Dynamics of vortex elements.

Nikolai Kornev

March 16, 2017

1 Computational vortex method. Main equations.

The computational vortex method is a grid free numerical method used in the fluid mechanics [1]. The most important advantage of this method is reduced artificial or numerical viscosity. According to this method, a continuous vortex field is represented as a set of N vortex elements each of which has the position vector $\mathbf{x}_j$, strength $\mathbf{\Gamma}_j$ and radius σ_j. The motion of vortex elements is described by the trajectory equation:

$$\frac{d\mathbf{y}_j}{dt} = \mathbf{V}(\mathbf{y}) \tag{1}$$

where the velocity $\mathbf{V}$ at any point $\mathbf{y}$ is calculated as a sum of velocity induced by vortex elements:

$$\mathbf{V}(\mathbf{y}) = \sum_{j=1}^{N} \mathbf{r} \times \mathbf{\Gamma}_j e^{-\pi\rho/2}, \tag{2}$$

where $\mathbf{r} = \mathbf{y} - \mathbf{x}_j$, $\rho = |\mathbf{r}|^2/\sigma^2$. Since at each time instant t the displacement of N points is calculated which depends on N contributions we have the classical N-body problems with N^2 operations.

During the motion the strength of the vortex element is changed according to the vortex transport equation

$$\frac{d\mathbf{\Gamma}_k}{dt} = (\mathbf{\Gamma}_k \cdot \nabla) \sum_{j=1}^{N} \mathbf{r} \times \mathbf{\Gamma}_j e^{-\pi\rho/2}, \tag{3}$$

Again, we have the classical N-body problems with N^2 operations. Equations (1) and (3) can be solved using the simple Euler method. From (3) we get $\mathbf{\Gamma}_k^*(t + \Delta t)$.

Due to vortex tube deformation the size of vortex elements is changed. It could be estimated from the condition

$$\mathbf{\Gamma}_k^*(t + \Delta t)\sigma_k^{2*}(t + \Delta t) = \mathbf{\Gamma}_k(t)\sigma_k^2(t) \tag{4}$$

From (4) we get $\sigma_k^*(t+\Delta t)$. Influence of viscosity is taken into account using the following simple model:

$$\sigma_k(t+\Delta t) = \sigma_k^*(t+\Delta t) + 2\nu\pi\Delta_t; \mathbf{\Gamma}_k(t+\Delta t) = \mathbf{\Gamma}_k^*(t)\left(\frac{\sigma_k^*(t+\Delta t)}{\sigma_k(t+\Delta t)}\right)^5 \qquad (5)$$

2 Task.

The task is to accelerate the computations using GPU, OMP and MPI options. The part of the code to be parallelized contains two embedded loops (see comments "this loop can be done parallel").

References

[1] Cottet, G-H. and Koumoutsakos, P.D. (2000) Vortex Methods: Theory and Practice, Cambridge University Press: New York.

Fortran code
This algorithm was implemented in Fortran Code.

```
* Short description of the code ***********************************************************
**********
*
*  A set of "Number" vortons is generated randomly within the box [0,1]*[0,1]*[0,1]
*  Each vorton has a strength "Omega v" and a radius "Radius"
*  Strength is a random vector with components r_x,r_y,r_z.  Each component is a random number
 in the range [-0.5,0.5]
*
*  The initially generated vortons move according to the Computational Vortex Method procedure
 "Ntime" time steps
*  The time step "Delta t" is constant
*  All vortons have the same radius "Radius"
*
*  Output:
*  After simulation we print the distribution of vortons and their strength
*  The velocity and vorticity fields in the plane x=0.5, 0<y<1 , 0<z<1  are also printed
*  The total time of simulations is also printed
*
*  Written by Nikolai Kornev on 14 January 2013
*  Last change 14 January 2013
*******************************************************************************************
**********

****************************************** Description of variables ***********************
**********

      real,allocatable::
     &Vortex(:,:),                                                    ! coordinates of vortons
     &Omega_v(:,:),                                                   ! strength vector of vorton
s
     &VortexN(:,:),                                                   ! coordinates of vortons (W
orking field)
     &Omega vN(:,:),                                                  ! strength vector of vorton
s  (Working field)
     &Sigma(:)                                                        ! radius of vortons  (Worki
ng field)

      Real StatisticalMoments(4)                  ! New Line 1

      open(774,file='Velocities.dat')

      open(773,file='MaxValue.dat')
******************************************  Input data   Begin  ****************************
*****************

      Ntime=1000000                                   ! Number of time steps
      Delta t=0.001                               ! Time step
      Radius=0.1                                  ! Radius of vorton (the same for all)
      Number=1000                                 ! number of vortons
      anu=0.00001
      dpnut=2.0*3.1416*Delta_t*anu

      Do i order=1,4                       ! New Line  2
      StatisticalMoments(i order)=0.000    ! New Line  3
      End Do                               ! New Line  4

******************************************  Input data  End  *******************************
**************



* description of variables ******************************************************************
**************
      allocate(Vortex(Number,3),VortexN(Number,3),Sigma(Number),
     &Omega_v(Number,3),Omega_vN(Number,3),stat=ierr)

* description of variables ******************************************************************
***************


* generation of initial distribution of vortons Begin

      Do ivorton=1,Number

      call Random(xxx)           ! Any generator of random numbers Uniform within the range [0
,1]
      call Random(yyy)
      call Random(zzz)
      Vortex(ivorton,1)=xxx
      Vortex(ivorton,2)=yyy
      Vortex(ivorton,3)=zzz
```

```
      call Random(Omega x)
      call Random(Omega y)
      call Random(Omega z)
      Omega v(ivorton,1)=(Omega x-0.5)
      Omega v(ivorton,2)=(Omega y-0.5)
      Omega v(ivorton,3)=(Omega z-0.5)
      Sigma  (ivorton)  = Radius
      End Do
* generation of initial distribution of vortons End

        time0=0
        call system clock(jcount1,jcount rate1,jcount_max1)
        time0=(jcount1+0.0)/(jcount_rate1+0.0)

      do itime=1,Ntime                                ! loop over time steps
      write(6,*)itime,Amagni,Energy

      Do ivorton=1,Number      ! this loop can be done parallel

*       calculation of the velocity and tensor S_ij induced at vorton with number "ivorton"
        Vxc=0.0
        Vyc=0.0
        Vzc=0.0
            dvxdxmov=0.0
            dvxdymov=0.0
            dvxdzmov=0.0
            dvydxmov=0.0
            dvydymov=0.0
            dvydzmov=0.0
            dvzdxmov=0.0
            dvzdymov=0.0
            dvzdzmov=0.0
          Do induced=1,Number                            ! this loop can be done parallel
            vxx=Vortex(ivorton,1)-Vortex(induced,1)
            vyy=Vortex(ivorton,2)-Vortex(induced,2)
            vzz=Vortex(ivorton,3)-Vortex(induced,3)
            radiika=vxx*vxx+vyy*vyy+vzz*vzz
            t1=vyy*   Omega v(induced,3)- vzz*Omega v(induced,2)
            t2=vzz*   Omega v(induced,1)- vxx*Omega v(induced,3)
            t3=vxx*   Omega v(induced,2)- vyy*Omega v(induced,1)
      Om22P=3.1416/Sigma(induced)/Sigma(induced)/2.0

            ssss=Exp(-radiika*Om22P)
*        Here are the velocities
            Vxc           = Vxc+ssss*t1
            Vyc           = Vyc+ssss*t2
            Vzc           = Vzc+ssss*t3

*       Here are the strain rate tensor S_ij

            dssss_dr=(-Om22P)*ssss

            dvxdxmov=dssss dr*vxx*t1                         +dvxdxmov
            dvxdymov=dssss dr*vyy*t1+Omega v(induced,3)*ssss +dvxdymov
            dvxdzmov=dssss_dr*vzz*t1-Omega_v(induced,2)*ssss +dvxdzmov

            dvydxmov=dssss dr*vxx*t2-Omega v(induced,3)*ssss +dvydxmov
            dvydymov=dssss dr*vyy*t2                         +dvydymov
            dvydzmov=dssss_dr*vzz*t2+Omega_v(induced,1)*ssss +dvydzmov

            dvzdxmov=dssss dr*vxx*t3+Omega v(induced,2)*ssss +dvzdxmov
            dvzdymov=dssss dr*vyy*t3-Omega v(induced,1)*ssss +dvzdymov
            dvzdzmov=dssss_dr*vzz*t3                         +dvzdzmov
          End do

*        new coordinates of the vorton "ivorton"  calculated using explicit Euler method
        VortexN(ivorton,1)=Vortex(ivorton,1)+Delta t*Vxc
        VortexN(ivorton,2)=Vortex(ivorton,2)+Delta t*Vyc
        VortexN(ivorton,3)=Vortex(ivorton,3)+Delta_t*Vzc

*        new strengths of the vorton "ivorton"  calculated using explicit Euler method
        domxdt=dvxdxmov*Omega v(ivorton,1)+dvxdymov*Omega_v(ivorton,2)+
     &         dvxdzmov*Omega v(ivorton,3)
        domydt=dvydxmov*Omega v(ivorton,1)+dvydymov*Omega_v(ivorton,2)+
     &         dvydzmov*Omega v(ivorton,3)
        domzdt=dvzdxmov*Omega v(ivorton,1)+dvzdymov*Omega_v(ivorton,2)+
     &         dvzdzmov*Omega v(ivorton,3)
        Omega vN(ivorton,1)=Omega v(ivorton,1)+domxdt*Delta t
        Omega vN(ivorton,2)=Omega v(ivorton,2)+domydt*Delta t
        Omega_vN(ivorton,3)=Omega_v(ivorton,3)+domzdt*Delta_t
      End do
```

```
* mapping to the cube back-----------------------------------------------------------------
          Do ivorton=1,Number
      do kkk=1,3
      If(VortexN(ivorton,kkk).lt.0.0)
     &VortexN(ivorton,kkk)=-VortexN(ivorton,kkk)
      If(VortexN(ivorton,kkk).gt.1.0)
     &VortexN(ivorton,kkk)=VortexN(ivorton,kkk)-1.0
      end do
          End do
* mapping to the cube back-----------------------------------------------------------------

* the old parameters became new ones
      Amagni=0.0
          Do ivorton=1,Number
          Vortex(ivorton,1)=VortexN(ivorton,1)
          Vortex(ivorton,2)=VortexN(ivorton,2)
          Vortex(ivorton,3)=VortexN(ivorton,3)
          Amagnit old=Sqrt(
     &Omega v(ivorton,1)**2+Omega_v(ivorton,2)**2+
     &Omega v(ivorton,3)**2)
          Omega v(ivorton,1)=Omega vN(ivorton,1)
          Omega v(ivorton,2)=Omega vN(ivorton,2)
          Omega v(ivorton,3)=Omega_vN(ivorton,3)
          Amagnit new=Sqrt(
     &Omega v(ivorton,1)**2+Omega_v(ivorton,2)**2+
     &Omega v(ivorton,3)**2)
      Sigma(ivorton)=Sigma(ivorton)*Sqrt(Amagnit_old/Amagnit_new)

      If(Amagnit new.ge.Amagni)then
      Amagni=Amagnit new
      Energy=(Amagnit new**2)*(Sigma(ivorton)**5)
      Speed max=Amagnit new*Sigma(ivorton)
      Sigmas=Sigma(ivorton)
      end if
          End do


* we print the maximum vorticity energy velocity radius
      write(773,773)itime*Delta t,Amagni,Energy,Speed_max,Sigmas
773   format(2f12.4,f12.7,2f12.4)

* Incorporation of viscosity
          Do ivorton=1,Number
      sigma old=Sigma(ivorton)
      Sigma(ivorton)=Sigma(ivorton)+dpnut
      Faktor=(sigma old/Sigma(ivorton))**5
          Omega v(ivorton,1)=Omega v(ivorton,1)*Faktor
          Omega v(ivorton,2)=Omega v(ivorton,2)*Faktor
          Omega v(ivorton,3)=Omega_v(ivorton,3)*Faktor
          End do
* Incorporation of viscosity

*eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
*       calculation of the velocity and tensor S_ij induced at the centre of the computational
 domain
          Vx=0.0

          Do induced=1,Number
            vxx=0.5-Vortex(induced,1)

            vyy=0.5-Vortex(induced,2)

            vzz=0.5-Vortex(induced,3)

            radiika=vxx*vxx+vyy*vyy+vzz*vzz

            t1=vyy*   Omega_v(induced,3)- vzz*Omega_v(induced,2)

            ssss=Exp(-radiika*Om22P)

           Vx=Vx+(ssss*t1)

          End do


      write(774,773)itime*Delta_t,Vx

      Do ier=1,4

      StatisticalMoments(ier)=StatisticalMoments(ier)+Vx**ier

      End Do
```

```
*eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

      End do                                            ! end of time loop

*      From this it is just printing results------------------------------------------------------
----
      close(773)
      close(774)

      call system clock(jcount1,jcount rate1,jcount max1)
      time1=(jcount1+0.0)/(jcount rate1+0.0)


      write(6,*)' It took',(time1-time0)/60.,' minutes'

* This part is not important This is just printing
* Print Print  Begin
      open(1,file='coordinates.dat')
      open(2,file='strengths.dat')
      Do ivorton=1,Number
      write(1,*)Vortex(ivorton,1),Vortex(ivorton,2),Vortex(ivorton,3)
      write(2,*)Omega_v(ivorton,1),Omega_v(ivorton,2),Omega_v(ivorton,3)
      End do
      close(1)
      close(2)

      open(1,file='sliceVelocity.dat')
      open(2,file='sliceVorticity.dat')
      NNN=100
      Delta=1./(NNN-1.)
      xxx=0.5
      do iy=1,NNN
      do iz=1,NNN
      yyy=(iy-1.)*Delta
      zzz=(iz-1.)*Delta

        Vxc=0.0
        Vyc=0.0
        Vzc=0.0
            dvxdymov=0.0
            dvxdzmov=0.0
            dvydxmov=0.0
            dvydzmov=0.0
            dvzdxmov=0.0
            dvzdymov=0.0

          Do induced=1,Number
            vxx=xxx-Vortex(induced,1)
            vyy=yyy-Vortex(induced,2)
            vzz=zzz-Vortex(induced,3)
            radiika=vxx*vxx+vyy*vyy+vzz*vzz
            t1=vyy*   Omega v(induced,3)- vzz*Omega v(induced,2)
            t2=vzz*   Omega v(induced,1)- vxx*Omega v(induced,3)
            t3=vxx*   Omega v(induced,2)- vyy*Omega_v(induced,1)
            ssss=Exp(-radiika*Om22P/2.)
*         Here are the velocities
            Vxc           = Vxc+ssss*t1
            Vyc           = Vyc+ssss*t2
            Vzc           = Vzc+ssss*t3

*        Here are the strain rate tensor S ij
            dssss_dr=(-Om22P)*Exp(-radiika*Om22P/2.)

            dvxdymov=dssss dr*vyy*t1+Omega v(induced,3)*ssss +dvxdymov
            dvxdzmov=dssss_dr*vzz*t1-Omega_v(induced,2)*ssss +dvxdzmov

            dvydxmov=dssss dr*vxx*t2-Omega v(induced,3)*ssss +dvydxmov
            dvydzmov=dssss_dr*vzz*t2+Omega_v(induced,1)*ssss +dvydzmov

            dvzdxmov=dssss dr*vxx*t3+Omega v(induced,2)*ssss +dvzdxmov
            dvzdymov=dssss_dr*vyy*t3-Omega_v(induced,1)*ssss +dvzdymov

          end do
      write(1,1)yyy,zzz,Vxc,Vyc,Vzc
      write(2,1)yyy,zzz,dvzdymov-dvydzmov,dvxdzmov-dvzdxmov,
     &dvydxmov-dvxdymov
      end do
      end do
      close(1)
      close(2)
```

```
1     format(5f12.4)

      Umean=StatisticalMoments(1)/Ntime
      Variance=Sqrt(StatisticalMoments(2)/Ntime)
      Skew=(StatisticalMoments(3)/Ntime)/(Variance**3)
      Curt=(StatisticalMoments(4)/Ntime)/(Variance**4)
      write(6,*)Umean,Variance,Skew,Curt

* Print Print  End

      deallocate(Vortex,VortexN,Omega_v,Omega_vN,Sigma,stat=ierr)

      end
```