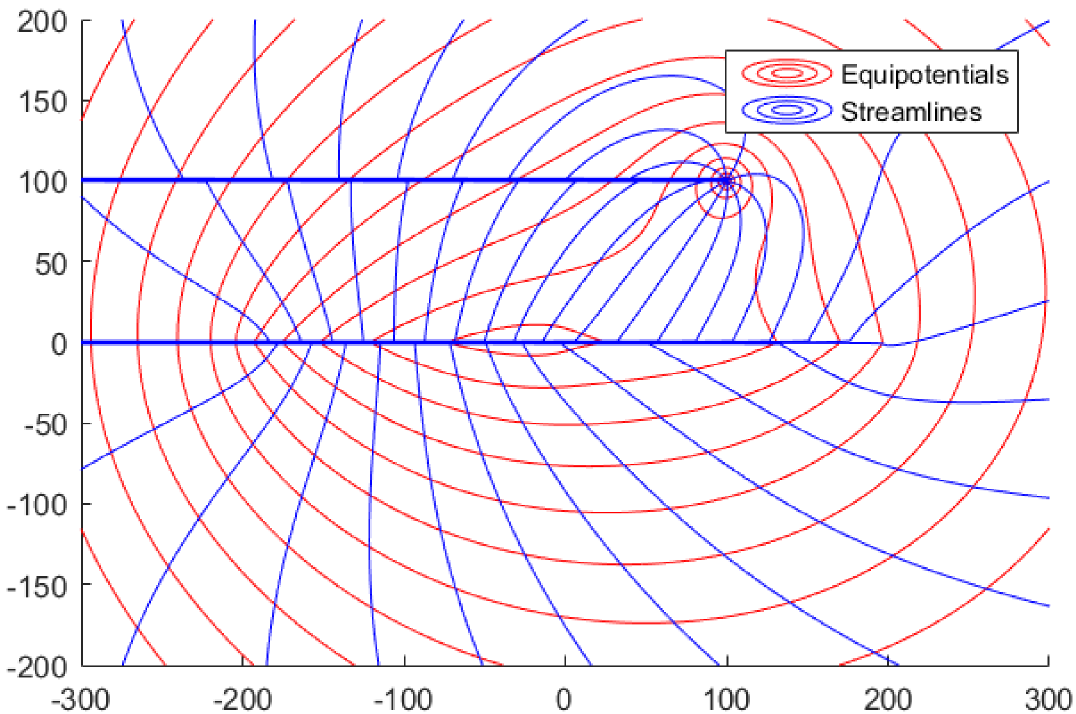
1. With well on :
   Strength of linesink running -2d to 0 : 7.8 m^2/d
   Strength of linesink running 0 to 2d: 4.3m^3/d
   Total extraction = 2420 m^3/d

   Without well on:
   Strength of linesink running -2d to 0 : 8.3 m^3/d
   Strength of linesink running 0 to 2d: 6.8m^3/d
   Total extraction =3020 m^3/d

2. See matlab output. The head at these points does match the reference values.



3. Extraction without well – extraction with well = 600 m^3/d

   Therefore the well is taking 600 m^3/d from the canal, so 200 m^3/d come from infinity

   Code:

HW05_Run.m	`k=10;`

	`` d=100; zs=-2*d; ze=2*d; Qx0=0.4; %Qx0=0; z0=1000; fi0=25; Phi0=.5*k*fi0^2; fi1=28; Phi1=.5*k*fi1^2; %% Solve for Strength of Line Sink Phi_M = [Phi0;Phi0;Phi1]; %last entry is the far field LS_c = [-d;d;z0]; %location of each refrence point in Phi_M alpha = 0; %angle of uniform flow relative to vertical LS_end=[zs,0;0,ze]; %endpoints of each line sink zw = [d+1i*d;-d+1i*d;-d-1i*d;d-1i*d]; Q = [800;00;00;00]; rw = [0.1;0.1;.1;.1]; b = Populate_b(Phi_M,Qx0,LS_c,alpha, zw,rw,Q); A = Populate_A(LS_end,LS_c); s = A\b; ContourMe_flow_net(-300,300 , 300, -200, 200, 200, @(z)Omega_total(z,Qx0,alpha, s, LS_end,zw,rw,Q),30); head_at_center_1 = sqrt (2* real(Omega_total(-d,Qx0,alpha, s, LS_end,zw,rw,Q))/k) head_at_center_2 = sqrt (2* real(Omega_total(d,Qx0,alpha, s, LS_end,zw,rw,Q))/k) head_at_refrence = sqrt(2*real(Omega_total(z0,Qx0,alpha, s, LS_end,zw,rw,Q))/k) ```

Calculate_L.m	``` function [L] = Calculate_L(z1,z2) L=sqrt((real(z1)-real(z2))^2 +(imag(z1)- imag(z2))^2); end ```
Calculate_Z.m	``` function [Z] = Calculate_Z(z,z1,z2) Z=(z-.5*(z2+z1))/(.5*(z2-z1)); end ```
Contour_me_flownet.m	``` function [Grid] = ContourMe_flow_net(xfrom, xto, Nx, yfrom, yto, Ny, func,nint) %== ============================ % ContourMe_I(xfrom, xto, Nx, yfrom, yto, Ny, func) (01.23.09) % % Contour the imaginary part of the specified complex function. % % Arguments: % % xfrom starting x-value for the domain % xto ending x-value for the domain % Nx number of grid columns % % yfrom starting y-value for the domain % yto ending y-value for the domain % Ny number of grid rows % % func function to contour; must take one complex argument. % % Returns: % ```

	`` % Grid Ny x Nx matrix of values of func at the rid nodes. % % Example Usage: % % G = ContourMe_I(1,2,11,1,2,11,@(z)Omega(1,-1,z)); %== Grid = zeros(Ny,Nx); X = linspace(xfrom, xto, Nx); Y = linspace(yfrom, yto, Ny); for row = 1:Ny for col = 1:Nx Grid(row,col) = func(complex(X(col), Y(row))); end end Bmax=max(imag(Grid)); Bmin=min(imag(Grid)); Cmax=max(Bmax); Cmin=min(Bmin); D=Cmax-Cmin; del=D/nint; Bmax=max(real(Grid)); Bmin=min(real(Grid)); Cmax=max(Bmax); Cmin=min(Bmin); D=Cmax-Cmin; nintr=round(D/del); figure; hold on contour(X, Y,real(Grid),nintr,'r'); contour(X, Y,imag(Grid),nint,'b'); legend('Equipotentials','Streamlines') axis square axis equal %hold on %contour(X, Y,real(Grid),nintr); %contour(X, Y,imag(Grid),nint); %axis equal ```

LS.m	<pre>function [Omega] = LS(Z,L) if abs(Z+1)<10^-5 || abs(Z-1) <10^-5 Omega = 0; else Omega = L/(4*pi) * ((Z+1)*log(Z+1)-(Z- 1)*log(Z-1)-2); end end</pre>
Omega_ls.m	<pre>function [Omega] =Omega_ls(z,Qx0,alpha, s, LS_end,zw,rw,Q) LS_array = nan(length(s),1); LS_array(length(LS_array),1) = 1; for m = 1:length(LS_array)-1 z1=LS_end(m,1); z2=LS_end(m,2); Z=Calculate_Z(z,z1,z2); L=Calculate_L(z1,z2); LS_array(m,1) = LS(Z,L); end Omega = -Qx0*z*exp(-1i*alpha) + dot(LS_array , s)+ Omega_well(z,zw,rw,Q); end</pre>
Omega_total. m	<pre>function [Omega] =Omega_total(z,Qx0,alpha, s, LS_end,zw,rw,Q) LS_array = nan(length(s),1); LS_array(length(LS_array),1) = 1; for m = 1:length(LS_array)-1 z1=LS_end(m,1); z2=LS_end(m,2); Z=Calculate_Z(z,z1,z2); L=Calculate_L(z1,z2);</pre>

	`` LS_array(m,1) = LS(Z,L); end Omega = -Qx0*z*exp(-1i*alpha) + dot(LS_array , s); for j=1:numel(zw) Omega = Omega + Omega_well(z,zw(j),rw(j),Q(j)); end end ```
Omeag_well.m	``` function [Omega] = Omega_well(z,z0,rw,Q) rsq=(z-z0)*conj(z-z0); if rsq>rw^2 Omega=Q/(2*pi)*log(z-z0); else Omega = 0; end ```
Phi_g	``` function [Phi] = Phi_g(Qx0,z,alpha) Phi = real(-Qx0*z*exp(-1i*alpha)); end ```
Populate_A.m	``` function [A] = Populate_A(LS_end,LS_c) [h,l] = size(LS_end); height=h+1; width=h+1; A=nan(height,width); A(:,width)=1; for m=1:height for j=1:width-1 z1=LS_end(j,1); z2=LS_end(j,2); z=LS_c(m); Z=Calculate_Z(z,z1,z2); L=Calculate_L(z1,z2); ```

	`` A(m,j)= real(LS(Z,L)); end end end ```
Populate_B.m	``` function [b] = Populate_b(Phi_M,Qx0,LS_c,alpha, zw,rw,Q) b = zeros(numel(Phi_M),1); for i=1:numel(Phi_M) b(i)=real(Phi_M(i)- Phi_g(Qx0,LS_c(i),alpha)); for j=1:numel(zw) b(i) = b(i) - real(Omega_well(LS_c(i),zw(j),rw(j),Q(j))); end end end ```
Real_ls.m	``` function [Phi] = Real_LS(Z,L) Phi = real(LS(Z,L)); end ```