Exceptional service in the national interest

Sandia National Laboratories

XFEM Development in the ALEGRA Code

Tom Voth, Irina Tezaur, John Niederhaus

Sandia National Laboratories

ALEGRA Winter Workshop December 10-12, 2019

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2019-14900PE

Summary of CY2019 progress

A focus on XFEM robustness!

- Revisit algorithms and assumptions to improve XFEM robustness:
  - Mass conservation equation/density update (T. Voth)
  - Void insertion (T. Voth)
  - Ensure solvability of contact linear system (I. Tezaur)
- Assess state of 2D XFEM code with ARL problems of interest (J. Neiderhaus)

Summary of CY2019 progress

A focus on XFEM robustness!

- **Revisit algorithms and assumptions to improve XFEM robustness:**
  - **Mass conservation equation/density update (T. Voth)**
  - Void insertion (T. Voth)
  - Ensure solvability of contact linear system (I. Tezaur)
- Assess state of 2D XFEM code with ARL problems of interest (J. Neiderhaus)

Mass conservation equation

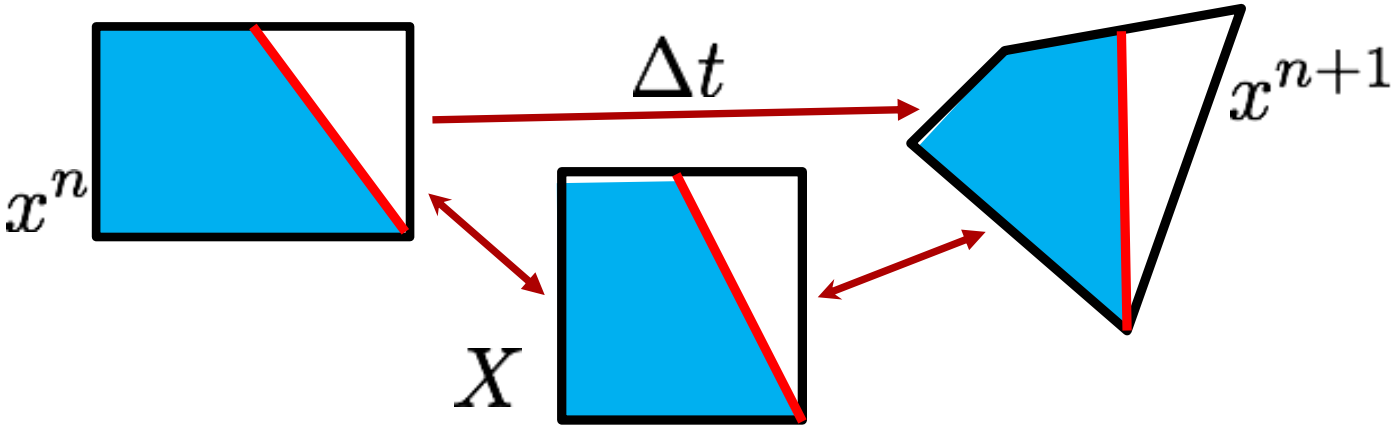
- XFEM mass conservation used UFEM constant volume fraction assumption:
  - Volume fraction is assumed to be unchanged during Lagrangian Step
  - Final (end of Lagrangian step) material volume is simply the product of element volume and material volume fraction.
  - Density update is then identical to that for UFEM:

$$\boxed{\rho^{n+1} = \rho^{n} V_e^{n} / V_e^{n+1}}$$

- Issues include:
  - Required scaling remap intersection volumes (and masses, momentum, etc).
  - Can (often?) results in artificial compression or expansion of materials and bad states.
- But, there's a better, more consistent way…

Mass conservation equation (cont'd)

- Discard constant volume fraction assumption all together.
- Final material volume is given the interface description at the end of the Lagrangian step and easily computed.



- Density update then becomes: $\rho^{n+1} = \rho^n V_{e,mat}^n / V_{e,mat}^{n+1}$
- Result is elimination of artificial compression / expansion and more robust material state updates.
- Uncovered an algorithmic issue with void insertion for XFEM

Summary of CY2019 progress

A focus on XFEM robustness!

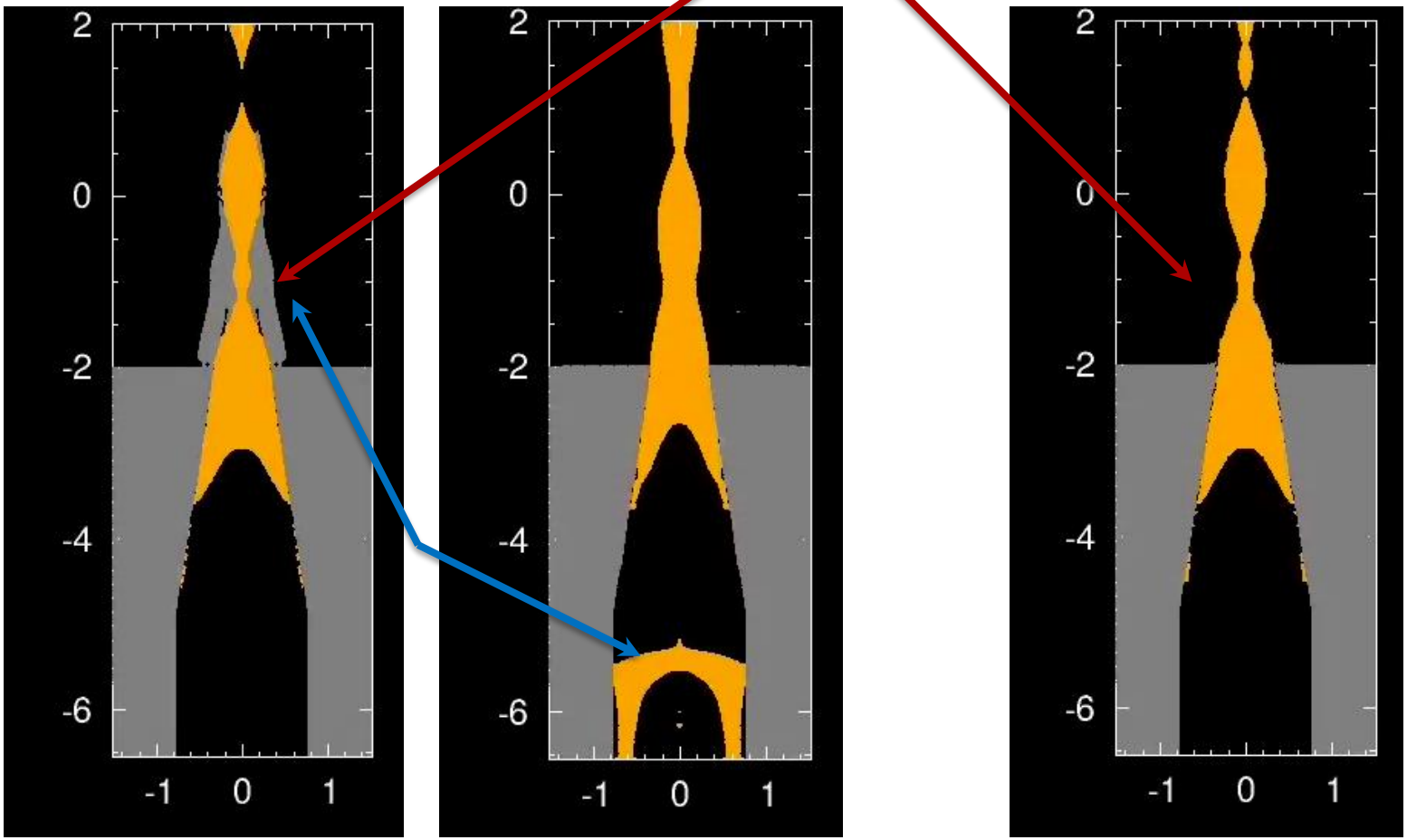
- **Revisit algorithms and assumptions to improve XFEM robustness:**
  - Mass conservation equation/density update (T. Voth)
  - **Void insertion (T. Voth)**
  - Ensure solvability of contact linear system (I. Tezaur)

- Assess state of 2D XFEM code with ARL problems of interest (J. Neiderhaus)

Void insertion impacts robustness

without void insert

with void insert



UFEM

XFEM

UFEM

Void insertion

- UFEM void Insertion algorithm was unchanged for XFEM:
  - UFEM algorithm adapted for XFEM.
  - Happens during Lagrangian step, reducing material volume and inserting void to take up remaining element volume.

- … but XFEM computes interfaces at the start (end) of the Lagrangian (Remap) step:
  - Used to prevent material interpenetration (contact enforcement).
  - Used for material by material remap.

- … and XFEM algorithms (remap and contact) assume the interface description does not change during the Lagrangian step (deformations allowed but material/master-element description is unchanged)

Void insertion (cont'd)

- However, void insertion violates XFEM's unchanged interface assumption:
  - Void is added that is not "known" to XFEM. Innocuous by itself.
  - "Fractured" material in cell does not know its volume has changed (interface location has not been changed).
  - As "fractured" material state has changed but XFEM is unaware of its volume change, we have a mass source/sink.

- Why did this (appear) to work (e.g. we're conserving mass)?
  - As noted, XFEM has been enforcing constant volume fraction through the Lagrangian step (CONSTANT VOLUME FRACTION algorithm).
  - Void insertion scales volume fraction to insert void.
  - Remap volumes (swept or intersected) scaled to ensure remapped volumes same as those computed with volume fraction (constant plus void insertion mods) at the end of the Lagrangian step. Results in conservation and hides Void Insertion "bug".

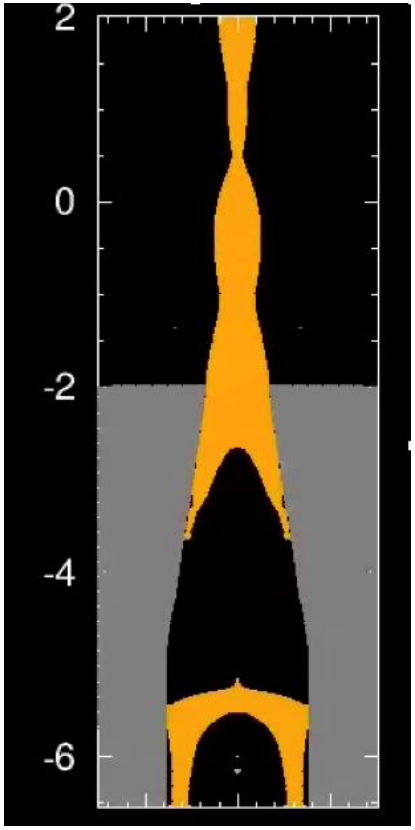
Void insertion (cont'd)

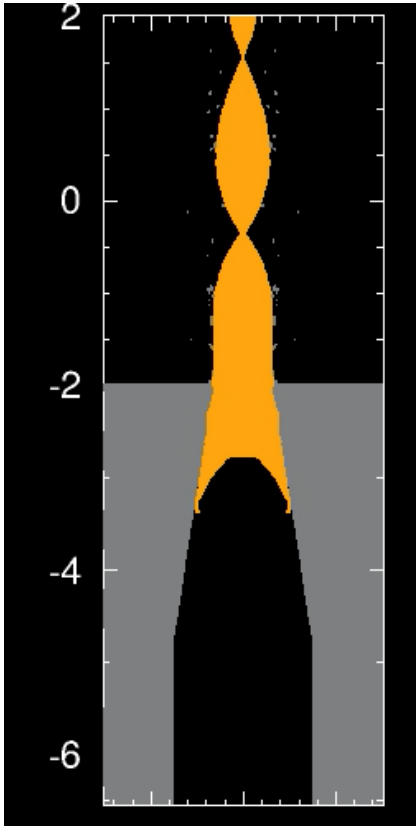
- Requirements for fix?
  - Void insertion cannot be part of the Lagrangian step.
  - Void insertion must occur while enrichments are in flux / being created and before interfaces are finalized.

- Fixed by moving XFEM's occurrence of void insertion to during remap step and before next Lagrangian step.
  - Approach suggested by M. Wong (original author of void insertion code).
  - Essentially hijack UFEM's void insertion clean-up that occurs after remap.
  - Void insertion for XFEM now only happens in remap where interface description is allowed to change.

Void insertion (cont'd)

XFEM results for DTE problem …





… without void Insertion. … with void Insertion.

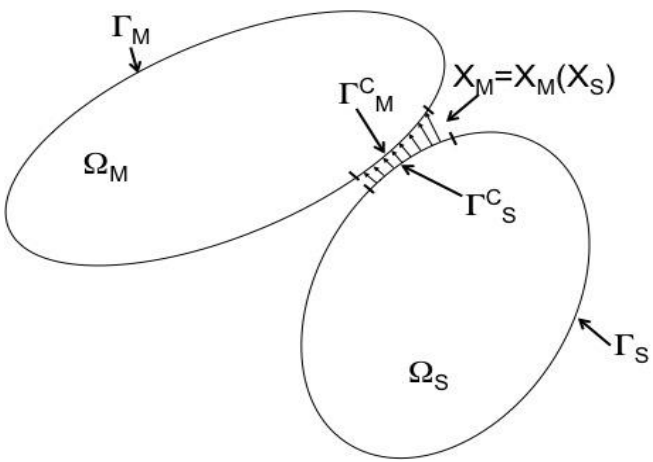
Summary of CY2019 progress

A focus on XFEM robustness!

- **Revisit algorithms and assumptions to improve XFEM robustness:**
  - Mass conservation equation/density update (T. Voth)
  - Void insertion (T. Voth)
  - **Ensure solvability of contact linear system (I. Tezaur)**

- Assess state of 2D XFEM code with ARL problems of interest (J. Neiderhaus)

XFEM contact problem



- **Contact constraint** enforces **gap rate** to be **zero** when **inter-penetration** between two surfaces is detected (right figure):

$$\boldsymbol{g}_{\text{rate}} = \boldsymbol{G}\boldsymbol{v}^{n+1/2} = \boldsymbol{0}$$

- Contact constraint is enforced via **Lagrange multipliers** $\boldsymbol{\lambda}$, which gives rise to a **KKT system** for momentum balance + contact constraint:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{a}^n + \boldsymbol{f}^{int} + \boldsymbol{f}^{ext} + \boldsymbol{G}^T\boldsymbol{\lambda} &= \boldsymbol{0} \\ \boldsymbol{G}\boldsymbol{v}^{n+1/2} &= \boldsymbol{0} \end{aligned}$$

- To be solvable, the discretization must satisfy the ***inf-sup*** **(LBB) condition**.

The ***XFEM contact implementation*** in ALEGRA does **not** ensure *a priori* satisfaction of the *inf-sup* condition!

➢ This is one source of **robustness issues** with XFEM!

Discrete *inf-sup* (LBB) condition*

$$\boldsymbol{M}\boldsymbol{a}^{n} + \boldsymbol{f}^{int} + \boldsymbol{f}^{ext} + \boldsymbol{G}^{T}\boldsymbol{\lambda} = \boldsymbol{0}$$
$$\boldsymbol{G}\boldsymbol{v}^{n+1/2} = \boldsymbol{0}$$

- Three conditions for ***discrete inf-sup stability:***
  - ☐ The mass matrix $\boldsymbol{M}$ must be non-singular.
  - ☐ The constraint matrix $\boldsymbol{G}$ cannot have more rows than the mass matrix $\boldsymbol{M}$.
  - ☐ The constraint matrix $\boldsymbol{G}$ must have full row rank (contact constraints must be linearly independent).

* Necessary (but not always sufficient) condition for *inf-sup* stability at the continuous level.

Discrete *inf-sup* (LBB) condition*

$$\boldsymbol{M}\boldsymbol{a}^{n} + \boldsymbol{f}^{int} + \boldsymbol{f}^{ext} + \boldsymbol{G}^{T}\boldsymbol{\lambda} = \boldsymbol{0}$$
$$\boldsymbol{G}\boldsymbol{v}^{n+1/2} = \boldsymbol{0}$$

- Three conditions for ***discrete inf-sup stability:***

  ☑ The mass matrix $\boldsymbol{M}$ must be non-singular.

  ☑ The constraint matrix $\boldsymbol{G}$ cannot have more rows than the mass matrix $\boldsymbol{M}$.

  ☒ The constraint matrix $\boldsymbol{G}$ must have full row rank (contact constraints must be linearly independent).

* Necessary (but not always sufficient) condition for *inf-sup* stability at the continuous level.

Lin. dept. constraints are a real problem!

Sandwich Problem

$$GM^{-1}G^T y = Gv^*$$

There are 140 singular/near-singular matrices out of 2659 total matrices.
--> Matrix matrix31.mm is singular with rank deficiency 2 and condition number Inf.
----> Matrix matrix31.mm has duplicate rows!
------> Identical Rows:
19 30
--> Matrix matrix32.mm is near-singular with rank deficiency 1 and condition number 1e16.
--> Matrix matrix37.mm is near-singular with rank deficiency 1 and condition number 5e16.
....

Lin. dept. constraints are a real problem!

Sandwich Problem

$$GM^{-1}G^T y = Gv^*$$

There are 140 singular/near-singular matrices out of 2659 total matrices.
--> Matrix matrix31.mm is singular with rank deficiency 2 and condition number Inf.
----> Matrix matrix31.mm has duplicate rows!
------> Identical Rows:
19 30
--> Matrix matrix32.mm is near-singular with rank deficiency 1 and condition number 1e16.
--> Matrix matrix37.mm is near-singular with rank deficiency 1 and condition number 5e16.
....

<u>Two remedies explored simultaneously:</u>

- ☐ Develop algorithm to ***detect*** and ***remove on-the-fly*** linearly dependent and nearly-linearly dependent constraints
- ☐ Understand cause of linearly dependent constraints and ***eliminate*** them ***at the source***

Lin. dept. constraints are a real problem!

Sandwich Problem

$$GM^{-1}G^T y = Gv^*$$

```
There are 140 singular/near-singular matrices out of 2659 total matrices.
--> Matrix matrix31.mm is singular with rank deficiency 2 and condition number Inf.
----> Matrix matrix31.mm has duplicate rows!
------> Identical Rows:
        19          30
--> Matrix matrix32.mm is near-singular with rank deficiency 1 and condition number 1e16.
--> Matrix matrix37.mm is near-singular with rank deficiency 1 and condition number 5e16.
....
```

<u>Two remedies explored simultaneously:</u>

- ☐ Develop algorithm to ***detect*** and ***remove on-the-fly*** linearly dependent and nearly-linearly dependent constraints
- ☐ Understand cause of linearly dependent constraints and ***eliminate*** them ***at the source***
  - ➢ On-the-fly detection algorithm expected to be needed to deal with ***nearly linearly dependent constraints***

Lin. dept. constraints are a real problem!

Sandwich Problem

$$GM^{-1}G^T y = Gv^*$$

```
There are 140 singular/near-singular matrices out of 2659 total matrices.
--> Matrix matrix31.mm is singular with rank deficiency 2 and condition number Inf.
----> Matrix matrix31.mm has duplicate rows!
------> Identical Rows:
        19          30
--> Matrix matrix32.mm is near-singular with rank deficiency 1 and condition number 1e16.
--> Matrix matrix37.mm is near-singular with rank deficiency 1 and condition number 5e16.
....
```

<u>Two remedies explored simultaneously:</u>

- ☑ Develop algorithm to ***detect*** and ***remove on-the-fly*** linearly dependent and nearly-linearly dependent constraints (→ QR algorithms!)
- ☐ Understand cause of linearly dependent constraints (→ bug in skinning alg.) and ***eliminate*** them ***at the source*** (→ TODO)
  - ➢ On-the-fly detection algorithm expected to be needed to deal with ***nearly linearly dependent constraints***

Full QR algorithm for contact solve

<u>Basic Idea:</u> (1) perform QR decomposition on $\boldsymbol{M}^{-1/2}\boldsymbol{G}^T$;
(2) remove linearly dept./nearly-linearly dept. constraints from $\boldsymbol{R}$;
(3) solve upper triangular full-rank system given by $\widetilde{\boldsymbol{R}}$.

1: Select a QR tolerance $\epsilon \geq 0$ (suggested default value: $\epsilon = 1.0 \times 10^{-10}$).
2: Form the scaled constraint matrix $\mathbf{G}_m \equiv \mathbf{G}\mathbf{M}^{-1/2}$.
3: Compute the QR decomposition of the constraint matrix $\mathbf{G}_m^T = \mathbf{Q}\mathbf{R}$, where

$$\mathbf{Q} \equiv \begin{pmatrix} \mathbf{Q}_1, & \mathbf{Q}_2 \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{pmatrix}.$$

4: Find the set of indices $\{i\}$ for which $|R_1(i,i)| < \epsilon$.
5: Zero out the rows in the set $\{i\}$ of $\mathbf{R}_1$ and the corresponding entries of $\mathbf{b} \equiv \mathbf{M}^{1/2}\mathbf{v}^*$.
6: Set $R_1(i,i) = 1.0$ for the rows that were zeroed out. Call the resulting matrix $\tilde{\mathbf{R}}_1$.
7: Solve $\tilde{\mathbf{R}}_1\boldsymbol{\lambda} = \mathbf{Q}_1\mathbf{b}$ for the Lagrange multipliers.

Sandwich problem

	Aztec tol = 1e-5	Aztec tol = 1e-6	Aztec tol = 1e-7	Aztec tol = 1e-8	Aztec tol = 1e-9	Aztec tol = 1e-10	Aztec tol = 1e-11
No QR	died	died	died	died	died	died	finished
Full QR, qr tol = 1e-6	finished	finished	finished	finished	not run	not run	not run

QR algorithm makes XFEM solver much more ***robust***!

Sandwich problem

	Aztec tol = 1e-5	Aztec tol = 1e-6	Aztec tol = 1e-7	Aztec tol = 1e-8	Aztec tol = 1e-9	Aztec tol = 1e-10	Aztec tol = 1e-11
No QR	died	died	died	died	died	died	finished
Full QR, qr tol = 1e-6	finished	finished	finished	finished	not run	not run	not run

QR algorithm makes XFEM solver much more ***robust***!

	# cycles	# linear iters	# of "numerical loss of precision"s
No QR, Aztec tol = 1e-11	3571	185,435	751
Full QR, qr tol = 1e-6, Aztec tol = 1e-5	3511	37,723	11

Moreover, there are ***5×*** ***fewer linear iterations**** w/ QR!

* Note that the current implementation of QR in ALEGRA uses a direct solver. Results with iterative solver are shown here for illustrative purposes.

Full QR algorithm for contact solve

<u>Basic Idea:</u> (1) perform QR decomposition on $\boldsymbol{M}^{-1/2}\boldsymbol{G}^T$;
(2) remove linearly dept./nearly-linearly dept. constraints from $\boldsymbol{R}$;
(3) solve upper triangular full-rank system given by $\widetilde{\boldsymbol{R}}$.

1: Select a QR tolerance $\epsilon \geq 0$ (suggested default value: $\epsilon = 1.0 \times 10^{-10}$).
2: Form the scaled constraint matrix $\mathbf{G}_m \equiv \mathbf{G}\mathbf{M}^{-1/2}$.
3: Compute the QR decomposition of the constraint matrix $\mathbf{G}_m^T = \mathbf{Q}\mathbf{R}$, where

$$\mathbf{Q} \equiv \begin{pmatrix} \mathbf{Q}_1, & \mathbf{Q}_2 \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{pmatrix}.$$

4: Find the set of indices $\{i\}$ for which $|R_1(i,i)| < \epsilon$.
5: Zero out the rows in the set $\{i\}$ of $\mathbf{R}_1$ and the corresponding entries of $\mathbf{b} \equiv \mathbf{M}^{1/2}\mathbf{v}^*$.
6: Set $R_1(i,i) = 1.0$ for the rows that were zeroed out. Call the resulting matrix $\tilde{\mathbf{R}}_1$.
7: Solve $\tilde{\mathbf{R}}_1\boldsymbol{\lambda} = \mathbf{Q}_1\mathbf{b}$ for the Lagrange multipliers.

☺ ***Pro:*** contact system will be nonsingular, with smaller condition number*.

* $\kappa(\widetilde{\boldsymbol{R}}) \leq \sqrt{\kappa(\boldsymbol{G}\boldsymbol{M}^{-1}\boldsymbol{G}^T)}$.

Full QR algorithm for contact solve

<u>Basic Idea:</u> (1) perform QR decomposition on $\boldsymbol{M}^{-1/2}\boldsymbol{G}^T$;
(2) remove linearly dept./nearly-linearly dept. constraints from $\boldsymbol{R}$;
(3) solve upper triangular full-rank system given by $\widetilde{\boldsymbol{R}}$.

1: Select a QR tolerance $\epsilon \geq 0$ (suggested default value: $\epsilon = 1.0 \times 10^{-10}$).
2: Form the scaled constraint matrix $\mathbf{G}_m \equiv \mathbf{G}\mathbf{M}^{-1/2}$.
3: Compute the QR decomposition of the constraint matrix $\mathbf{G}_m^T = \mathbf{Q}\mathbf{R}$, where

$$\mathbf{Q} \equiv \begin{pmatrix} \mathbf{Q}_1, & \mathbf{Q}_2 \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{pmatrix}.$$

4: Find the set of indices $\{i\}$ for which $|R_1(i,i)| < \epsilon$.
5: Zero out the rows in the set $\{i\}$ of $\mathbf{R}_1$ and the corresponding entries of $\mathbf{b} \equiv \mathbf{M}^{1/2}\mathbf{v}^*$.
6: Set $R_1(i,i) = 1.0$ for the rows that were zeroed out. Call the resulting matrix $\tilde{\mathbf{R}}_1$.
7: Solve $\tilde{\mathbf{R}}_1\boldsymbol{\lambda} = \mathbf{Q}_1\mathbf{b}$ for the Lagrange multipliers.

☺ ***Pro:*** contact system will be nonsingular, with smaller condition number*.

☹ ***Con:*** computing QR decomp. is expensive – total CPU time may go up.

* $\kappa(\widetilde{\boldsymbol{R}}) \leq \sqrt{\kappa(\boldsymbol{G}\boldsymbol{M}^{-1}\boldsymbol{G}^T)}$.

Hybrid QR algorithm to reduce CPU time

1: Form ***original contact system*** given by $\boldsymbol{GM}^{-1}\boldsymbol{G}^{T}$.

2: Try to solve this system using ***Amesos*** (Trilinos direct solver).

3: If Amesos solve fails (which will happen if $\boldsymbol{GM}^{-1}\boldsymbol{G}^{T}$ is exactly singular), ***perform QR*** to ***identify/remove linearly dependent constraints*** and solve resulting system given by $\widetilde{\boldsymbol{R}}_1$.

- If we solve this using ***Amesos*** as well, we ***remove*** the linear solver tolerance parameter.

Hybrid QR algorithm to reduce CPU time

1: Form ***original contact system*** given by $\boldsymbol{GM}^{-1}\boldsymbol{G}^{T}$.

2: Try to solve this system using ***Amesos*** (Trilinos direct solver).

3: If Amesos solve fails (which will happen if $\boldsymbol{GM}^{-1}\boldsymbol{G}^{T}$ is exactly singular), ***perform QR*** to ***identify/remove linearly dependent constraints*** and solve resulting system given by $\widetilde{\boldsymbol{R}}_1$.

- If we solve this using ***Amesos*** as well, we ***remove*** the linear solver tolerance parameter.

<u>Discussion:</u>

😐 Exact ***linear dependencies*** are ***removed***.

- Algorithm can be extended to remove ***near linear dependencies***.

☺ ***CPU time*** comparable to or ***lower than*** CPU time with no QR.

☺ By using direct solve, Aztec ***linear solver tolerance*** parameter is ***no longer required***.

Usage of QR algorithms in ALEGRA

- Two ***new run-time options*** have been added to allow the user to specify which ***QR algorithm*** to use, and the ***QR tolerance*** parameter (see discussion beginning on p. 331 of the ALEGRA User Manual).

```
XFEM
   [MSPAIR, int, int]
   :
   [additional MSPAIR keywords]
   :
   [QUADRATURE POINTS int (2)]
   [OVERFILL TOLERANCE real (5.)]
   [AZTEC SET, int(0)]
   [QR ALGORITHM, {(NONE)|HYBRID|FULL}]
   [QR TOLERANCE, real (1.0e-10)]
   [SOLVER CHECK, {(ON)|OFF}]
   [MATERIAL BMATRIX, {ON|(OFF)}]
   [GHOST PENALTY STABILIZATION, {ON|OFF}]
   [GHOST PENALTY PARAMETER, real (0.01)]
   [GHOST PENALTY FREE EDGE, {ON|OFF}]
END
```

Note: AztecOO (Trilinos direct solver) is currently not a valid option with full or hybrid QR; if Aztec sublist is present in an input file using full or hybrid QR, the following error will be thrown:

```
ERROR:
UnsDynamics::Check_Errors_And_Set_Values
   AZTECOO is not a valid option when
using full or hybrid qr for contact
solve! The solver that is used is Amesos.
```

By using ***Amesos*** (Trilinos direct solver), we take the linear solver tolerance out of the equation!

- Please see ***Cup*** and ***Sandwich*** Training problems (Benchmarks/Training/AlegraXFEM) for example of ***full*** and ***hybrid QR*** usage, respectively.

Summary of CY2019 progress

A focus on XFEM robustness!

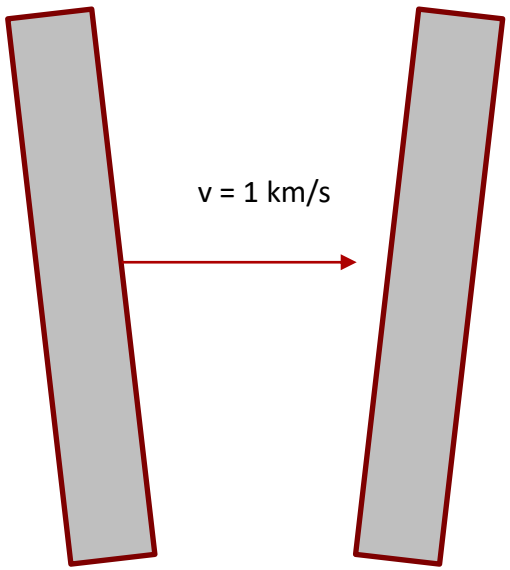
- Revisit algorithms and assumptions to improve XFEM robustness:
  - Mass conservation equation/density update (T. Voth)
  - Void insertion (T. Voth)
  - Ensure solvability of contact linear system (I. Tezaur)

- **Assess state of 2D XFEM code with ARL problems of interest (J. Neiderhaus)**

Test cases, November 2019

- For testing of XFEM after (a) change to density update and (b) addition of QR factorization algorithms.
  - Oblique plate slap (2 materials, metal)
  - Oblique cylinder impact on layered target (3 materials, metal and polymer)
  - APM2 onto layered aluminum (4/5? Materials, all metal)
  - PTI with cover plate (3 materials, including ceramic)
- These are all still at relatively coarse resolution, but could be refined for further testing.

Oblique plate slap problem

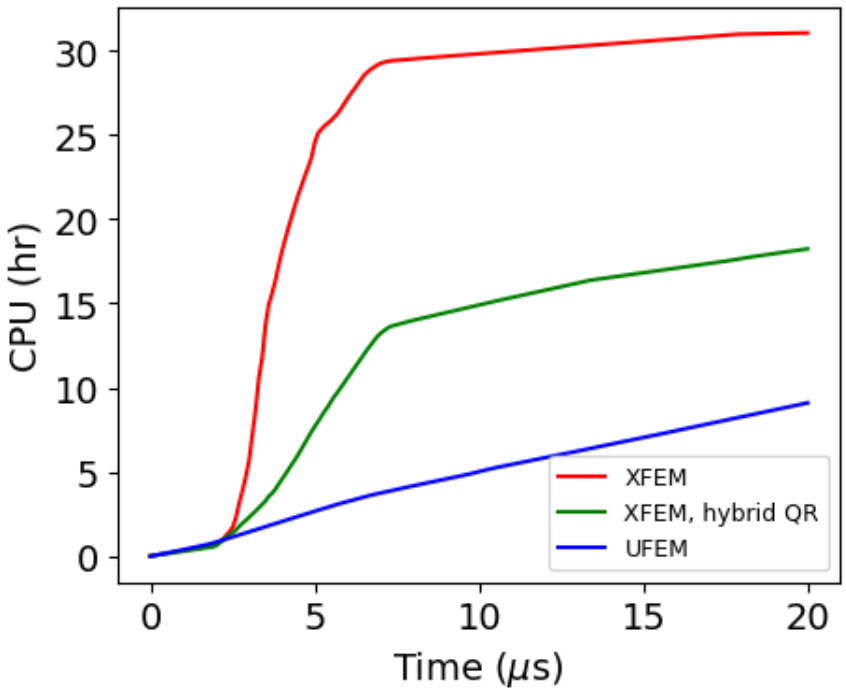


- Test originally created by Bob Doney
- Plates inclined in opposite directions at 2.5° each
- Both plates are aluminum
- EOS, strength, void insertion with force fracture
- Simulation should allow recoil and separation (and sliding, if plates are not inclined symmetrically).
- **225K elements**

→ *Last attempt to run this problem was in 2014! Persistently failed with Aztec iterations > maximum.*

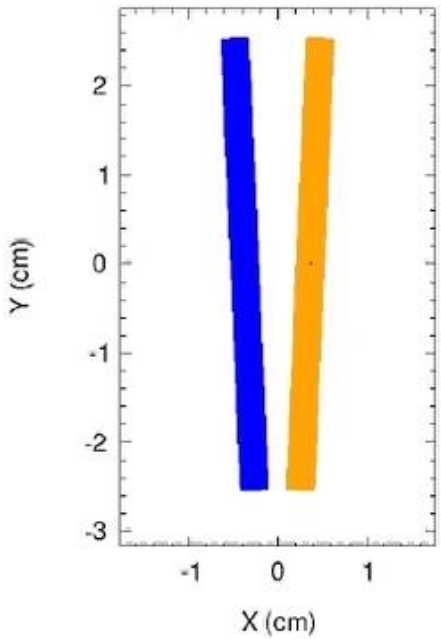
Oblique plate slap problem results

- All XFEM runs fail with element inversion at intial contact time unless discard is added for all material volfrc < 1e-4.
- Force fracture and phase control provide 3% speedup (used on all runs shown here).
- Hybrid QR provides near 2× speedup over regular XFEM.
- Did not run full QR.
- Ran on 16 cores.



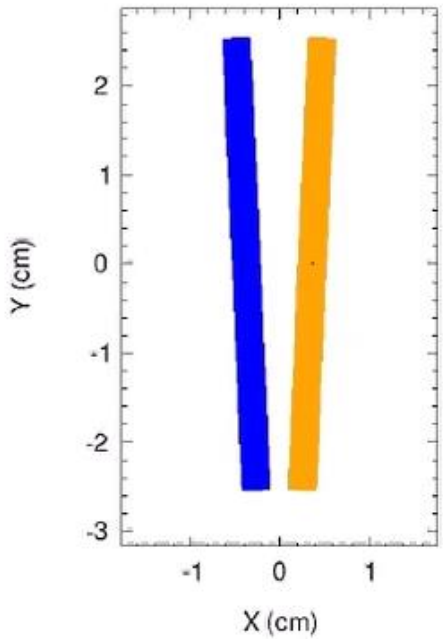
Oblique plate slap problem results: movies

UFEM



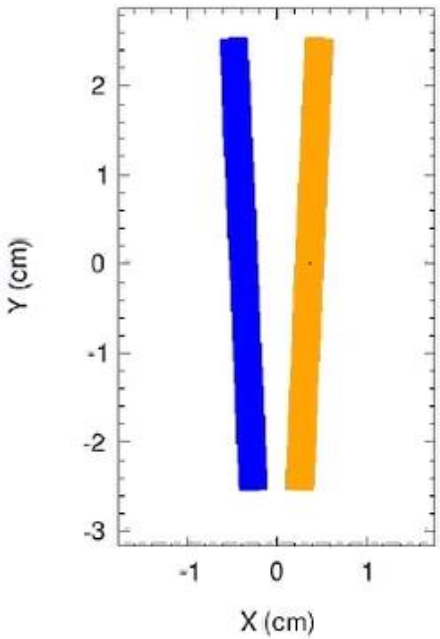
Time = 0.00 μs

XFEM



Time = 0.00 μs

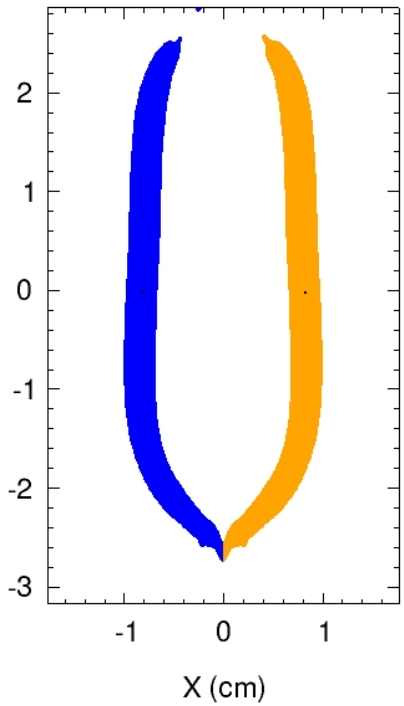
XFEM, hybrid QR



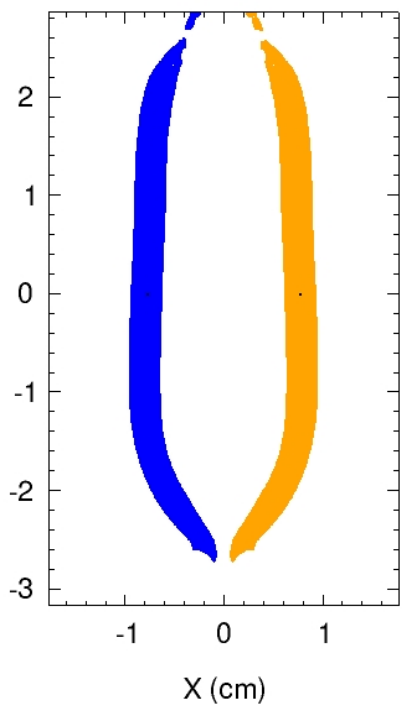
Time = 0.00 μs

Oblique plate slap problem results: final frame

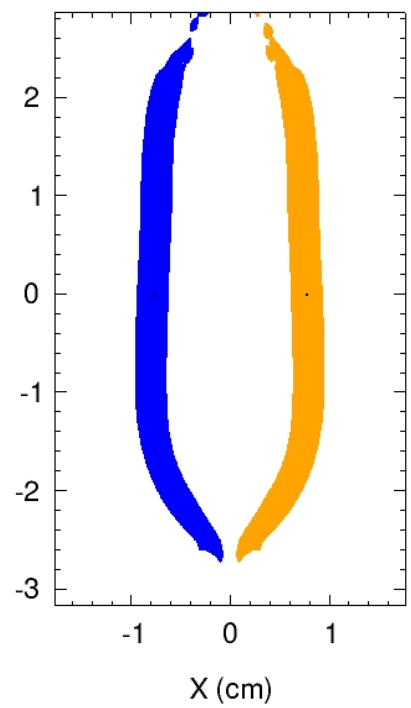
UFEM



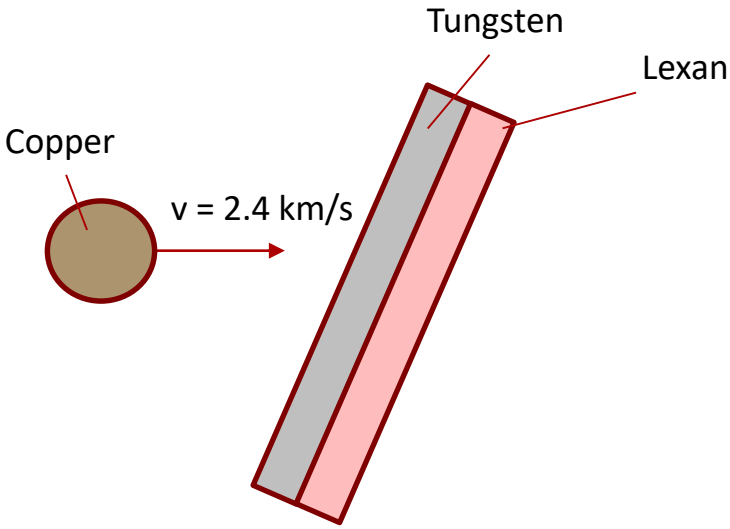
XFEM



XFEM, hybrid QR



Oblique cylinder impact on layered target

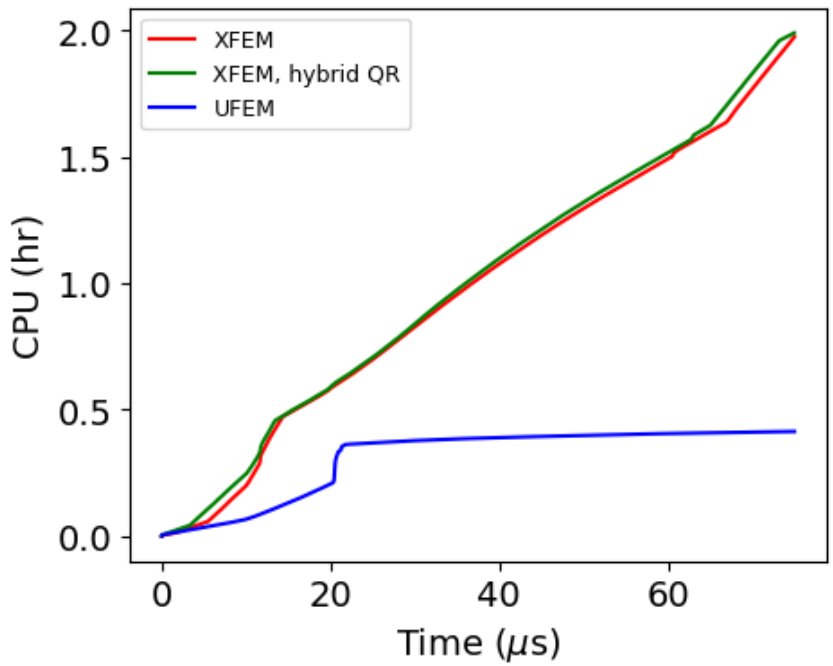


- Adjacent plates inclined 25°
- EOS and strength
- Simulation should allow recoil and separation due to difference in shock impedance of two target plate materials
- **148K elements**

→ *Last studied extensively in ALEGRA in November of 2009!!! Abandoned because of interface ordering issues, and never revived after automatic priorities were added.*

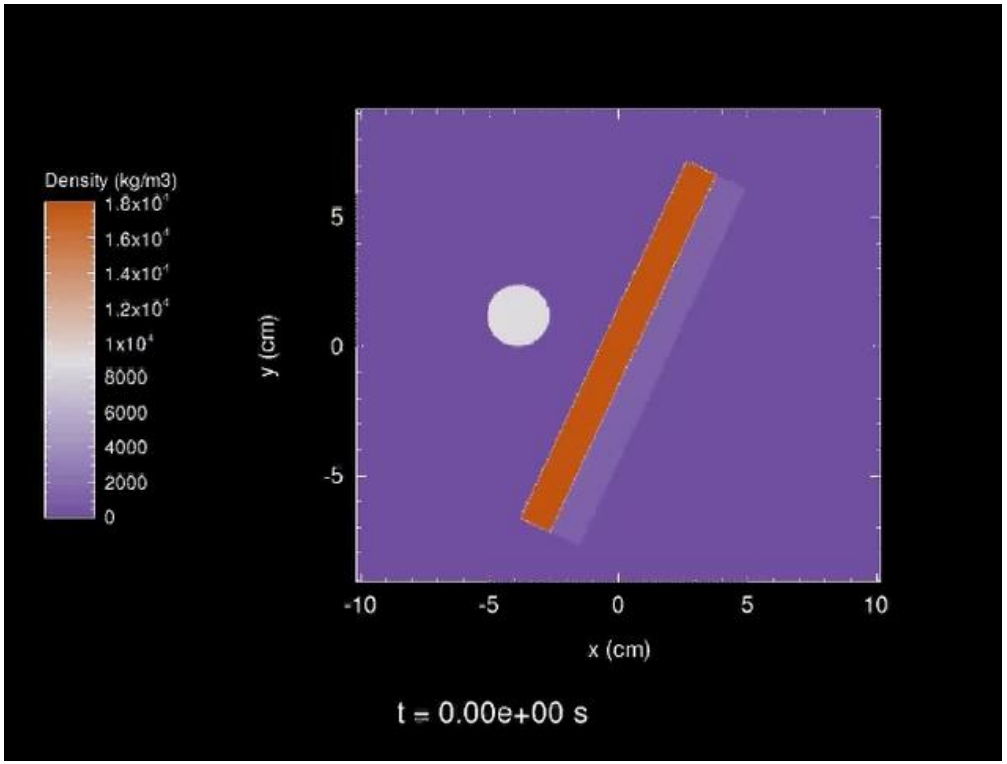
Oblique cylinder impact results

- All XFEM runs are successful after modern syntax is added (auto priorities, Aztec contact solver, no enforcement tolerances, etc.).
- Force fracture and phase control are necessary to allow UFEM simulation to run to completion. Not needed for XFEM!
- Hybrid QR provides no speedup for this problem.
- Ran on 8 cores.

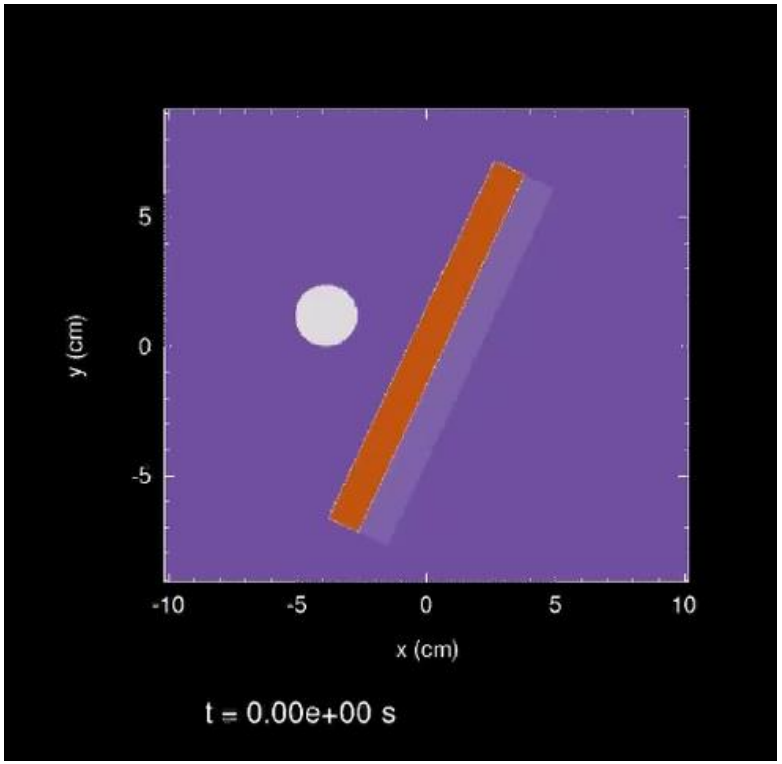


Oblique cylinder impact results: movies

UFEM

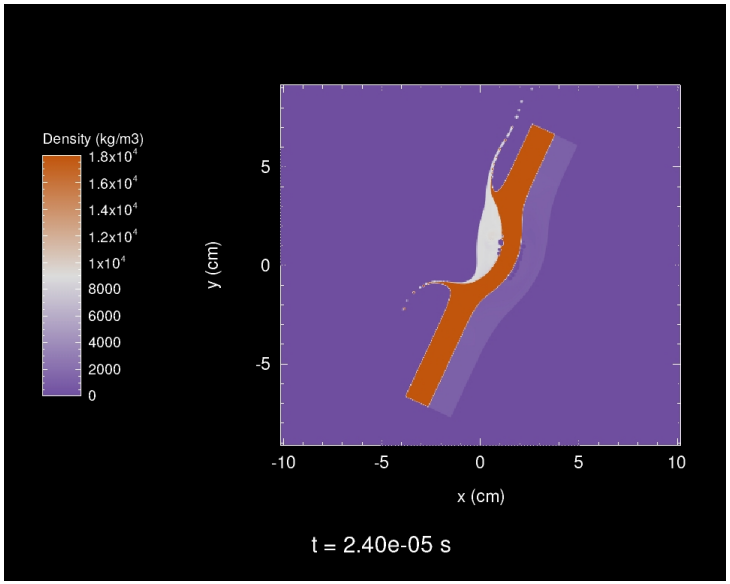


XFEM, hybrid QR

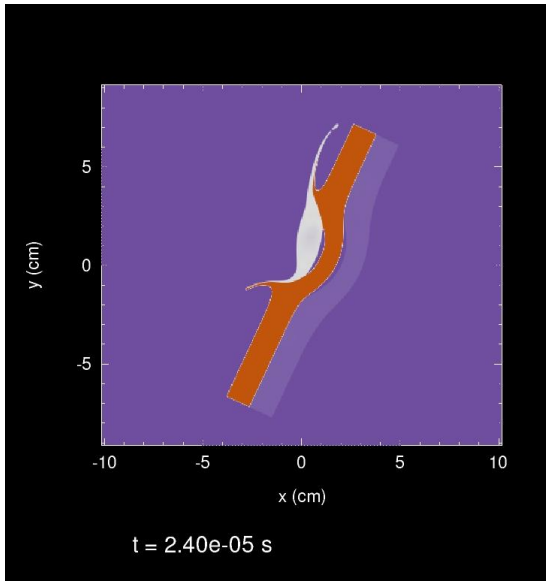


Oblique cylinder impact results: density at 24 μs

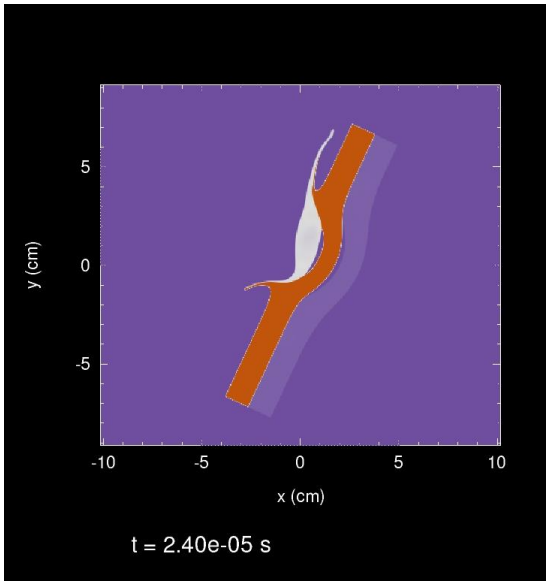
UFEM



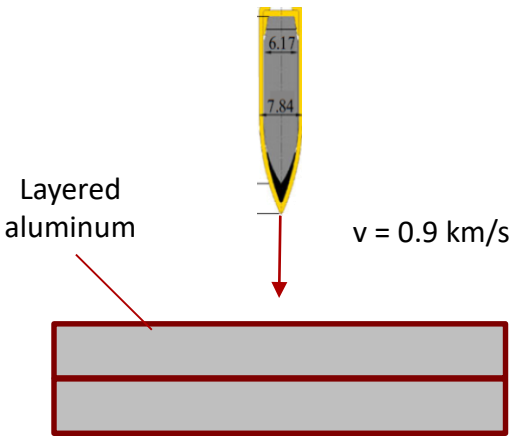
XFEM



XFEM, hybrid QR

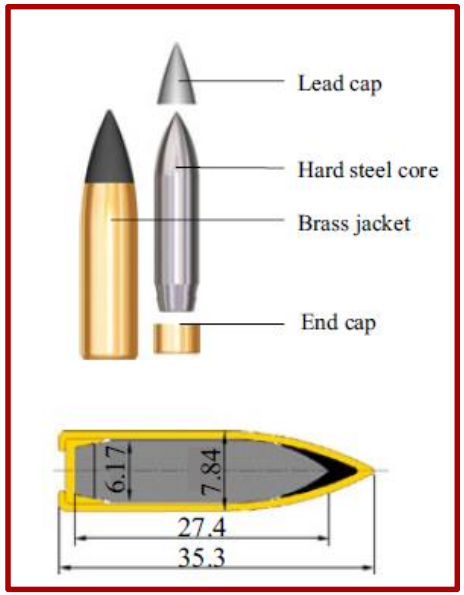


APM2 impact on layered aluminum



- Normal impact of APM2 round.
- Two 20-mm-thick Al plates, no adhesion.
- Axisymmetric r-z coordinates
- 4 materials, 5 material ID's.
- Jacket should separate from core and remain in the plates.
- **83K elements**

APM2 round
(0.30 cal / 7.62mm)



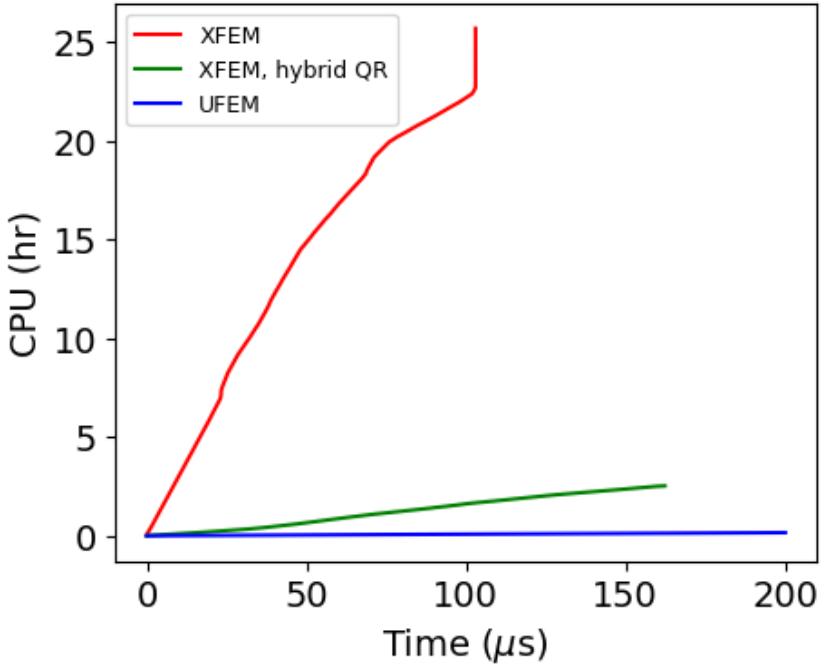
Forrestall et al, 2010
https://doi.org/10.1007/s11340-009-9262-5

→ *Last studied extensively in ALEGRA in 2014.*

→ *apm2-xfem test is still in the repo, TDD status currently.*

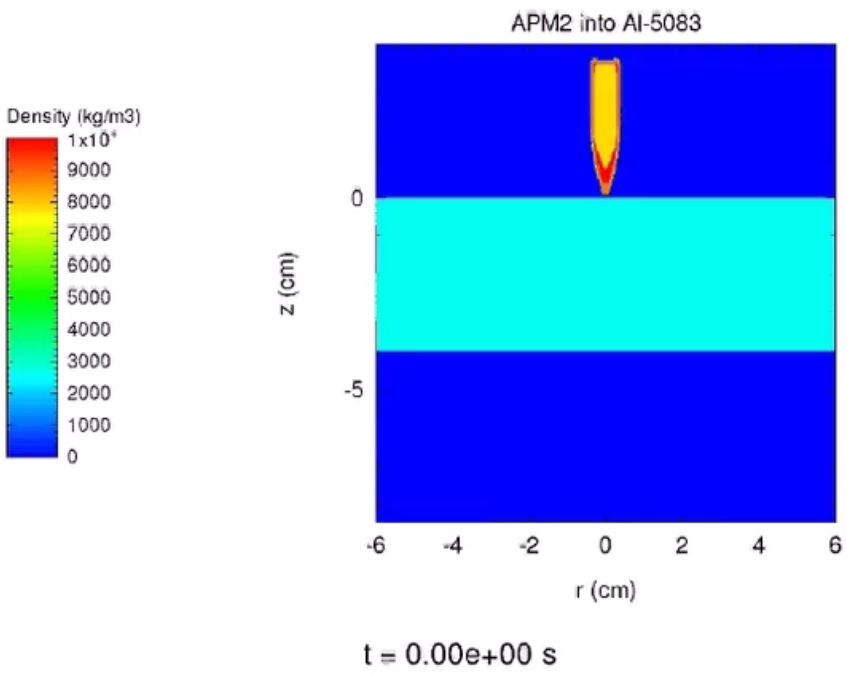
APM2 impact results

- XFEM simulations fail with element inversion persistently.
- With force fracture, phase control, and discard for volfrc < 1.e-4, the failure is deferred late enough to see the desired behavior.
- Hybrid QR provides both speedup and apparently better stability.
- Regular and hybrid simulations ran on different days on cee-compute007 – load might have played a role.
- Ran on 16 cores.

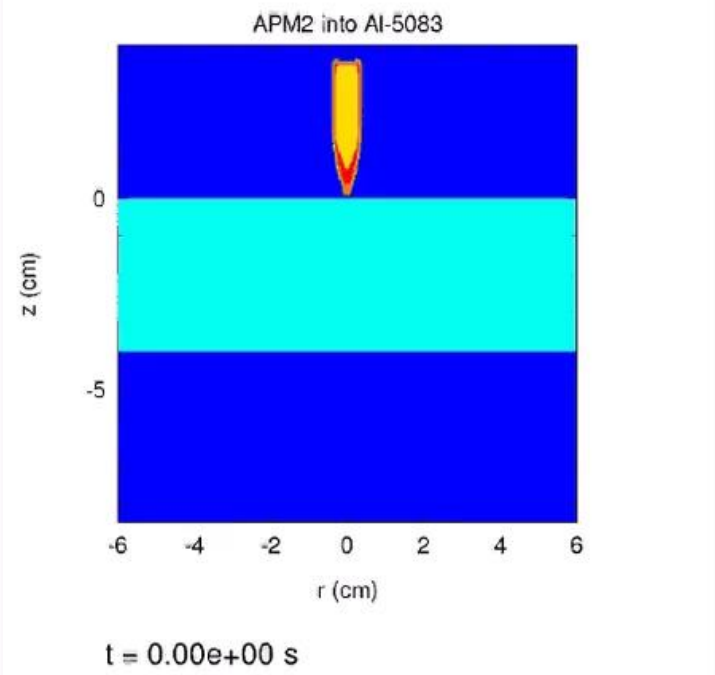


APM2 impact results: movies

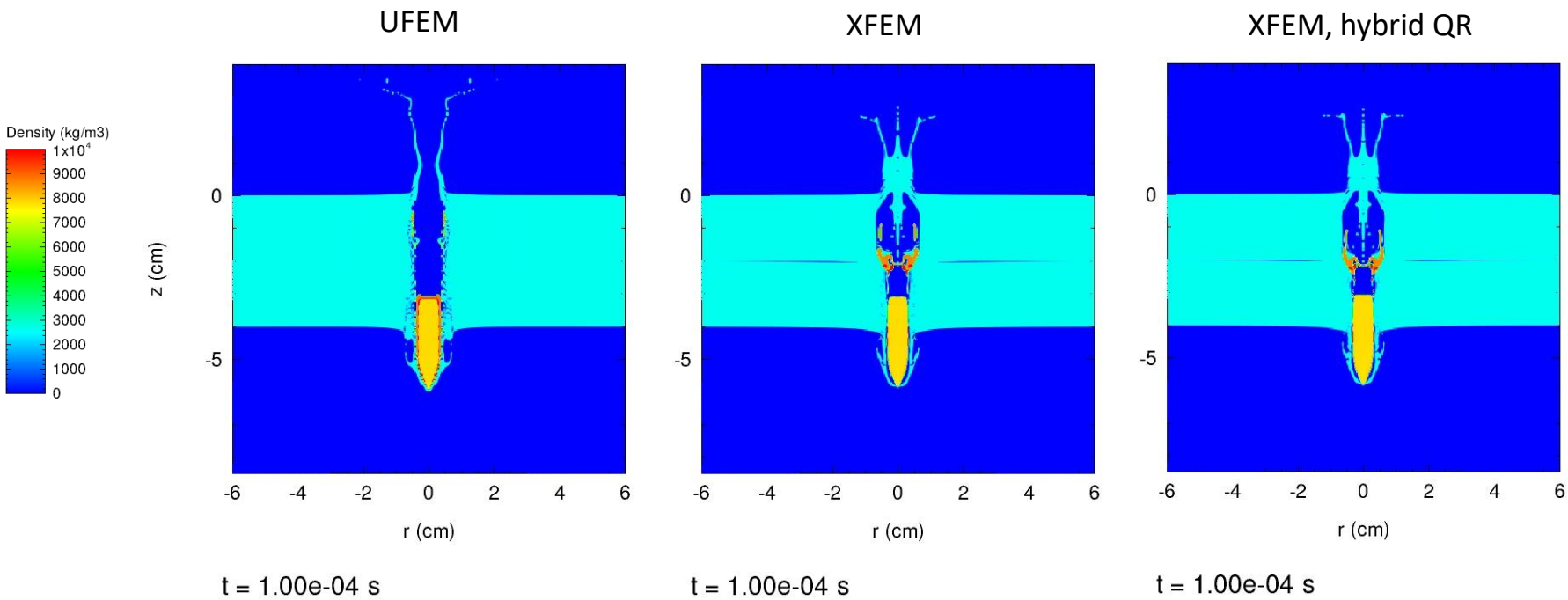
UFEM



XFEM, hybrid QR



APM2 impact results: frame at 100 μs



Summary

Sandia National Laboratories

CY2019 XFEM improvements:

- Discovered and resolved issue with ***density update***
- Discovered and resolved issue with ***void insertion***
- Discovered issue with ***singularity*** of ***XFEM contact solve***, and implemented several ***remedies***.

CY2019 Results:

- Fixes resulted in ***robustness improvements*** and ***reduced CPU times*** for 2D XFEM simulations.

*With this year's **improvements**, we are now able to **run 2D XFEM cases** that were **abandoned** years ago!!*

Ongoing/future work involving XFEM

Work in progress (WIP):

- Continue ***evaluation*** of XFEM on test cases of interest to ARL
- ***Journal article*** on XFEM for multi-material Eulerian SM in ALEGRA
  - ***QR algorithms*** for ensuring discrete *inf-sup* stability and their connection to various regularization algorithms in optimization is one ***novel contribution***.

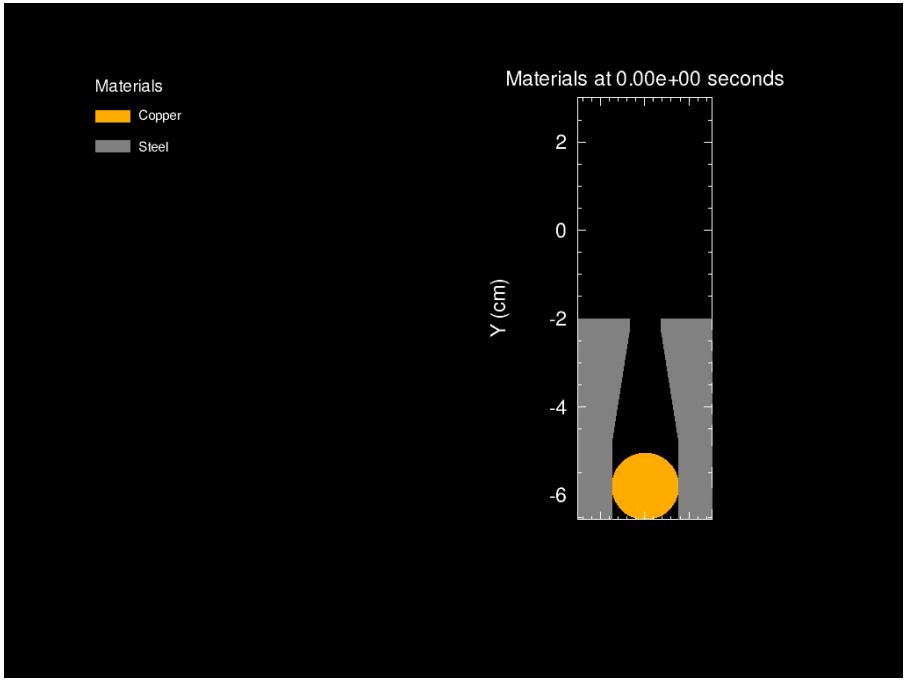
Possible future work:

- ***Continue*** XFEM performance improvements
- ***Higher-order remap*** for XFEM intersection remap
- Move 2D robustness/accuracy improvements to ***3D***
- Fix bug in ***skinning*** algorithm.
- ***Computational improvements*** to full/hybrid ***QR implementations***:
  - Switch to ***sparse QR library*** – QR will scale like $O(N^2)$ vs. $O(N^3)$
  - Modify ***hybrid QR logic*** to perform QR more often (e.g. when contact system is ill-conditioned but not exactly singular)

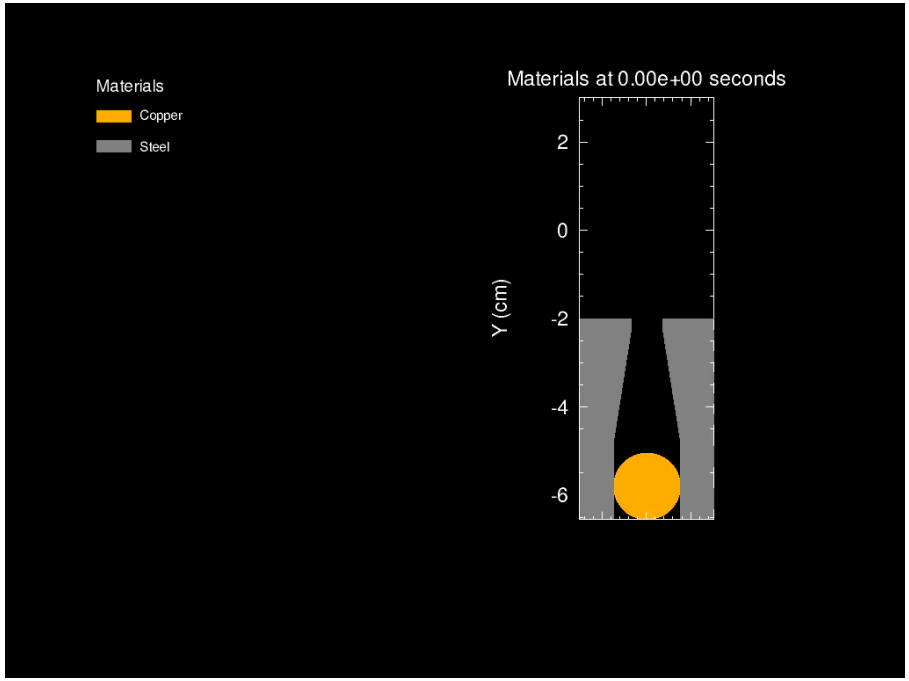
Backup Slides

Motivation for XFEM higher-order remap

XFEM, No QR, Oct. 2019 Code



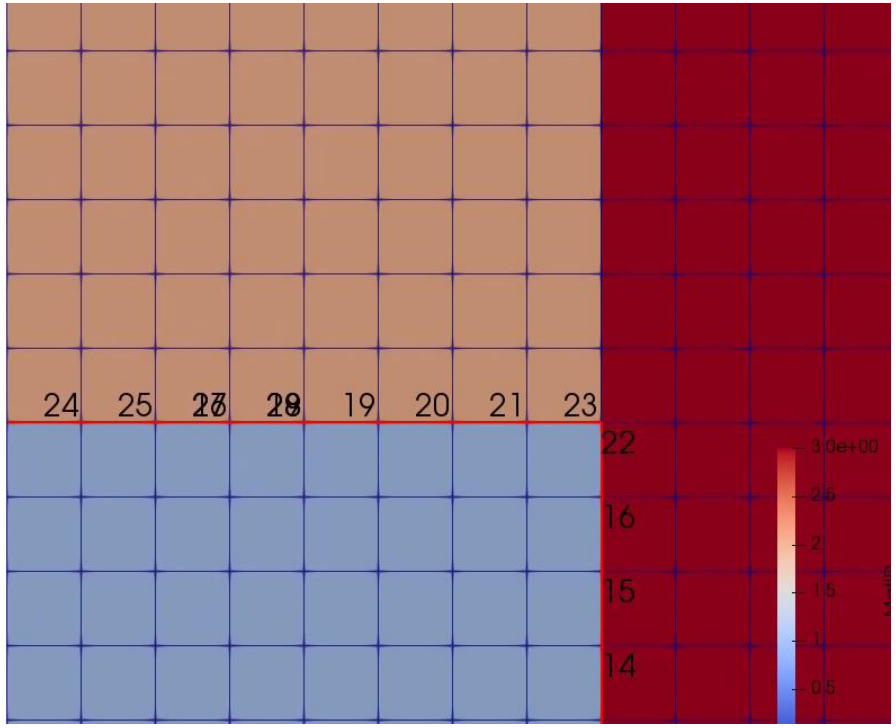
UFEM, Oct. 2019 Code



- ***"Sticking"*** projectile behavior in XFEM solution (left) is due to ***first-order remap*** algorithm in XFEM.
  - Would be remedied by implementing ***higher-order remap***.

Identifying cause of linearly dependent constraints: skinning algorithm

	Sandwich	Cup
QR with qr tol = 1e-10	27, 29	48, 51, 56, 58
QR with qr tol = 1e-5	26, 27, 28, 29, 38	48, 51, 55, 56, 57, 58, 59
Tuples of identical/near identical constraints (* = identical constraints)	{18, 29}*, {17, 27, 26}	{46, 48}, {50, 51, 57, 58}, {49, 55, 56}



Identifying cause of linearly dependent constraints: skinning algorithm

	Sandwich	Cup
QR with qr tol = 1e-10	27, 29	48, 51, 56, 58
QR with qr tol = 1e-5	26, 27, 28, 29, 38	48, 51, 55, 56, 57, 58, 59
Tuples of identical/near identical constraints (* = identical constraints)	{18, 29}*, {17, 27, 26}	{46, 48}, {50, 51, 57, 58}, {49, 55, 56}

Identifying cause of linearly dependent constraints: skinning algorithm

The culprit is the **skinning algorithm**, which adds redundant and nonsensical constraints.

	Sandwich	Cup
QR with qr tol = 1e-10	27, 29	48, 51, 56, 58
QR with qr tol = 1e-5	26, 27, 28, 29, 38	48, 51, 55, 56, 57, 58, 59
Tuples of identical/near identical constraints (* = identical constraints)	{18, 29}*, {17, 27, 26}	{46, 48}, {50, 51, 57, 58}, {49, 55, 56}

Identifying cause of linearly dependent constraints: skinning algorithm

The culprit is the **skinning algorithm**, which adds redundant and nonsensical constraints.

	Sandwich	Cup
QR with qr tol = 1e-10	27, 29	48, 51, 56, 58
QR with qr tol = 1e-5	26, 27, 28, 29, 38	48, 51, 55, 56, 57, 58, 59
Tuples of identical/near identical constraints (* = identical constraints)	{18, 29}*, {17, 27, 26}	{46, 48}, {50, 51, 57, 58}, {49, 55, 56}

Fixing skinning algorithm is **longer-term endeavor.**