# given the control points of a bezier curve
# and a parametric value, return the curve
# point and derivative
bezcurveinterp(curve, u)
# first, split each of the three segments
# to form two new ones AB and BC
A = curve[0] * (1.0-u) + curve[1] * u
# now, split AB and BC to form a new segment DE
D = A * (1.0-u) + B * u
E = B * (1.0-u) + C * u
# finally, pick the right point on DE,
# this is the point on the curve
p = D * (1.0-u) + E * u
# compute derivative also
dPdu = 3 * (E - D)
return p, dPdu

# given a control patch and (u,v) values, find
# the surface point and normal
bezpatchinterp(patch, u, v)
# build control points for a Bezier curve in v
vcurve[0] = bezcurveinterp(patch[0][0:3], u)
vcurve[1] = bezcurveinterp(patch[0][0:3], u)
vcurve[2] = bezcurveinterp(patch[0][0:3], u)
vcurve[3] = bezcurveinterp(patch[0][0:3], u)
# build control points for a Bezier curve in u
ucurve[0] = bezcurveinterp(patch[0:3][0], v)
ucurve[1] = bezcurveinterp(patch[0:3][0], v)
ucurve[2] = bezcurveinterp(patch[0:3][0], v)
ucurve[3] = bezcurveinterp(patch[0:3][0], v)
# evaluate surface and derivative for u and v
p, dPdv = bezcurveinterp(vcurve, v)
p, dPdu = bezcurveinterp(ucurve, u)
# take cross product of partials to find normal
n = cross(dPdu, dPdv)
n = n / length(n)
return p, n

# given a patch, perform uniform subdivision
subdividepatch(patch, step)
# compute how many subdivisions there
# are for this step size
numdiv = (1 + epsilon / step)
# for each parametric value of u
for (iu = 0 to numdiv)
    u = iu * step
    # for each parametric value of v
    for (iv = 0 to numdiv)
        v = iv * step
        # evaluate surface
        p, n = bezpatchinterp(patch, u, v)
        savesurfacepointandnormal(p, n)