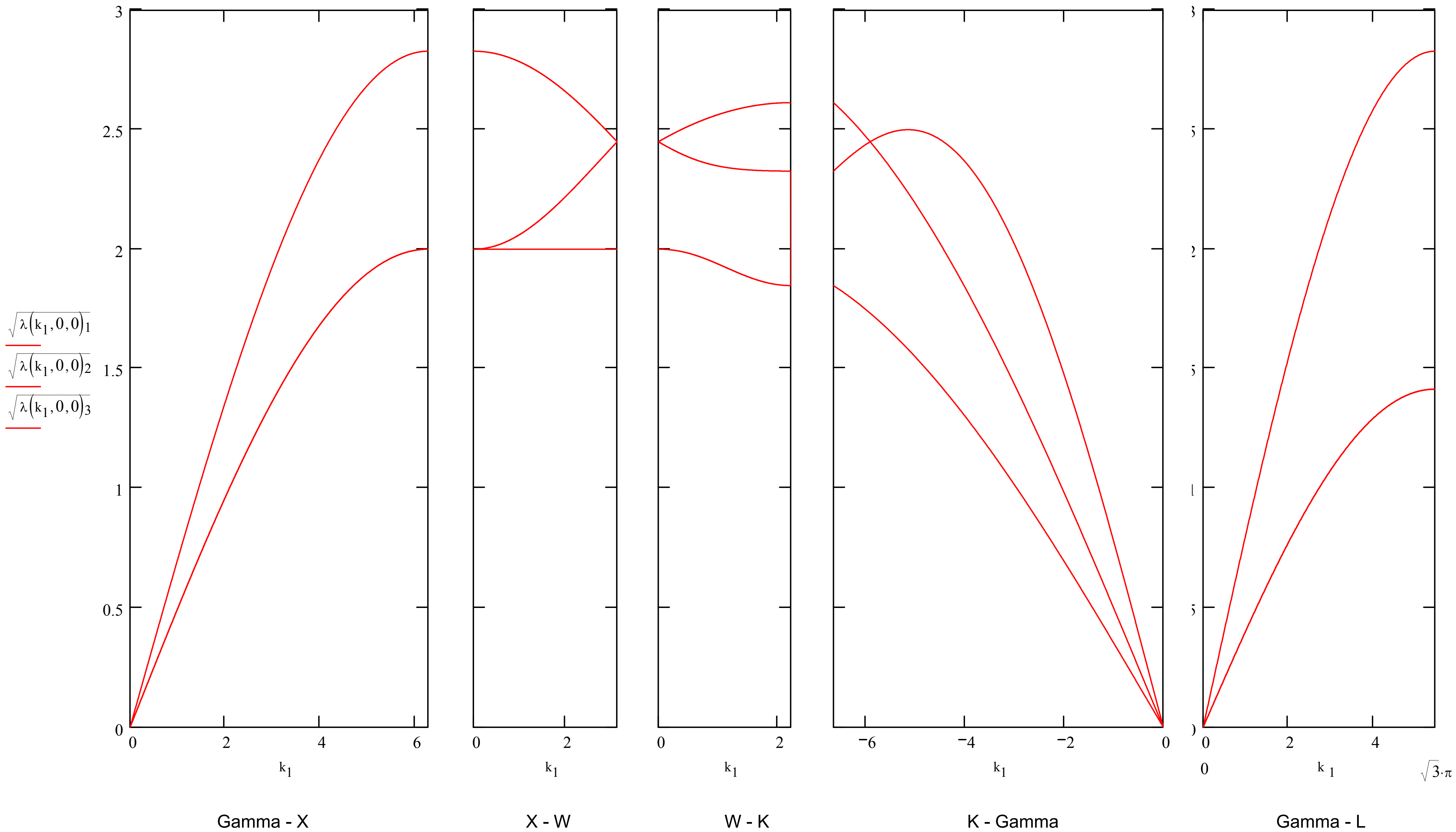
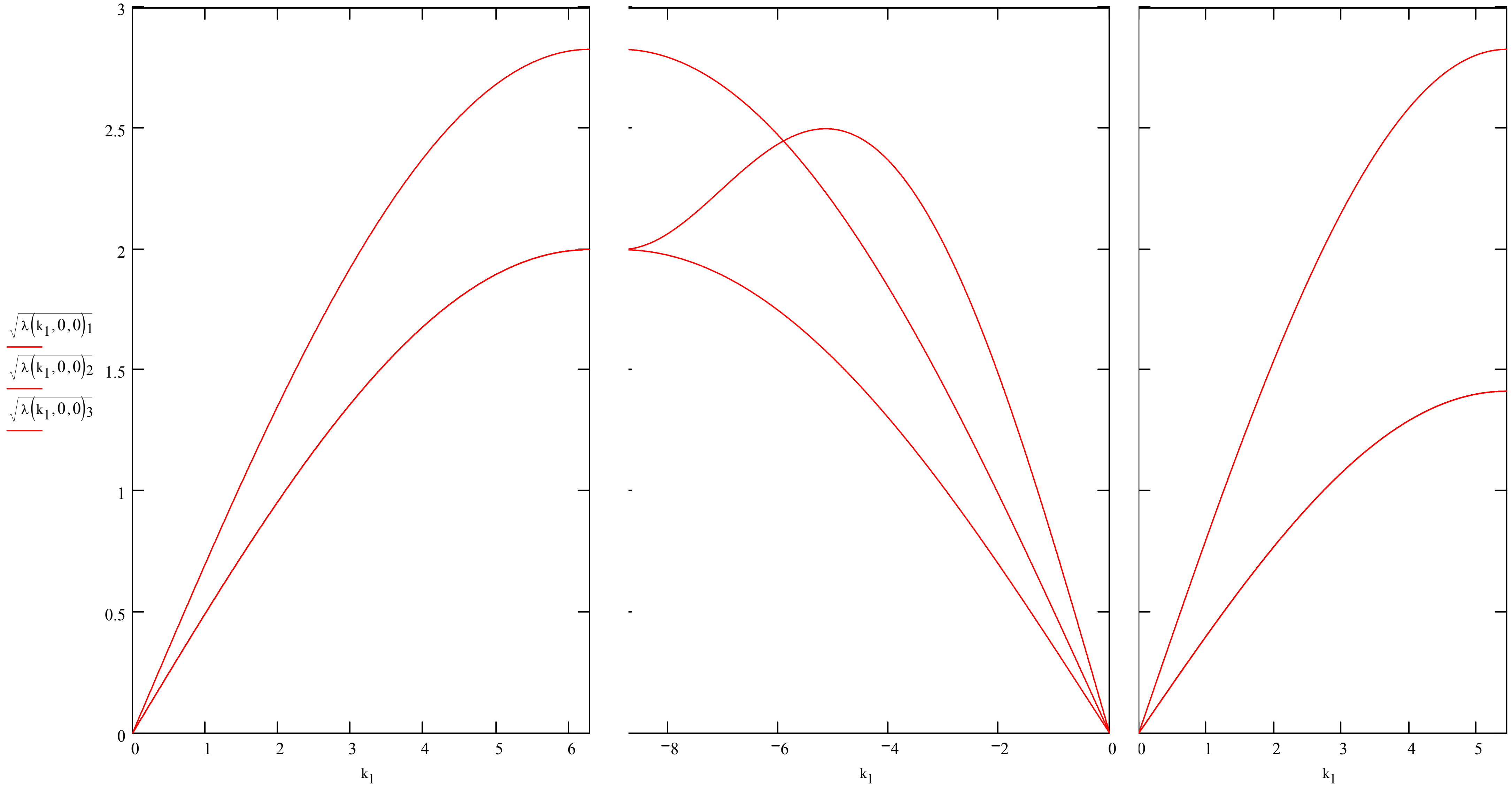
$$M\left(k_x,k_y,k_z\right) := \begin{bmatrix} 4-\cos\left[\left(k_x+k_y\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_x-k_y\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_x+k_z\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_x-k_z\right)\cdot\frac{a_0}{2}\right] & -\cos\left[\left(k_x+k_y\right)\cdot\frac{a_0}{2}\right]+\cos\left[\left(k_x-k_y\right)\cdot\frac{a_0}{2}\right] & -\cos\left[\left(k_x+k_z\right)\cdot\frac{a_0}{2}\right]+\cos\left[\left(k_x-k_z\right)\cdot\frac{a_0}{2}\right] \\ -\cos\left[\left(k_x+k_y\right)\cdot\frac{a_0}{2}\right]+\cos\left[\left(k_x-k_y\right)\cdot\frac{a_0}{2}\right] & 4-\cos\left[\left(k_x+k_y\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_x-k_y\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_y+k_z\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_y-k_z\right)\cdot\frac{a_0}{2}\right] & -\cos\left[\left(k_y+k_z\right)\cdot\frac{a_0}{2}\right]+\cos\left[\left(k_y-k_z\right)\cdot\frac{a_0}{2}\right] \\ -\cos\left[\left(k_x+k_z\right)\cdot\frac{a_0}{2}\right]+\cos\left[\left(k_x-k_z\right)\cdot\frac{a_0}{2}\right] & -\cos\left[\left(k_y+k_z\right)\cdot\frac{a_0}{2}\right]+\cos\left[\left(k_y-k_z\right)\cdot\frac{a_0}{2}\right] & 4-\cos\left[\left(k_x+k_z\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_x-k_z\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_y+k_z\right)\cdot\frac{a_0}{2}\right]-\cos\left[\left(k_y-k_z\right)\cdot\frac{a_0}{2}\right] \end{bmatrix}$$

$$\lambda\left(k_x,k_y,k_z\right) := \mathrm{eigenwerte}\left(M\left(k_x,k_y,k_z\right)\right)$$





100

110

111

```
% solid state physics
% calculation of dispersion relations for monoatomic 3D lattices
% Philipp Thaler
% 4.5.09


% parameters
a = 4.49;      % length of a conventional unit cell in A
C = 4/2;      % spring force constant in N/m
m = 0.344*10^-24;      % mass in kg

% postitions of the examined atom and the nearest neighbors
% fcc:
an = a/2 * [0,0,0; 1,1,0; 1,0,1; 0,1,1; -1,-1,0; -1,0,-1; 0,-1,-1; 1,-1,0; 1,0,-1; 0,1,-1;
-1,1,0; -1,0,1; 0,-1,1];

% wavevector to plot 100-110-111 for fcc
nk = 100;
k0 = linspace(0,0,nk);
k100 = linspace(0,2*pi,nk)/a;
k110 = linspace(-2*pi*sqrt(2),0,nk)/a;
k111 = linspace(0,pi*sqrt(3),nk)/a;
kx = [k100,   k110/sqrt(2),  k111/sqrt(3)];
ky = [k0,     k110/sqrt(2),  k111/sqrt(3)];
kz = [k0,     k0,            k111/sqrt(3)];
kk = [k100,   k100(end)+fliplr(-k110), k100(end)-k110(1)+k111];

% calculate the frequencies
w = NaN(3,length(kk));
for n = 1:length(kk)
    k = [kx(n), ky(n), kz(n)];
    w(:,n) = disp_solver(an, m, C, k);
end

%plot fcc:
figure(1)
plot(kk, w(1,:),'b-',kk, w(2,:),'b-',kk, w(3,:),'b-')
hold on
%Points Gamma, X, K, Gamma,L
ymax = round(10*1.2*max(w(:)))/10;
plot([0,0],[0,ymax],'r-')
plot([2*pi,2*pi]/a,[0,ymax],'r-')
plot([2*pi+2*pi*sqrt(2)-sqrt(9/2)*pi,2*pi+2*pi*sqrt(2)-sqrt(9/2)*pi]/a,[0,ymax],'r-')
plot([2*pi+2*pi*sqrt(2),2*pi+2*pi*sqrt(2)]/a,[0,ymax],'r-')
plot([2*pi+2*pi*sqrt(2)+pi*sqrt(3),2*pi+2*pi*sqrt(2)+pi*sqrt(3)]/a,[0,ymax],'r-')

xlim([kk(1) kk(end)])
ylim([0 ymax])
hold off
```

```
% solver for the eigenvalue problem of 3d dispersion relations
% philipp thaler
% 8.5.09

function w = disp_solver(an, m, C, k)

%%%%%%%%%%
% solver %
%%%%%%%%%%

% directions of the neighbors
ar =an(2:end,:);
arnorm = zeros(size(ar,1),1);
for n = 1:size(ar,1)
    arnorm(n) = norm(ar(n,[1,2,3]));
    ar(n,[1,2,3]) = ar(n,[1,2,3])./arnorm(n);
end

% Newton's law
% m * d2u/dt2 = sum( C * delta_u)

% approach
% u = u0 * exp(-iwt) * exp(iak)

% split in 3 dimensions, using the approach for u
% -m/C * w^2 * ux + sum(delta_u) = 0
% -m/C * w^2 * uy + sum(delta_u) = 0
% -m/C * w^2 * uz + sum(delta_u) = 0

% delta_u = u(k',l',m')-u(k,l,m)
% for nearest neighbor #1 k'=k+1, l'=l+1, m'=m
% --> delta_u = u0 * exp(-iwt) * exp(i(k*kx+l*ky+m*kz)) * (exp((k'-k)kx + (l'-l)ky +
(m'-m)kz))-1)

for n = 1:size(ar,1)
    % x-direction, sum(delta_u):
    delta_uxx = (ar(:,1).*ar(:,1)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    delta_uxy = (ar(:,1).*ar(:,2)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    delta_uxz = (ar(:,1).*ar(:,3)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    % y-direction, sum(delta_u):
    delta_uyx = (ar(:,2).*ar(:,1)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    delta_uyy = (ar(:,2).*ar(:,2)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    delta_uyz = (ar(:,2).*ar(:,3)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    % z-direction, sum(delta_u):
    delta_uzx = (ar(:,3).*ar(:,1)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    delta_uzy = (ar(:,3).*ar(:,2)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
    delta_uzz = (ar(:,3).*ar(:,3)).*(exp(i*sum(an(2:end,:).*repmat(k,size(ar,1),1),2))-1);
end


M =
-real([sum(delta_uxx,1),sum(delta_uxy,1),sum(delta_uxz,1);sum(delta_uyx,1),sum(delta_uyy,1),
sum(delta_uyz,1);sum(delta_uzx,1),sum(delta_uzy,1),sum(delta_uzz,1)]);
% eigs(M) = w^2 * m/C
w = real(sqrt(eigs(M)*C/m));
```