Sindh Univ. Res. Jour. (Sci.Ser.), Vol: 35 (2) 35-42 (2003)

A MODIFIED LOG-REMESHING TECHNIQUE FOR THE COMPUTATION OF COMPLEX INDUSTRIAL FREE-SURFACE PROBLEMS

Meena Kumari Rajani and M. S. Chandio*

Institute of Mathematics and Computer Science, University of Sindh, Jamshoro, Pakistan.

Abstract

In this paper we have proposed a remeshing algorithm for the computation of free-surfaces based upon a fixed-connectivity approach. A complex industrial problem is investigated and is compared against the existing remeshing techniques. Various remeshing approaches are used and compared and a best approach is suggested. Our proposed algorithm has shown a great improvement in the solution accuracy and stability against the literature.

Keywords: Free-surface, Remeshing Algorithm, filament-stretching.

Introduction

Various research groups are engaged in solving free-surface problems. Free-surface problems are common and arise in many important industrial applications. These free-surface flows exhibit an additional complication, that is the temporal position of the free-surface is unknown a priori. Various remeshing techniques have been used in order to cope with the complex free-surface geometries. Yao and McKinley [1], in their work, introduced an elliptic mapping and remeshing technique (see Thompson et al. [2]) to redistribute the internal nodes, according to the displacement of the moving boundaries.

In the current paper, we have investigated various remeshing strategies for free-surface problems. A complex filament-stretching problem is considered to develop a robust remeshing algorithm for the computation of free-surface problems, for details of the numerical computation, the reader is referred to [3,4,5].

Our approach has been to devise algorithms that mitigate the numerical difficulties encountered during processing the free-surface problems. To avoid temporal mesh distortion and to maintain smooth and consistent nodal distribution, an efficient remeshing algorithm is introduced at each time-step.

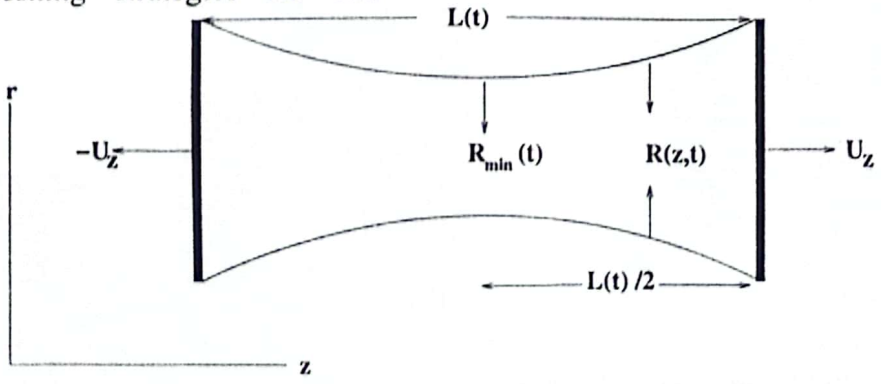


Figure 1. Domain of whole filament sample

* Author for Correspondence

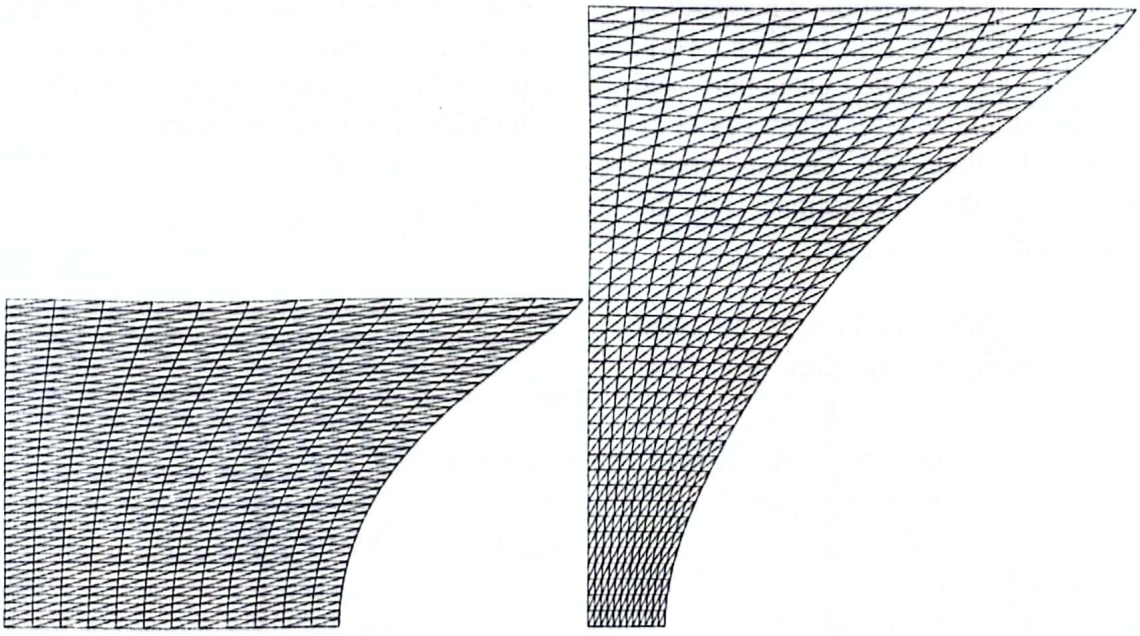
Remeshing strategies

The initial mesh plays an important role in simulating any flow problem. The success of implementation of the numerical algorithm depends in part upon the accuracy afforded by the mesh. In the beginning of the process, a rectangle mesh is considered. It is conspicuous that the initial mesh will deform axially, in time, due to axial stretching and radially due to the pinning boundary conditions and the presence of the free-surface. In order to avoid mesh distortion, a robust remeshing strategy is required. Two different methods are commonly employed to achieve this goal: (i) a fixed-connectivity approach; and (ii) an automatic (adaptive)-mesh generation. In the fixed-connectivity approach, the distribution of interior nodes is made by a predetermined rule using logarithmic, tan-hyperbolic (tanh), or uniform remeshing transformations. A second approach, used in recent years, is automatic mesh regeneration, particularly suitable for domains of arbitrary shape. This is achieved either by removing or inserting new elements until the mesh has acquired the desired nodal density in each region of the domain. In the present work we have allotted to employ a fixed-connectivity approach.

Due to the underpinning boundary conditions and the presence of the free surface, the mesh experiences large deformation both radially (near the interface of the free-surface and the top-plate) and axially, see fig. 1.
Here, for axial remeshing, where the mesh undergoes large stretching, we have employed, three different remeshing techniques, viz, uniform, tan-hyperbolic (tanh) and logarithmic remeshing rules. However, uniform remeshing is performed radially.

Uniform Remeshing

Redistribution of mesh points and mesh concentration in the uniform-scheme is maintained by distributing the internal and free-surface nodes uniformly.



(a) (b)

Figure No.2: Uniform remeshing: mesh distortion at time (a) 0.4 and (b) 0.9 secs

In this technique, we observed a slight swelling near the end plates at the beginning of stretching, however, the mesh sees big deformation at larger times, see fig.2. This deformation consequently causes the rupture of mesh in the middle of the filament and forced the solution to diverge.

Tan hyperbolic (tanh) remeshing

Redistribution of mesh points and mesh concentration in the tanh-scheme are maintained by a specified control parameter (for more details, the reader is referred to implementation of elliptic-mappings, see Thompson et al. [2]). A typical such mapping function may be expressed as:

$$H_i^{t^{n+1}} = 1 - \frac{\tanh(\beta(1 - H_i^{t^n}))}{\tanh\beta} \qquad (1)$$

where, H_i, $0 \le H_i \le 1$, is axial node-position of node i, and β is a mesh-density control-parameter.

With the tanh-scheme, large aspect-ratios have resulted about the mesh mid-plane region. This causes filament break-up away from the mid-plane, at large time levels, see fig.3. Nevertheless, this scheme performs well at relatively small Hencky-strain levels. Since the problem under consideration, does experience large deformation, a log-remeshing algorithm is employed instead [2]. Since the shape of the free-surface is highly curved near the top moving-plate, so there, a finer mesh is deemed necessary to retain mesh aspect-ratio, and hence accuracy and stability.

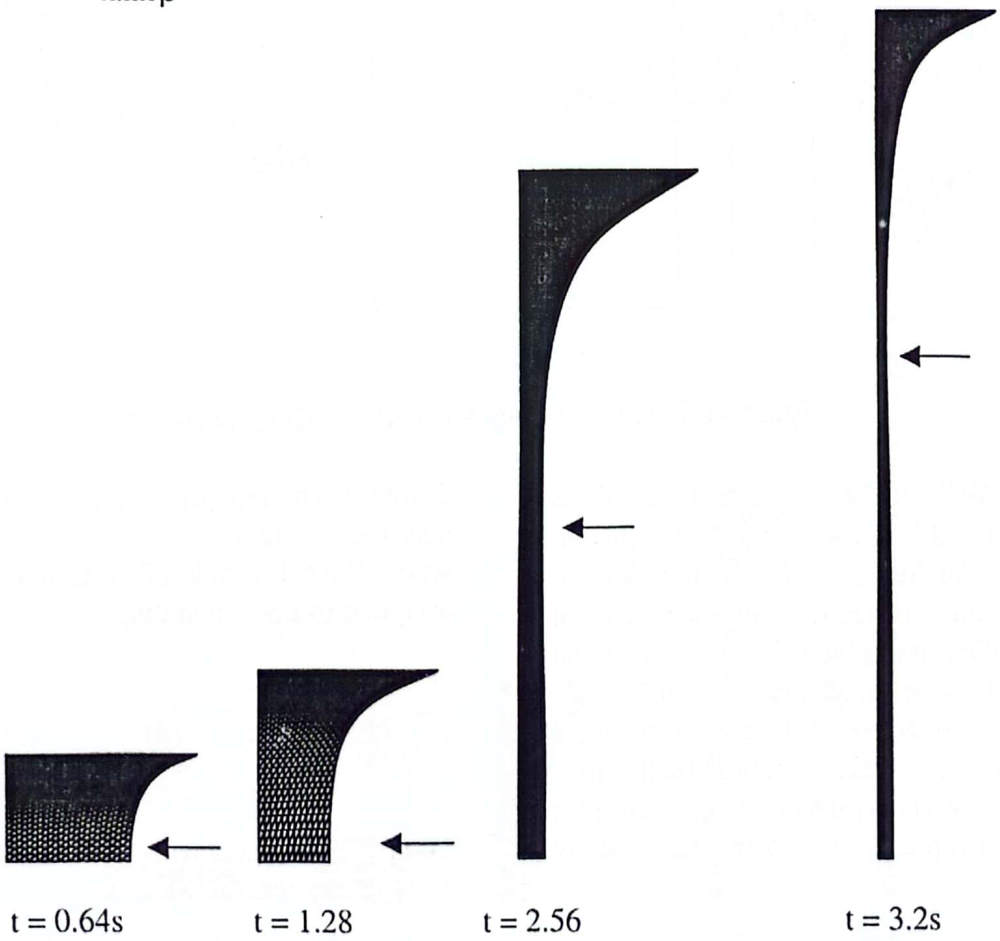


Figure 3. Tanh-remeshing: mesh stretching at various times, arrows indicating minimum radius position.

Modified log remeshing technique

We have made a slight modification in the log-remeshing approach, see Thompson *et.al* [2]. Since the mesh undergoes large stretching along the length of a filament, a modified log-remeshing approach is implemented for axial remeshing, where the mesh undergoes large stretching. This mapping function is given by

$$\Delta h^{n+1} = \Delta h^{n} + \frac{h_1 * 2 * elem}{n!} \qquad (2)$$

In a modified log-remeshing approach, the mesh is split into two sections along its length. Figure 4 provides a schematic diagram of mesh sections and boundaries at times t^n and t^{n+1}. Let L^{t^n} represent the mesh (filament) length at time t^n, and LS1 and LS_2 axial lengths of sections S_1 and S_2, respectively. This is expressed in the form

$$LS_1 = \frac{L^{t^n}}{fact}, \text{ and } LS_2 = L^{t^n} - LS_1, \quad (3)$$

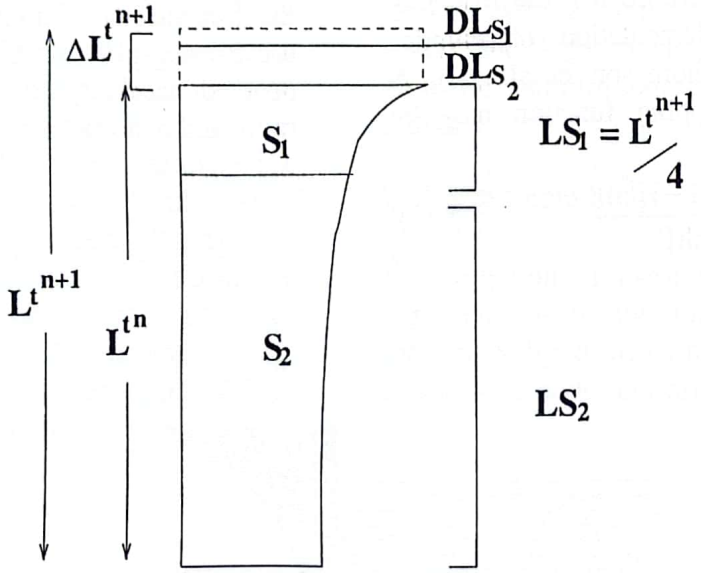


Figure 4: Schematic diagram with mesh sections

where, Δh^{n+1} is the change in length at time t +1, Δh^n at time t, h_1 is the current length of the filament, elem is the element number and n is the total number of axial nodes. This technique has shown great improvement at small times in contrast to previous discussed remeshing strategies. However, in this conventional log-remeshing strategy, mesh-density degrades rather too rapidly. In this manner and by design, high element-density is maintained near the top-plate.

where 'fact' is a ratio of mesh-length assigned to LS_1, such that

$$L^{t^n} = LS_1 + LS_2 \qquad (4)$$

Temporal displacements in sections, S_1 and S_2, are evaluated as

$$DL_{S_1} = \frac{\Delta L^{t^{n+1}}}{\alpha} \text{ and } DL_{S_2} = \Delta L^{t^{n+1}} - DL_{S_1}, \quad (5)$$

where $\Delta L^{t^{n+1}} = L^{t^{n+1}} - L^{t^n}$, α is a displacement ratio, and DL_{S_1}, DL_{S_2} are relative displacements of sections S_1 and S_2, respectively. Here

$$\Delta L^{t^{n+1}} = DL_{S_1} + DL_{S_2}. \quad (6)$$

With the use of these segment-displacement ratios, the length of sections S_1 and S_2 may be computed at time t^{n+1}, so that, as in equation (3)

$$LS_1 = \frac{L^{t^{n+1}}}{fact} + DL_{S_1}, \text{and } LS_2 = L^{t^{n+1}} - LS_1. \quad (7)$$

Once the respective displacements and ratios of each section are determined, a logarithmic mesh-mapping is performed per section, at every time-step, hence avoiding excess mesh distortion. This mapping function is given by

$$H_i^{t^{n+1}} = H_i^{t^n} + \frac{2i\Delta H^{t^{n+1}}}{telm(telm+1)} \quad (8)$$

where, telm is the total number of elements along the mesh-length, and $\Delta H_i^{t^{n+1}}$ takes the form of DL_{S_1} and DL_{S_2} in their respective sections. See appendix A for source code.

A rectangular mesh is used to perform the numerical simulation. The simulation commences from a simple mesh, but soon encounters large deformation that persists in time due to the transient nature of the problem. To avoid excessive mesh distortion at large deformation, and to retain smooth internal node distribution, the above cited low-cost remeshing algorithm [3] is applied at each time-step. A high level of field solution accuracy is maintained and reported, up to large times, see fig.5. This concurs well and exceeds findings in the literature.

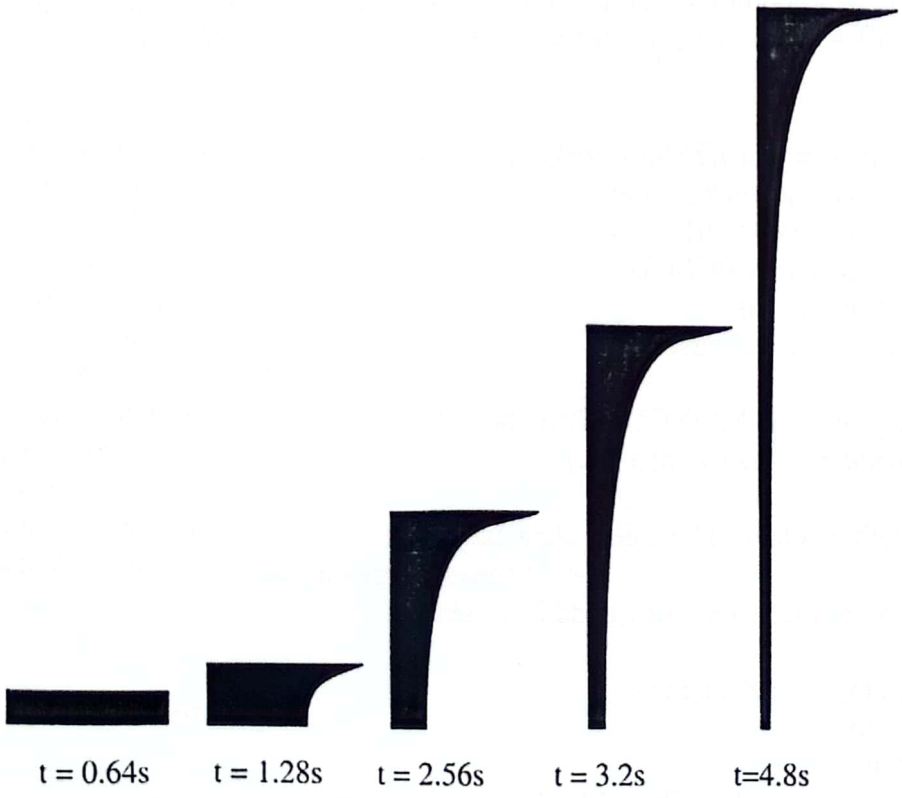


Figure 5. Modified log-remeshing scheme: mesh stretching at various times

Appendix 'A'

A subroutine for updating mesh (i) Axially (ii) Radially free boundary

```
!-------------------------------------------------------------------------------
!    Refer plate and free surface routines
!-------------------------------------------------------------------------------
!===============================================================================
!
      subroutine  re_admesh_logsec
      USE common_h
!
      Implicit None
!
      Integer :: ind1,i_nod,j_nod,k_nod,i_inc,j_inc,k_inc,nodd,elem_inc
      double precision :: s1,s2,x_co,y_co,x1_co,x2_co,y1_co,y2_co,  &
                          xdiff,ydiff,xratio,yratio,xcord,ycord,    &
                          xdiff1,ydiff1,strtt,endd,nfact,delh, h1,h2
         Integer :: Densenod, CoarseEl
!
! update nodes ( vetices + midside )  axially
          Densenod = DenseEl  * 2
          CoarseEl = ZFSMOVEL - DenseEl
         nfact = CoarseEl * ( CoarseEl + 1 )
!
!
        xcord = xyarray(1,1)                    ! get new position of x
        ycord = xyarray(1,2)                    ! get new position of y
         nodd = gl_mns_n(1)                      ! get nodd number
        x1_co = coord(nodd,1); y1_co = coord(nodd,2) ! get previous positions
        xdiff = xcord - x1_co; ydiff = ycord - y1_co ! find diff. b/w coord
!
!
          yratio = ydiff/(ZFSMOVEL * alphadd)
            h1 = yratio * DenseEl
            h2 = ydiff - h1
      do i_inc = 1,TOPUNOD
          delh = 0.0d0
          elem_inc = CoarseEl
!
        do j_inc = SYMMNOD-2,Densenod,-2
          nodd = vertexx(i_inc,j_inc)
!
          delh = delh + (h2 * 2.d0 * elem_inc)/nfact
                                          ! corresponding nodes
          coord(nodd,2) = coord(nodd,2) + delh
!
          elem_inc = elem_inc - 1
        enddo
      enddo
       nfact = DenseEl * ( DenseEl + 1 )
      do i_inc = 1,TOPUNOD
        delh = 0.0d0
```

```
          elem_inc = DenseEl
!
        do j_inc =  Densenod-1,1,-2
          nodd = vertexx(i_inc,j_inc)
!
          delh = delh + (h1 * 2.d0 * elem_inc)/nfact
                                              ! corresponding nodes
          coord(nodd,2) = coord(nodd,2) + delh + h2
!
          elem_inc = elem_inc - 1
        enddo
      enddo
!
! update midside nodes (Vertically)
      do i_inc = 1,TOPUNOD
        do j_inc = 2,SYMMNOD-2,2
         i_nod = vertexx(i_inc,j_inc-1)
         j_nod = vertexx(i_inc,j_inc+1)
         k_nod = vertexx(i_inc,j_inc)
         x1_co = coord(i_nod,1) ; y1_co = coord(i_nod,2)
         x2_co = coord(j_nod,1) ; y2_co = coord(j_nod,2)
           s2 = (y1_co + y2_co)/2.0d0
           s1 = (x1_co + x2_co)/2.0d0;
         coord(k_nod,1) = s1;
         coord(k_nod,2) = s2
        enddo  !end of j_inc loop
      enddo     !end of i_inc loop
! update remaining midside nodes (horizontally)
!
      do i_inc = 1, ZPLTMOVEL
        do j_inc = 1, SYMMNOD
         i_nod = vertexx(i_inc,j_inc)
         j_nod = vertexx(i_inc+1,j_inc)
         k_nod = midsid(i_inc,j_inc)
         x1_co = coord(i_nod,1) ; y1_co = coord(i_nod,2)
         x2_co = coord(j_nod,1) ; y2_co = coord(j_nod,2)
           s2 = (y1_co + y2_co)/2.0d0
           s1 = (x1_co + x2_co)/2.0d0;
         coord(k_nod,1) = s1;
         coord(k_nod,2) = s2
        enddo  !end of j_inc loop
      enddo     !end of i_inc loop
!
!
      return
      end subroutine  re_admesh_logsec
!
!
!=================================================================
```

Conclusion

In the present work a modified log-remeshing approach is proposed that mitigates mesh distortion at early stages and maintains the accuracy and stability in the solution. In the present work, we have used a fixed mesh-connectivity approach. This does not discount adaptive mesh generation techniques that may be implemented in order to retain suitable mesh-aspect ratios by inserting/removing elements. Attention may be paid to the improvement of free-surface location techniques, based upon Eulerian, Lagrangian or mixed ALE schemes.

References

[1]. M. Yao and G. H. McKinley, *Numerical simulation of extensional deformations of viscoelastic liquid bridges in filament stretching devices,* J. Non-Newt. Fluid Mech., 74:47-88 (1998).

[2]. J. F. Thompson, Z. U. A. Warsi and C. W. Mastion, *Numerical grid generation: foundations and applications,* North-Holland (1985).

[3]. M. S. Chandio, H. Mattalah, M. F. Webster, *Numerical Simulation of Viscous Filament Stretching Flows*, Under review Num. Meth. Fluid Flow, 2003, available Research Report Series, CSR-22/2002, Department of Computer Science, University of Wales Swansea, U.K, http://www-compsci.swan.ac.uk/reports.

[4]. P. Townsend and M. F. Webster, *An algorithm for the three-dimensional transient simulation of non-Newtonian fluid flows*, Proc. NUMETA 87, Martinus Nijhoff, Publishers, Dordrecht, 2:T12/1, (1987).

[5]. D. M. Hawken, H. R. Tamaddon-Jahromi, P. Townsend and M. F. Webster, *A Taylor-Galerkin Based Algorithm for Viscous Incompressible Flow*, Int. J. Num. Meth. Fluids, 10:327-351 (1990).