ALGORITHM:

The SVD rank-1 modification algorithm is described by Matthew Brand
in the paper at, <http://www.stat.osu.edu/~dmsl/thinSVDtracking.pdf>.
The algorithm works as follows:

#. Extend V so that both its last column and row are the zero vector.
#. Compute a and b so that they perform the appropriate transformation.
#. If updating:
#.  a = c, b^T = [0,...,0,1]
#. Else if downdating:
#.  a = X[:,-1], b^T = [0,...,0,1]
#. Else if revising:
#.  a = X[:,-1] - c, b^T = [0,...,0,1]
#. Else: #recentering
#.  a = X * (I – (1/q) * (l * l^T)) # l = [1,...,1]
#. Compute m, p, Ra and P.
#. m = U^T * a
#. p = a - U * m
#. Ra = ||p||
#. P = Ra^(-1) * p
#. Compute n, q, Rb and Q, similarly to the previous step with substitution of b for a.
#. Compute K.
#. K = [S 0] * [m ] *[n ]^T
#.  [0 0]  [Ra]  [Rb]
#. Diagonalize K.

WARNING::

During testing, this code showed strong deviations, since it is an approximation, from
the true thin SVD of the matrix X. However, testing was conducted with a small matrix X,
and may be working perfectly fine.
"""

```
V = np.vstack([V, np.zeros(V.shape[1])])
if down or type(c) == type(np.array([])):
    b = np.zeros(V.shape[0])
    b[-1] = 1
    b = np.reshape(b, (b.shape[0], 1))
    if down:
        a = np.reshape(np.multiply(X[:,-1], -1), (-1, 1))
    elif add:
        a = np.reshape(c, (-1, 1))
    else:
        a = np.reshape(X[:,-1] - c, (-1, 1))
else:
    ones = np.zeros(V.shape[0])
    ones = np.add(b, 1)
    b = np.reshape(ones, (-1, 1))
    a = np.reshape(np.multiply((-1/X.shape[1]), np.dot(X, b)), (-1, 1))

m = np.reshape(np.dot(np.transpose(U), a), (-1, 1))
p = np.reshape(a - np.dot(U, m), (-1, 1))
Ra = np.linalg.norm(p)
```

```
P = np.reshape(np.multiply((1 / Ra), p), (-1, 1))
n = np.reshape(np.dot(np.transpose(V), b), (-1, 1))
q = b - np.dot(V, n)
Rb = np.linalg.norm(q)
Q = np.reshape(np.multiply((1 / Rb), q), (-1, 1))

k = S
K = np.zeros((k.shape[0] + 1, k.shape[0] + 1))
K[:-1,:-1] = k
stack = np.vstack(np.append(m, Ra))
t = np.reshape(np.append(n, Rb), (1, -1))
dot = np.dot(stack, t)
K = np.add(K, dot)

D, P = np.linalg.eig(K)

return (np.transpose(np.linalg.inv(P)), np.diag(D), P)
```