Jack Lange

5/6/18

1.1

Part one

$$\Omega = -Q_{x0}\left(z - \frac{R^2}{z}\right) + \frac{Q}{2\pi}\ln\frac{z}{R} + C$$

$$\Phi_{lake} = Re\left(-Q_{x0}\left(z_1+R - \frac{R^2}{z_1+R}\right) + \frac{Q}{2\pi}\ln\frac{z_1+R}{R} + C\right)$$

$$\Phi_{ref} = Re\left(-Q_{x0}\left(z_{ref} - \frac{R^2}{z_{ref}}\right) + \frac{Q}{2\pi}\ln\frac{z_{ref}}{R} + C\right)$$

$$\Phi_{lake} = Re(\ldots) + \Phi_{ref} - Re(\ldots)$$

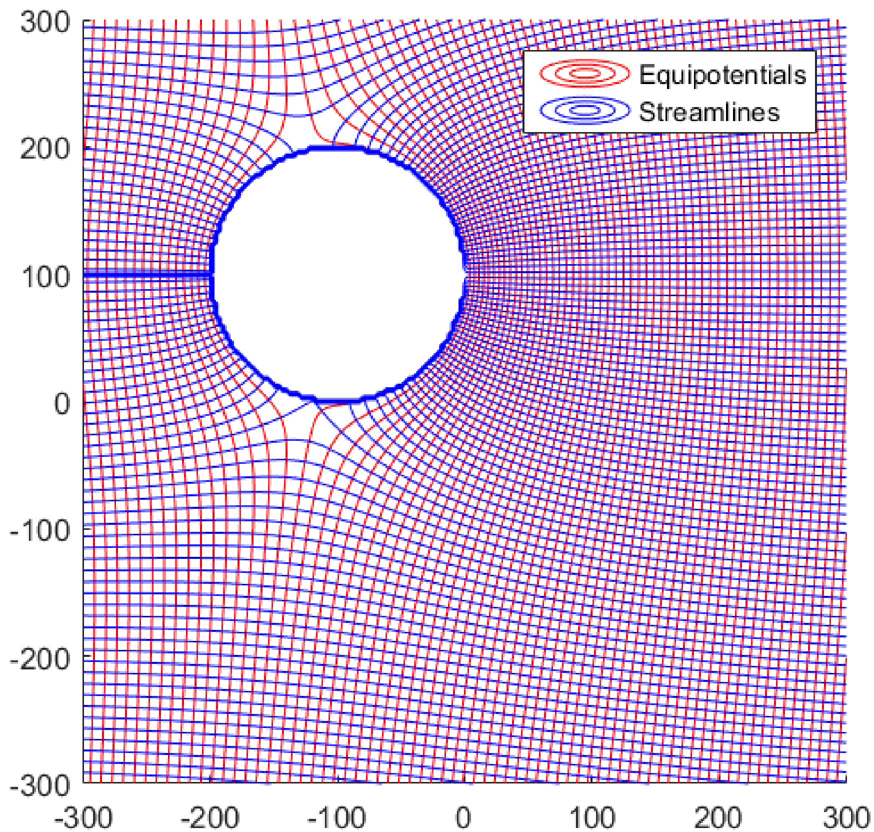
$$\Phi_{lake} = Re(@\, z_1+r) - Re(@\, z_{ref}) + \Phi_{ref} + Q\left(\frac{1}{2\pi}\ln \ldots - \frac{1}{2\pi}\ln \ldots\right)$$

$$\Downarrow$$

$$-Q\,\mathrm{real}\left(\frac{1}{2\pi}\ln\frac{z_1+R}{R} - \frac{1}{2\pi}\ln\left(\frac{z_{ref}}{r}\right)\right) = \Phi_{ref} - \Phi_{lake} + R(@\, z_1+R) - R(@\, z_{ref})$$

$$Q = \frac{\mathrm{Real}\left(-Q_{x0}\left(\left(z_1+R - \frac{R^2}{z_1+R}\right) - \left(z_{ref} - \frac{R^2}{z_{ref}}\right)\right)\right) + \Phi_{ref} - \Phi_{lake}}{\mathrm{real}\left(\frac{1}{2\pi}\ln\frac{z_1+R}{R} - \frac{1}{2\pi}\ln\frac{z_{ref}}{R}\right)}$$

The system of two equations presented above is used to solve for the value of Q and the constant, given a potential at the lake and a far field potential. The program used to generate a flow net for this situation is simple, requiring a couple lines to solve the above system of equations, and a function to calculate Omega given Q, C and the rest of the conditions. The code and resulting flow nets are presented below. The potential along the boundary of the lake is exactly correct.



Part11_runfile	`z1 = -100 + 1i *100;` `R1 = 100;` `Phi_lake = 100;` `refz = 1000;` `refPhi = 50;` `Qx0 = .4;` `z0 = [z1, refz];` `Phi = [Phi_lake, refPhi];` `nLakes = 1;` `Q = real(refPhi - Phi_lake -Qx0* ((z1+R1 - (R1*R1/(z1+R1)))-(refz - (R1*R1/(refz)))))/real((1/2*pi) * (log((z1+R1)/R1) - log(refz/R1)));`

	`` C= real(Phi_lake + Qx0 *(z1 + R1 - R1*R1 /(z1+R1)) - (Q/(2*pi)) * log((z1+R1)/R1)); ContourMe_flow_net(-300,300,200,-300,300,200, @(z)Omega_total_1(z,z1,R1,Q , C,Qx0),100); ```
Omega_total_1.m	``` function [omega] = Omega_total_1(z,z1,R1,Q , C,Qx0) %OMEGA_TOTAL_1 Summary of this function goes here % Detailed explanation goes here rsq = R1 *R1; if ((z-z1)*conj(z-z1)) < rsq omega = NaN; else omega = -Qx0 *((z-z1)- R1*R1 /(z-z1)) + (Q/(2*pi)) * log((z-z1)/R1)+C; end ```

1.2

The code for this portion and part 2 is roughly the same, I will first present the parts that are shared:

BZ_of_z	``` function [Z] = BZ_of_z(z,zm,rm) %UNTITLED Summary of this function goes here % Detailed explanation goes here Z = (z-zm)/rm ; end ```

Cauchy_integral_phi	<pre>function [a] = Cauchy_integral_phi(N_in,m,z_of_Z,Omega_of_z) if N_in<2*m N=2*m; else N=N_in; end deltheta=2*pi/N; theta_0=0.5*deltheta; Int=zeros(N,m+1); a=zeros(1,m+1); for nu=1:N n=nu-1; theta=theta_0+n*deltheta; Z=exp(1i*theta); z=z_of_Z(Z); Omega=Omega_of_z(z); for j=1:m+1 mu=j-1; Int(nu,j)=real(Omega)*exp(-1i*mu*theta); end end for j=1:m+1 a(j)=0; for n=1:N a(j)=a(j)+Int(n,j); end a(j)=2*a(j)/N; end a(1)=.5*a(1); end</pre>
Omega_lake	<pre>function [Omega] = Omega_lake(Z,z,a,Q,z_ref,z0,N_coef) if Z*conj(Z)<0.999 Omega = complex(NaN,NaN); else if N_coef==0 Omega = 0; else Omega = Q/(2*pi)*log((z-z0)/(abs(z_ref-z0))); for i = 1:N_coef Omega = Omega + a(i+1)*Z^-i; end end</pre>

	`end`
Omega_total	<pre>function [Omega] = Omega_total(z, m_not ,z0, R, a, Q, z_ref, M, N_coef, W0, C) Omega = -W0*z + C; for m = 1:M if m ~= m_not Z = BZ_of_z(z,z0(m),R(m)); Omega = Omega + Omega_lake(Z,z,a(m,:),Q(m),z_ref,z0(m),N_coef); end end end</pre>
Solve_lakes_fulit	<pre>function [a,Q,C] = solve_lakes_fulit(Qx0,Phi0,Phi_lake,M,N,m,z0,R,chi_far,z_ref) error=1e6; NIT=0; C=Phi0; a=zeros(M,m+1); Q_old=zeros(1,M); erQmax=0; eramax=0; eramaxr=0; a_old=a; Qsum=0; Q=zeros(1,M); asum=zeros(1,M); while error>1e-5 && NIT<100 for mm=1:M a(mm,:)=- conj(Cauchy_integral_phi(N,m,@(chi)z_of_Z(chi,z0(mm),R(mm)),@(z)O mega_total(z,mm,z0,R,a,Q,z_ref,M,m,Qx0,C))); Q(mm)=2*pi*(Phi_lake(mm)+real(a(mm,1)))/log(1/abs(chi_far(mm))); C=Phi0-Omega_total(z_ref,0,z0,R,a,Q,z_ref,M,m,Qx0,0); erQ=abs(Q(mm)-Q_old(mm)); Qsum=Qsum+abs(Q(mm)); if erQ>erQmax</pre>

	`` erQmax=erQ; end end for mm=1:M for kk=1:m+1 era=abs(a(mm,kk)-a_old(mm,kk)); asum(mm)=asum(mm)+abs(a(mm,kk)); if era>eramax eramax=era; end end erar=M*eramax/asum(mm); if erar>eramaxr eramaxr=erar; end end NIT=NIT+1 erQmaxr=M*abs(erQmax/Qsum); if erQmaxr>eramaxr error=erQmaxr; else error=eramaxr; end error a_old=a; Q_old=Q; Qsum=0; asum=zeros(1,M); erQmax=0; eramax=0; eramaxr=0; end end ```
Z_of_Z	``` function [z] = z_of_Z(Z,zm,rm) %UNTITLED Summary of this function goes here % Detailed explanation goes here z=Z *rm + zm ; end ```

A brief explanation of each function:

BZ _of_z : transforms a coordinate in z space to a coordinate in the unit circle space of the give lake

Z_of_z: transforms coordinates in big Z space of a given lake back to z space

Omega _lake: calculates the complex potential of a lake given the lakes a coefficients (calculated using the Cauchy integral), and the lake discharge Q

Omega_total: calculates the contribution of each lake, and the uniform flow at a point z

Solve_lakes_fulit: takes and iterative approach to solving for the interdependent quantities a, Q and C. The function iterations go as follows: solve for the taylor coefficients at a lake, solve for Q of that lake and C using the new coefficients. This process is repeated until either 100 iterations occur, or the values of the taylor coefficients and Q aren't changing more than a given tolerance in each iteration.

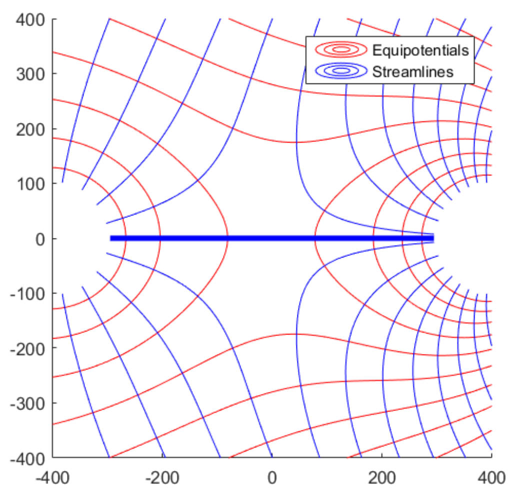
For question 1.2, only the first term of the Taylor coefficients is required, so the program was run using the following runfile:

Runfile 1.2

```
M = 2;
N = 5;
m =1;

Qx0 = .4;
z0 =[-400 , 400 ] ;
R = [100, 100];


Phi_lake=[150,200];

Phi0 = 50;
z_ref = -1000;



chi_far = zeros (M,1);
for mm  = 1:M

   chi_far(mm) = BZ_of_z(z_ref, z0(mm),R(mm));

end

[a ,Q,C]
=solve_lakes_fulit(Qx0,Phi0,Phi_lake,M,N,m,z0,R,chi_far,z
_ref) ;
```

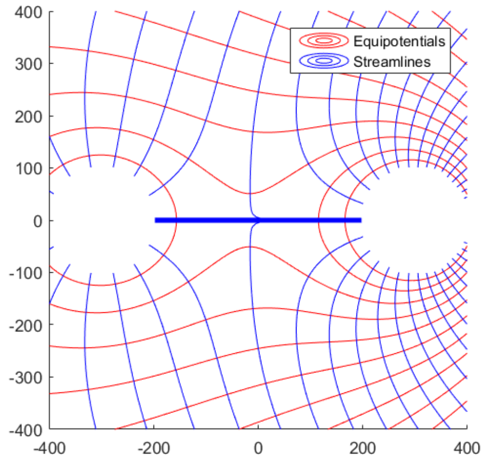
	`` ContourMe_flow_net(-400,400,100,- 400,400,100,@(z)Omega_total(z, 0 ,z0, R, a, Q, z_ref, M,m, Qx0, C),50); %ContourMe_R_int(-400,400,100,- 400,400,100,@(z)Omega_total(z, 0 ,z0, R, a, Q, z_ref, M,m, Qx0, C),60); Potential_at_lake_1 = real(Omega_total(z0(1) + R(1), 0 ,z0, R, a, Q, z_ref, M,m, Qx0, C)) Potential_at_lake_2 = real(Omega_total(z0(2) + R(2), 0 ,z0, R, a, Q, z_ref, M,m, Qx0, C)) ```

The following flownets were generated by moving the lakes together. The left lake has given potential of 150, the right lake has a potential of 200.



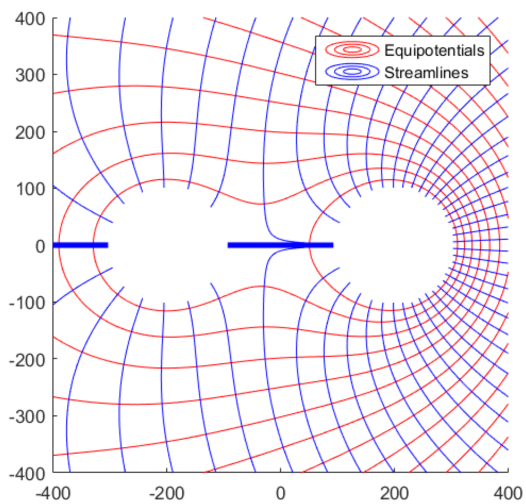
Phi lake 1 = 152

Phi lake 2 = 200



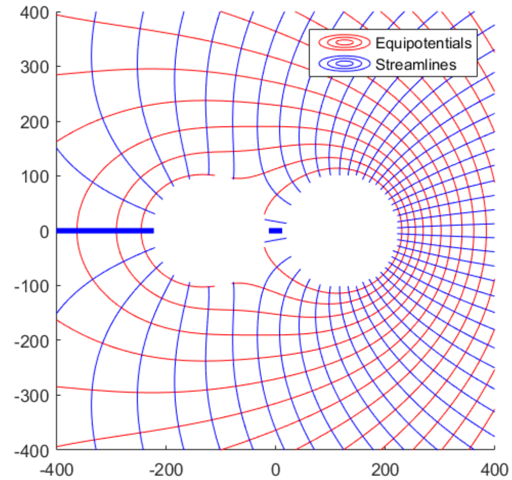
Phi lake 1 = 153

Phi lake 2 = 201



Phi lake 1 = 157

Phi lake 2 = 202



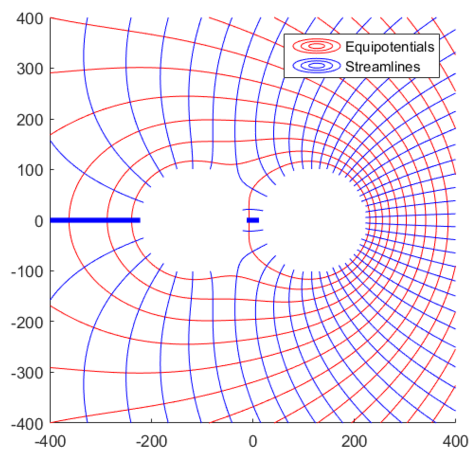
Phi lake 1 = 170

Phi lake 2 = 202

As the lakes move closer together the potential along their boundary deviates from its given value when only one term in the taylor expansion is used.

Part 2

This part of the program uses the same code as 1.2, but the variable m, which dictates the number of taylor coefficients is increased to 20. This ensures that the potential along each lake boundary is exactly the given potential.



Phi lake 1 = 150

Phi lake 2 = 200

The flownet is noticeably different than the version where only 1 taylor series term is used, especially the area between the lakes. Furthermore, the potential along the lake boundary is exactly the given value.