1. Regional Stress

1.1. GeoTaos::CFS working sheet for regional stress

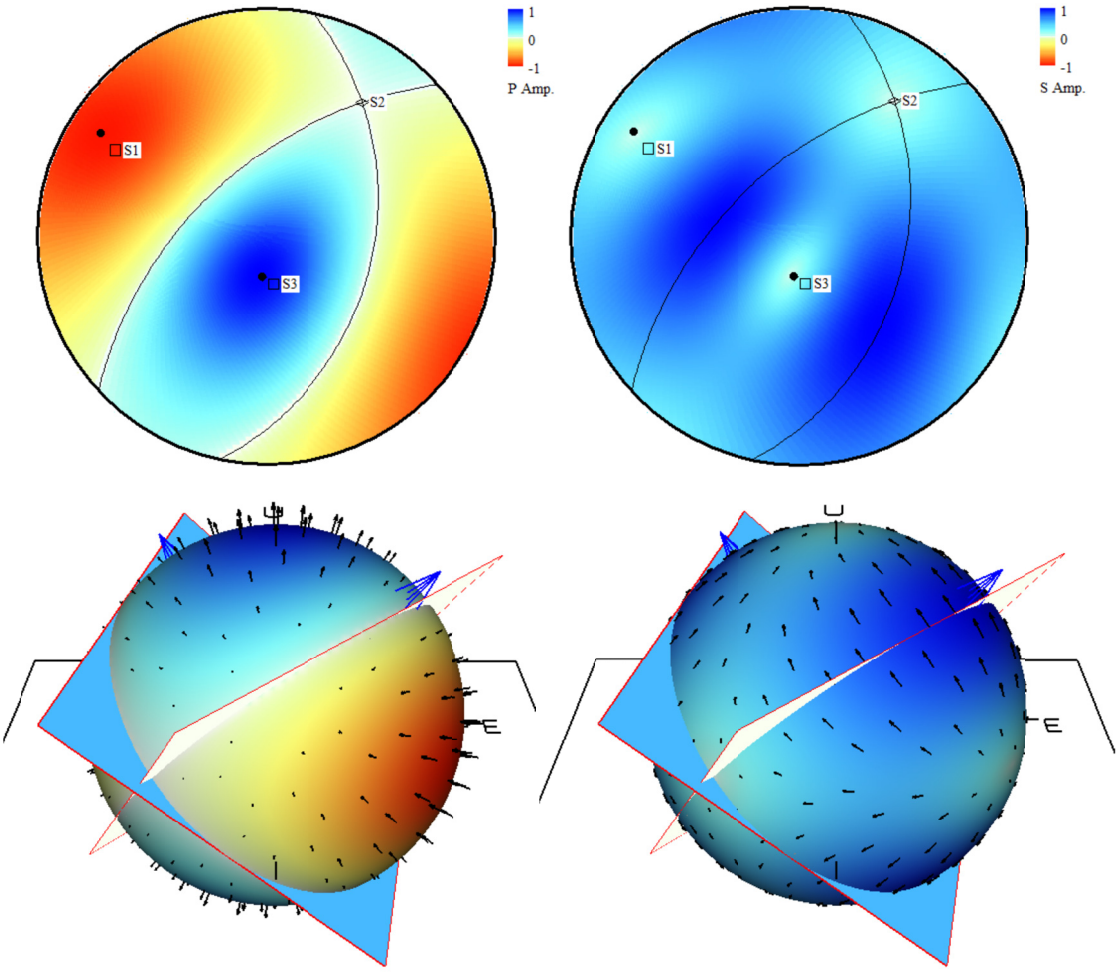
5	Regional background stress.				
6	Pattern	4: 3+Lithostatic			Select a pattern matching your
7	Sigma-#	Sigma-1	Sigma-2	Sigma-3	The principal stress axes
8	Az, N+	120	35	-7.440136	deg. Azimuth and dip of the prin
9	Ad,H+	15	-18.018135	6621427	S1, S2, S3 form a right-hand co
10	Stress	40	30	20	MPa, S1>=S2>=S3, magnitude c
?		1,2(S)=>2(D),3..	1,3(S)=>2,3(D)	Check.	Make/check direction of princi
12	Opt.Faul..	0: P (LHS projection)			Draw radiation parttern of the
13					

Part of [Coulomb] working sheet

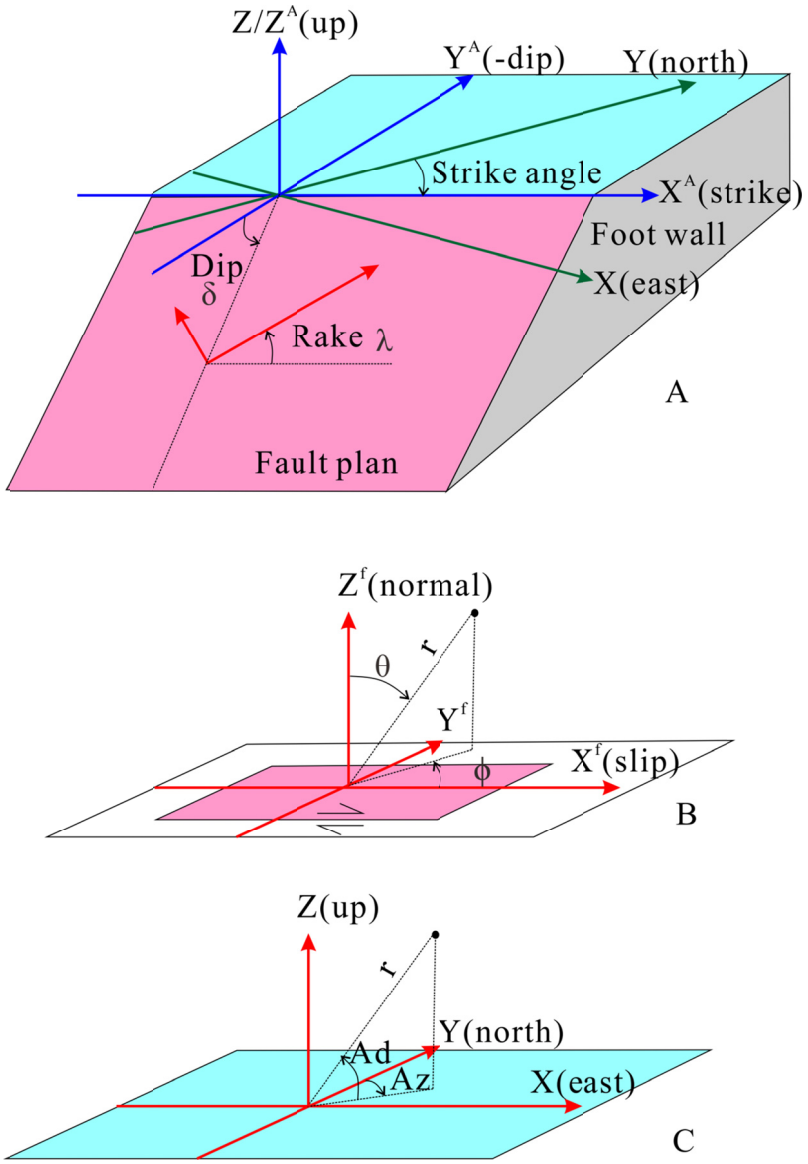
Row-11: "1,2(S)=>2(D),3". Known Az, Ad of the 1st principal stress and Az of the 2nd principal stress calculate all left angles.

Row-11: "1,3(S)=>2,3(D)". Known Az, Ad of the 1st principal stress and Az of the third principal stress calculate all left angles.

Row-12: Determine the optimally oriented fault from the directions of the principal stresses and the given frictional coefficient by grid (3, B), and draw the selected radiation pattern.



1.2. GeoTaos: Coordinate systems used in GeoTaos



Geographic coordinate system:

Cartesian coordinates: X(east), Y(north), Z(up).

Spherical polar coordinates: r, θ, ϕ.

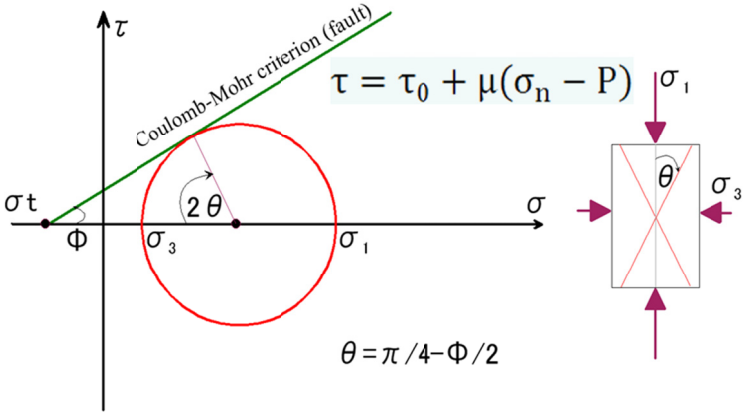
Fault coordinate system: X^f(slip direction), Y^f, Z^f(normal)

Aki & Richard convention (2002, P.101) for the fault orientation parameters. The strike angle (ϕ_S) is measured clockwise round from North, with the fault dipping down to the right of the strike direction: $0 \le \phi_S < 2\pi$. The dip angle is measured down from the horizontal: $0 \le \delta < 2\pi$. The slip direction is specified by the rake λ, measured in the fault plan as the angle between the directions of strike and slip.

In the calculation of deformation due to dislocation along a fault embedded in a half infinite space the fault system (Okada, 1982) is used. Where, X, Y, Z correspond to dip, strike, and up directions, respectively.

1.2. GeoTaos: Draw focal spheres of the optimally-oriented fault under a given regional stress

When the directions of the three principal stresses and the frictional coefficient (μ) are given, following the Coulomb-Mohr criterion, one can define an optimally-oriented fault (ϕ_S, δ, λ) so that the angle between the directions of slip and the maximum principal stress is θ=45-atan(μ).



Codes for making the optimally-orientated fault

```
XL_Vector3D vS1, vS3, vB, vN;
vS1.Make_Rad(prnStrs[i1].fStrike, prnStrs[i1].fDip); //Vector for Sigma-1
vS3.Make_Rad(prnStrs[i3].fStrike, prnStrs[i3].fDip); //Vector for Sigma-3

float fThe = (PI_2 - atan(pThis->faultCoulomb.fFrict))/2;      //Angle between slip and Sigma-1
vN = -vP*sin(fThe) + vT*cos(fThe);                            //Normal of the fault
vN.Normalize();
if(vN.z < 0) vN *= -1;
float fStrike = -atan2(vN.y, vN.x);                           //Strike angle of the fault
float fDip    =  acos(vN.z);                                  //Dip angle of the fault

XL_EQMT eq;
float fRake = eq.CalculateRake_Rad(fStrike, fDip, prnStrs[i1].fStrike, prnStrs[i1].fDip);
eq.Create(fStrike, fDip, fRake);   //create the fault

int nR = 8000;
CPoint p0 = CPoint(x0+nR, y0-nR);
eq.DrawRadiation(pDC, p0, nR, nFlag);                  //Draw ration pattern
```

Codes for drawing slip vectors in the focus sphere of the optimally-orientated fault

```
// calculate fault vectors vStrike, vDip, vNorm, and vSlip in X(E-W), Y(S-N), Z(U-D) coordinate system
XL_EQMT::Fault_Vectors(fStrike, fDip, fRake, vStrike, vDip, vNorm, vSlip);

XL_Transform3D t3;
//for transform vector from fault coordinates to X(E-W), Y(S-N), Z(U-D) coordinates
t3.CreateTransformMatrix(vSlip, XL_Vector3D::interline(vNorm, vSlip), vNorm);
triangles = XL_Sphere::createsphere(2);      //generate homogeneous triangles in the sphere
i = triangles.begin();
while (i != triangles.end()) {
       Triangle3D t = *i++;
       //the center of the triangle
```

```
        vCal.x = (t.v0[0] + t.v1[0] + t.v2[0]);
        vCal.y = (t.v0[1] + t.v1[1] + t.v2[1]);
        vCal.z = (t.v0[2] + t.v1[2] + t.v2[2]);
        vCal.Normalize();

        fThe = acos(vCal.z);                            //[0, PI]
        fFai = atan2(vCal.y, vCal.x);                   //[-PI, PI]

        //radiation vector
        if(nFlag==0) {     // P radiation
                fAmp = sin(2.0*fThe)*cos(fFai)*0.5;
                vRad = vCal * fAmp;
        }else{
                fSR  =  cos(2.0*fThe)*cos(fFai)*0.5;
                fSH  = -cos(fThe)*sin(fFai)*0.5;
                vRad.x = cos(fThe)*cos(fFai) * fSR;
                vRad.y = cos(fThe)*sin(fFai) * fSR;
                vRad.z = -sin(fThe) * fSR;
                vRad.x += -sin(fFai) * fSH;
                vRad.y +=  cos(fFai) * fSH;
        }
        //so far radiation is calculated in the fault coordinate system
        //transform calculation point and radiation vector to (E, N, U) system
        t3.Transform(&vCal);
        float fRp = (nFlag==0)? fR : fR+0.02;
        x = vCal.x * fRp;
        y = vCal.y * fRp;
        z = vCal.z * fRp;
        t3.Transform(&vRad);
        if (nFlag==0 && fAmp<0)
                XLgl_DrawArrow2(x-vRad.x, y-vRad.y, z-vRad.z, vRad.x, vRad.y, vRad.z);
        else
                XLgl_DrawArrow2(x, y, z, vRad.x, vRad.y, vRad.z);

}
```