Activities Report

Development of new XNS features

by Leszek Seweryn

November 17, 2010

Supervisor: Mehdi Behbahani
# Contents

1 Introduction ................................................................. 1
   1.1 Mac OS ................................................................. 1
   1.1.1 Searching ......................................................... 1
   1.1.2 Shell appearance ............................................... 2
   1.1.3 Useful shortcuts ............................................... 2
   1.1.4 Useful commands ............................................... 3
   1.2 XNS ................................................................. 3
       1.2.1 Files in XNS ................................................ 3
       1.2.2 Parameters in XNS ......................................... 3
       1.2.3 Functions in XNS ........................................... 5
       1.2.4 Functions in ewd ........................................... 5
       1.2.5 Simulation on Hydra ...................................... 9
   1.3 JuGene ................................................................. 11
       1.3.1 The ssh aliases .............................................. 11
       1.3.2 File transfer ................................................ 11
       1.3.3 mprm generation ............................................ 14
       1.3.4 XNS simulations ............................................ 14
       1.3.5 run.* script ................................................. 14
   1.4 Fortran ................................................................. 14
       1.4.1 Vectors ......................................................... 14
       1.4.2 Common blocks .............................................. 15
       1.4.3 Save attribute .............................................. 15
       1.4.4 mpi subroutines ............................................ 15

2 Performance issue of xns-stock (Thrombosis model - shear dependent k-values) ................................................................. 16
   2.1 Problem statement ................................................. 16
   2.2 Solution ............................................................. 16
   2.3 Simulation performance comparison ................................ 18

3 Concatenation of simulation results (Aortic flow) ......................... 21
   3.1 Problem statement ................................................. 21
   3.2 Solution ............................................................. 21

4 Conversion of the VRML 1.0 surface mesh data into mxyz format(Human heart project) ................................................................. 22
   4.1 Problem statement ................................................. 22
4.2 Solution ................................................. 22
  4.2.1 Simple case ....................................... 22
  4.2.2 More complex case ................................. 22

5 Surface grid points displacement calculation (Human heart project) 24
  5.1 Problem statement ................................... 24
  5.2 Solution ................................................ 24
    5.2.1 Surface shift vector approach .................. 24
    5.2.2 Volume shift vector approach .................. 25
    5.2.3 The xns.in modifications ....................... 25
    5.2.4 The newxrigid.F file ............................. 26
    5.2.5 Simulation attempts - human heart project ......... 26

6 Simulation data post-processing .................................. 28
  6.1 remove_spacetime_records script ........................ 28
  6.2 compareXyz application .................................. 29
  6.3 find_xyz application ..................................... 29
  6.4 movePointCoordinates application .......................... 30
  6.5 Visualization in Ensight ................................. 31
  6.6 The mixd to *.grd conversion tool ...................... 31

7 Shear stress calculation and output .................................. 33
  7.1 Problem statement ................................... 33
  7.2 Solution ................................................ 33
    7.2.1 Background ....................................... 33
    7.2.2 XNS ............................................... 34
    7.2.3 XNS version history for shear stress implementation 35

8 Time approximated factor for rngdexp and rngdrot expressions .......... 36
  8.1 Problem statement ................................... 36
  8.2 Solution ................................................ 36
    8.2.1 XNS ............................................... 36
    8.2.2 XNS version history for fapprox implementation .... 36

9 Shear stress dependent reactivity rates .................................. 37
  9.1 Problem statement ................................... 37
  9.2 Solution ................................................ 37
    9.2.1 XNS ............................................... 37
    9.2.2 XNS version history ................................. 37

10 Mechanical platelet activation due to high shear stress values .... 38
  10.1 Problem statement ................................... 38
  10.2 Solution ................................................ 38
    10.2.1 XNS ............................................... 38
    10.2.2 XNS version history ................................. 38
11 Previous XNS modifications

11.1 Changes introduced by Christian Waluga till 2008-10-13 (version xns-waluga)  39
11.2 Changes introduced by Sebastian Stock till 2009-04-30 (version xns-stock) .  41
1 Introduction

1.1 Mac OS

1.1.1 Searching

find . | xargs grep 'ewdzero' > find_ewdzero

Checks files recursively in the current directory looking for "ewdzero" keyword, pipes the output to find_ewdzero file.

1.1.2 Shell appearance

Location: \home\seweryn\02v6."3+s\n
When working with multiple bash shells it seems to be far more convenient to use dark background and be able to visually differentiate between different shells for different purposes. For this purpose a useful script was prepared and is presented in figure 1.1.

```
#!/bin/bash

# USAGE: ./console name foreground-color background-color
echo Please specify: NAME FOREGROUND_COLOR BACKGROUND_COLOR
read NAME FG BG

# -T option sets the name of the terminal session
# -fg option sets the foreground color
# -bg option sets the background color
# -cr option sets the cursor color
# -g option sets the initial (x,y) position of the new terminal session
xterm -T $NAME -bg $BG -fg $FG -cr purple -g 280x70 &
```

Figure 1.1: console

In order to set a default appearance of the shell the .Xdefault file has to be created (modified) in the home directory. Following lines have to be appended:
1 Introduction

XTerm*background: black
XTerm*foreground: white

1.1.3 Useful shortcuts

`cmd+Shift+3`:
Captures the whole screen.

`cmd+Shift+4`:
Captures selected area.

`cmd+Ctrl+Shift+4`:
Captures selected area. Open the "Preview" application press `cmd+N` to paste the picture and `cmd+S` to save it in the *.PNG format.

`cmd+Shift+4+Spacebar`:
Captures current window.

`Left_Shift+numeric++`:
Increases the font size in xterm.

1.1.4 Useful commands

`man colors`:
Command can be used to display all the colors that are available in MAC OS.

`open <file>`:
Can be used in xterm to open a file. Use `open -a <application>` to start an application.

`cat <file1> <file2> > <file3>`:
Concatenates file1 with file2 within file3, where file1 occurs first.

`cat <file2> » <file1>`:
Appends file2 at the end of file1.

`chmod -R <u,g,o> <-,+,-> <r,w,x> <directory, file>`:
Recursively changes the access rights of the given file or directory for the owner(u), group(g) or others(o). The rights can be reset(=), added(+) or removed(-). They allow to read(r), write(w) or execute(x) the object in question.

`chmod 755 <file>`:
Modifies access rights of the file. The owner gets full access(read=4, write=2, execute=1), the group and others are only allowed to read and execute but not to modify the file.
1.2 XNS

1.2.1 Files in XNS

blkadvsd.F:
Forms element matrices and residuals for 2D-3D semi-discrete AD.

ewd.h:
Contains declaration of memory allocation functions and parameters, as well as CRAY pointers.

file.h:
Contains I/O directory and file names.

global.h:
Contains global variable and parameter names in XNS.

1.2.2 Parameters in XNS

facemap:
Array that maps local face node numbers to local element node numbers.

irng:
The i-th reference node group.

mdf:
Maximum number of degrees of freedom.

mef:
Maximum number of element faces.

men:
Maximum number of element nodes.

mreg:
Maximum number of reference element groups.

mrng:
Maximum number of reference node groups.

msd:
Maximum number of spatial dimensions.

nde:
The number of independent components of stress tensor (nde = 3 if nsd = 3).

ndf:
Number of degrees of freedom per node.
ne: Number of elements.

nec: Number of elements per partition.

necr: Number of \textit{irng} elements per PE.

nef: Number of element faces.

nelr: Number of elements in the current partition whose face belongs to a given \textit{rng}.

nen: Number of nodes per element.

ner: Number of elements in all partitions whose face belongs to a given \textit{rng}.

nnc: Number of nodes per partition.

nnf: Number of face nodes.

nnl: Number of nodes touched by partition.

nsd: Number of coordinates per node (space dimensions).

nxc: Number of coordinates per partition.

x(nsd, nnl): Coordinates of nodes touched by partition.

xe(nsd, nen): Stores element coordinates.

xg(nsd, nnc): Coordinates of nodes within partition.
1.2.3 Functions in XNS

`elementSurfaceElementType(elemtype)`:
Location: `element.F`
Determines the surface element type based on the volume element type - `elemtype`.

`ewdfeqquad(elemtype, iquadoh, nquad, wq, wq, nsd)`:
Location: `quad.F`
Retrieves the coordinates of the quadrature points for the given element.

`facemap(mef, mnf, nelemtype)`:
Maps local face node numbers to local element node numbers.

`iel2iec(nelr)`:
If any of the faces of the element has a boundary code, the number of the element in the partition is stored for the given number of elements with at least one face belonging to `irng`.

`iel2ief1(nelr)`:
If any of the faces of the element has a boundary code, its face number is stored for given element number.

`rng(nef, nec)`:
Returns the boundary code (reference node group) or 0 if no boundary info is assigned.

1.2.4 Functions in ewd

`ewdlocalize(ndf, ie, bm, b, id)`:
Location: `ewdcomm.F`
Performs single element gather.

`ewdfeshape(elemtype,"abc",nquad,xq,xe,nsd,tsd,1,sq,det,sh,zf,dt)`
Location: `shape.F`

0: `sq[tsd,nen,nquad]` - shape function values
0: `det[dsd,nquad]` - Jacobian determinants
0: `sh[tsd,nen,nquad]` - shape function derivatives
0: `zf[tsd,lsd,nquad]` - Jacobian entries
Figure 1.2: nec and nef parameters
Introduction

I: \text{xq[msd,nquad]} - quadrature point coordinates

\text{ewdgather}(x, xg, m, ndf, isall, id):
  Location: \text{ewdcomm.F.}
  0: \text{x[ndf,ndc]} - element level data
  I: \text{xg[ndf,nnc]} - node level data
  I: \text{mg[ndf,ndc]} - element level mask
  I: \text{isall} - if non-masked

\text{ewdgather1}(xm, xg, mg, ndf, isall, id):
  Location: \text{ewdcomm.F.}
  Converts data from node level to partition level
  0: \text{xm[ndf,nnl]} - partition level data
  I: \text{xg[ndf,nnc]} - node level data
  I: \text{mg[ndf,nnc]} - node level mask

\text{ewdgather2}(x, xm, m, ndf, isall, id):
  Location: \text{ewdcomm.F.}
  0: \text{x[ndf,ndc]} - element level data
  I: \text{xm[ndf,nnl]} - partition level data
  I: \text{m[ndf,ndc]} - node level mask

\text{ewdgetafacemask}(irng, ..., mi):
  Location: \text{ewdmesh.F.}
  Returns the mask \text{mi} that contains all nodes that belong to a given \text{irng}.
  \text{ml(1:nen, 1:nec)} = 0.0
  !For every element in the partition
  DO iec=1, nec
    !For every face of this element
    DO ifl=1, nef
      !If the face of the element belongs to a given \text{irng}
      IF( \text{rng(ifl, iec).EQ.irng } ) THEN
        !Get all nodes belonging to a given face of the given element
        !Set the mask \text{m} to 1.0 for these entries
        \text{ml( facemap(ifl, 1:nnf), iec )} = 1.0
      END IF
    END DO
  END DO

The \text{facemap(ifl, nnf)} returns the element node number for the given face and face node number.
In this manner facemap(1,1) = 3, facemap(1,2) = 2, facemap(1,3) = 1, etc.

ewdgetelemmask(ireg, ..., m):
    Location: ewdmesh.F.
    Returns the mask m that contains all elements that belong to a given ireg (reference element group).

ewdreadint(ienspace, nenspace*nec, iifn, off, .false., dbio):
    Location: ewdFile.F.
    ienspace - output matrix
    nenspace*nec - size of output
    iifn - file descriptor
    off - offset (boolean)
    .false. - whether to append a p-based suffix
    dbio - debug

ewdreduceint(<size>, <whereToCheck>, <output>, "sum", "min", "max"):
    Location: ewdcollect.F. "min":
    Parses the integer vector and returns the smallest integer value within it. Later, the same is done for all processes and the final result is returned. "max":
Returns the max integer value of a (vector) matrix. "sum":
Returns the sum of all entries in the matrix. The ewdreduceint(1, nelr, ner, "sum")
sums nelr over all partitions.

**ewdrhsadd**(nde, ie, p, b, cid1):
Location: ewdcomm.F.
Performs single element assembly.

0: bm[ndf,nnl] - partition level data
I: b[ndf,ndc] - element level data
I: ndf
I: ie - element number
I: cid1 - communication ID

**ewdscatter**(xg, x, m, ndf, isall, isovr, id):
Location: ewdcomm.F.

0: xg[ndf,nnc] - node level data
I: x[ndf,ndc] - element level data
I: m[ndf,ndc] - element level mask
I: isovr - if overwrite at destination flag

**ewdscatter1**(xm, x, m, ndf, isall, isovr, id):
Location: ewdcomm.F.

0: xm[ndf,nnl] - partition level data
I: x[ndf,ndc] - element level data
I: m[ndf,ndc] - element level mask

**ewdscatter2**(xg, xm, mg, ndf, isall, isovr, id):
Location: ewdcomm.F.

0: xg[ndf,nnc] - node level data
I: xm[ndf,nnl] - partition level data
I: mg[ndf,nnc] - node level mask

**ewdzero**(n, x, incx):
Initializes a double precision vector with 0s; n - size of the vector, x - vector, incx -
increment index step (normally '1')

### 1.2.5 Simulation on Hydra

The following steps are used to start a simulation:
Input:
The following XNS input files have to be gathered:

- `xns.in` contains boundary information.
- `mrng` lists number of nodes and elements (semi-discrete, steady on cases).
- `minf.space` lists number of nodes and elements (space-time, steady off cases).
- `mien` contains connectivity array (nen elements).
- `mxyz.space` contains nodal coordinates -nsd entries, e.g. 3 for (x,y,z) (semi-discrete, typically "steady on" cases).
- `mxyz` contains nodal coordinates -nsd entries, e.g. 3 for (x,y,z) (space-time, typically "steady off" cases).
- `run.4` simulation script for 4 processing nodes.

The `mrng` binary file can be read using the following command `hexdump -e "%i %i %i %i %i\n" mrng | less`. The `mxyz` binary file can be read using the following command `hexdump -e "%E %E %E\n" mxyz | less`.

Partition:
The partition utility can be used to partition the usual mien FEM connectivity domain into a set of contiguous sub-domains. For 4 partitions the syntax is: `partition -hex 4`. Other options include: `-tet`, `-qua`, `-tri`. As a result (for four partitions) three files are generated by the partition utility:

- `dual` for every element stores the number of partition it belongs to.
- `mipe.0004` stores the unique node number, e.g. for 800 elements, those belonging to the 1st partition are numbered: 1, 2, 3, ..., those belonging to the 2nd partition are numbered: 201, 202, ...

Submit:
Use `qsub run.4` to submit scenario for simulation purposes.

Status:
Use `qstat` to check the current status of the simulation. The following script will execute the qstat command every 5s:

```
#!/bin/bash
while true
do
  qstat
  sleep 5
done
```

Deletion:
Use `qdel <pid>` to delete simulation with the specific process identifier.
Introduction

mxyz.out:
File contains mesh coordinates for the last iteration step simulated.

1.3 JuGene

1.3.1 The ssh aliases

The 2 useful aliases can be created in .cshrc:

jugenecompile 'ssh hac094@jugene.fz-juelich.de':
To access the account where the code should be modified and compiled.

jugenerun 'ssh hac098@jugene.fz-juelich.de':
To access the account where simulations should be run.

1.3.2 File transfer

Location: \home\seweryn\WORKBENCH\

The scp command can be used for file transfer purposes:

scp <file> hac094@jugene.fz-juelich.de:LESZEK:
Copies the ‘file’ from the local folder to ‘LESZEK’ directory on JuGene.

scp -r hac094@jugene.fz-juelich.de:LESZEK home/seweryn:
Copies ‘LESZEK’ directory to the ‘home/Seweryn/’ directory on the local machine.

To make the file transfer to the hac094 easier the sendjugene script was prepared. The syntax is: sendjugene <file-or-directory-to-move>. If no parameter is specified the script exits with an error message. Otherwise, it prompts the user to specify in which direction the transfer should take place and whether the object is a file or directory. After successful transfer an appropriate message is displayed. Note that the existence of ‘LESZEK’ directory is a requirement for a successful transfer. The script is presented in figure 1.4.

A similar script (sendjugenerun) is also available for data transfer with the hac094. After some time both scripts were merged into one (send2jugene) script which is presented in figure 1.5.
Figure 1.4: sendjugene
Figure 1.5: Script enabling data transfer between both hac098 and hac094 accounts.
1.3.3 mprm generation

1. Modification of the \texttt{xns.in} file
   - \texttt{nprmsave on}
   - \texttt{nts 0}
   - \texttt{nit 0}

2. Appropriate XNS simulation

3. Modification of the \texttt{xns.in} file
   - \texttt{nprmread on}

1.3.4 XNS simulations

\texttt{llsubmit run.2040:}
\quad Submits simulation scenario.

\texttt{llq -u hac098:}
\quad Checks the status of the simulation (all the jobs submitted by hac098 are displayed).

\texttt{llcancel jugene3b.12883.0:}
\quad Cancels job with ID jugene3b.12883.0.

1.3.5 run.* script

Several useful options can be added to the run.* script:

Ask JuGene to notify by an e-mail whenever the job started or finished.

\begin{verbatim}
# @ notify_user = lese@o2.pl
# @ notification = always
\end{verbatim}

1.4 Fortran

1.4.1 Vectors

\begin{verbatim}
real, dimension (3)::v (or the old notation: real::v(3)):
\quad Declares a vector v with 3 components v(1), v(2), v(3).

real, dimension(0:2)::v:
\quad Enforces the initial index of 0 (v(0), v(1), v(2)).
\end{verbatim}
1.4.2 Common blocks

```plaintext
common /<name>/ list-of-variables
Common blocks are used to share variables between subroutines.
real a,b
Common /coeff/ a,b
```

Subroutine sub()
real a,b
common /coeff/ a,b

1.4.3 Save attribute

It is used to make sure that specific parameters (and their values) are visible during the next call to a given subroutine.

1.4.4 mpi subroutines

```plaintext
mpi_allreduce(itmp, iout, 1, MPI_INTEGER, MPI_MIN, MPI_COMM)
Combines all values from all processes and distributes the result back to all processes.
```

- `itmp` - send buffer
- `iout` - receive buffer (output)
- `1` - number of elements
- `MPI_INTEGER` - type
- `MPI_MIN` - Operation handle
- `MPI_COMM` - Communicator handle
2 Performance issue of xns-stock  
(Thrombosis model - shear dependent k-values)

2.1 Problem statement

Sebastian Stock ran simulations on HYDRA with a max number of 16 processing nodes. In such scenarios the code seems to be quite well optimized and outperforms slightly Christian Waluga’s version. When running on JUGENE, however, with a larger number of processing elements (e.g. 2040) the ts/h ratio decreases by a factor of 20. The critical segment of code that is responsible for such a degradation of performance can be found in reactive.F file, in the blkfluxreactive() subroutine:

```fortran
call ewdlexset(xi(xsd), xi(ysd), xi(zsd), ti, dt, its, iit)
do ik=1,maxk
   call ewdlex(kexp(irmg,ik), 1, ki(ik), ierr)
end do
if (ierr.lt.0) call ewderr("blkfluxreactive: ewdlex failed ", ierr)
```

2.2 Solution

**Location:** \home\seweryn\WORKBENCH\xns-working-version\xns-seweryn-1.0\n
The changes to XNS that are described below cause improvement of the performance of xns-stock when running on JUGENE. The xns-stock version makes extensive use of the ewdlex() function which parses string expressions when calculating new kaa, kas, krs values. This is done for every its and every iit highly decreasing the ts/h ratio.

Current version of the XNS assumes constant nodal coordinates of the element during a single simulation. For this reason the recalculation of the k-values has to be done in fact only one single time when both its and iit are 1.

The improvement described below assumes k-value calculations that are independent of time. Changes made to the xns-stock source code include following files:

**global.h**

Following global parameters have been introduced:
• maxquad:
  Stores the number of quadrature points that are later used in
  blkfluxreactive() (reactive.F) subroutine. It is used in hypo.F
  for memory allocation of the m_kValues matrix.

• maxnecr:
  Stores the largest number of irng elements found among all reactive
  irng ’s for the given PE. Evaluated in hypo.F.

• countReactive:
  Stores the amount of reactive irng ’s. Evaluated in hypo.F.

hypo.F
Following global parameters have been introduced:

• irng:
  Index of the rng under investigation.

• istat:
  Stores results of memory allocation calls.

• vectorIndex:
  Index of the v_irngMapping vector.

• elemtypeh:
  Stores the type of the volume element used.

• v_irngMapping:
  Indexes all reactive irng’s. It is used when accessing the m_kValues matrix
  in reactive.F.

• xq, wq:
  Dummy parameters for the ewdfequad() subroutine.

• m_kValues(countReactive, maxquad, maxnecr, maxk):
  Matrix that stores k values for every quadrature point of the elements with reactive
  nodes.

The hypo() subroutine has been modified. Using the elementSurfaceElementType() subroutine
the appropriate surface element type is found based on the volume element type and
stored under elemtypeh parameter. The elemtypeh is used as input for the ewdfequad() to
find the amount of quadrature points (maxquad). The rngdreactive matrix is used to
find the amount of reactive irngs (countReactive). This enables memory allocation for the
v_irngMapping . What follows is a loop which parses all reactive irngs looking for the one
with largest number of elements. Each reactive irng number is stored one by one in the
v_irngMapping . Finally the m_kValues is allocated and passed later together with the
v_irngMapping to the newd() subroutine.

reactive.F
The blkfluxreactive() subroutine has been modified. The declarations of v_irngMapping
and m_kValues matrices were added. The v_irngMapping is parsed in order to find the
location index of the current irng (vectorIndex) which at the same time indicates appropriate data location in the m_kValues. If both its and iit are 1, the coordinates of the quadrature points are calculated and the ewdlex function is called to find the k values at these points. The calculated values are written to the m_kValues. If both its or iit are different from 1, the k values are read from the m_kValues and used later for flux calculations.

blkrhs.F
The blkrhs() subroutine has been modified in such a way that it includes declarations of v_irngMapping and m_kValues matrices. The call to blkfluxreactive() subroutine was modified.

newd.F
The declarations of v_irngMapping and m_kValues matrices have been introduced. The call to blkrhs() subroutine was modified.

2.3 Simulation performance comparison

Location:  hac098@jugene.fz-juelich.de\LESZEK\PERFORMANCE_STUDY\n
In the following, the simulation performance on JUGENE is compared for the xns-waluga xns-stock and xns-seweryn-1.0 XNS versions. Both xns-stock and xns-seweryn-1.0 have proven to provide identical results. The previous, however, runs over 20 times slower in the case of Taylor Couette system and almost 12 times slower for Stagnation point flow analysis. Compared to xns-waluga, the xns-seweryn-1.0 achieves 77% in the case of Taylor Couette and 95% for Stagnation point flow analysis. Detailed simulation results are presented in the following table:

<table>
<thead>
<tr>
<th>TAYLOR COUETTE - 2040 PEs</th>
<th>Performance - ts/h</th>
</tr>
</thead>
<tbody>
<tr>
<td>xns-stock</td>
<td>79.47</td>
</tr>
<tr>
<td>xns-seweryn</td>
<td>1848.86</td>
</tr>
<tr>
<td>xns-waluga</td>
<td>2387.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STAGNATION POINT FLOW - 2048 PEs</th>
<th>Performance - ts/h</th>
</tr>
</thead>
<tbody>
<tr>
<td>xns-stock</td>
<td>36.27</td>
</tr>
<tr>
<td>xns-seweryn</td>
<td>431.67</td>
</tr>
<tr>
<td>xns-waluga</td>
<td>452.15</td>
</tr>
</tbody>
</table>

Taylor Couette simulation performance results for xns-stock can be depicted from the xns.log file

<table>
<thead>
<tr>
<th>Miscellaneous</th>
<th>= 91.12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>= 9.22</td>
</tr>
<tr>
<td>Output</td>
<td>= 118.80</td>
</tr>
<tr>
<td>Form LHS and RHS (fluid)(())</td>
<td>= 46.61</td>
</tr>
<tr>
<td>GMRES solver (fluid)(())</td>
<td>= 11.80</td>
</tr>
</tbody>
</table>
Matrix-vector product........(fluid)(^) = 38.02
Gather..........................(fluid)(^) = 22.45
Scatter.........................(fluid)(^) = 27.14
Alignment...............(fluid)(^) = 0.01
Miscellaneous................(fluid)(^) = 4338.57

Total.....( 79.47 ts/hr)............. = 4703.75

Taylor Couette simulation performance results for xns-seweryn-1.0 can be depicted from the xns.log file

Miscellaneous............................ = 92.29
Input.................................... = 9.92
Output................................... = 172.48
Form LHS and RHS.......(fluid)(^) = 46.31
GMRES solver......................(fluid)(^) = 12.23
Matrix-vector product..........(fluid)(^) = 38.18
Gather.........................(fluid)(^) = 22.39
Scatter........................(fluid)(^) = 27.12
Alignment.....................(fluid)(^) = 0.01
Miscellaneous................(fluid)(^) = 46.52

Total.....( 1848.86 ts/hr)............. = 467.46

Taylor Couette simulation performance results for xns-waluga can be depicted from the xns.log file

Miscellaneous............................ = 89.92
Input.................................... = 13.66
Output................................... = 100.19
Form LHS and RHS.......(fluid)(^) = 46.74
GMRES solver......................(fluid)(^) = 11.99
Matrix-vector product..........(fluid)(^) = 38.53
Gather.........................(fluid)(^) = 22.46
Scatter........................(fluid)(^) = 27.10
Alignment.....................(fluid)(^) = 0.01
Miscellaneous................(fluid)(^) = 2.48

Total.....( 2387.05 ts/hr)............. = 353.08

Stagnation point flow simulation performance results for xns-stock can be depicted from the xns.log file
2 Performance issue of xns-stock (Thrombosis model - shear dependent k-values)

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td>113.21</td>
</tr>
<tr>
<td>Input</td>
<td>22.43</td>
</tr>
<tr>
<td>Output</td>
<td>123.83</td>
</tr>
<tr>
<td>Form LHS and RHS (fluid)(~)</td>
<td>31.64</td>
</tr>
<tr>
<td>GMRES solver (fluid)(~)</td>
<td>37.82</td>
</tr>
<tr>
<td>Matrix-vector product (fluid)(~)</td>
<td>71.63</td>
</tr>
<tr>
<td>Gather</td>
<td>81.83</td>
</tr>
<tr>
<td>Scatter</td>
<td>89.84</td>
</tr>
<tr>
<td>Alignment</td>
<td>0.01</td>
</tr>
<tr>
<td>Miscellaneous (fluid)(~)</td>
<td>3657.54</td>
</tr>
<tr>
<td><strong>Total..( 36.27 ts/hr)...........</strong></td>
<td><strong>4229.76</strong></td>
</tr>
</tbody>
</table>

Stagnation point flow simulation performance results for xns-seweryn-1.0 can be depicted from the xns.log file

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td>179.49</td>
</tr>
<tr>
<td>Input</td>
<td>23.78</td>
</tr>
<tr>
<td>Output</td>
<td>383.28</td>
</tr>
<tr>
<td>Form LHS and RHS (fluid)(~)</td>
<td>126.85</td>
</tr>
<tr>
<td>GMRES solver (fluid)(~)</td>
<td>151.06</td>
</tr>
<tr>
<td>Matrix-vector product (fluid)(~)</td>
<td>286.58</td>
</tr>
<tr>
<td>Gather</td>
<td>326.61</td>
</tr>
<tr>
<td>Scatter</td>
<td>358.47</td>
</tr>
<tr>
<td>Alignment</td>
<td>0.03</td>
</tr>
<tr>
<td>Miscellaneous (fluid)(~)</td>
<td>84.76</td>
</tr>
<tr>
<td><strong>Total..( 431.67 ts/hr)...........</strong></td>
<td><strong>1920.90</strong></td>
</tr>
</tbody>
</table>

Stagnation point flow simulation performance results for xns-waluga can be depicted from the xns.log file

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td>177.27</td>
</tr>
<tr>
<td>Input</td>
<td>29.13</td>
</tr>
<tr>
<td>Output</td>
<td>102.00</td>
</tr>
<tr>
<td>Form LHS and RHS (fluid)(~)</td>
<td>127.57</td>
</tr>
<tr>
<td>GMRES solver (fluid)(~)</td>
<td>149.37</td>
</tr>
<tr>
<td>Matrix-vector product (fluid)(~)</td>
<td>286.62</td>
</tr>
<tr>
<td>Gather</td>
<td>327.63</td>
</tr>
<tr>
<td>Scatter</td>
<td>362.10</td>
</tr>
<tr>
<td>Alignment</td>
<td>0.02</td>
</tr>
<tr>
<td>Miscellaneous (fluid)(~)</td>
<td>20.59</td>
</tr>
<tr>
<td><strong>Total..( 452.15 ts/hr)...........</strong></td>
<td><strong>1582.31</strong></td>
</tr>
</tbody>
</table>
3 Concatenation of simulation results (Aortic flow)

3.1 Problem statement

The mxyz.space file contains a large number of 3-dimensional coordinates of nodes of a mesh that has been used for simulation of aortic flow. The data.out-space file contains sets of 4 resulting simulation values for each node. Both files have to be combined to provide all the information in one single file, which was required by Helmholtz Institute in order to do a comparison between simulation and PiV data.

3.2 Solution

Location: \home\seweryn \WORKBENCH\PROGRAMS\mixd2one\n
A simple Fortran application is the solution to the problem. As it uses the ewd libraries, a symbolic link has been established in the directory mentioned above. The current version of the application asks the user to specify the amount of nodes for which the data is to be combined and the name of the output data file. If the files exist in the current directory (the mxyz.space file containing the x,y,z coordinates and the data.out-space), the matrices are allocated for data from both files and finally the data is read. What follows is specification of the format in which data is to be written and the corresponding write operation:

```
format= "((E14.5, 6(3X, E14.5)))"
OPEN (UNIT=1, FILE=h_combined, ACCESS='SEQUENTIAL',STATUS='NEW')
do inn=1,nn
write(1,format) (combined(isd,inn), isd=1,7)
end do
CLOSE(1)
```
4 Conversion of the VRML 1.0 surface mesh data into mxyz format (Human heart project)

4.1 Problem statement

The Helmholtz surface mesh data is saved in VRML 1.0 format. The coordinates of the nodes should then be saved in the binary format in the same manner as used by X3w for input in the mxyz file. The VRML format does not specify directly the number of nodes that are part of the surface mesh.

4.2 Solution

4.2.1 Simple case

Location: \home\seweryn\WORKBENCH\PROGRAMS\vrml2mxyz-1.0\n
Execution: ./vrml2mxyz -g -tri <mesh_files>

A Fortran application provides solution to the problem. As it uses the ewd libraries, a symbolic link has been established in the directory mentioned above. The application is to be called in the following way: ./vrml2mxyz <-g option if debug required> -tri <mesh file name>. The current version of the application works only for meshes made of tetrahedral elements. The VRML file is parsed line by line for the "point" token. The following lines contain nodes coordinates. The first run counts the lines with the coordinates until the "[" token is found. Having the number of nodes, the VRML data file is parsed again and the corresponding coordinates are read and written to mxyz and minf files.

4.2.2 More complex case

Location: \home\seweryn\WORKBENCH\PROGRAMS\vrml2mxyz-1.1\n
Execution: ./vrml2mxyz -g -tri Later versions of the mesh data files received from the Helmholtz institute contain mesh information in three separate blocks of one file (inflow, outflow, wall). Some nodal information is to be found in two of these blocks simultaneously (as one node may be part of the e.g. inflow and wall mesh at the same time). These three
blocks of data have to be parsed, concatenated, doublets removed and finally written to a single output file.

The program starts by locating the three blocks of data and writing it into one matrix. In the following, the coordinates of the last node are taken and the rest of the matrix is parsed. If the same coordinates occur somewhere else in the matrix, the coordinates of the last node are being removed. The procedure continues for every node in the matrix:

!Coordinates have been read.
!The $xx$ matrix may contain nodes that are shared by 2 surfaces.

doubleCounter = 0
DO inn=1, nn
!take the element from the tail
DO inn2=1, (nn-inn)
IF( xx(1,inn2).EQ.xx(1,nn-inn+1)
 & .AND.xx(2,inn2).EQ.xx(2,nn-inn+1)
 & .AND.xx(3,inn2).EQ.xx(3,nn-inn+1) ) THEN
IF (debug) CALL ewdmsg("vrml2mxyz: Double surface element: ", (nn-inn+1))
IF (debug) CALL ewdmsg("vrml2mxyz: and: ", inn2)
doubleCounter = doubleCounter + 1
xx(1,nn-inn+1) = emptyLine
exit
END IF
END DO
END DO
5 Surface grid points displacement calculation (Human heart project)

5.1 Problem statement

Given the volume mesh file (in binary or FieldView format), its surface mesh file (in binary format) and the surface mesh file for the transformed mesh (in binary format), the displacement vector for the volume mesh should be found. The nodes from the first surface mesh have to appear in the same order in the second surface mesh.

5.2 Solution

5.2.1 Surface shift vector approach

Location: /home/seweryn/WORKBENCH/PROGRAMS/shiftVector
Execution: ./shiftVector

During the execution the applications prompts for the file names of the volume and both surface meshes. Additionally, the user has to specify the minf file name which contains the number of nodes in the surface mesh. If all these files are successfully found in the current directory, the matrices are allocated and the data read in.

The following algorithm calculates the displacement vector:

1. Choose the ith node from the first surface grid,
2. Find the node with the same coordinates in the volume mesh (save the node position index),
3. Subtract coordinates from those of the ith node in the second surface grid,
4. Write the displacement vector to the "mshift" file,
5. Assign -999's to corresponding volume node (so that later search skips this position).

Following assumptions have been made during the development of the algorithm:

- There is no -999 coordinate in the volume grid,
- The maximum allowed accuracy difference is 5.0E-5.

The resulting displacement and position vectors are saved in mshift and mposition files.
5.2.2 Volume shift vector approach

**Location:** \home\seweryn\WORKBENCH\PROGRAMS\shiftVector_volume

**Execution:** ./shiftVector

Volume shift vector is the modification of the approach described in the sub-chapter above in which the shift vector is calculated for all surface nodes and zero shift is assigned to the remaining volume nodes. As a result a file is obtained of the same size as the mxyz in which shift information is assigned to every node in the same order.

In addition, the information about the maximal shift found is displayed in a dynamic manner:

```
IF( SQRT(xyz_shift(1,inn_surface)*xyz_shift(1,inn_surface)
    & + xyz_shift(2,inn_surface)*xyz_shift(2,inn_surface)
    & + xyz_shift(3,inn_surface)*xyz_shift(3,inn_surface)) .GT.
    & ishift_max
    & )THEN

ishift_max = SQRT(xyz_shift(1,inn_surface)*xyz_shift(1,inn_surface)
    & + xyz_shift(2,inn_surface)*xyz_shift(2,inn_surface)
    & + xyz_shift(3,inn_surface)*xyz_shift(3,inn_surface))

inn_max = inn_volume
if (debug) write(0,fmt=999) ishift_max
END IF
```

5.2.3 The xns.in modifications

The xns.in file requires following additional information in order to the able to modify the initial mesh:

- **mshift mshift**  
  Specifies the name of the file with the shift vector information (volume shift vector approach).

- **moving on**  
  Indicates that the mesh is allowed to change.

- **surface_movement shift**  
  Specifies the type of movement.

- **rngxset 11111**  
  Specifies which irng’s are allowed to change their coordinates and in which directions these changes are possible.

- **smallshift 1.0**  
  If smaller that 1.0, splits the mesh movement between consecutive time steps. In each time step the coordinates are shifted by smallshift $\times$ shift.
5.2.4 The newxrigid.F file

Main changes concern the newxrigid.F file. The new coordinates are calculated in the following manner:

```
IF(its.NE.1.AND.iit.EQ.1) THEN
  DO inl=1,nnl
    inltop = inl
    mchanged(inltop) = 1.0
    z(xsd,inltop) = xshift(xsd,inltop)
    z(ysd,inltop) = xshift(ysd,inltop)
    IF (nsd.eq.3) then
      z(zsd,inltop) = xshift(zsd,inltop)
    END IF
  END DO
ENDIF
```

5.2.5 Simulation attempts - human heart project

The simulation input files have been prepared basing on the two heart mesh files that were received from the Helmholtz institute. The initial test case surface mesh was composed of 94248 nodes.

- **Mesh with 94248 nodes**
  When using the shiftVector application to generate the mshift file, the largest shift vector found, had the magnitude of 10.677355. It was associated with the node number 31. Trying to shift the node coordinates in one time step during the simulation, results in negative element Jacobian determinants (tangled mesh) already in the second time step (no shift calculations successful). In later investigations, the whole shift vector was introduced gradually, applying in each time step only 0.1 of the total shift. In such a case the mesh becomes tangled in third time step, meaning that 0.2 of the total shift can be simulated successfully. Further decrease of the applied shift per time step answers the question of how large displacement can be applied, but no improvement is observed. In each case the element that has the negative determinant can be found near (160.20, -150.56, 14.566) coordinates. Its node number is 115140. The initial heart mesh has been investigated at these coordinates showing very small elements in that neighborhood. The mesh composed of small elements can handle only small shifts without altering its correctness.

- **Mesh with 7395 nodes**
  For a less precise mesh (and larger elements) the simulator managed to perform 5 time steps resulting in 0.5 of the total shift. The element that has the negative determinant can be found near (159.11, -145.36, 17.121) coordinates which indicates the same position as for the larger mesh.
5 Surface grid points displacement calculation (Human heart project)

Investigations of the skewness of the elements of the grid show element quality near the above-mentioned coordinates which can be best depicted in figures 5.2.5 and 5.2.5.

From the figures above it can be seen that the quality of the elements near the (150,-145,15) coordinates (most of the indicated red elements) is poor. Further work should include attempts to improve their quality.
6 Simulation data post-processing

6.1 remove_spacetime_records script

**Location:** \home\seweryn\WORKBENCH\PROGRAMS\n
**Execution:** ./remove_spacetime_records <input_file> <output_file> <number_timeSteps+1> <number_ndf> <nn>

The script removes the spacetime records from the data file specified as input for visualization purposes. The script uses the dd command that shall be shortly described here. The dd command is used to copy and convert raw data. It accepts a number of parameters, the most important one being:

if=<input_file> - path of the input file,
of=<output_file> - path of the output file,
bs=<number_bytes2read> - sets the block size to be operated on,
skip=<number_blocks2skip> - skips every ith block of data in the input,
count=<number_blocks2copy> - number of blocks to be copied,
seek=<number_blocks2skip> - number of blocks to be skipped in the output file,

By the use of the dd command, it is possible to retrieve the mesh coordinates for a single time step (from the mxyz.all_space file):

```
#input file containing mesh coordinates for all time steps
if=mxyz.all_space
#output file containing mesh coordinates for a single time step
of=result_ts_$1
#block size: #coordinates * nn * realTypeSize
bs=$(3*191*8)
#ith time step
skip=$($1-1)

dd if=$if of=$of bs=$bs skip=$($1-1) count=1 seek=0
```
6.2 compareXyz application

Location: \home\seweryn\WORKBENCH\PROGRAMS\n
Execution: ./compareXyz

The program has been written to enable investigation of the surface mesh changes (correctness) after a simulation in which shift vector was used.

Required input from the user:

- modified mxyz file handle (binary),
- node number nn (integer value) - first nn nodes are taken for comparison purposes,
- original mxyz file handle (binary),
- mshift file handle that contains coordinates displacement,
- delta (real value) that describes maximal allowed deviation,
- shift × fraction (real value) that specifies the fraction of the shift (from the mshift data file) that should be implemented

The program parses nodal coordinates and filters out those nodes that belong to the surface mesh. For these nodes the coordinates from both mxyz files are compared against the shift fraction value. If the difference is greater than delta, the program terminates and the following information is printed to the shell:

- volume node number,
- original and modified coordinates,
- shift × fraction value,
- the difference between the modified and original coordinate values

6.3 find_xyz application

Location: \home\seweryn\WORKBENCH\PROGRAMS\n
Execution: ./find_xyz

The program helps to locate grid node numbers with coordinates similar to those specified by the user.

Required input from the user:

- mxyz file handle (binary)
- grid node number
- searched coordinates (x, y, z - real values) written in one line
• **delta** as maximal allowed deviation

The program parses all nodal coordinates comparing them with the ones passed by the user. If a match occurs (maximal allowed deviation is not exceeded) the coordinates are printed to the screen together with the volume node number and the sum of deviations.

### 6.4 movePointCoordinates application

**Location:** `\home\seweryn\WORKBENCH\PROGRAMS\`

**Execution:** `./moveCoordinates -g <mxyz.space_file_handle>`

The program changes the coordinates of the specific surface nodes of the mesh.

Required input from the user:

- grid node number (according to the mesh file passed as input)
- **delta** as maximal allowed deviation used to filter out those nodes that belong to the surface grid

The program parses all nodal coordinates filtering out those nodes that belong to the surface of the cylinder with the radius \( r_0 = 0.75 \) (only x and y coordinates are considered, delta is used here). From the cylinder surface nodes the ones that satisfy:

\[
15.40 \leq |z| \leq 15.40 + 4.0
\]

are being selected. For these the new radius is being calculated according to the following formula:

\[
r_{\text{new}} = r_0 - \frac{0.35}{0.75} \times \frac{r_0}{4} \times \left[ 1 - \cos \left( 2 \times \pi \times \left( \frac{|z| - 15.40}{4.0} \right) \right) \right]^2
\]

The change of node positions is calculated as follows:

\[
x = x \times \left( \frac{r_{\text{new}}}{r_0} - 1.0 \right)
\]

\[
y = y \times \left( \frac{r_{\text{new}}}{r_0} - 1.0 \right)
\]

\[
z = 0.0
\]

The program counts the amount of surface nodes found and the number of surface nodes shifted and prints this information to the shell upon completion. Also, the `mshift.space` and `mshift` files are generated.
6.5 Visualization in Ensight

The following steps are required to visualize the mesh geometry after the ith time step of the simulation:

- **dd:**
  Run the dd command to separate the mesh coordinates for the required time step.

- **mixd2ensight.in:**
  Modify the **mixd2ensight.in** file:

  ```
  debug on
  steady off
  source minf.space
  elemtype tet
  #the file containing required mesh coordinates
  mxyz result_ts_20
  mien mien
  mrng mrng
  data data.all_space
  nrec 0
  nsd 3
  ndf 3
  nen 4
  nrng 11
  srng 11 wall
  srng 10 wall
  srng 5 inflow
  srng 4 outflow
  ```

- **mixd2ensight:**
  Run the **mixd2ensight** command locally, the **out.case** will be generated.

- **ensight90:**
  Run the **ensight90** command locally.

6.6 The mixd to *.grd conversion tool

The **main3D** application can be used to convert the mesh information saved in the **mixd** back to the ***.grd** format which can be later investigated in Gridgen. During execution, following input has to be specified:

- **Inf. - Datei:**
  minf.space
6 Simulation data post-processing

- Koord. - Datei: mxyz.space
- mien:
  mien
- RNG:
  mrng
- Outptu - Datei:
  mesh.grd
7 Shear stress calculation and output

7.1 Problem statement

The shear stress at the nodes of the mesh should be calculated and periodically printed out for investigation purposes.

7.2 Solution

7.2.1 Background

The stress tensor is given by:

\[ T = \mu \times \left( \begin{array}{ccc} 2 \times \frac{\partial u}{\partial x} & \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial z} \right) & \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial z} \right) & 2 \times \frac{\partial v}{\partial x} & \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) & \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) & 2 \times \frac{\partial w}{\partial x} \end{array} \right) \]  \hspace{1cm} (7.1)

or shortly:

\[ T = \left( \begin{array}{ccc} t_{11} & t_{12} & t_{13} \\ t_{21} = t_{12} & t_{22} & t_{23} \\ t_{31} = t_{13} & t_{32} = t_{23} & t_{33} \end{array} \right) \]  \hspace{1cm} (7.2)

In case of 2 dimensional cases this can be reduced to:

\[ T = \mu \times \left( \begin{array}{ccc} 2 \times \frac{\partial u}{\partial x} & \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & 2 \times \frac{\partial v}{\partial y} \end{array} \right) \]  \hspace{1cm} (7.3)

or:

\[ T = \left( \begin{array}{cc} t_{11} & t_{12} \\ t_{21} = t_{12} & t_{22} \end{array} \right) \]  \hspace{1cm} (7.4)
The second invariant of stress tensor is calculated as:

\[ II_T = -\frac{1}{2} \times (t_{11}^2 + t_{22}^2 + t_{33}^2 + 2 \times t_{12}^2 + 2 \times t_{13}^2 + 2 \times t_{23}^2) \]  

(7.5)

or for the 2D case:

\[ II_T = -\frac{1}{2} \times (t_{11}^2 + t_{22}^2 + 2 \times t_{12}^2) \]  

(7.6)

The shear stress is given by the following equation:

\[ \tau_{\text{scalar}} = (-II_T)^{\frac{1}{2}} = \left( \frac{1}{2} \times T : T \right)^{\frac{1}{2}} \]  

(7.7)

### 7.2.2 XNS

The stress tensor calculation is already implemented in `recoverstress` subroutine (`recover.F` file). It can be activated by setting the `recover on` or `rechot on` flags in the `xns.in` file. The `t[nde,nnl]` matrix contains the corresponding node values of the stress tensor. The calculation of the shear stress is achieved by introduction of the `si` matrix:

```fortran
real *8 :: si(nnl)
p没见过 (siptr, si)
```

The matrix is initialized and all its values nullified:

```fortran
siptr = ewdrealloc(siptr, nnl*fsize)
call ewdzero(nnl, si, 1)
```

What follows is the shear stress calculation for each node:

```fortran
DO inl=1, nnl
   si(inl)= t(1,inl)**2 + t(3,inl)**2 + 2*t(2,inl)*t(2,inl)
   IF(nsd.EQ.3) THEN
      si(inl) = si(inl) + t(6,inl)**2 + 2*t(4,inl)*t(4,inl)
      + 2*t(5,inl)*t(5,inl)
   END IF
   si(inl) = DSQRT(0.5*si(inl))
END DO
```

What remains is the periodic output of the shear stress for each node at the end of each time step which is realized in `statejotsi` subroutine. Following piece of code writes the `ss.all` file and sets the appropriate offset for the next write operation:
call ewdwritefloat(dsi, nnc, "ss.all", djotsoff, .false., dbio)
djotsoff = djotsoff + nn

### 7.2.3 XNS version history for shear stress implementation

- **xns-seweryn-3.0**: first implementation of shear stress calculation
- **xns-seweryn-3.1**: improved shear stress periodic output implementation
- **xns-seweryn-3.2**: fixed memory allocation problems for JuGene environment (\texttt{mg}, \texttt{zg} matrices)
- **xns-seweryn-3.3**: added \texttt{smallshift<real>} parameter to \texttt{xns.in} to enable smaller shift steps
- **xns-seweryn-3.4**: further improvements of the shear stress periodic output implementation
8 Time approximated factor for rngdexp and rngdrot expressions

8.1 Problem statement

The *rngdexp* and *rngdrot* expressions are to be extended by a time varying factor. It is to be specified in the *xns.in* file in a comfortable and transparent manner.

8.2 Solution

8.2.1 XNS

Modified: parseinput.F, genbc.F

Following syntax is required in order to use the *fapprox* feature:

```
fapprox <irng> <idf> <napprox>
approx <tupper> <approx expression> //F1
approx <tupper> <approx expression> //F2
.
.
approx <tupper> <approx expression> //Fnapprox
```

The parser implemented in the *parseinput.F* activates the feature whenever the *fapprox* keyword is found in the *xns.in* file. The same line should include the *rng* and *df* numbers (the feature can be used for a single *rng* – *df* pair) as well as the number of functions that are used. Following *napprox* lines have to begin with the *approx* keyword which is followed by the upper time instance (for which the given approximation function should be used) and the expression that can be a function of time or constant. The lower time instance is 0.0 for the first function (f1) or tupper_{n−1} for f_n. Whenever the *tupper_{napprox}* is reached during simulation, all the *tupper* values are updated and the approximation procedure restarted.

8.2.2 XNS version history for fapprox implementation

*xns-seweryn-4.5* - the most recent, working version
9 Shear stress dependent reactivity rates

9.1 Problem statement

The reactivity rates should depend on the shear stress values obtained in chapter 7.

9.2 Solution

9.2.1 XNS

Modified: parseinput.F, reactive.F

Following syntax is required in order to use shear stress dependent reactivity rates (additional requirements from chapter 7 have to be met as well):

\[
\text{ssexp} \ <\text{irng}> \ <\text{ik1}> \ <\text{minfty1}> \ <\text{ssexp1}>
\text{ssexp} \ <\text{irng}> \ <\text{ik2}> \ <\text{minfty2}> \ <\text{ssexp2}>
\text{ssexp} \ <\text{irng}> \ <\text{ik3}> \ <\text{minfty3}> \ <\text{ssexp3}>
\]

The \text{ssexp} can be any function of available variables or (and) a function of \text{STRESS} parameter. The reactivity rates are evaluated for every quadrature point of an element within the \text{blkfluxreactive} subroutine.

9.2.2 XNS version history

\text{xns-seweryn-5.3} - the most recent, working version
10 Mechanical platelet activation due to high shear stress values

10.1 Problem statement

Mechanical platelet activation due to high shear stress values should be implemented.

10.2 Solution

10.2.1 XNS

Modified: parseinput.F, blkadvndst.F

Following syntax is required in order to use mechanical platelet activation feature (additional requirements from chapter 7 have to be met as well):

\[
\text{sscritical} <\text{sscritical}> <\text{ssconvf}> <\text{coeff1-coeff7}>
\]

The activation rate of platelets due to mechanical shear stress \((k_{sa})\) is approximated by the following dimensionless formula:

\[
k_{sa} = \text{coeff1} + \text{coeff2} ||\dot{\gamma}|| + \text{coeff3} ||\dot{\gamma}||^2 + \text{coeff4} ||\dot{\gamma}||^3
+ \text{coeff5} ||\dot{\gamma}||^4 + \text{coeff6} \ln (\text{coeff7} ||\dot{\gamma}||),
\]

\[
||\dot{\gamma}|| = \frac{\text{STRESS}}{\text{sscritical ssconvf}}.
\]

The \text{sscritical} parameter describes the critical concentration of ADP. The values are evaluated for each quadrature point of the element. Another implementations approach would include string expressions and lex parses but this would highly deteriorate the XNS performance which has already been observed and investigated in chapter 2.

10.2.2 XNS version history

xns-seweryn-6.0 - the most recent, working version
11 Previous XNS modifications

11.1 Changes introduced by Christian Waluga till 2008-10-13 (version xns-waluga)

Changes in ewd:

date unknown:

- ewdlex.i, ewdlex.y:
  Christian implemented three new variables to use in expressions in the xns.in file: r, phi and [i]. r is the radius calculated from \( r = \sqrt{x^2 + y^2} \) and phi accordingly as \( \text{atan2}(y,x) \). Be aware that these functions only make sense in the XY-layer! By using \([i]\) one can refer to the i-th degree of freedom.

Changes in XNS:

2008-10-13:

- hypo.F:
  Fixed input of s.all in restart-mode.

2008-10-06:

- blkadvndst.F:
  Reordered loops in blkadvndst to reduce cache misses.

- parseinput.F, initialize.F, global.h, blkadvndst.F, reactive.F:
  Removed discontinuity capturing from N-DOF code, (removed: dc.F)

- hardcoded reaction terms for platelet model (modified: ).

2008-10-01:

  Implemented adherent platelet 'height'.

2008-09-22:

  Made name of s.all customizable.

- reactive.F, blkrhs.F:
  Solved little problem with big consequences.
11 Previous XNS modifications

2008-09-09:
  • blkadvndst.F:
    Moved computation of discontinuity coefficient to dc.F. Created: dc.F.

2008-09-08:
  • global.h, initialize.F, parseinput.F, genvars.F:
    Added saveambient option.

2008-08-29:
  • blkadvndst.F:
    Tuned blkadvndst and converted it to F90.

2008-08-18:
  • genvars.F:
    Tuned genu for steady cases.
  • hypo.F, reactive.F:
    Improved linearization of surface platelet coverage integral.
  • blkrhs.F:
    Fixed zero-initialisation error in blkrhs that caused the reactive wall BCs not to work properly on JUGENE.

2008-08-14:
  • reactive.F, hypo.F, initialize.F, file.h:
    Implemented periodic output of surface coverage \( S(\vec{x}, t) \) to file s.all.

2008-07-18:
    Implemented reactive wall boundary condition (added: reactive.F)

2008-07-08:
  • tested instationary ambient input file feature on JUGENE
    (testcase: taylor_couette_hex_3dFine/20080710-001).

2008-07-08:
  • fixed bug in parallel version of ambient file input (see 2008-06-27).

2008-06-30:
  • blkadvndst.F:
    Extended discontinuity capturing to arbitrary number of DOF (modified: ). Has to be tested! Set tau_dc_factor 0.0 to deactivate.

2008-06-27:
11 Previous XNS modifications

- genvars.F:
  Extended ambient parameter treatment in genu to read a steady (ambient steady) or unsteady (ambient unsteady) velocity field from the file specified in daux.

2008-06-24:
- blkadvndst.F:
  Extended stabilization terms to arbitrary number of DOF.

2008-06-21:
- ewdlex.l, ewdlex.y, ewdparse.c:
  Implemented polar coordinate variables r, phi and DOF-array-operator [i] in ewdlex.

2008-06-19:
- copied blkadvsd.F to blkadvndsd.F
- newd.F:
  Replaced call to blkadvst with blkadvndst.
- blkadvndst.F:
  Extended blkadvndst to arbitrary number of DOF.

11.2 Changes introduced by Sebastian Stock till 2009-04-30 (version xns-stock)

2009-04-30:
- blkadvndst.F:
  Found little bug with big consequences in blkadvndst.F. Modifications described in the source code.
- parseinput.F:
  Some finishings: keyword reactive is obsolete now. RNG is known as reactive if a kexp is defined for this rng. The \(<M>\)-parameter is to be given within the kexp-term. Usage: kexp <irng> <kdo> <M> <expr>.

2009-04-11:
- reactive.F:
  Improvement in genkratemask: changed krs2(nn1) array to kml(maxk,nn1) array. Accordingly replaced krs2, kas and kaa by kml(ik,in1).
- reactive.F:
  Replaced krs2, kas and kaa in subroutine blkfluxreactive by ki(ik) which stores the k-rates for every element.
11 Previous XNS modifications

2009-04-08:

- **global.h, initialize.F, parseinput.F:**
  Removed krs2 again, therefor changed kexp to kexp(mrng,maxk). Introduced ik and maxk in global.h. Finally made changes in parseinput.F in order to read different expressions for $k_{rp}$, $k_{ap}$ and $k_{aa}$ for every rng. As well, for a better understanding in the source code, you can call the different $k$-rates by their DOF: krsdof, kasdof and kaadof.

2009-03-27:

- **global.h, initialize.F, parseinput.F:**
  Introduced new keyword kexp to define in xns.in for each rng. Up to now only for $k_{rs}$ which is known as krsexp(irng).

- **reactive.F, hypo.F:**
  Added new subroutine genkratemask. Values of krsexp are stored in array krs2, according to array rml for every reactive node on the layer. Subroutine is called from hypo.F.

- **reactive.F:**
  Made change in subroutine updates: substituted krs by krs2.

- **reactive.F:**
  Changed krs to krs2 in subroutine blkfluxreactive. Therefore krs2 is calculated from krsexp for every element (more or less copied from subroutine blkflux).