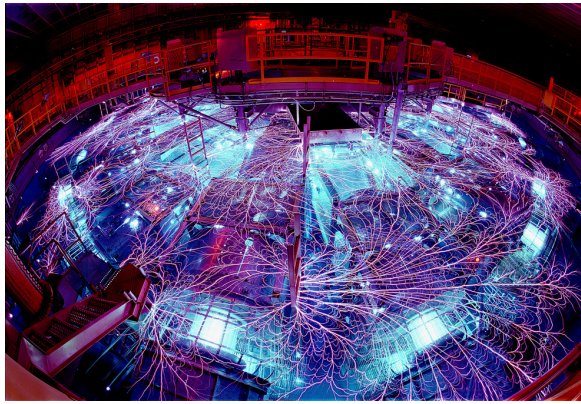
Exceptional service in the national interest

Sandia National Laboratories

SAND2014-18594PE





Making LAPACK and `libflame` Live in harmony

Kyungjoo Kim

Computer Science Research Institute, Sandia National Laboratories

September 25, 2014

U.S. DEPARTMENT OF ENERGY | National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2014-00000

Sandia National Laboratories

Use case of DLA in my application

- High order Moving Least Squares (MLS)

FLAPACK

- Introduction
- FLAPACK
- LAPACK test suite
- Using FLAPACK
- Conclusion

Sandia National Laboratories

High order Moving Least Squares

This work is supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program as part of the Colloboratory on Mathematics for Mesoscopic Modeling of Materials (CM4), under Award Number DE-SC0009247.

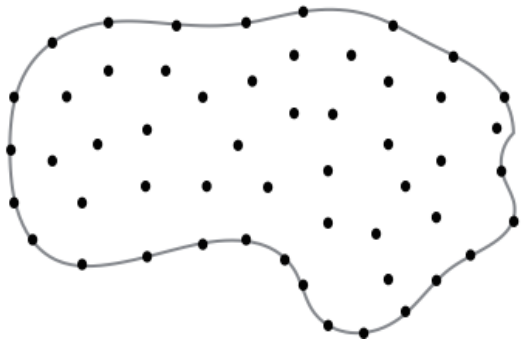
High order Moving Least Squares (MLS)

Mesh: a list of points with their connectivities.

Why meshfree methods ?

- Generating a suitable mesh is a challenging task.
- Easy to handle large deformation, moving boundary and fluid structure interaction problems..
- By advecting points in Lagrangian form, the non-linear advection term in Navier Stokes equations can be removed.
- Need to construct basis functions for each particle in every timestep with updated particle positions.

High order Moving Least Squares (MLS)

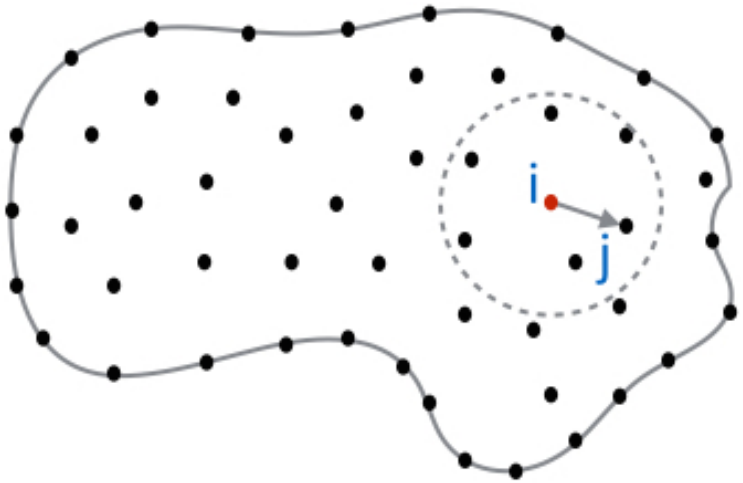


Meshfree: points are scattered on the domain.

Why meshfree methods ?

- Generating a suitable mesh is a challenging task.
- Easy to handle large deformation, moving boundary and fluid structure interaction problems..
- By advecting points in Lagrangian form, the non-linear advection term in Navier Stokes equations can be removed.
- Need to construct basis functions for each particle in every timestep with updated particle positions.

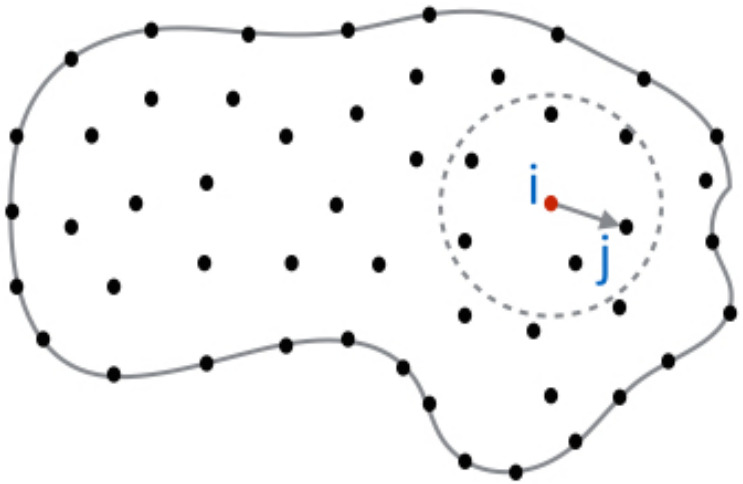
Computing MLS basis functions



Consider a set of points $\mathbf{X}_\Omega = \{\mathbf{x}_i\}_{i \in 1,\ldots N} \subset \Omega$. We seek an approximant of a function of the form:

$$u_h(\mathbf{x}) = \sum_{j=1}^{N} \phi_j(\mathbf{x}) u(\mathbf{x}_j)$$

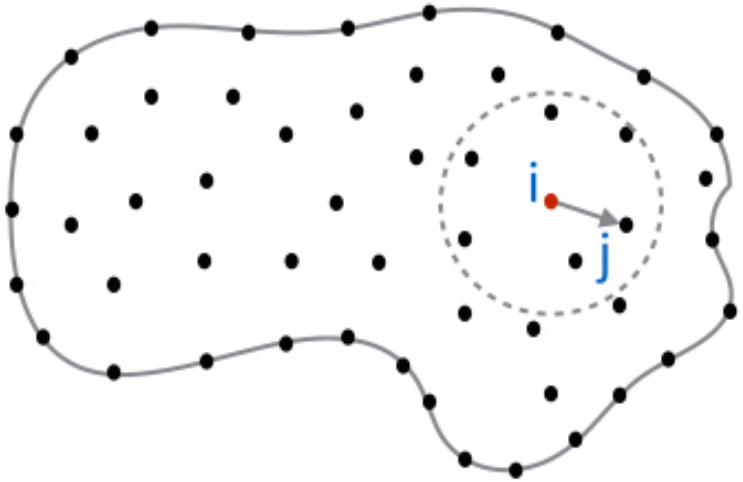
where the ϕ_j are shape functions associated with each point.



For a given function $u(\mathbf{x})$ known only at discrete values in the cloud of points, we construct approximation of shape function using polynomials:

$$\hat{\phi}(\mathbf{x}) = \mathbf{p}(\mathbf{x})^T \mathbf{c}$$

where $\mathbf{p}^T = [1, x, y, ...p_n]$ and $\mathbf{c} = [c_0, c_1, c_2, ...c_n]^T$.



The unknown coefficient vector $\mathbf{c}$ is determined by minimizing the following function for each particle at $\mathbf{x}_i$:

$$J(\mathbf{c}) = \sum_j \left(u_j - \hat{\phi}_i\right)^2 W(r_{ij})$$
$$= \sum_j \left(u_j - \mathbf{p}(\mathbf{x}_i)^T \mathbf{c}\right)^2 W(r_{ij}).$$

Solving for the minimization of a SPD quadratic form, the solution is given for each particle at $\mathbf{x}_i$:

$$\mathbf{c} = \left(\sum_j \mathbf{p}_j W(r_{ij}) \mathbf{p}_j^T\right)^{-1} \sum_j \mathbf{p}_j W(r_{ij}) u_j.$$

Computational aspects

- The cost for solving basis functions for each particle increases with $O(p^9)$ for 3D problems.
- Although the computation is completely local, the cost is comparable to the cost of solving global linear systems (Krylov solver preconditioned by Algebraic MultiGrid).
- This has to be recomputed for every time iteration.

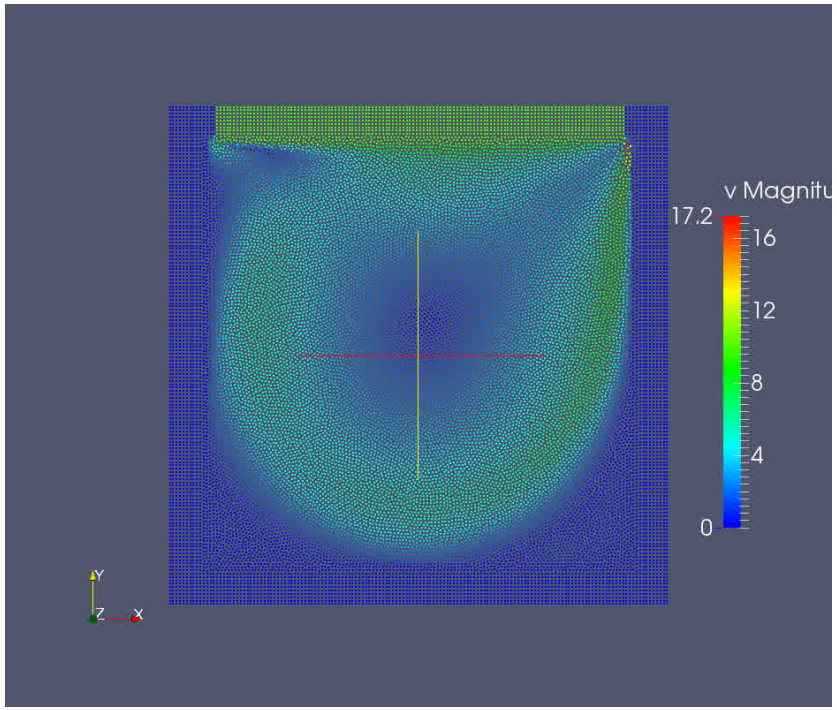
```
// as many as possible
for each timestep:
  // # of particles ~ millions
  for each particle i in the problem domain:
    // # of neighbors ~ p^d
    for each particle j in the neighborhood of i:
       // rank one update ~ p^(2·dim)
       M+ = p_j W(r_ij) p_j^T
    end of j
    // invert M ~ p^(3·dim)
    c = M^-1 rhs
  end of i
  // implicit time integration solving global systems of equations
end of timestep
```

Massively parallel 3D implicit MLS code

LAMMPS (a classical molecular dynamics code) handles particle data, parallel data distribution, ghosting.

Trilinos provides distributed parallel linear algebra: linear and non-linear solvers, preconditioners.

The code has been developed for distributed memory architectures and wish to explore hybrid node-level parallelism.



2D incompressible Navier Stokes equations: Lid driven cavity

Discussion

Sandia National Laboratories

Fast rank one updates and matrix inversion:

- Portable performance to many-core architectures.
- Multithreaded capability to solve a large number of small problems in parallel.
- High-level resource control that can group a small number of threads and assign the group of threads to small problems.
- Standization of LAPACK interface including data layout and characteristic features of modern architectures.

Sandia National Laboratories

FLAPACK

This work is supported by NSF award ACI-1148125/1340293 SI2-SSI : A Linear Algebra Software Infrastructure for Sustained Innovation in Computational Chemistry and other Sciences.

Sandia National Laboratories

Algorithm: $A := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$
where A_{TL} is 0×0

while $n(A_{TL}) < n(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$$A_{11} := \{L\backslash U\}_{11} = \text{LU\_UNB}(A_{11})$$
$$A_{12} := L_{11}^{-1} A_{12}$$
$$A_{21} := A_{21} U_{11}^{-1}$$
$$A_{22} := A_{22} - A_{21} A_{12}$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

Overview of `libflame`

- A family of algorithms for each operations is formally derived.
- Object-based APIs with high-level matrix abstraction: index-free notations.
- High performance library.
- Arbitrary row and column strides thanks to BLIS interface.
- Multi-threaded runtime task parallelism via algorithms-by-blocks: SuperMatrix.
- Completely in C; no FORTRAN compiler needed.

```
FLA_Error FLA_LU_blk_var5( FLA_Obj A, int nb_alg )
{
  FLA_Obj ATL, ATR,      A00, A01, A02,
          ABL, ABR,      A10, A11, A12,
                         A20, A21, A22;
  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,    0, 0, FLA_TL );

  while ( FLA_Obj_width( ATL ) < FLA_Obj_width( A ) ) {
    b = min( FLA_Obj_length(ABR), nb_alg );
    FLA_Repart_2x2_to_3x3(
       ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
    /* ************* */   /* ******************** */
                           &A10, /**/ &A11, &A12,
       ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
       b, b, FLA_BR );
    /*-----------------------------------------*/
    FLA_LU_unb_var5( A11 );
    FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
              FLA_NO_TRANSPOSE, FLA_UNIT_DIAG,
              FLA_ONE, A11,
                       A12 );
    FLA_Trsm( FLA_RIGHT, FLA_UPPER_TRIANGULAR,
              FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A11,
                       A21 );
    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_NO_TRANSPOSE,
              FLA_MINUS_ONE, A21,
                             A12,
              FLA_ONE,       A22 );
    /*-----------------------------------------*/
    FLA_Cont_with_3x3_to_2x2(
       &ATL, /**/ &ATR,        A00, A01, /**/ A02,
                               A10, A11, /**/ A12,
    /* *************** */   /* ****************** */
       &ABL, /**/ &ABR,        A20, A21, /**/ A22,
       FLA_TL );
  }
  return FLA_SUCCESS;
}
```

Overview of **`libflame`**

- A family of algorithms for each operations is formally derived.
- Object-based APIs with high-level matrix abstraction: index-free notations.
- High performance library.
- Arbitrary row and column strides thanks to BLIS interface.
- Multi-threaded runtime task parallelism via algorithms-by-blocks: SuperMatrix.
- Completely in C; no FORTRAN compiler needed.

Problems

`libflame`:

- The library supports an important subset of LAPACK functionality.
- For example, banded matrices are not supported.
- Full LAPACK functionality is important as many applications already rely on it.

LAPACK:

- FORTRAN compilers may not be available for new (or experimental) architectures: e.g., TI DSP.
- LAPACK evolves with new interfaces and new libraries.
- This often requires non-trivial modifications in application codes to adopt new features.
- Outdated standard FORTRAN-style interface; future achitectures require improved data layout and interface for better portability.

Sandia National Laboratories

FLAPACK delivers portable performance of `libflame` *through the LAPACK interface.*

What have been done...

- Entire LAPACK sources (ver. 3.5.0) were converted to C using `f2c` translator.
- For functionality supported in `libflame`, the LAPACK interface becomes a wrapper to `libflame`.
- Numerical properties of the algorithms in `libflame` were carefully examined via LAPACK test suite.

```
Replaced by libflame (also include unblocked versions)
------------------------------------------------------
(sdcz)getrf - LU with partial pivoting
(sdcz)hegst - Reduction of generalized eigenproblem.
(sdcz)lauum - Triangular matrix multiplication
(sdcz)potrf - Cholesky
(sdcz)potri - SPD inversion
(sdcz)trtri - Triangular inversion

(sd  )gebrd, orgbr, ormbr - Bidiagonalization
(sd  )sytrd, orgtr, ormtr - Symmetric tridiagonalization
(sd  )gelqf, orglq, ormlq - LQ
(sd  )geqpf, geqp3        - QR with column pivoting
(sd  )geqrf, orgqr, ormqr - QR
(sd  )gesvd               - SVD
------------------------------------------------------
```

Sandia National Laboratories

```
Leverage libflame (also include unblocked versions)
-----------------------------------------------------
(sdcz)gesv, gesvx, gesvxx - Solution to a system of linear equations.
(sdcz)hegv, hegvd, hegvx  - Hermitian eigenproblem.
(sdcz)pbtrf               - Cholesky for banded matrix..
(sdcz)pftrf               - Cholesky for RFP format.
(sdcz)pftri               - SPD inversion for RFP format.
(sdcz)posv, posvx, posvxx - Solution to SPD matrix.
(sdcz)sygv, sygvd, sygvx  - Generalized SPD eigenproblem.
(sdcz)getri               - Inverse of general matrix
(sdcz)sytri2x             - Inverse of Sym Indefinite matrix.

(sd  )gels, gelsd, gelss  - Least square problem.
      gelsx, gelsy
(sd  )gesdd, gejsv        - SVD.
(sd  )syev, syevd, syevr, - Symmetric eigenproblem.
      syevx
(sd  )ggsvp               - Preprocessing for SVD.
(sd  )gegs, gges, ggesx,  - Non-symmetric eigenproblem.
      ggev, ggevx
(sd  )ggqrf               - Generalized QR.
(sd  )ggrqf               - Generalized RQ.
(sd  )orcsd, orcsd2by1    - CS decomposition.
(sd  )ggglm               - General Gauss-Markov linear model.
(sd  )gglse               - Linear equality-constrained least square problem.
-----------------------------------------------------
```

Sandia National Laboratories

Transpose-free Transpose

- Matrix can be virtually tranposed by swapping column and row strides.

$$A(i,j) = A[i*cs + j*rs];$$

$$A^T(i,j) = A[i*rs + j*cs];$$

- BLIS supports arbitrary column and row strides.
- High-level matrix abstraction encapsulates stride information.
- Often, a single case needs to be implemented and it supports other operations: e.g., QR(LQ), Tridiagonalization, Bidiagonalization, SVD, etc.

Note that one routine corresponds to one operation in LAPACK implementation.

```
function [ U, s, V ] = FLA_Svd( A, transu, jobu, transv, jobv )
  if ( FLA_Obj_length( A ) > FLA_Obj_width( A ) )
    [ U, s, V ] = FLA_Svd_upper( A )
  else
    FLA_Obj_flip( A )
    [ V, s, U ] = FLA_Svd_upper( A )
    FLA_Obj_flip( A )
  end if
```

A single implementation of `FLA_SVD_upper` reused for several SVD operations based on different `trans` and `job` flags.

Control tree

- A family of algorithms is obtained from rigorous principles of formal derivation.
- Algorithms are implemented by harnessing other algorithms using control trees.
- This provides great tuning flexibility (algorithmic combination of and maintainability.

```
function [ U, s, V ] = FLA_Svd_upper( A, Svd_ctrl )
  if ( FLA_Obj_is_tall_rectangular( A, Svd_ctrl->Crossover ) == FALSE )
    [ A, T, S ] = FLA_Bidiag( A, Svd_ctrl->Bidiag_ctrl )
    [ s, U, V ] = FLA_Bsvd_upper( A, T, S, Svd_ctrl->Bsvd_ctrl );
  else
    [ U, R ] = FLA_QR( A, Svd_ctrl->QR_ctrl );
    [ R, T, S ] = FLA_Bidiag( R, Svd_ctrl->Bidiag_ctrl );
    [ s, R, V ] = FLA_Bsvd_upper( R, T, S, Svd_ctrl->Bsvd_ctrl );
    [ U ] = FLA_Gemm( U, R, Svd_ctrl->Gemm_ctrl );
  end if
```

Algorithm describes workflow and control tree includes its building blocks.

Control tree

- A family of algorithms is obtained from rigorous principles of formal derivateinodn.
- Algorithms are implemented by harnessing other algorithms using control trees.
- This provides great tuning flexibility (algorithmic combination of and maintainability.

```
Svd_ctrl
  + Crossover
  + Bidiag_ctrl
    + Unblocked (fused) + Blocked alg. variants
  + Bsvd_ctrl
    + Alg. variants
    + Max number of iterations + Blocksize
  + QR_ctrl
    + Unblocked alg. + Blocked alg. variants
    + Blocksize
  + Gemm_ctrl
    + Alg. variants
```

- A control tree describes algorithm variants.
- Then, algorithm variants become performance parameters for various matrix shapes and architectures.
- This tuning space cannot be explored in LAPACK implementation.

LAPACK test suite

Sandia National Laboratories

LAPACK provides rigorous testsuite based

- Algorithms in `libflame` formally derived, but their numerical stability is not verified.
- LAPACK provides rigorous testsuite with various test matrices:
  e.g., zero, identity, underflow/overflow, rank deficiency, clustered or evenly distributed eigenvalues, etc.
- The testsuite also test input/output

Test with BLIS

- All passed with a single failure on CTFSM:
  triangular solve where A is Rectangular Fully Packed (RFP) format.
- Inverse scale in TRSM might cause the problem.

Setup FLAPACK

```
$ cd libflame

./configure \
    --enable-max-arg-list-hack \
    --enable-lapack2flame \
    --disable-vector-intrinsics \
    --disable-ldim-alignment \
```

Remark

- `libflame` and BLIS framework may require leading dimension alignment or storage alignment.
- This may not be treated correctly when FORTRAN interface is used with user-provided buffer storage.

Using FLAPACK

Sandia National Laboratories

In FORTRAN, two lines of modification:

```
CALL FLA_INIT
...
CALL DPOTRF( UPLO, N, AFAC, LDA, INFO )
...
CALL FLA_FINALIZE
```

`FLA_INIT` and `FLA_FINALIZE` are optional.

In C, native `libflame` interface is recommended (not CLAPACK interface).

Conclusion

Sandia National Laboratories

- Full LAPACK layer is now available in `libflame`.
- The library is verified against LAPACK test suite.
- Features of `libflame` (e.g., SuperMatrix and GPU interface) are deliverable through the FLAPACK interface.