Implementation of linear viscoelasticity model in PDLAMMPS

R. Rahman and J.T. Foster
Center for Simulation, Visualization and
Real-time prediction (SiViRt)
The University of Texas at San Antonio
San Antonio, TX 78249
Email: rezwanur.rahman@utsa.edu,
john.foster@utsa.edu

June 26, 2013

1 Introduction

In this documentation the discussion is focused on implementation of linear viscoelasticity model in PDLAMMPS [PLPS08],[Pli95]. The peridynamic viscoelasticity formulation used in this work was developed by John Mitchell at Sandia National Lab [Mit11]. In the PDLAMMPS a new pair-peri-style is added in order to incorporate the viscoelasticity. The new source codes `pair-peri-ves.cpp` and `pair-peri-ves.h` are introduced. Besides, `fix-peri-neigh.cpp` and `fix-peri-neigh.h` are modified for introducing viscoelasticity.

2 Algorithm and implementation

In order to include the viscoelasticity the total extension state can be decomposed into two parts [Mit11] [1]:

[1] For detail please see the document on implementation of viscoelasticity in state based peridynamics model [Mit11]

$$\text{Total extension: } e(Y) = e^i(Y) + e^d(Y) \tag{1}$$

$$\text{Volumetric extension: } e^i(Y) = \frac{\theta(Y)|X|}{3} \tag{2}$$

$$\text{Deviatoric extension: } e^d(Y) = |Y| - |X| - \frac{\theta(Y)|X|}{3} \tag{3}$$

Here, $|Y|$, $|X|$ and θ are the reference state, defromation state and dilation state, respectively. The deviatoric extension ca be written as:

$$e^d(Y) = e^{de}(Y) + e^{db(i)}(Y) \tag{4}$$

Here, $e^{de}(Y)$ and $e^{db(i)}$ are the elastic and back parts of the deviatoric extension. Considering viscoelastic effect the peridynamic force state can be written as:

$$t = t^i + t^d = -\frac{3p}{m}\omega\underline{x} + (\alpha_\infty + \alpha_i)e^d - \alpha_i\omega e^{db} \tag{5}$$

p, k, α_i, t^i and t^d are hydrostatic pressure, bulk modulus, elastic properties volumetric and deviatoric scaler force states, respectively. $\alpha_\infty + \alpha_i = \frac{15\mu}{m}$; where, μ, m are the shear modulus and weighted volume, respectively. The influence function: $\omega\langle\xi\rangle = \frac{1}{\|\xi\|}$, $\|\xi\|$ is the scaler reference state. At the current or t_{n+1} timestep the scaler force state based on LPS is:

$$t^i_{n+1} = 3k\left(\frac{\theta(i)}{m(i)}\omega_+ + \frac{\theta(i)}{m(i)}\omega_-\right)e\nu(x_j - x_i)V_j \tag{6}$$

Based on viscoelasticity the deviatoric scale force state can be written as:

$$\begin{aligned} t^d_{n+1} &= 15(1-\lambda)\mu\left(\frac{\omega_+}{m(i)} + \frac{\omega_-}{m(i)}\right)e\nu(x_j - x_i)V_j \\ &+ 15\mu\lambda\left(\frac{\omega_+}{m(i)} + \frac{\omega_-}{m(i)}\right)\left(e - e^{db}_{n+1}\right)\nu(x_j - x_i)V_j \end{aligned} \tag{7}$$

Here, V_j, e^{db}_{n+1}, $\nu(x_j - x_i)$ are volume fraction, back extension at current timestep and particle volume scaling factor, respectively. λ varies within zero to one. The back extension at current state can be calculated as [2] [Mit11]:

[2] https://software.sandia.gov/trac/peridigm/

$$
\begin{aligned}
e^{db}_{n+1} &= (1-\text{decay})\, e^{d}_{n} + \text{decay} \cdot e^{db}_{n} + \beta \Delta e^{d} && (8)\\
\Delta e^{d} &= e^{d}_{n+1} - e^{d}_{n} && (9)\\
\text{decay} &= exp\left(-\frac{dt}{\tau_b}\right) && (10)\\
\beta &= 1 - \frac{\tau_b}{dt}\,(1-\text{decay}) && (11)
\end{aligned}
$$

At any current timestep (i.e. $n+1$), for a bond between i^{th} and j^{th} PD nodes, the deviatoric extension is stored on the fly. The value is stored in `deviatorextension[i][j]`. Similarly, the back extension is also stored in `deviatorBackextension[i][j]`. Both of the arrays initialized with zero. In the `fix_peri_neigh.cpp` an integer flag based switch is used in order to trigger the LPS, PMB or VES (Viscoelastic solid) models. The `class FixPeriNeigh` looks like:

```
FixPeriNeigh::FixPeriNeigh(LAMMPS *lmp,int narg, char **arg) :
  Fix(lmp, narg, arg)
{

  //Get the pair information
  Pair *anypair01 = force->pair_match("peri/pmb",1);
  Pair *anypair02 = force->pair_match("peri/lps",1);
  Pair *anypair03 = force->pair_match("peri/ves",1);

  isPMB = 0;  //Check if PMB
  isLPS = 0;  //Check if LPS
  isVES = 0;  //Check if VES

  //Select the flag based on which model is being used
  if (anypair01 != NULL)
  {
  isPMB=1;
  }
  else if (anypair02 != NULL)
  {
  isLPS=1;
```

```
  }
  else
  {
  isVES=1;
  }

  restart_global = 1;
  restart_peratom = 1;
  first = 1;

  // perform initial allocation of atom-based arrays
  // register with atom class
  // set maxpartner = 1 as placeholder

  maxpartner = 1;
  npartner = NULL;
  partner = NULL;
  if (isVES == 1){
    deviatorextention = NULL;
    deviatorBackextention = NULL;
  }
  r0 = NULL;
  vinter = NULL;
  wvolume = NULL;

  grow_arrays(atom->nmax);
  atom->add_callback(0);
  atom->add_callback(1);

  // initialize npartner to 0 so atom migration is OK the 1st time

  int nlocal = atom->nlocal;
  for (int i = 0; i < nlocal; i++) npartner[i] = 0;

  // set comm sizes needed by this fix

  comm_forward = 1;
}
```

There are three integer flags `isPMB` , `isLPS`, `isVES` .PMB, LPS and VES stand for bond based PD model, linear peridynamic solid and viscoelastic peridynamic solid, respectively. Based on the input file any one of these flags are set to one based on:

```
  Pair *anypair01 = force->pair_match("peri/pmb",1);
  Pair *anypair02 = force->pair_match("peri/lps",1);
  Pair *anypair03 = force->pair_match("peri/ves",1);
```

The stored values are used in:

```
PairPeriVES::compute(int eflag, int vflag)

double PairPeriVES::single(int i, int j, int itype, int jtype,
                           double rsq, double factor_coul,
                           double factor_lj,double &fforce)
```

At any current step the total scaler force state t is calculated and the current deviatoric extension and back extension are stored in order to use in next step. λ and time constant τ_b are material dependent parameters. These are defined in the LAMMPS input script. This will be discussed in the next section. The new LAMMPS pair style `pair_peri_ves` is defined with addition arguments in the input file (λ and τ_b):

```
void PairPeriVES::settings(int narg, char **arg)
{
  if (narg) error->all(FLERR,"Illegal pair_style command");
}

/* ----------------------------------------------------------------------
   set coeffs for one or more type pairs
------------------------------------------------------------------------- */

void PairPeriVES::coeff(int narg, char **arg)
{
  if (narg != 9) error->all(FLERR,"Incorrect args for pair coefficients");
  if (!allocated) allocate();
```

```
  int ilo,ihi,jlo,jhi;
  force->bounds(arg[0],atom->ntypes,ilo,ihi);
  force->bounds(arg[1],atom->ntypes,jlo,jhi);

  double bulkmodulus_one = atof(arg[2]);
  double shearmodulus_one = atof(arg[3]);
  double cut_one = atof(arg[4]);
  double s00_one = atof(arg[5]);
  double alpha_one = atof(arg[6]);
  double mlambdai_one = atof(arg[7]);    // New
  double mtaui_one = atof(arg[8]);      //New

  int count = 0;
  for (int i = ilo; i <= ihi; i++) {
    for (int j = MAX(jlo,i); j <= jhi; j++) {
      bulkmodulus[i][j] = bulkmodulus_one;
      shearmodulus[i][j] = shearmodulus_one;
      cut[i][j] = cut_one;
      s00[i][j] = s00_one;
      alpha[i][j] = alpha_one;
      m_lambdai[i][j] = mlambdai_one;    // New
      m_taubi[i][j] = mtaui_one;         // New
      setflag[i][j] = 1;
      count++;
    }
  }

  if (count == 0) error->all(FLERR,"Incorrect args for pair coefficients");
}
```

3 LAMMPS command for PD VES

There is no significant change in the LAMMPS input script for viscoelastic model. For PD VES the LAMMPS commands are:

```
pair_style peri/ves
pair_coeff i j Bulk_modulus Shear_modulus s00 alpha lambda tau
```

The LAMMPS compilation with the VES is same. The updated `Install.sh` should be used.

4 Conclusion

The LAMMPS implementation of peridynamic viscoelastic model is still in beta phase. Any bug or issue can be informed to the authors through `rezwanur.rahman@utsa.edu` .

5 Acknowledgment

The authors are thankful to Steve Plimpton and S. M. Raiyan Kabir for their useful suggestion regarding the code development for PDLAMMPS implementation of viscoelasticity.

References

[Mit11] John A. Mitchell. A non-local, ordinary-state-based viscoelasticity model for peridynamics. *Sandia National Lab Report*, 8064:1–28, 2011.

[Pli95] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J Comp Phys*, 117:1–19, 1995.

[PLPS08] Michael L. Parks, Richard B. Lehoucq, Steven J. Plimpton, and Stewart A. Silling. Implementing peridynamics in molecular dynamics code. *Computer physics communication*, 179:777–783, 2008.