

```

# Implementation of a basic simplex method for finding solutions of
# linear programming problems.
#
# Author: Paul J. Atzberger
#
#
import numpy as np

def pivot(tab_data, extras):
    setB, setN, tabM, vec_c = tuple(map(tab_data.get,
                                        ['setB', 'setN', 'tabM', 'vec_c']));
    #setB, tabM = tuple(map(data.get, ['setB', 'tabM']));
    m = tabM.shape[0]-1; n = tabM.shape[1]-1;

    # identify the pivot: q index and p index
    sN = tabM[m, setN]; # get s_i for i not in B
    qq = np.argmin(sN, keepdims=True)[0]; # find most negative s_q < 0.
    q = setN[qq];
    y = tabM; # reference full table using y notation
    yq = y[0:m, q]; II, = np.nonzero(yq > 0);
    if len(II) == 0:
        tab_data['simplex_msg'] = 'LP is unbounded';
        return False;

    y0 = y[0:m, n];
    yi0_over_yiq_select = y0[II]/yq[II];
    pp = np.argmin(yi0_over_yiq_select, keepdims=True)[0]; # smallest yi0/yiq.
    p = II[pp]; # get index value

    setN[qq] = setB[p]; # put index p in set N
    setB[p] = q; # put index q in set B

    # -- update the tableau

    # we update the y_i, y0 representations using
    # in-place row operations
    ypq = y[p, q]; # pivot value
    y[p, :] = y[p, :]/ypq; # modify row p by 1/ypq
    for i in range(0, m):
        if (i != p): # i != p
            yiq = y[i, q]; yij = y[i, :]; # row i
            # compute: yij_new = (yij - (yiq/ypq)*ypj)
            ff = yiq; # factor is yiq since ypq already modified by 1/ypq
            y[i, :] = yij - ff*y[p, :]; # new row value

    # update the last row
    tabM[m, setB] = 0.0;
    tabM[m, setN] = vec_c[setN] - np.matmul(vec_c[setB], y[0:m, setN]);
    tabM[m, n] = -np.sum(vec_c[setB]*y[0:m, n]);

```

```

return True;

def check_sN_pos(tab_data,extras=None):
    setB,setN,tamM = tuple(map(tab_data.get,['setB','setN','tabM']));

    m = tamM.shape[0]-1; n = tamM.shape[1]-1;
    sN = tamM[m,setN];
    if np.all(sN >= 0):
        return True;
    else:
        return False;

def simplex_method(tab_data,extras=None):

    if extras is not None:
        max_steps, = tuple(map(extras.get,['max_steps']));
    else:
        max_steps = None;

    if max_steps is None:
        max_steps = int(1e2);

    for i in range(0,max_steps):
        if check_sN_pos(tab_data,extras):
            tab_data['simplex_msg'] = 'success';
            return True; # optimal solution
        else: # pivot
            flag_pivot = pivot(tab_data,extras);

            if flag_pivot == False:
                return False;

    tab_data['simplex_msg'] = 'max number of steps exceeded';
    return False;

def transform_to_canonical(tab_data,extras=None):
    setB,setN,tamM,vec_c = tuple(map(tab_data.get,
                                     ['setB','setN','tabM','vec_c']));

    m = tamM.shape[0]-1; n = tamM.shape[1]-1;

    B = tamM[0:m,setB]; N = tamM[0:m,setN];
    B_inv_N = np.linalg.solve(B,N);
    B_inv_b = np.linalg.solve(B,tamM[0:m,n]);

    tamM[0:m,setB] = np.eye(m);
    tamM[0:m,setN] = B_inv_N;
    tamM[0:m,n] = B_inv_b;

```

```

y = tabM;
tabM[m,setB] = 0.0;
tabM[m,setN] = vec_c[setN] - np.matmul(vec_c[setB],y[0:m,setN]);
tabM[m,n] = -np.sum(vec_c[setB]*y[0:m,n]);

def print_tableau(tab_data,label=""):
    setB,setN,tabM,vec_c = tuple(map(tab_data.get,
                                     ['setB','setN','tabM','vec_c']));
    print(label);
    print("setB = " + str(setB));
    print("setN = " + str(setN));
    print("tabM = \n" + str(tabM));
    print("vec_c = \n" + str(vec_c));

# -----
print("="*80);
print("Simplex Method");
print("");
print("Author: Paul J. Atzberger");
print("-"*80);

# Example data (solution done by hand in lecture)
tabM = np.array([[2,1,1,0,3],[1,4,0,1,4],[-7,-6,0,0,0]],dtype=np.float64);
setB = np.array([2,3],dtype=int); setN = np.array([0,1],dtype=int); # note index base is zero
vec_c = np.array([-7,-6,0,0],dtype=np.float64);
tab_data = {'setB':setB,'setN':setN,'tabM':tabM,'vec_c':vec_c};

# show tableau
print_tableau(tab_data,"tableau (initial):"); print("");

# transform the tableau into canonical form
# (if already canonical form, returns the same tableau)
transform_to_canonical(tab_data);

# show tableau
print_tableau(tab_data,"tableau (transformed to canonical)"); print("");

# perform the simplex method
extras = {'max_steps':int(1e2)};

flag_simplex = simplex_method(tab_data,extras);

setB,setN,tabM = tuple(map(tab_data.get,['setB','setN','tabM']));
m = tabM.shape[0]-1; n = tabM.shape[1]-1;

# show the final results (even if failed)
print_tableau(tab_data,"tableau (final):"); print("");

```

```

# show the final solution or message
if flag_simplex:
    print("."*80);
    print("found optimal solution:");
    cT_x = -tabM[m,n]; # objective function value
    print("c*x = " + str(cT_x));
    x_star = np.zeros(n);
    x_star[setB] = tabM[0:m,n]; # x_star[setN] = 0.0;
    print("x_star = \n" + str(x_star));
else:
    print("."*80);
    print("simplex method failed.");
    print("reason: " + tab_data['simplex_msg']);

print("-"*80);
print("="*80);
#-----

```