

2D geometry

Define $\text{orient}(A, B, C) = \overline{AB} \times \overline{AC}$. CCW iff > 0 .

Define $\text{perp}((a, b)) = (-b, a)$. The vectors are orthogonal.

For line $ax + by = c$ def $\bar{v} = (-b, a)$.

Line through P and Q has $\bar{v} = \overline{PQ}$ and $c = \bar{v} \times P$.

$\text{side}_l(P) = \bar{v}_l \times P - c_l$ sign determines which side P is on from l .

$\text{dist}_l(P) = \text{side}_l(P) / \|\bar{v}_l\|$ squared is integer.

Sorting points along a line: comparator is $\bar{v} \cdot A < \bar{v} \cdot B$.

Translating line by \bar{t} : new line has $c' = c + \bar{v} \times \bar{t}$.

Line intersection: is $(c_l \bar{v}_m - c_m \bar{v}_l) / (\bar{v}_l \times \bar{v}_m)$.

Project P onto l : is $P - \text{perp}(v) \text{side}_l(P) / \|\bar{v}\|^2$.

Angle bisectors: $\bar{v} = \bar{v}_l / \|\bar{v}_l\| + \bar{v}_m / \|\bar{v}_m\|$

$c = c_l / \|\bar{v}_l\| + c_m / \|\bar{v}_m\|$.

P is on segment AB iff $\text{orient}(A, B, P) = 0$ and $\overline{PA} \cdot \overline{PB} \leq 0$.

Proper intersection of AB and CD exists iff $\text{orient}(C, D, A)$

and $\text{orient}(C, D, B)$ have opp. signs and $\text{orient}(A, B, C)$ and

$\text{orient}(A, B, D)$ have opp. signs. Coordinates:

$$\frac{A \text{orient}(C, D, B) - B \text{orient}(C, D, A)}{\text{orient}(C, D, B) - \text{orient}(C, D, A)}.$$

Circumcircle center:

```
pt circumCenter(pt a, pt b, pt c) {
    b = b-a, c = c-a; // consider coordinates relative to A
    assert(cross(b,c) != 0); // no circumcircle if A,B,C
        aligned
    return a + perp(b*sq(c) - c*sq(b))/cross(b,c)/2;
```

Circle-line intersect:

```
int circleLine(pt o, double r, line l, pair<pt,pt> &out) {
    double h2 = r*r - l.sqDist(o);
    if (h2 >= 0) { // the line touches the circle
        pt p = l.proj(o); // point P
        pt h = l.v*sqrt(h2)/abs(l.v); // vector paral to l, of
            len h
        out = {p-h, p+h};
    }
    return 1 + sgn(h2);
```

Circle-circle intersect:

```
int circleCircle(pt o1, double r1, pt o2, double r2, pair<
    pt,pt> &out) {
```

```
    pt d=o2-o1; double d2=sq(d);
    if (d2 == 0) {assert(r1 != r2); return 0;} // concentric
        circles
    double pd = (d2 + r1*r1 - r2*r2)/2; // = |O_1P| * d
    double h2 = r1*r1 - pd*pd/d2; // = h^2
    if (h2 >= 0) {
        pt p = o1 + d*pd/d2, h = perp(d)*sqrt(h2/d2);
        out = {p-h, p+h};
    }
    return 1 + sgn(h2);
```

Tangent lines:

```
int tangents(pt o1, double r1, pt o2, double r2, bool
    inner, vector<pair<pt,pt>> &out) {
    if (inner) r2 = -r2;
    pt d = o2-o1;
    double dr = r1-r2, d2 = sq(d), h2 = d2-dr*dr;
    if (d2 == 0 || h2 < 0) {assert(h2 != 0); return 0;}
    for (double sign : {-1,1}) {
        pt v = (d*dr + perp(d)*sqrt(h2)*sign)/d2;
        out.push_back({o1 + v*r1, o2 + v*r2});
    }
    return 1 + (h2 > 0);
```

3D geometry

$\text{orient}(P, Q, R, S) = (\overline{PQ} \times \overline{PR}) \cdot \overline{PS}$. S above PQR iff > 0 .

For plane $ax + by + cz = d$ def $\bar{n} = (a, b, c)$.

Line with normal \bar{n} through point P has $d = \bar{n} \cdot P$.

$\text{side}_\Pi(P) = \bar{n} \cdot P - d$ sign determines side from Π .

$\text{dist}_\Pi(P) = \text{side}_\Pi(P) / \|\bar{n}\|$.

Translating plane by \bar{t} makes $d' = d + \bar{n} \cdot \bar{t}$.

Plane-plane intersection of has direction $\bar{n}_1 \times \bar{n}_2$ and goes

through $((d_1 \bar{n}_2 - d_2 \bar{n}_1) \times \bar{d}) / \|\bar{d}\|^2$.

Line-line distance:

```
double dist(line3d l1, line3d l2) {
    p3 n = l1.d*l2.d;
    if (n == zero) // parallel
        return l1.dist(l2.o);
    return abs((l2.o-l1.o)|n)/abs(n);
```

Spherical to Cartesian: $(r \cos \varphi \cos \lambda, r \cos \varphi \sin \lambda, r \sin \varphi)$.

Sphere-line intersection:

```
int sphereLine(p3 o, double r, line3d l, pair<p3,p3> &out)
    {
        double h2 = r*r - l.sqDist(o);
        if (h2 < 0) return 0; // the line doesn't touch the
            sphere
```

```
p3 p = l.proj(o); // point P
p3 h = l.d*sqrt(h2)/abs(l.d); // vector parallel to l,
    of length h
    out = {p-h, p+h};
    return 1 + (h2 > 0);
```

Great-circle distance between points A and B is $r \angle AOB$.

Spherical segment intersection:

```
bool properInter(p3 a, p3 b, p3 c, p3 d, p3 &out) {
    p3 ab = a*b, cd = c*d; // normals of planes OAB and OCD
    int oa = sgn(cd|a),
        ob = sgn(cd|b),
        oc = sgn(ab|c),
        od = sgn(ab|d);
    out = ab*cd*od; // four multiplications => careful with
        overflow !
    return (oa != ob && oc != od && oa != oc);
}
bool onSphSegment(p3 a, p3 b, p3 p) {
    p3 n = a*b;
    if (n == zero)
        return a*p == zero && (a|p) > 0;
    return (n|p) == 0 && (n|a*p) >= 0 && (n|b*p) <= 0;
}
struct directionSet : vector<p3> {
    using vector::vector; // import constructors
    void insert(p3 p) {
        for (p3 q : *this) if (p*q == zero) return;
        push_back(p);
    }
};
directionSet intersSph(p3 a, p3 b, p3 c, p3 d) {
    assert(validSegment(a, b) && validSegment(c, d));
    p3 out;
    if (properInter(a, b, c, d, out)) return {out};
    directionSet s;
    if (onSphSegment(c, d, a)) s.insert(a);
    if (onSphSegment(c, d, b)) s.insert(b);
    if (onSphSegment(a, b, c)) s.insert(c);
    if (onSphSegment(a, b, d)) s.insert(d);
    return s;
}
```

Angle between spherical segments AB and AC is angle between $A \times B$ and $A \times C$.

Oriented angle: subtract from 2π if mixed product is negative.

Area of a spherical polygon:

$$r^2[\text{sum of interior angles} - (n-2)\pi].$$