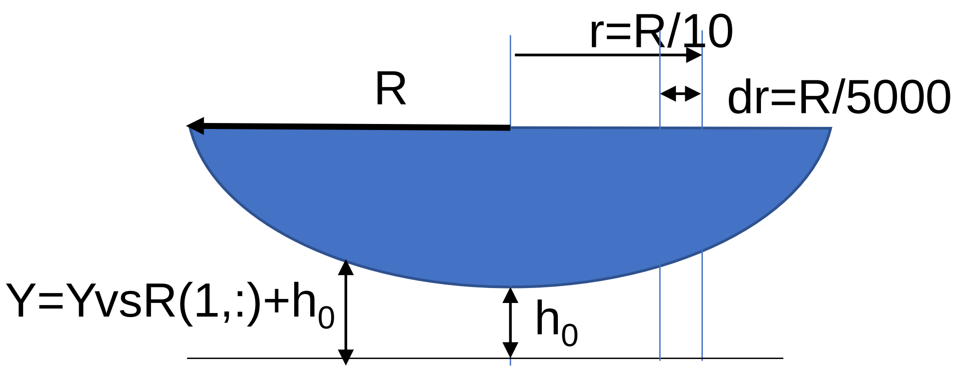
Tutorial for the code (elastic half space version)

```
3       %% Experiment parameters
4
5  -    E=2.1*1e6;% PDMS moudlus---unit: Pa
6  -    hinitial=400E-9;% Initial central separation---unit:m
7  -    eta= 0.2;% Viscosity---unit: pa.s
8  -    R=0.007;% Radius of curvature---unit:m
9  -    k=1021; % spring constant---unit:N/m
10 -    v=3.5E-6;% drive velocity---unit:m/s
11 -    poi=0.5; % Poisson's ratio
12
13 -    ShiftLength = 0*1e-9;
14 -    Hamaker = 10e-20;
15 -    electrolyte_conc = 0.001*6.022e23*1000;  %1mM in 1/m^3
16 -    surface_potential = -0.05;               %V
17 -    boltzman_roomtemp = 4.14e-21;            %J
18 -    electron_charge = -1.602e-19;            %C
19 -    permittivity = 8.854e-12;                %C^2/J/m
20 -    dielectric = 80;                         %unitless
21 -    debye_length =(permittivity*dielectric*boltzman_roomtemp/(2*e
22 -    w_doublelayer = 64*electrolyte_conc*boltzman_roomtemp*tanh(el
```

```
25      %% Computation domain
26
27 -    Rs=[0:R/5000:R/10]; % mesh size in radius
28 -    hguess1=zeros(1,length(Rs));% mesh size for hguess
29 -    p1guess=zeros(1,length(Rs));%mesh size for pressure p
30 -    hcalc=zeros(1,length(Rs));% mesh size for hcalc
31 -    wguess=zeros(1,length(Rs));% mesh size for deformation w
32 -    disjoining_p_VDW = zeros(1,length(Rs));% mesh size for Van der waals force
33 -    disjoining_p_DL = zeros(1,length(Rs));% mesh size for double layer force
34 -    allruns = zeros(400,1);%matrix to store run number
35 -    allforce=zeros(400,1);%matrix to store force
36 -    allhvst=zeros(400,length(Rs)); %matrix to store the central separation h
37 -    allwvst=zeros(400,length(Rs)); %matrix to store the deformation w
38 -    allpvst=zeros(400,length(Rs));%matrix to store the pressure
39 -    allxvst=zeros(400,length(Rs));%matrix to store x=h-w
40 -    allVDW=zeros(400,length(Rs));%matrix to store Van der Waals force
41 -    allDL=zeros(400,length(Rs));%matrix to store double layer force
42 -    allhcalc = zeros(3000,length(Rs));%matrix to store hcalc
43 -    allhguess1 = zeros(3000,length(Rs));%matrix to store hguess
```

line 5-22: set the real experimental parameters to the code

line 27-43: set the mesh size for the computational parameters



Calculate hydrodynamic pressure $\hat{p}$ from $\hat{h}guess$ and lubrication equation

$$\frac{\partial \hat{h}}{\partial \hat{t}} = \frac{1}{12\hat{r}} \cdot \frac{\partial}{\partial \hat{r}} \left(\hat{r}\hat{h}^3 \frac{\partial \hat{p}}{\partial \hat{r}} \right)$$

Line 109-113: solve the $\hat{r}\frac{\partial \hat{h}}{\partial \hat{t}}$ in the lubrication equation.

Line 115-118: $\frac{\partial \hat{p}}{\partial \hat{r}}$ is calculated

Line 119-125: $\hat{p}$ along the radius is calculated

```
109         dhdt1=(hguess1-hpreviousSol)./tincredless;
110         inside = dhdt1.*Rsdless;
111         fittinginside = spline(Rsdless,inside);
112         integ = fnint(fittinginside);
113         calcint = fnval(integ,Rsdless);
114 
115         for a = 1:length(p1guess)
116             dp1finder(a) = -12*calcint(a)/(Rsdless(a)*(hguess1(a))^3);
117             dp1finder(1) = 0;
118         end
119            p1guess(length(Rs)) = (3*(allxvst(indexer-1,1)-(hguess1(1)-wguess(1)*
120 
121         for a=length(p1guess)-1:-1:1
122             p1guess(a)=p1guess(a+1)+((dp1finder(a)+dp1finder(a+1))/2)*(Rsdless(a
123 
124         end
125         p1guess(1) = p1guess(2);
```

Calculate $\hat{h}calc$ according to force balance equation and compare $\hat{h}calc$ with $\hat{h}guess$ to verify if the difference is within the tolerance criteria

Line 141-152: Normal elastic deformation calculated

$$\hat{w} = \varepsilon \int_0^\infty \frac{2}{\hat{\xi}} X(\hat{\xi}T)\hat{Z}(\hat{\xi})J_0(\hat{\xi}\hat{r})d\hat{\xi}$$

Line 153-154: Calculate h according to force balcane

Line 159-162: Verify the difference between $\hat{h}calc$ and $\hat{h}guess$

```
141        for i = 1:length(rangedless)
142            Z(1,i) = trapz(Rsdless,Rsdless.*p1guess.*besselj(0,rangedless(1,i)*Rsdless));
143        end
144
145        gamma = 3-4*poi;
146        X = (gamma*(1-exp(-4*rangedless*thicknessdless))-4*rangedless.*thicknessdless.*exp(-2*rangedle
147
148
149        for j = 1:length(Rsdless)
150            wguess(1,j) = trapz(rangedless,2*(1-poi.^2).*X.*Z.*besselj(0, rangedless.*Rsdless(1,j)));
151
152        end
153        force=trapz(Rsdless,2*pi*Rsdless.*(p1guess+disjoining_p_DL-disjoining_p_VDW))+(allxvst(indexer
154        hcalc=force/springparam-min(time,timestop)+h0+elasticparam*wguess;
155        xxx = hcalc-wguess*elasticparam;
156
157
158
159        if max(abs(hcalc-hguess1))*hinitial< 1E-10
160            criteria=true;
161            allruns(indexer,1) = runs;
162
163
164        else
```

A method to update the new $\hat{h}guess$ to decrease the difference between $\hat{h}calc$ and $\hat{h}guess$

```
171        if min(hcalc)<0 || min(hguess1)<0
172            ratio = ratio + (1-ratio)*0.0008;
173            hguess1 = allhguess1(runs,:)*ratio+allhcalc(runs,:)*(1-ratio
174            criteriacount = 0;
175        end
176
177        if runs > 5
178
179            if (min(hguess1-allhguess1(runs-1)) > 0) && (min(allhgues
180                ratio = ratio + (1-ratio)*0.0008;
181                hguess1=allhguess1(runs,:)*ratio+allhcalc(runs,:)*(1-r
182                criteriacount = 0;
183            elseif (max(hguess1-allhguess1(runs-1,:)) < 0) && (max(al
184                ratio = ratio + (1-ratio)*0.0008;
185                hguess1=allhguess1(runs,:)*ratio+allhcalc(runs,:)*(1-r
186                criteriacount = 0;
187            else
188                criteriacount = criteriacount +1;
189            end
190
191
192            if (criteriacount >350) && (max(abs(hcalc-hguess1)*hinitia
193                ratio = ratio-ratio/20000;
194                criteriacount = 0;
195            end
196
197        end
198         hguess1=hguess1*ratio+hcalc*(1-ratio);
```

Line 171-195: criteria to update the ratio for updating “hguess”

Line 171-174: when the minimum of $\hat{h}calc$ or $\hat{h}guess$ get negative value, the ratio will be increased to help the convergence.

Line 179-185: when $\hat{h}calc$ and $\hat{h}guess$ have no intersection, , the ratio will be increased to help the convergence.

Line 192-195: when the result have a good sign of convergence, the ratio will be decreased speed up the convergence.

Flow chart for the code

0. Start with an h_{guess}

1. Given an h_{guess}, find $\partial h/\partial t$ from current and previous solution

2. Calculate $r\frac{\partial h}{\partial t}$

3. $I = \int r\frac{\partial h}{\partial t}dr$ using spline fit

4. Calculate $\frac{\partial p}{\partial r} = I \cdot \frac{12\eta}{rh^3}$

5. Calculate $p(r)$

6. Integrate $\mathrm{F} = \int p(r) \cdot 2\pi r dr$ to calculate force

7. Calculate w from expression for deformation

(Evaluate X, Z)

8. Use F from step 6 and force balance (spring and hydrodynamics) to get h_{calc}

8. Compare h_{guess} and h_{calc}

9. If within tolerance, move to next time step, reset ratio to default

10. If not within tolerance, update guess value, modify ratio to converge

Start from 0. for new iteration/exit iteration