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

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

C:\Users\Jack\Documents\GW modeling\Original work\GW modeling hw 5\figure.tif

1. 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 | 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 |
| Omega\_ls.m | 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 |
| Omega\_total.m | 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);    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 |
|  |  |
|  |  |
|  |  |
|  |  |