.ipynb

March 6, 2021

1 Simplex Example

1.0.1 EE 578B - Winter 2021

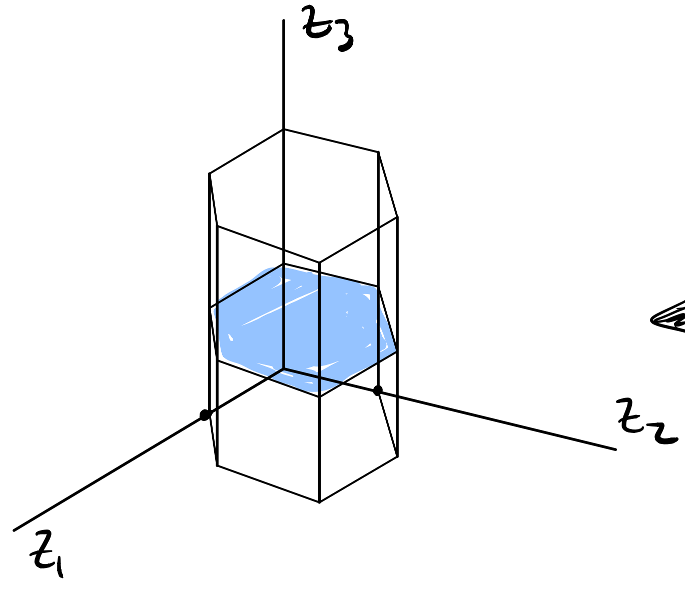
$$r = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 1 \\ 1 \end{bmatrix}$$

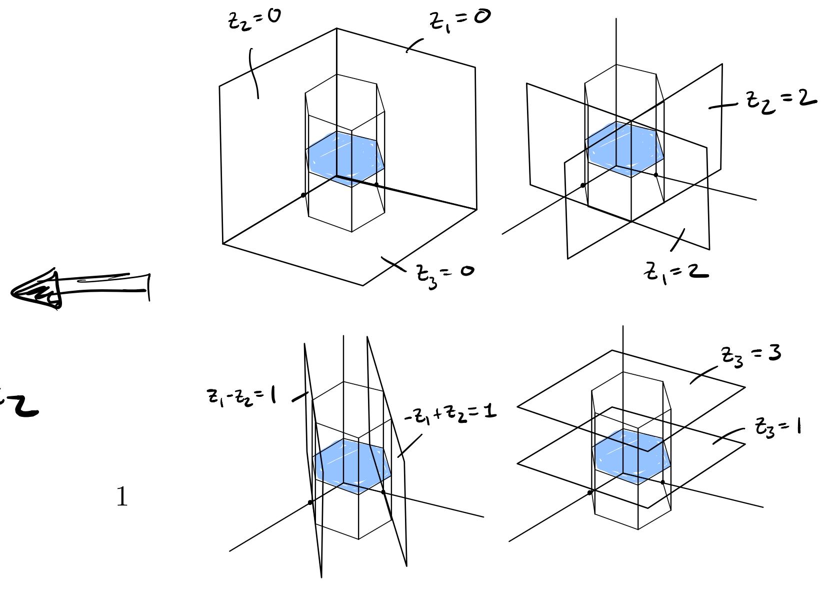
1.1 Note: in lecture there were some sign mistakes in the last 2 rows of A which is why things didn't work.

1.2 Initial Tableau

$$T = \begin{bmatrix} 1 & -r^T & 0 \\ 0 & A & b \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 3 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Constraints





2 NOTE: rows will be ZERO INDEXED in notes...

3 ALSO: All cells must be run in order since you're editting T...

3.1 Note: General way to find an initial feasible solution

3.1.1 In the general, case where an initial feasible solution is not obvious, we temporarily add an extra set of slack variables to form the tableau...

$$T_{\text{init}} = \begin{bmatrix} 1 & \mathbf{1}^T & \mathbf{0}^T & 0 \\ \mathbf{0} & I & A & b \end{bmatrix}$$

3.1.2 and take the initial slack variable columns as the solution. The objective function in this tableau represents trying to minimize the sum of the slack variables. We then perform row operations till the added columns are no longer basis columns.

[121]:
```
import numpy as np
r = np.array([1.,1.,1.,0.,0.,0.,0.,0.]);
b = np.array([[1.,2.,2.,3.,1.,1.]]).T;
A = np.array([[0.,0.,1.,0.,0.,0.,0.,0. ],
              [1.,0.,0.,1.,0.,0.,0.,0. ],
              [0.,1.,0.,0.,1.,0.,0.,0. ],
              [0.,0.,1.,0.,0.,1.,0.,0. ],
              [-1.,1.,0.,0.,0.,0.,1.,0.],
              [ 1.,-1.,0.,0.,0.,0.,0.,1.]]);

T = np.block([[1.,-r,0],
              [np.zeros([6,1]),A,b]]);
col_labels = '       z1  z2  z3  s1  s2  s3  s4  s5  b'
print(col_labels)
print(T)
```

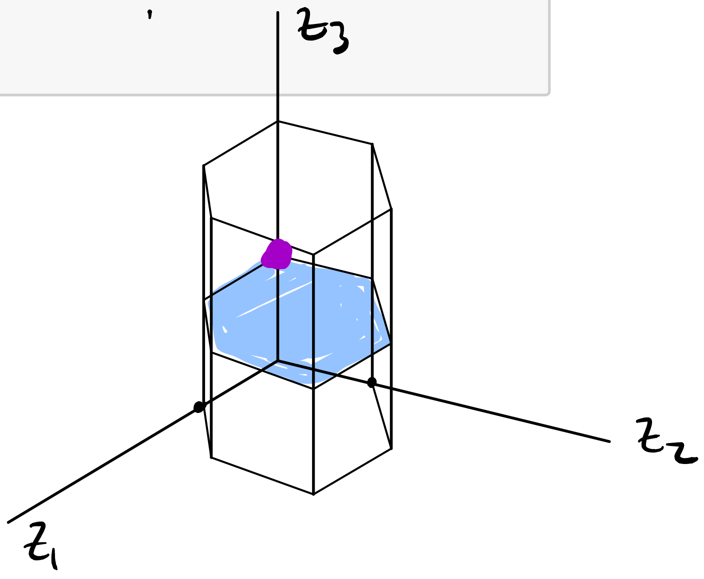
```
       z1  z2  z3  s1  s2  s3  s4  s5  b
[[ 1. -1. -1. -1. -0. -0. -0. -0. -0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  1.  0.  0.  0.  0.  2.]
 [ 0.  0.  1.  0.  0.  1.  0.  0.  0.  2.]
 [ 0.  0.  0.  1.  0.  0.  1.  0.  0.  3.]
 [ 0. -1.  1.  0.  0.  0.  0.  1.  0.  1.]
 [ 0.  1. -1.  0.  0.  0.  0.  0.  1.  1.]]
```

3.1.3 Finding an initial identity - feasible solution

... row 4 = row4 - row 1 and cashing out... ... row 0 = row0 + row1

[122]:
```python
T[4] = T[4] - T[1]
T[0] = T[0] + T[1]
print(col_labels)
print(T)
print('Current reward: ',T[0,-1])
print('Basis columns: z3 s1  s2  s3  s4  s5')
```

```
       z1  z2  z3  s1  s2  s3  s4  s5  b
[[ 1. -1. -1.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  1.  0.  0.  0.  0.  2.]
 [ 0.  0.  1.  0.  0.  1.  0.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  2.]
 [ 0. -1.  1.  0.  0.  0.  0.  1.  0.  1.]
 [ 0.  1. -1.  0.  0.  0.  0.  0.  1.  1.]]
Current reward:  1.0
Basis columns: z3 s1  s2  s3  s4  s5
```

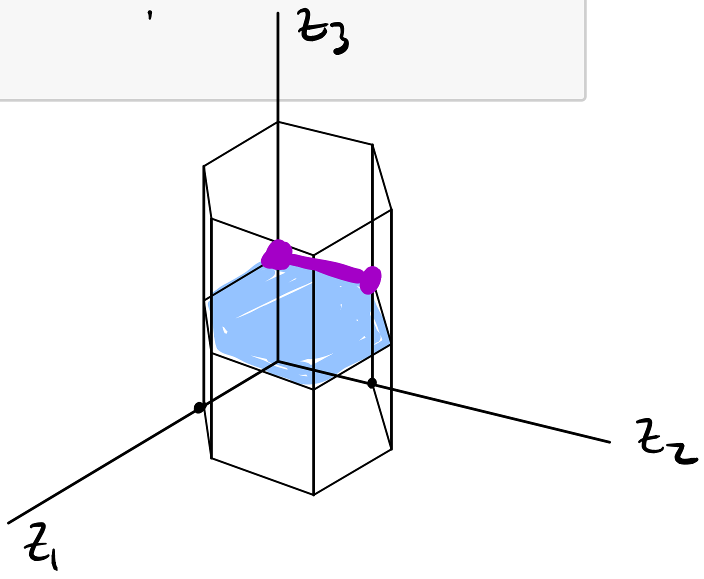


4 Add column z2...

Column z1 or z2 will improve solution since objective row values is negative... Row 3 or Row 5 are initially options to swap in (because values in z2 column are positive) Considering ratio with values in b column gives that Row 5 is the only option Row reduce z2 column to identity ...row 3 = row 3 - row 5 ...row 6 = row 6 + row 5 cash out... ...row 0 = row0 + row 5

[123]:
```python
T[3] = T[3] - T[5]
T[6] = T[6] + T[5]
T[0] = T[0] + T[5]
print(col_labels)
print(T)
print('Current reward: ',T[0,-1])
print('Basis columns: z2 z3 s1 s2 s3 s5')
```

```
       z1  z2  z3  s1  s2  s3  s4  s5  b
[[ 1. -2.  0.  0.  0.  0.  0.  1.  0.  2.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  1.]
 [ 0.  1.  0.  0.  1.  0.  0.  0.  0.  2.]
 [ 0.  1.  0.  0.  0.  1.  0. -1.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  2.]
 [ 0. -1.  1.  0.  0.  0.  0.  1.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  1.  2.]]
Current reward:  2.0
Basis columns: z2 z3 s1 s2 s3 s5
```

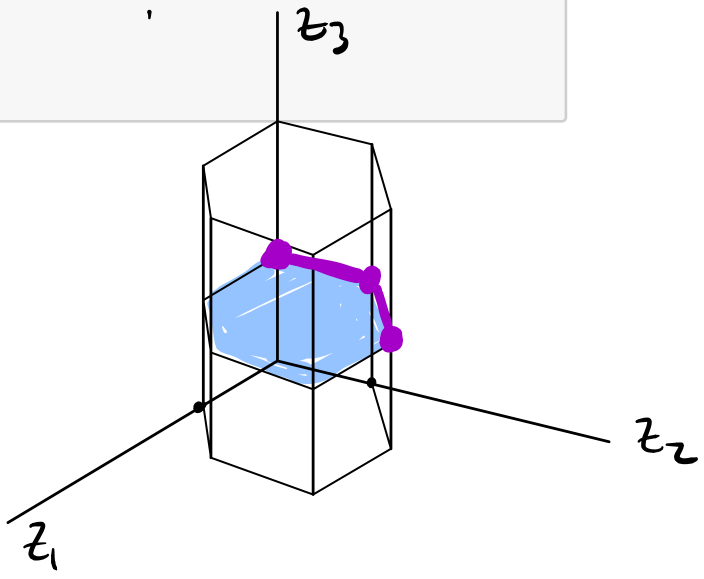


5 Add column z1...

Only option add Column z1... Row 2 or Row 3 are initially options to swap in Ratio with b column values gives only row3 is possible... Row reduce z1 column to identity ...row 2 = row 2 - row 3 ...row 5 = row 5 + row 3 cash out... ...row 0 = row0 + row 6

[124]:
```
T[2] = T[2] - T[3]
T[5] = T[5] + T[3]
T[0] = T[0] + 2*T[3]
print(col_labels)
print(T)
print('Current reward: ',T[0,-1])
print('Basis columns: z1 z2 z3 s1 s3 s5')
```

```
       z1  z2  z3  s1  s2  s3  s4  s5  b
[[ 1.  0.  0.  0.  0.  2.  0. -1.  0.  4.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  1. -1.  0.  1.  0.  1.]
 [ 0.  1.  0.  0.  0.  1.  0. -1.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  2.]
 [ 0.  0.  1.  0.  0.  1.  0.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  1.  2.]]
Current reward:  4.0
Basis columns: z1 z2 z3 s1 s3 s5
```

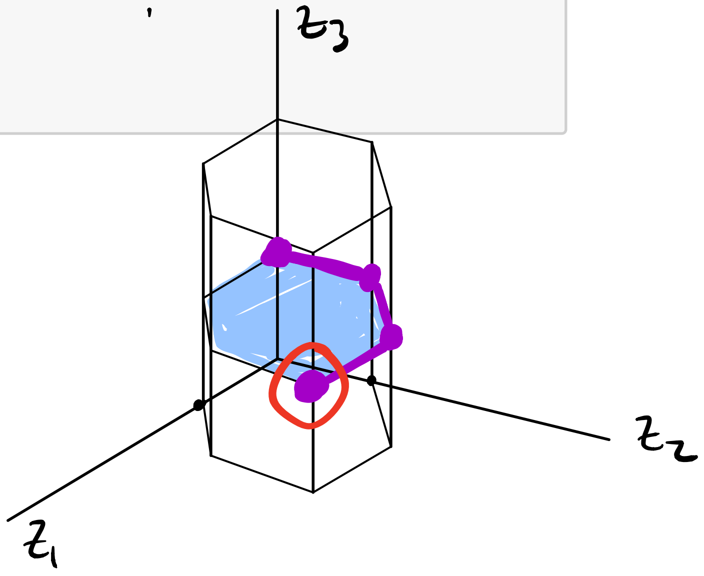


6 Add column s4...

Only option add Column s4... Row 2 or Row 6 are initially options to swap in Ratio with b column values gives only row 2 is possible... Row reduce s4 column to identity ...row 3 = row 3 + row 2 ...row 6 = row 6 - row 2 cash out... ...row 0 = row0 + row 2

[125]:
```
T[3] = T[3] + T[2]
T[6] = T[6] - T[2]
T[0] = T[0] + T[2]
print(col_labels)
print(T)
print('Current reward: ',T[0,-1])
print('Basis columns: z1 z2 z3 s3  s4  s5')
```

```
       z1  z2  z3  s1  s2  s3  s4  s5  b
[[ 1.  0.  0.  0.  1.  1.  0.  0.  0.  5.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  1. -1.  0.  1.  0.  1.]
 [ 0.  1.  0.  0.  1.  0.  0.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  2.]
 [ 0.  0.  1.  0.  0.  1.  0.  0.  0.  2.]
 [ 0.  0.  0.  0. -1.  1.  0.  0.  1.  1.]]
```



```
Current reward:  5.0
Basis columns: z1 z2 z3 s3  s4  s5
```

6.1 DONE...

```
[ ]:
```