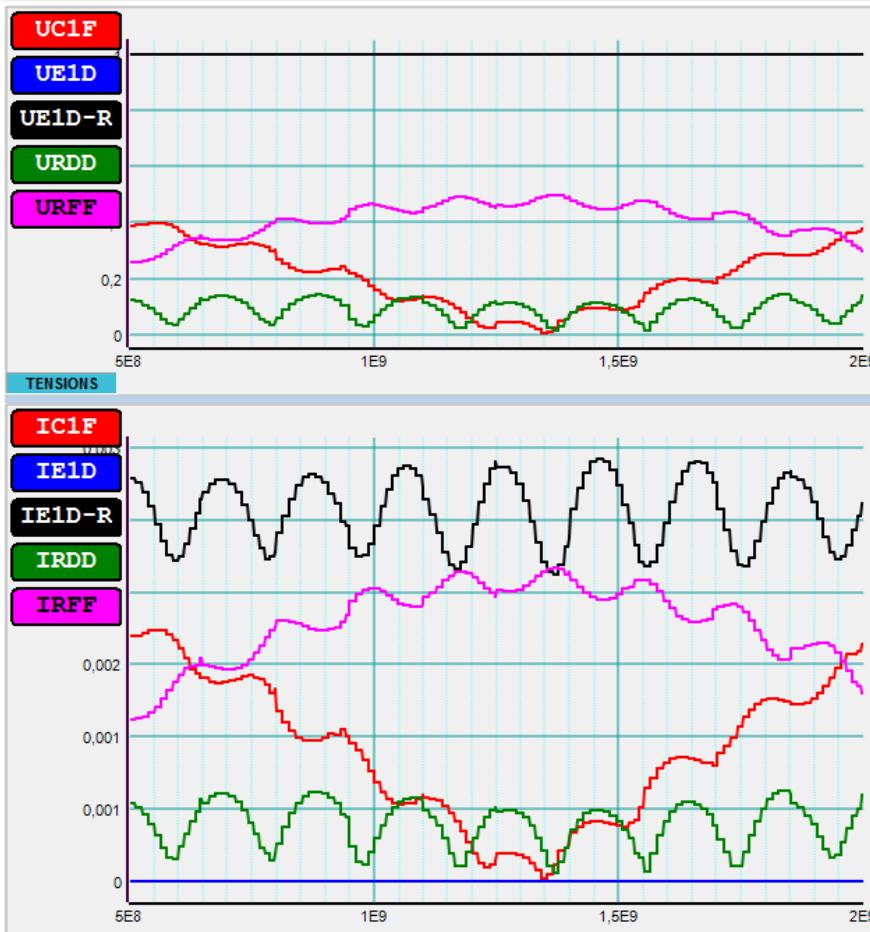


Figure 2: Affichage des résultat avec algorithme incomplet préconditionné utilisant un seuil de $1e^{-2}$

puis les indices en numérotation C (i.e. commençant à l'indice 0). L'option `-v` permet de contrôler le niveau de verbosité et l'option `-i <precision>` permet d'activer le mode hybride direct/itératif avec le critère d'arrêt donné pour la partie itérative.

Les matrices sont fournies au format binaire, elles contiennent la matrice au format CSR ou CSC. Elles commencent par indiquer la taille de la matrice `n`, par un entier sur 4 octet. Puis le tableau `colptr` de taille `n+1` est décrit par autant d'entiers sur 4 octets chacun, numéroté avec la numérotation Fortran (i.e. commençant à l'indice 1). Vient ensuite la liste des colonnes de taille `colptr[n]-1`, en numérotation Fortran. et le tableau des valeurs en double précision sur 8 octets. La première matrice contient les valeurs constantes, la seconde les valeurs en ω et la troisième les valeurs en $\frac{1}{\omega}$.

Le second membre quand à lui ne contient que sa taille suivi de `n` valeur en double complexe sur 16 octets chacune.

3.2.2 Valeurs de retour

Le binaire peut retourner différentes valeurs entières:

- 0 :** Si aucune erreur ne s'est produite;
- 2 :** Si il y a eu un problème de lecture de la matrice ou du second membre;
- 3 :** Si les dimensions des différentes matrices ou du vecteur ne correspondent pas;
- 4 :** Si le format de la matrice est incorrect;
- 6 :** Si la mémoire est insuffisante;
- 7 :** Si la matrice ne valide pas les tests de `pastix_checkMatrix()`;
- 8 :** Si les solutions n'ont pas pu être écrites sur disque;
- 9 :** Si une erreur a été faite dans les options;
- 10 :** Si une erreur a eu lieu pendant le prétraitement;
- 11 :** Si une erreur a eu lieu pendant la factorisation ou la résolution;
- 12 :** Si le fichier contenant la liste des indices à écrire dans les solutions ne peut pas être lu.

Dans le cas d'une exécution distante la sortie est copiée par le script appelant dans un fichier `return.txt`.

3.3 Appel distants

Les appels au binaire `algotech` peuvent se faire soit en local soit via la plate-forme de calcul extreme-factory. Dans le second cas, l'interfaçage peut être fait de deux manières, soit via l'interface web où l'on peut choisir son application, ses données et ses ressources puis visualiser les résultats, soit via une connexion SSH où l'on pourra exécuter la commande `xf_Run` qui soumettra un calcul de manière équivalente à ce que propose l'interface web. C'est ce second modèle que nous utiliserons ici puisqu'il nous permet de nous connecter de manière transparente pour l'utilisateur, toute la partie interface graphique étant fournie par l'application Pack'ElecBuilder.

Ainsi nous utilisons SSH par l'intermédiaire de Putty pour envoyer nos données (`pscp.exe`), exécuter la commande `xf_Run` (`plink.exe`), et récupérer nos résultats (`pscp.exe`).

La commande `xf_Run` prend pour arguments :

- s AlgotechPastix** : pour sélectionner l'application à utiliser. Ici, le binaire `algotech`;
- v <version>** : pour indiquer la version de l'application à utiliser;
- i <inputDirectory>** : pour indiquer le chemin du dossier contenant les matrices;
- q compute.q** : pour indiquer la queue au gestionnaire de travaux;
- n <nodeNbr>** : pour indiquer le nombre de processus MPI;
- C <threadNbr>** : pour indiquer le nombre de coeurs par processus MPI;
- j <jobName>** : pour indiquer le nom du job;

-w hh:mm:ss : pour indiquer le temps d'exécution maximal du job;

Ensuite, les options que nous voudrions passer au binaire `algotech` sont précédées par deux tirets : `xf_Run <opts_xf_Run> -- <opts_algotech>`.

4 Conclusion

Ainsi nous avons décrit ici deux solutions pour l'interfaçage de Pack'ElecBuilder avec PASTIX pour pouvoir utiliser la même solution pour les calculs locaux et distants, nous avons décidé d'utiliser la solution permettant le déport de la boucle en temps dans le binaire `algotech`. En effet, cette solution s'adapte à la fois au calcul locaux et distants. C'est donc celle-ci que nous utiliserons. La version SSH pourra évoluer dans les mois à venir lorsqu'une interface de programmation aura été proposée par BULL pour permettre de remplacer les connexions SSH et de se passer d'outils tels que Putty.

A Fichier d’interfaçage avec le Delphi

Le fichier `S_pastix.pas` est le fichier permettant de décrire les fonctions et constantes permettant les appels de la bibliothèque PaStiX depuis le code en Delphi. Il a été construit à partir de `common/src/api.h` de PaStiX et des descriptions de fonctions se trouvant dans `sopalin/src/pastix.h`.

```

1 unit S_pastix
2
3 interface
4
5 uses S_recc_ksc ;
6
7 const
8
9 { Acces au tableau iparm}
10 {
11   enum: IPARM_ACCESS
12
13   Integer parameters tabular accessors
14
15   IPARM MODIFY_PARAMETER      — Indicate if parameters have been set by user
16           Default: API_YES          IN
17   IPARM_START_TASK            — Indicate the first step to execute (see PaStiX
18           steps)      Default: API_TASK_ORDERING    IN
19   IPARM_END_TASK              — Indicate the last step to execute (see PaStiX
20           steps)      Default: API_TASK_CLEAN     IN
21   IPARM_VERBOSE                — Verbose mode (see Verbose modes)
22           Default: API_VERBOSE_NO    IN
23   IPARM_DOF_NBR               — Degree of freedom per node
24           Default: 1                  IN
25   IPARM_ITERMAX               — Maximum iteration number for refinement
26           Default: 250             IN
27   IPARM_MATRIX_VERIFICATION  — Check the input matrix
28           Default: API_NO          IN
29   IPARM_MC64                   — MC64 operation <pastix.h> IGNORE
30           Default: 0                  IN
31   IPARM_ONLY_RAFF              — Refinement only
32           Default: API_NO          IN
33   IPARM_CSCD_CORRECT          — Indicate if the cscd has been redistributed
34           after blend  Default: API_NO          IN
35   IPARM_NBITER                 — Number of iterations performed in refinement
36           Default: -                OUT
37   IPARM_TRACEFMT               — Trace format (see Trace modes)
38           Default: API_TRACE_PICL   IN
39   IPARM_GRAPHDIST              — Specify if the given graph is distributed or
40           not        Default: API_YES         IN
41   IPARM_AMALGAMATION_LEVEL    — Amalgamation level
42           Default: 5                  IN
43   IPARM_ORDERING                — Choose ordering
44           Default: API_ORDER_SCOTCH  IN
45   IPARM_DEFAULT_ORDERING       — Use default ordering parameters with \scotch or
46           \metis  Default: API_YES         IN
47   IPARM_ORDERING_SWITCH_LEVEL  — Ordering switch level (see \scotch User's
48           Guide)      Default: 120        IN
49   IPARM_ORDERING_CMIN          — Ordering cmin parameter (see \scotch User's
50           Guide)      Default: 0          IN
51   IPARM_ORDERING_CMAX          — Ordering cmax parameter (see \scotch User's
52           Guide)      Default: 100000    IN
53   IPARM_ORDERING_FRAT          — Ordering frat parameter (see \scotch User's
54           Guide)      Default: 8          IN
55   IPARM_STATIC_PIVOTING        — Static pivoting
56           Default: -                OUT
57   IPARM_METIS_PFACTOR          — \metis pfactor
58           Default: 0                  IN
59   IPARM_NNZEROS                 — Number of nonzero entries in the factorized
60           matrix      Default: -          OUT
61   IPARM_ALLOCATED_TERMS        — Maximum memory allocated for matrix terms
62           Default: -          OUT
63   IPARM_BASEVAL                  — Baseval used for the matrix
64           Default: 0                  IN
65   IPARM_MIN_BLOCKSIZE          — Minimum block size
66           Default: 60                 IN
67   IPARM_MAX_BLOCKSIZE          — Maximum block size
68           Default: 120                IN
69   IPARM_SCHUR                    — Schur mode
70           Default: API_NO

```

```

    IN
IPARM_ISOLATE_ZEROS      - Isolate null diagonal terms at the end of the
    matrix      Default: API_NO           IN
44 IPARM_RHSD_CHECK        - Set to API_NO to avoid RHS redistribution
    Default: API_YES            IN
    IN
IPARM_FACTORIZATION       - Factorization mode (see Factorization modes)
    Default: API_FACT_LDLT     IN
46 IPARM_NNZEROS_BLOCK_LOCAL - Number of nonzero entries in the local block
    factorized matrix Default: -          OUT
    IN
IPARM_CPU_BY_NODE         - Number of CPUs per SMP node
    Default: 0                  IN
48 IPARM_BINDTHRD          - Thread binding mode (see Thread binding modes)
    Default: API_BIND_AUTO     IN
    IN
IPARM_THREAD_NBR          - Number of threads per MPI process
    Default: 1                  IN
50 IPARM_DISTRIBUTION_LEVEL - Distribution level IGNORE
    Default: -                  IN
    IN
IPARM_LEVEL_OF_FILL        - Level of fill for incomplete factorization
    Default: 1                  IN
52 IPARM_IO_STRATEGY        - IO strategy (see Checkpoints modes)
    Default: API_IO_NO         IN
    IN
IPARM_RHS_MAKING          - Right-hand-side making (see Right-hand-side
    modes)      Default: API_RHS_B        IN
54 IPARM_REFINEMENT         - Refinement type (see Refinement modes)
    Default: API_RAF_GMRES     IN
    IN
IPARM_SYM                 - Symmetric matrix mode (see Symmetric modes)
    Default: API_SYM_YES      IN
56 IPARM_INCOMPLETE         - Incomplete factorization
    Default: API_NO            IN
    IN
IPARM_ABS                 - ABS level (Automatic Blocksize Splitting)
    Default: 1                  IN
58 IPARM_ESP                 - ESP (Enhanced Sparse Parallelism)
    Default: API_NO            IN
    IN
IPARM_GMRES_IM             - GMRES restart parameter
    Default: 25                IN
60 IPARM_FREE_CSCUSER       - Free user CSC
    Default: API_CSC_PRESERVE
    IN
    IN
IPARM_FREE_CSCPASTIX      - Free internal CSC (Use only without call to
    Refin. step) Default: API_CSC_PRESERVE IN
62 IPARM_OOC_LIMIT           - Out of core memory limit (Mo)
    Default: 2000              IN
    IN
IPARM_OOC_THREAD           - Out of core thread number IGNORE
    Default: 1                  IN
    IN
IPARM_OOC_ID               - Out of core run ID IGNORE
    Default: -                  OUT
    OUT
IPARM_NB_SMP_NODE_USED     - Number of SMP node used IGNORE
    Default: -                  IN
    IN
IPARM_THREAD_COMM_MODE     - Threaded communication mode (see Communication
    modes)      Default: API_THREAD_MULT   IN
    IN
IPARM_NB_THREAD_COMM       - Number of thread(s) for communication
    Default: 1                  IN
    IN
68 IPARM_FILL_MATRIX          - Initialize matrix coefficients (for test only)
    IGNORE Default: -           IN
    IN
IPARM_INERTIA               - Return the inertia (symmetric matrix without
    pivoting)  Default: -           OUT
    OUT
IPARM_ESP_NBTASKS           - Return the number of tasks generated by ESP
    Default: -                  OUT
    OUT
IPARM_ESP_THRESHOLD          - Minimal block sizee to switch in ESP mode (128
    * 128)    Default: 16384      IN
    IN
IPARM_DOF_COST               - Degree of freedom for cost computation (If
    different from IPARM_DOF_NBR) Default: 0          IN
    IN
IPARM_MERGE_REFINEMENT       - Enable refinement in MURGE
    Default: API_YES           IN
    IN
IPARM_STARPU                 - Use StarPU runtime
    Default: API_NO            IN
    IN
IPARM_AUTOSPLIT_COMM         - Automaticaly split communicator to have one MPI
    task by node  Default: API_NO           IN
    IN
IPARM_FLOAT                  - Indicate the floating point type IGNORE
    Default: -                  INOUT
    INOUT
IPARM_PID                    - Pid of the first process (used for naming the
    log directory) Default: -1          OUT
    OUT
IPARM_ERROR_NUMBER            - Return value
    Default: -                  IN
    IN
    OUT
IPARM_CUDA_NBR               - Number of cuda devices
    Default: 0                  IN
    IN

```

```

80    IPARM_TRANSPOSE_SOLVE      — Use transposed matrix during solve
81    IPARM_STARPU_CTX_DEPTH     Default: API_NO           IN
82    IPARM_STARPU_CTX_NBR       Default:3            — Tree depth of the contexts given to StarPU
83    IPARM_PRODUCE_STATS        Default:-1          INOUT
84    IPARM_GPU_CRITERIUM       API_NO             IN
85    IPARM_SIZE                 Default:0          IN
86    IPARM_SIZE                 Iparm Size         IGNORE
87    IPARM_SIZE                 Default :           IN
88    IPARM_MODIFY_PARAMETER     = 0 ;
89    IPARM_START_TASK           = 1 ;
90    IPARM_END_TASK             = 2 ;
91    IPARM_VERBOSE              = 3 ;
92    IPARM_DOF_NBR              = 4 ;
93    IPARM_ITERMARX             = 5 ;
94    IPARM_MATRIX_VERIFICATION = 6 ;
95    IPARM_MC64                 = 7 ;
96    IPARM_ONLY_RAFF             = 8 ;
97    IPARM_CSCD_CORRECT         = 9 ;
98    IPARM_NBITER                = 10 ;
99    IPARM_TRACEFMT              = 11 ;
100   IPARM_GRAPHDIST             = 12 ;
101   IPARM_AMALGAMATION_LEVEL   = 13 ;
102   IPARM_ORDERING              = 14 ;
103   IPARM_DEFAULT_ORDERING     = 15 ;
104   IPARM_ORDERING_SWITCH_LEVEL = 16 ;
105   IPARM_ORDERING_CMIN         = 17 ;
106   IPARM_ORDERING_CMAX         = 18 ;
107   IPARM_ORDERING_FRAT         = 19 ;
108   IPARM_STATIC_PIVOTING       = 20 ;
109   IPARM_METIS_PFACTOR         = 21 ;
110   IPARM_NNZEROS               = 22 ;
111   IPARM_ALLOCATED_TERMS       = 23 ;
112   IPARM_BASEVAL                = 24 ;
113   IPARM_MIN_BLOCKSIZE         = 25 ;
114   IPARM_MAX_BLOCKSIZE         = 26 ;
115   IPARM_SCHUR                = 27 ;
116   IPARM_ISOLATE_ZEROS         = 28 ;
117   IPARM_RHSD_CHECK             = 29 ;
118   IPARM_FACTORIZATION          = 30 ;
119   IPARM_NNZEROS_BLOCK_LOCAL   = 31 ;
120   IPARM_CPU_BY_NODE            = 32 ;
121   IPARM_BINDTHRD              = 33 ;
122   IPARM_THREAD_NBR             = 34 ;
123   IPARM_DISTRIBUTION_LEVEL    = 35 ;
124   IPARM_LEVEL_OF_FILL           = 36 ;
125   IPARM_IO_STRATEGY             = 37 ;
126   IPARM_RHS_MAKING              = 38 ;
127   IPARM_REFINEMENT              = 39 ;
128   IPARM_SYM                     = 40 ;
129   IPARM_INCOMPLETE              = 41 ;
130   IPARM_ABS                     = 42 ;
131   IPARM_ESP                     = 43 ;
132   IPARM_GMRES_IM                = 44 ;
133   IPARM_FREE_CSCUSER             = 45 ;
134   IPARM_FREE_CSCPASTIX          = 46 ;
135   IPARM_OOC_LIMIT                = 47 ;
136   IPARM_OOC_THREAD               = 48 ;
137   IPARM_OOC_ID                  = 49 ;
138   IPARM_NB_SMP_NODE_USED         = 50 ;
139   IPARM_THREAD_COMM_MODE         = 51 ;
140   IPARM_NB_THREAD_COMM            = 52 ;
141   IPARM_FILL_MATRIX              = 53 ;
142   IPARM_INERTIA                 = 54 ;
143   IPARM_ESP_NBTASKS              = 55 ;
144   IPARM_ESP_THRESHOLD             = 56 ;
145   IPARM_DOF_COST                  = 57 ;
146   IPARM_MURGE_REFINEMENT          = 58 ;
147   IPARM_STARPU                   = 59 ;
148   IPARM_AUTOSPLIT_COMM            = 60 ;
149   IPARM_FLOAT                     = 61 ;
150   IPARM_PID                      = 62 ;

```

```

152 IPARM_ERROR_NUMBER = 63 ;
IPARM_CUDA_NBR = 64 ;
IPARM_TRANSPOSE_SOLVE = 65 ;
154 IPARM_STARPU_CTX_DEPTH = 66 ;
IPARM_STARPU_CTX_NBR = 67 ;
IPARM_PRODUCE_STATS = 68 ;
156 IPARM_GPU_CRITERIUM = 69 ;
IPARM_SIZE = 128 ; { Need to be greater or equal to 64 for
backward compatibility }

160 { Acces au tableau dparm }
162 {
164   Enum: DPARM_ACCESS
166     Floating point parameters tabular accessors
168       DPARM_FILL_IN - Fill-in
169         Default: - OUT
170       DPARM_MEM_MAX - Maximum memory (-DMEMORY_USAGE)
171         Default: - OUT
172       DPARM_EPSILON_REFINEMENT - Epsilon for refinement
173         Default: 1e^(-12) IN
174       DPARM_RELATIVE_ERROR - Relative backward error
175         Default: - OUT
176       DPARM_EPSILON_MAGN_CTRL - Epsilon for magnitude control
177         Default: 1e^(-31) IN
178       DPARM_ANALYZE_TIME - Time for Analyse step (wallclock)
179         Default: - OUT
180       DPARM_PRED_FACT_TIME - Predicted factorization time
181         Default: - OUT
182       DPARM_FACT_TIME - Time for Numerical Factorization step (wallclock)
183         Default: - OUT
184       DPARM_SOLV_TIME - Time for Solve step (wallclock)
185         Default: - OUT
186       DPARM_FACT_FLOPS - Numerical Factorization flops (rate!)
187         Default: - OUT
188       DPARM_SOLV_FLOPS - Solve flops (rate!)
189         Default: - OUT
190       DPARM_RAFF_TIME - Time for Refinement step (wallclock)
191         Default: - OUT
192       DPARM_SIZE - Dparm Size IGNORE
193         Default: -
194         DPARM_SIZE = 64 ; { Need to be greater or equal to 64 for
backward compatibility }

196 /* Etapes de resolution de PaStiX */
197 {
198   Enum: API_TASK
199     PaStiX step modes (index IPARM_START_TASK and IPARM_END_TASK)
200       API_TASK_INIT - Set default parameters
201       API_TASK_ORDERING - Ordering
202       API_TASK_SYMBFACT - Symbolic factorization
203       API_TASK_ANALYSE - Tasks mapping and scheduling
204       API_TASK_NUMFACT - Numerical factorization
205       API_TASK_SOLVE - Numerical solve
206       API_TASK_REFINE - Numerical refinement
207       API_TASK_CLEAN - Clean
208     }
209     { _POS_ 1 }
210
211
212

```

```

214 API_TASK_INIT      = 0 ;
215 API_TASK_ORDERING = 1 ;
216 API_TASK_SYMBFACT = 2 ;
217 API_TASK_ANALYSE  = 3 ;
218 API_TASK_NUMFACT  = 4 ;
219 API_TASK_SOLVE    = 5 ;
220 API_TASK_REFINE   = 6 ;
221 API_TASK_CLEAN    = 7 ;

222 /* Affichage de PaStiX */
223 {
224     Enum: API_VERBOSE

225     Verbose modes (index IPARM_VERBOSE)

226     API_VERBOSE_NOT      - Silent mode ; no messages
227     API_VERBOSE_NO        - Some messages
228     API_VERBOSE_YES       - Many messages
229     API_VERBOSE_CHATTERBOX - Like a gossip
230     API_VERBOSE_UNBEARABLE - Really talking too much...
231 }
232 { _POS_ 5 }

233 API_VERBOSE_NOT      = 0 ; { Nothing }
234 API_VERBOSE_NO        = 1 ; { Default }
235 API_VERBOSE_YES       = 2 ; { Extended }
236 API_VERBOSE_CHATTERBOX = 3 ;
237 API_VERBOSE_UNBEARABLE = 4 ;

238 /* Load strategy for graph and ordering */
239 {
240     Enum: API_IO

241     Check-points modes (index IPARM_IO)

242     API_IO_NO           - No output or input
243     API_IO_LOAD          - Load ordering during ordering step and symbol matrix
244             instead of symbolic factorisation.
245     API_IO_SAVE          - Save ordering during ordering step and symbol matrix
246             instead of symbolic factorisation.
247     API_IO_LOAD_GRAPH   - Load graph during ordering step.
248     API_IO_SAVE_GRAPH   - Save graph during ordering step.
249     API_IO_LOAD_CSC     - Load CSC(d) during ordering step.
250     API_IO_SAVE_CSC     - Save CSC(d) during ordering step.
251 }
252 { _POS_ 6 }

253 API_IO_NO           = 0 ;
254 API_IO_LOAD          = 1 ;
255 API_IO_SAVE          = 2 ;
256 API_IO_LOAD_GRAPH   = 4 ;
257 API_IO_SAVE_GRAPH   = 8 ;
258 API_IO_LOAD_CSC     = 16 ;
259 API_IO_SAVE_CSC     = 32 ;

260 /* Generation du second membre */
261 {
262     Enum: API_RHS

263     Right-hand-side modes (index IPARM_RHS)

264     API_RHS_B - User's right hand side
265     API_RHS_1 - $ \forall i ; X_i = 1 $
266     API_RHS_I - $ \forall i ; X_i = i $

267 }
268 { _POS_ 7 }

269 API_RHS_B = 0 ; { Utilisation du second membre fournit }
270 API_RHS_1 = 1 ; { Utilisation d'un second membre dont tous les coefficients
271                 valent 1 }
272 API_RHS_I = 2 ; { Utilisation d'un second membre tel que RHS(i) = i }
273 API_RHS_0 = 3 ; { Initialisation en mode ONLY_RAFF d'une solution X0(i) = 0 }

274 /* Type de raffinement utilise */
275 {
276     Enum: API_RAF

```

```

288     Refinement modes (index IPARM_REFINEMENT)
289
290     API_RAF_GMRES   - GMRES
291     API_RAF_GRAD    - Conjugate Gradient ($LL^t$ or $LDL^t$ factorization)
292     API_RAF_PIVOT   - Iterative Refinement (only for $LU$ factorization)
293     API_RAF_BICGSTAB - BICGSTAB
294 }
295 { _POS_ 8 }

296     API_RAF_GMRES = 0 ; { Utilisation de GMRES }
297     API_RAF_GRAD = 1 ; { Utilisation du gradient conjugué }
298     API_RAF_PIVOT = 2 ; { Utilisation de la méthode du pivot }
299     API_RAF_BICGSTAB = 3 ;

300 /* Type de facto utilisée (LLT ;LDLT ;LU) */
301 {
302     Enum: API_FACT

303     Factorization modes (index IPARM_FACTORISATION)

304     API_FACT_LL    - $LL^t$ Factorization
305     API_FACT_LDLT  - $LDL^t$ Factorization
306     API_FACT_LU    - $LU$ Factorization
307     API_FACT_LDLH  - $LDL^h$ hermitian factorization
308 }
309 { _POS_ 4 }

310     API_FACT_LL    = 0 ; { Factorisation de Cholesky }
311     API_FACT_LDLT  = 1 ; { Factorisation de Crout }
312     API_FACT_LU    = 2 ; { Factorisation LU }
313     API_FACT_LDLH  = 3 ;

314 /* Matrice symétrique ou non (0 : symétrique ; 1 : non) */
315 {
316     Enum: API_SYM

317     Symmetric modes (index IPARM_SYM)

318     API_SYM_YES   - Symmetric matrix
319     API_SYM_NO    - Nonsymmetric matrix
320     API_SYM_HER   - Hermitian
321 }

322 { _POS_ 3 }

323     API_SYM_YES = 0 ; { Matrice symétrique      }
324     API_SYM_NO  = 1 ; { Matrice non symétrique }
325     API_SYM_HER = 2 ;

326 /* Supressing user CSC(D) when not usefull anymore */
327 {
328     Enum: API_ERASE_CSC

329     CSC Management modes (index IPARM_FREE_CSCUSER and IPARM_FREE_CSCPASTIX)

330     API_CSC_PRESERVE - Do not free the CSC
331     API_CSC_FREE     - Free the CSC when no longer needed
332 }
333 { _POS_ 11 }

334     API_CSC_PRESERVE = 0 ;
335     API_CSC_FREE     = 1 ;

336 /* DMP communication mode */
337 {
338     Enum: API_THREAD_MODE

339     Communication modes (index IPARM_THREAD_COMM_MODE)

340     API_THREAD_MULTIPLE      - All threads communicate.
341     API_THREAD_FUNNELED      - One thread perform all the MPI Calls.
342     API_THREAD_COMM_ONE      - One dedicated communication thread will receive
343         messages.
344     API_THREAD_COMM_DEFINED  - Then number of threads receiving the messages is
345         given by IPARM_NB_THREAD_COMM.
346     API_THREAD_COMM_NBPROC   - One communication thread per computation thread

```

```

      will receive messages.

362  }
{ _POS_ 9 }

364
365  API_THREAD_MULTIPLE      = 1 ;
366  API_THREAD_FUNNELED     = 2 ;
367  API_THREAD_COMM_ONE     = 4 ;
368  API_THREAD_COMM_DEFINED = 8 ;
369  API_THREAD_COMM_NBPROC  = 16 ;

370
371  {* Thread binding }
372  {
373    Enum: API_BIND_MODE

374    Thread-binding modes (index IPARM_BINTHRD)

375
376    API_BIND_NO   - Do not bind thread
377    API_BIND_AUTO - Default binding
378    API_BIND_TAB  - Use vector given by pastix_setBind
379  }
{ _POS_ 12 }

380
381  API_BIND_NO      = 0 ; { Do not bind threads
382  API_BIND_AUTO    = 1 ; { Default thread binding
383  API_BIND_TAB     = 2 ; { Use tabular given by pastix_setBind to bind }

384
385  {* Boolean }
386  {
387    Enum: API_BOOLEAN

388    Boolean modes (All boolean except IPARM_SYM)

389
390    API_NO   - No
391    API_YES - Yes
392  }
{ _POS_ 2 }

393
394  API_NO  = 0 ;
395  API_YES = 1 ;

400
401  {* Trace format }
402  {
403    Enum: API_TRACEFMT

404    Trace modes (index IPARM_TRACEFMT)

405
406    API_TRACE_PICL      - Use PICL trace format
407    API_TRACE_PAJE       - Use Paje trace format
408    API_TRACE_HUMREAD   - Use human-readable text trace format
409    API_TRACE_UNFORMATED - Unformatted trace format
410  }
{ _POS_ 10 }

411
412  API_TRACE_PICL      = 0 ; { Use PICL trace format
413  API_TRACE_PAJE       = 1 ; { Use Paje trace format
414  API_TRACE_HUMREAD   = 2 ; { Use text trace format
415  API_TRACE_UNFORMATED = 3 ; { Use unformatted trace format

416
417  {
418    Enum: API_ORDER

419    Ordering modes (index IPARM_ORDERING)

420
421    API_ORDER_SCOTCH    - Use \scotch ordering
422    API_ORDER_METIS     - Use \metis ordering
423    API_ORDER_PERSONAL  - Apply user's permutation
424    API_ORDER_LOAD       - Load ordering from disk
425  }
{ _POS_ 11 }

426
427  API_ORDER_SCOTCH    = 0 ;
428  API_ORDER_METIS     = 1 ;
429  API_ORDER_PERSONAL  = 2 ;
430  API_ORDER_LOAD       = 3 ;

431
432  {
433    Enum: API_FLOAT

```

```

438 Ordering modes (index IPARM_ORDERING)
440
441 API_REALSINGLE - Use \scotch ordering
442 API_REALDOUBLE - Use \metis ordering
443 API_COMPLEXSINGLE - Apply user's permutation
444 API_COMPLEXDOUBLE - Load ordering from disk
445 }
446 { _POS_ 61 }

447 API_REALSINGLE = 0 ;
448 API_REALDOUBLE = 1 ;
449 API_COMPLEXSINGLE = 2 ;
450 API_COMPLEXDOUBLE = 3 ;

451 {
452 * Enum: API_GPU_CRITERIUM
453 *
454 * Criterium used to decide to put tasks on GPUs.
455 *
456 * API_GPU_CRITERION_UPDATES - Number of updates on the panel.
457 * API_GPU_CRITERION_CBLKSIZE - Size of the target panel.
458 * API_GPU_CRITERION_FLOPS - Number of FLOP involved in updates.
459 * API_GPU_CRITERION_PRIORITY - Priority computed in static scheduler.
460 }
461

462 API_GPU_CRITERION_UPDATES = 0 ;
463 API_GPU_CRITERION_CBLKSIZE = 1 ;
464 API_GPU_CRITERION_FLOPS = 2 ;
465 API_GPU_CRITERION_PRIORITY = 3 ;

466 {
467 * Enum: MODULES
468
469 Module Identification number.
470
471 If an error occurs ; error value is set to
472 MODULE + EER_NUMBER.
473
474 User can catch error by computing iparm[IPARM_ERROR_NUMBER]%100.
475
476 MODULE can be catch by computing iparm[IPARM_ERROR_NUMBER] - iparm[
477 IPARM_ERROR_NUMBER]%100.

478 MOD_UNKNOWN - Unknown module
479 MOD_SOPALIN - Numerical factorisation module
480 MOD_BLEND - Analysing module
481 MOD_SCOTCH - Scotch module
482 MOD_FAX - Symbolic factorisation module
483 MOD_ORDER - Order module
484 MOD_COMMON - Common module
485 MOD_SI -
486 MOD_GRAPH - Graph module
487 MOD_SYMBOL - Symbol structure module
488 MOD_KASS - Kass module
489 MOD_BUBBLE - Bubble
490 MOD_MERGE - Merge

491 }
492

493 MOD_UNKNOWN = 0 ;
494 MOD_SOPALIN = 100 ;
495 MOD_BLEND = 200 ;
496 MOD_SCOTCH = 300 ;
497 MOD_FAX = 400 ;
498 MOD_ORDER = 500 ;
499 MOD_COMMON = 600 ;
500 MOD_SI = 700 ;
501 MOD_GRAPH = 800 ;
502 MOD_SYMBOL = 900 ;
503 MOD_KASS = 1000 ;
504 MOD_BUBBLE = 1100 ;
505 MOD_MERGE = 1200 ;

506 {
507 * Enum: EER_NUMBERS
508
509 Error Numbers

```

```

514      NO_ERR           - No error
516      UNKNOWN_ERR     - Unknown error
518      ALLOC_ERR        - Allocation error
519      ASSERT_ERR        - Error in one assertion
520      NOTIMPLEMENTED_ERR - Not implemented feature
521      OUTOFGMEMORY_ERR   - Not enough memory (OOC)
522      THREAD_ERR         - Error with threads
523      INTERNAL_ERR       - Internal error
524      BADPARAMETER_ERR    - Bad parameters given
525      FILE_ERR          - Error in In/Out operations
526      BAD_DEFINE_ERROR    - Error with defines during compilation
527      INTEGER_TYPE_ERR    - Error with integer types
528      IO_ERR             - Error with input/output
529      MATRIX_ERR          - Wrongly defined matrix
530      FLOAT_TYPE_ERR      - Wrong type of floating point values
531      STEP_ORDER_ERR      - Error in ordering
532      MPI_ERR            - Error with MPI calls
533  }
534 { Need to conserve it MURGE compliant }

535      NO_ERR           = 0 ;
536      UNKNOWN_ERR     = 1 ;
537      ALLOC_ERR        = 2 ;
538      ASSERT_ERR        = 3 ;
539      NOTIMPLEMENTED_ERR = 4 ;
540      OUTOFGMEMORY_ERR   = 5 ;
541      THREAD_ERR         = 6 ;
542      INTERNAL_ERR       = 7 ;
543      BADPARAMETER_ERR    = 8 ;
544      FILE_ERR          = 9 ;
545      BAD_DEFINE_ERR      = 10 ;
546      INTEGER_TYPE_ERR    = 11 ;
547      IO_ERR             = 12 ;
548      MATRIX_ERR          = 13 ;
549      FLOAT_TYPE_ERR      = 14 ;
550      STEP_ORDER_ERR      = 15 ;
551      MPI_ERR            = 16 ;

552  procedure z_pastix(pastix_data: Pointer;
553                      mpi_comm: Integer;
554                      n: Integer;
555                      colptr: TArray<Integer>;
556                      rows: TArray<Integer>;
557                      values: TArray<Complex>;
558                      perm: TArray<Integer>;
559                      invp: TArray<Integer>;
560                      rhs: TArray<Complex>;
561                      nrhs: Integer;
562                      iparm: TArray<Integer>;
563                      dparm: TArray<Double>);
564                      cdecl; external 'libpastix';

565  function z_pastix_checkMatrix_2steps(data_check: pointer;
566                                         pastix_comm: integer;
567                                         verb : integer;
568                                         flagsym : integer;
569                                         flagcor : integer;
570                                         n : integer ;
571                                         colptr : TArray<Integer> ;
572                                         row : TArray<integer> ;
573                                         avals : TArray<complex> ;
574                                         loc2glob : pointer ;
575                                         dof : integer) : integer ;
576                                         cdecl ; external 'libpastix';

577  procedure z_pastix_checkMatrix_2steps_end(data_check : pointer ;
578                                         verb : integer ;
579                                         row : TArray<integer> ;
580                                         avals : TArray<complex> ;
581                                         dof : integer) ;
582                                         cdecl ; external 'libpastix';

583  procedure openblas_set_num_threads(nblas : integer); cdecl; external 'libopenblas';
584
585  implementation

```

590
end.



**RESEARCH CENTRE
BORDEAUX – SUD-OUEST**

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803