```python
def Jvec(self, m, v, u=None):
    # Set current model; clear dependent property $\mathbf{A}(\mathbf{m})$
    self.curModel = m
    sigma = self.curModel.transform # $\sigma = \mathcal{M}(\mathbf{m})$
    if u is None:
        # Run forward simulation if $u$ not provided
        u = self.fields(self.curModel)
    else:
        shp = (self.mesh.nC, self.survey.nTx)
        u = u.reshape(shp, order='F')

    D = self.mesh.faceDiv
    G = self.mesh.cellGrad
    # Derivative of model transform, $\frac{\partial \sigma}{\partial \mathbf{m}}$
    dsigdm_x_v = self.curModel.transformDeriv * v

    # Take derivative of $C(m,u)$ w.r.t. $m$
    dCdm_x_v = np.empty_like(u)
    # loop over fields for each transmitter
    for i in range(self.survey.nTx):
        # Derivative of inner product, $\left(\mathbf{M}^f_{1/\sigma}\right)^{-1}$
        dAdsig         = D * self.dMdsig( G * u[:,i] )
        dCdm_x_v[:, i] = dAdsig *  dsigdm_x_v

    # Take derivative of $C(m,u)$ w.r.t. $u$
    dCdu = self.A
    # Solve for $\frac{\partial u}{\partial m}$
    dCdu_inv = self.Solver(dCdu, **self.solverOpts)
    P        = self.survey.getP(self.mesh)
    J_x_v    = - P * mkvc( dCdu_inv * dCdm_x_v )
    return J_x_v
```