ORF524 - Final Exam

Bachir EL KHADIR

January 19, 2016

Problem 1

1. True
2. False
3. False
4. True
5. True
6. True

Problem 2

1. **True**.

   Let's write the optimization problem in the following form:

   $$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & Ax = b,\ x \geq 0 \end{aligned}$$

   We can assume, without loss of generality, that the problem is non-degenerate by removing some rows from $A = (A_B\ A_N)$.

   If a vertex $x = (x_B\ x_N) = (x_B\ 0)$ has an objective value no larger than the objective value of all its neighbours, its reduced vector cost $\bar{c} = c_N - c_B^T A_B^{-1} A_N$ is non negative.

   Proof. Let $j \in N$. Let d be a direction such that: $Ad = 0$ and $\forall i \in N\ d_i = 1_{i=j}$, eg: $d = (-A_B^{-1}A_j; 0, .., \underbrace{1}_{j}, .., 0)$. Then moving along the direction $\theta d, \theta \in \mathbb{R}^+$ we will eventually set one coordinate in B to 0 since the feasible set is bounded, therefore reaching a neighbouring vertex. Let's call θ^* the maximal move in that direction(> 0 by non-degeneracy). The change to the objective value is then $\theta^* \bar{c}_j$, which should be non-negative, and therefore $\bar{c}_j \geq 0$,and $c_N \geq c_B^T A_B^{-1} A_N$ □

   Now let $y = (y_B;\ y_N) = (A_B^{-1}(b - A_N^{-1} y_N);\ y_N)$ be a feasible point, then

   $$\begin{aligned} c^T(y - x) &= c_B^T(y_B - x_B) + c_N y_N \\ &\geq c_B^T(A_B^{-1}(b - A_N^{-1} y_N) - A_B^{-1} b) + c_B^T A_B^{-1} A_N y_N && \text{(since } y_N \geq 0 \text{ and } \bar{c} \geq 0) \\ &\geq 0 \end{aligned}$$

   Therefore x is optimal.

2. **True**.

   Let $\lambda \in (0,1)$, $c_1, c_2, c, b_1, b_2, b \in \mathbb{R}^n$ so that $c = \lambda c_1 + (1-\lambda)c_2, b = \lambda b_1 + (1-\lambda)b_2$

   - Convexity in c:

     Lemma 0.1. *For two sets A, B, if $a, b \in A, B$, then $\lambda a \le \lambda \sup A$, $a + b \le \sup A + \sup B$, so that $\sup(A+B) \le \sup A + \sup B$ and $\sup \lambda A \le \lambda \sup A$.*

$$\begin{aligned} V(b, \lambda c_1 + (1-\lambda)c_2) &= \max_{Ax\le b, x\ge 0} \lambda c_1^T x + (1-\lambda)c_2^T x \\ &\le \max_{Ax\le b, x\ge 0} \lambda c_1^T x + \max_{Ax\le b, x\ge 0} (1-\lambda)c_2^T x \\ &\le \lambda \max_{Ax\le b, x\ge 0} c_1^T x + (1-\lambda) \max_{Ax\le b, x\ge 0} c_2^T x \\ &= \lambda V(b, c_1) + (1-\lambda)V(b, c_2) \end{aligned}$$

   - Concavity in b:

     Let x_i be a feasible solution to $\max_{Ax_i \le b_i, x_i \ge 0} c^T x_i$ for $i = 1, 2$. Then $x = \lambda x_1 + (1-\lambda)x_2$ is a feasible solution to $\max_{Ax\le b, x\ge 0} c^T x$, and we have that $\lambda c^T x_1 + (1-\lambda)c^T x_2 = c^T x$ Which means that $\max_{Ax_i\le b_i, x_i \ge 0, i=1,2} \lambda c^T x_1 + (1-\lambda)c^T x_2 \le \max_{Ax\le b, x\ge 0} c^T x$ Since the max on the left invloves two independent variables, it can be distributed so that we have:

$$\lambda V(b_1, c) + (1-\lambda)V(b_2, c) \le V(b, c)$$

3. **False**.

   Take $f(x) = 1_{x>0}$, the $\operatorname{epi} f = (-\infty, 0] \times [0, \infty) \cup [0, \infty) \times [1, \infty)$, which is close as the union of two closed sets, but f is not continuous.

4. **False**. Take $f(x) = 1_{x>0}\frac{1}{x} + 1_{x\le 0}\infty$

   $\operatorname{epi} f$ is closed but $\operatorname{dom}(f) = (0, \infty)$ is not closed.

5. **False**. $f(n, -n) \to_{n\infty} 0$, but $||(n, -n)|| \to_n \infty$, so f is not coercive.

6. **True**.

   Without loss of generality, by expressing the problem in an appropriate basis $(e_1, ..., e_n)$, we can assume that Q is diagonal $\operatorname{diag}(\lambda_1, ..., \lambda_n)$ with $\lambda_1 \ge ... \ge \lambda_n \ge 0$

   - If there exist i such that $\lambda_i = 0$ and $b_i \ne 0$, then $Qe_n = 0$ and $f(\alpha e_n) = \alpha b_n \to_{\alpha\to\pm\infty} -\infty = f^*$, and the inequality is trivially verified. Otherwise, we can just dismiss the coordinates for which $\lambda_i = b_i = 0$ because they don't affect the objective function nor the gradient, so that we can assume Q is invertible.
   - Else, Q is invertible

$$f(x) = \frac{1}{2}(x + Q^{-1}b)'Q(x + Q^{-1}b) - \frac{1}{2}b'Q^{-1}b \ge -\frac{1}{2}b'Q^{-1}b = f(-Q^{-1}b) = f^*$$

     We have that:

$$x_{k+1} = x_k - \alpha\nabla f(x_k) = x_k - \alpha Q(x_k + bQ^{-1})$$

     Adding bQ^{-1} to both sides:

$$x_{k+1} + Q^{-1}b = (I - \alpha Q)(x_k + Q^{-1}b)$$

So:

$$||x_{k+1} + Q^{-1}b||^2 \leq ||I - \alpha Q||^2 ||x_k + Q^{-1}b||^2$$

Therefore

$$f(x_{k+1}) - f^* \leq \rho(f(x_k) - f^*)$$

By immediate induction, $f(x_k) - f^* \leq \rho^k(f(x_0) - f^*)$ with $\rho := ||I - \alpha Q||^2$.
For α small enough, the eigen values of $(I - \alpha Q)$ are all smaller than 1, and $\rho < 1$

Problem 3

1. The dual problem:

$$\begin{aligned} \underset{y}{\text{maximize}} \quad & (1+\lambda)y_1 + (-2+\lambda)y_2 + (2+\lambda)y_3 + (5+\lambda)y_4 \\ \text{subject to} \quad & y_1 + y_2 \leq -4 - \lambda \\ & -y_1 + 2y_2 - y_3 \leq 3 - 2\lambda \\ & -y_1 + y_2 - y_4 \leq 1 - \lambda \end{aligned}$$

Complementary conditions:

$$\begin{aligned} (1+\lambda+x_1-x_2-x_3)y_1 &= 0 \\ (-2+\lambda+x_1+2x_2+x_3)y_2 &= 0 \\ (2+\lambda-x_2)y_3 &= 0 \\ (5+\lambda-x_3)y_4 &= 0 \end{aligned}$$

2.

$$\lambda \geq 2$$

$V(\lambda) = 0$	$(-4-\lambda)x_1$	$+(3-2\lambda)x_2$	$+(1-\lambda)x_3$
$w_1 = 1+\lambda$	1	-1	-1
$w_2 = -2+\lambda$	1	2	1
$w_3 = 2+\lambda$	0	-1	0
$w_4 = 5+\lambda$	0	0	-1

$$\frac{3}{2} \leq \lambda \leq 2$$

$V(\lambda) = -\frac{1}{2}(-2+\lambda)(3-2\lambda)$	x_1	$+w_2$	$+x_3$

$$\frac{3}{2} \geq \lambda \geq 0$$

$V(\lambda) = (3-2\lambda)(1+\lambda)$	$(-1-3\lambda)x_1$	$-(3-2\lambda)z_1$	$+(3\lambda-2)x_3$
$x_2 = 1+\lambda$	1	-1	-1
$w_2 = 3\lambda$	3	2	-1
$w_3 = 1$	-1	1	1
$w_4 = 5+\lambda$	0	0	-1

3.

$$V(\lambda) = \begin{cases} 0 & \text{if } 2 \leq \lambda \\ -\frac{1}{2}(-2+\lambda)(3-2\lambda) & \text{if } \frac{3}{2} \leq \lambda \leq 2 \\ (3-2\lambda)(1+\lambda) & \text{if } 0 \leq \lambda \leq \frac{3}{2} \end{cases}$$

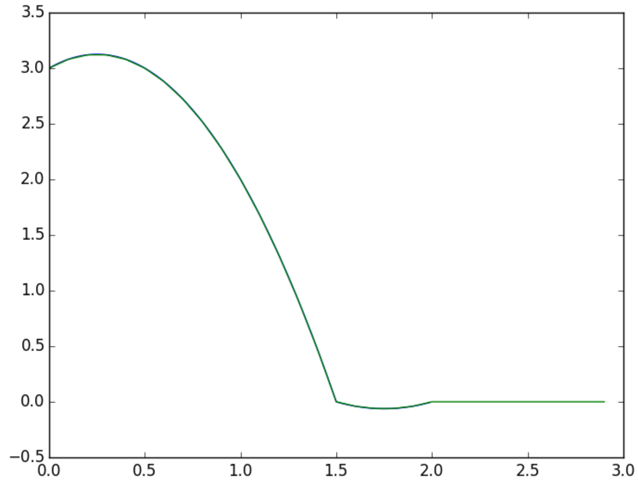


Figure 1: $V(\lambda)$

4.

$$V'(\lambda) = \begin{cases} 0 & \text{if } \frac{2}{\leq}\lambda \\ 2\lambda - \frac{7}{2} & \text{if } \frac{3}{2} < \lambda < 2 \\ -2\lambda + 1 & \text{if } 0 \leq \lambda < \frac{3}{2} \end{cases}$$

In 2 and $\frac{3}{2}$, V has only right and left derivative. $V'(2+) = 0, V'(2-) = \frac{1}{2}, V'(\frac{3}{2}+) = -\frac{1}{2}, V'(\frac{3}{2}-) = -2$

Problem 4

1. If $x, y \in \{0, 1\}$, then $x^2 + y^2 = x + y \in \{0, 1, 2\}$. Let $v := x + y$, then :

$$p(u) = \min_{a-u \leq v,\ v \in \{0,1,2\}} v = \begin{cases} 0 & \text{if } a \leq u \\ 1 & \text{if } a-1 \leq u < a \\ 2 & \text{if } a-2 \leq u < a-1 \\ \infty & \text{if } u < a-2 \end{cases}$$

The problem is not convexe because the feasible set is discrete and not reduced to a singleton. p is non-increasing, so it has a right limit everywhere, furthermore it is right-continuous, so: $\liminf_{x \to x_0} f(x) = \lim_{x \geq x_0} f(x) = f(x_0)$, so it is lower semi-continuous.

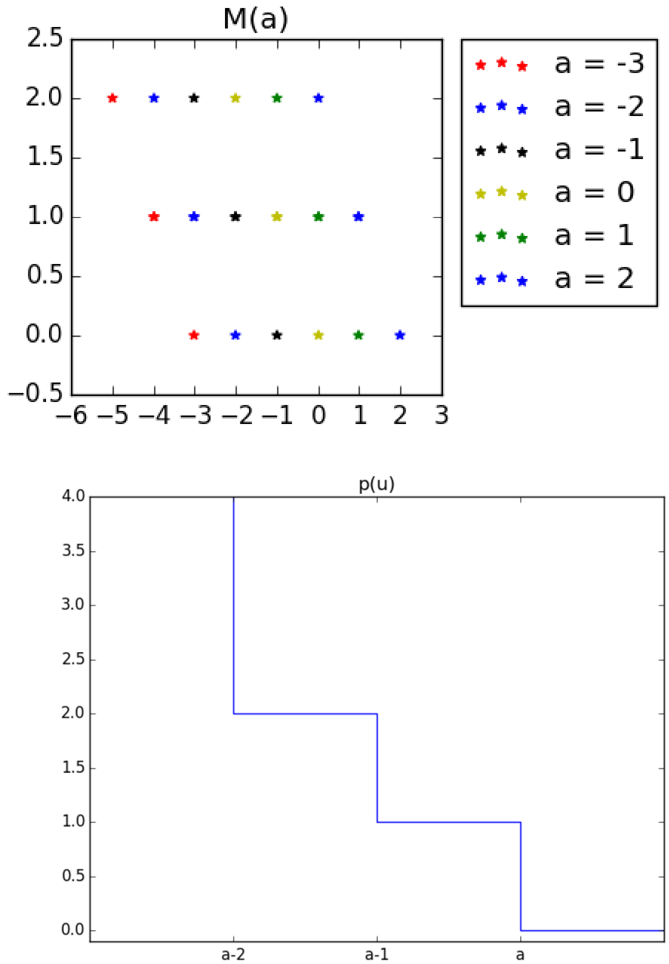


Figure 2: Sketch of $\bar{M}(a)$ and $p(u)$

2. The problem is feasible iff $a \leq 2$

   In the following graph I have drawn the supporting (and non parallel to the y axis) hyperplanes of epi p.

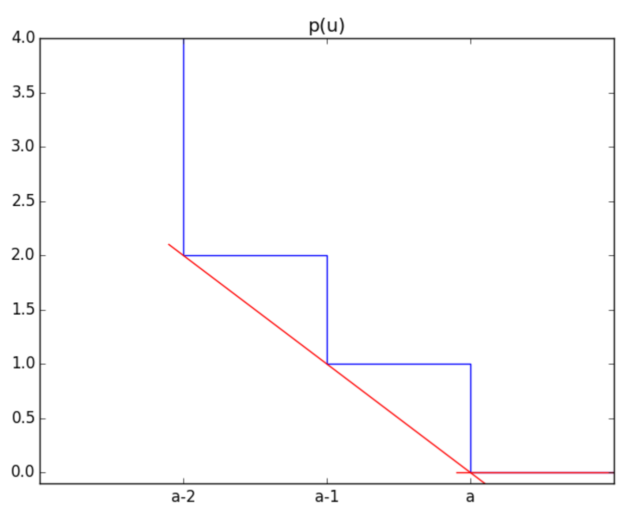


Figure 3: Supports of epi(p)

- The primal equals to the dual when the min common point of the $y-axis$ equals the max crossing support, ie when $a \leq 0$ or $a \in \{1, 2\}$.

- There is a duality gap otherwise, ie when $a \in (0, 2) \setminus \{1\}$
- There is uniqueness of the dual solution only if there is a unique supporting plane crossing the $y-$axis at a maximal point. It is the case whenever $a \neq 0, a \leq 2$.

3. Take $a = -1$. Let $X = \{0, 1\}$ The primal:

$$\max_{x+y\geq -1, x,y\in X} x^2 + y^2$$

, the optimal value is 0, and the optimal solution is $(0, 0)$

The dual:

$$\begin{aligned}
\min_{\lambda\leq 0} \max_{x,y\in X} -\lambda(1 + x + y) + x^2 + y^2 &= \min_{\lambda\leq 0} 2\left(\max_x x^2 - \lambda x\right) - \lambda && \text{(By symmetry)} \\
&= \min_{\lambda\leq 0} -\lambda + 2 \max_{x\in\{0,1\}} x(1 - \lambda) && (x^2 = x) \\
&= \min_{\lambda\leq 0} -\lambda + 0 && (1 - \lambda > 0) \\
&= 0 && \text{(When } \lambda = 0\text{)}
\end{aligned}$$

Problem 5

-
  - States: (i, j) meaning we have to do the multiplication of the matrices $M_i...M_j$, and $C(i, j)$ is the optimal cost to doing this multiplication
  - Action: Put the parentheses at position $k \in \{i, ..., j - 1\}$: $M_i...M_j = (M_i...M_k)(M_{k+1}...M_j)$
  - Cost: The cost to doing the multiplication $(M_i...M_k)(M_{k+1}...M_j)$, is $r_i c_k c_j + C(i, k) + C(k + 1, j)$

$$C(i, i) = 0$$
$$C(i, j) = \min_{k=i,..,j-1} r_i c_k c_j + C(i, k) + C(k + 1, j)$$

- Shortest path formulation:
  - Node space: $\{(k_{i_1}, ..., k_{i_r})|k_{i_j} \text{ all different } , r \in \{1, ..., n - 1\}\}$, $(k_{i_1}, ..., k_{i_r})$ denotes the order in which we put the parentheses:
    1. $(M_1...M_{k_{i_1}})(M_{k_{i_1}+1}...M_n)$
    2. $((M_1..M_{k_{i_2}})(M_{k_{i_2}+1}..M_{k_{i_1}}))(M_{k_{i_1}+1}...M_n)$ if $k_{i_2} < k_{i_1}$, $(M_1....M_{k_{i_1}})((M_{k_{i_1}+1}...M_{k_{i_2}})(M_{k_{i_2}+1}...M_n))$ otherwise.
    3. etc...
  - Starting node: () the empty tuple.
  - Final nodes: $(k_{i_1}, ..., k_{i_{n-1}})$ tuples with $n - 1$ elements.
  - Transitions: $(k_{i_1}, ..., k_{i_r}) \rightarrow (k_{i_1}, ..., k_{i_r}, k_{i_{r+1}})$ where $k_{i_{r+1}} \notin \{k_{i_1}, ..., k_{i_r}\}$
  - Cost: Let $a = \max\{k_{i_j}|k_{i_j} < k_{i_{r+1}}\}$, $b = \min\{k_i|k_{i_j} > k_{i_{r+1}}\}$. The cost is then $r_a c_{k_{i_{r+1}}} r_b$

  The problem with this formulation is that it takes an exponential number of nodes ($O(n^n)$)

- Linear programming formulation:

$$\underset{D}{\text{maximize}} \quad \sum_{i<j} D(i, j) \quad \text{subject to} \quad D(i, j) \leq r_i c_k c_j + D(i, k) + D(k + 1, j) \forall k \in \{i, ..., j - 1\}$$

  It is clear that C is a solution to this linear problem.(see lecture 22)

Problem 6

1. Look at the code.

2. Let $\mathcal{C}$ be the set of the sort trajectories, and $p = \frac{1}{|\mathcal{C}|}$

   - States: (t, S)
   - Randomness: $(t, S) \to (t+1, S+s)$, $s \sim \mathcal{U}(\mathcal{C})$.
   - Actions: Hold / Exec
   - Transitional cost: 0 if we hold, $S - K$ if we exercise.

3. Bellman equation:
$$V_k(S) = \max\{S - K, p \sum_{s \in C} V_{k+1}(S+s)\}$$
$$V_T(S) = (S-K)^+$$

4. **Value iteration:** Look at the code.

   LP formualation: Let $J(t, S)$ be the price of the option at time t is $S_t = S$, and we decide to adopt the strategy

   J verifies: $J(t, S) = \max\{E[J(t+1, S_{t+1})|S_t = S], S - K\} = [\max_\mu(P_\mu J + g_\mu)](t, S)$ where:
$$\mu(t, S) \in \{\text{HOLD}, \text{EXEC}\}$$
$$(P_\mu J)(t, S) = \begin{cases} p \sum_{s \in \mathcal{C}} J(t+1, S+s) & \text{if } \mu(t,S) = \text{HOLD} \\ 0 & \text{otherwise} \end{cases}$$
$$g_\mu(t, S) = \begin{cases} 0 & \text{if } \mu(t,S) = \text{HOLD} \\ S - K & \text{otherwise} \end{cases}$$

   The LP problem is:
$$\min e^T J \text{ s.t } \forall \mu \; J \geq P_\mu J + g_\mu$$

   At time t, S can take the following values $\{S_{t-1} + s,\ s \in \mathcal{C}\} = \{S_t^k, k \leq N_t\}$ where (S_t^k) is an increasing sequence. Let's denote by $\tilde{J}(t, k) := J(t, S_t^k)$ when $k \leq N_t$ and L otherewise where $L >> S_0$ is a very big constant. The problem can be written as:
$$\begin{aligned} \min \sum_{t,k} & \tilde{J}(t,k) \\ & \text{s.t } \forall t, k \in \{1...T-1\} \\ & \tilde{J}(t,k) \geq p \sum_{j \leq |\mathcal{C}|} \tilde{J}(t+1, k+j) \\ & \tilde{J}(t,k) \geq S_t^k - K \\ & \tilde{J}(T,k) = S_T^k - K \\ & \tilde{J}(t,k) = L \text{ when } k > N_t \end{aligned}$$

   Let $x_{t*T+k} = \tilde{J}(t,k)$, and define $A \in \mathcal{M}_{T^2,T^2}$, $B \in \mathcal{M}_{\frac{T(T-1)}{2},T^2}$ $U \in \mathbb{R}^{T^2}$ such that:

$$A := \begin{array}{r} \text{T(T-1)} \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right. \\ \text{T} \left\{ \begin{array}{c} \\ \\ \end{array} \right. \end{array} \begin{bmatrix} 0 & p & ... & p & \cdots \\ \vdots & & \ddots & \ddots & \cdots \\ 0 & & \cdots & p & p \\ 0 & 0 & 0 & 0 & 0 \\ & \cdots & \cdots & & \end{bmatrix}$$

$$B := \begin{array}{r} \text{T-1} \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right. \\ T-N_1 \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right. \\ \vdots \left\{ \right. \\ T-N_{T-1} \left\{ \right. \end{array} \left[\begin{array}{ccc|cccc|c|ccc} 0 & 1 & \cdots & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ \hline & & & 0 & 0 & 1 & \cdots & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ \hline & & & & & & & \ddots & & & \\ \hline & & & & & & & & 0 & \cdots & 1 \end{array} \right]$$

$U \in R^{T^2}$ such that : $U_{tT+k} := S_t^k - K$

The LP problem is equivalent to:

$$\begin{aligned} &\min e^T x \\ &\quad \text{s.t} \\ &x \geq Ax \\ &x \geq U \\ &Bx = L1_{\frac{T(T-1)}{2}} \end{aligned}$$

question6

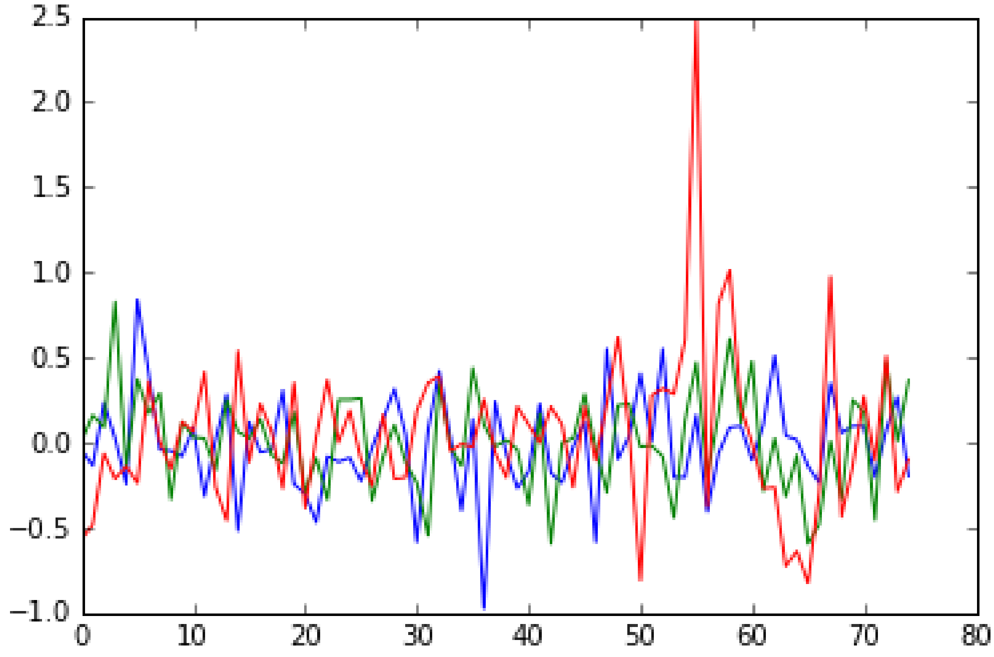
January 15, 2016

In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
# Load microsoft data
msft = pd.read_csv('microsoft.csv')
trajectories = np.diff(msft['MSFT.Adjusted'])
length_short_path = 30 # days
length = len(trajectories)
num_paths = int(length / length_short_path)
trajectories = trajectories[:length_short_path * num_paths]
trajectories.shape = (length_short_path, num_paths)
for i in range(3):
    plt.plot(trajectories[i, :])
plt.show()

K = 0.
T = 5
```



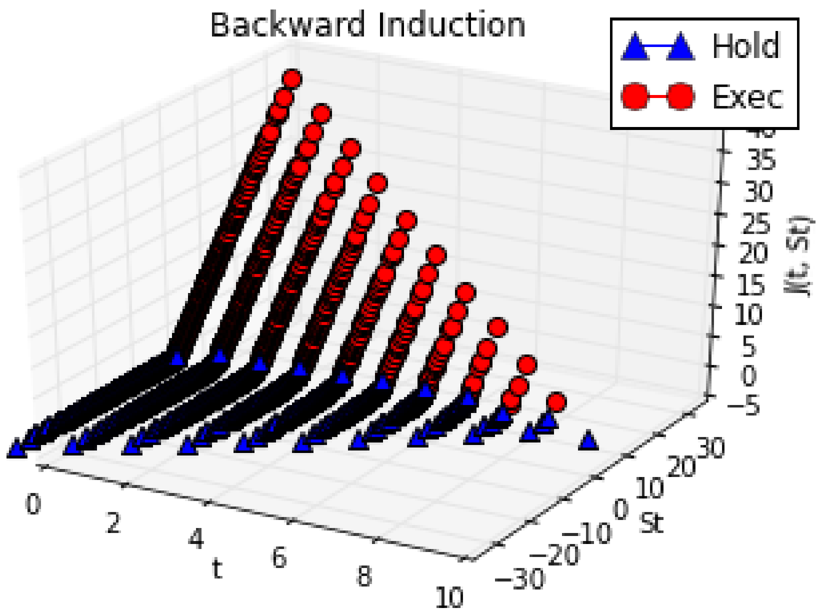
```
In [4]: outcomes = np.sort(np.sum(trajectories[:5,:], axis=1))
        print(outcomes)
        num_outcomes = len(outcomes)
```

```
[-4.404346 -0.829189 -0.79187   1.067787  3.531883]
```

```
In [5]: def generate_slates(n):
            St = np.array([0])
            S = [St]
            for i in range(n):
                Stnext = np.array([])
                for ds in outcomes:
                    Stnext = np.concatenate([Stnext, (St + ds)])
                St = unique(Stnext)
                St = np.sort(St)
                S.append(St)
            return S[::-1]

        def backward_induction(St, Jnext):
            expected_J = np.zeros_like(St)
            offset = len(Jnext) - len(St)
            p = 1. / offset
            for i in range(offset):
                expected_J += p* (Jnext[i:-(offset - i)] if offset !=  i else Jnext[i:])
            buffer = np.array([expected_J, St-K])
            control = np.argmax(buffer, axis=0)
            J = np.maximum(St-K, expected_J)
            return (J, control)

        def price_tree(n):
            slates = generate_slates(n)
            J = np.zeros(len(slates[0])+num_outcomes-1)
            for t, St in enumerate(slates):
                J, control = backward_induction(St, J)
                yield np.array([t*np.ones_like(St), St, J, control])
```

```
In [7]: from mpl_toolkits.mplot3d import Axes3D
        import matplotlib.pyplot as plt
        import matplotlib.lines as mlines

        T = 10
        def plot(pricing_method, name, img):
            fig = plt.figure()
            ax = fig.add_subplot(111, projection='3d')
            c = np.array(('b', 'r'))
            m = np.array(('^', 'o'))
            for points in pricing_method(T):
                for u in (0, 1):
                    xs, ys, z_tree, _ = points[:,abs(points[3] - u) <= 0.01]
                    ax.scatter(xs,
                               ys, z_tree, s=50,
                               c=c[u], marker=m[u],
```

```
                    depthshade=False)

    ax.set_xlabel('t'); ax.set_xlim((0, T))
    ax.set_ylabel('St'); ax.set_ylim((-T * outcomes[-1], T * outcomes[-1]))
    ax.set_zlabel('J(t, St)')
    labels_str = ('Hold', 'Exec')
    labels = [
        mlines.Line2D([], [], color=c[u], marker=m[u],
                      markersize=10, label=labels_str[u])
        for u in (0, 1)]
    plt.title(name)
    plt.legend(handles=labels)
    plt.savefig('q%s.png' % img)
    plt.show()
plot(price_tree, 'Backward Induction', 'tree')
```



In [0]:
```
#LP

# Get the relevent matrices
def get_A():
    row = np.zeros(T*T)
    for i in range(num_outcomes):
        row[T+i] = 1./num_oucomes
    A = [row]
    for _ in range(T*(T-1)-1):
        row = np.roll(row, 1)
        A.append(row)
    row = 0 * row
    for _ in range(T): A.append(row)
    return np.array(A)
```

```
def get_B():
    B = np.zeros( shape=(T*(T-1)/2, T*T) )
    i, j = 0, 0
    for St in generate_slates(T)[::-1]:
        Nt = len(St)
        j += Nt + 1
        for b in range(T-(Nt+1)):
            B[i, j] = 1
            i, j = i+1, j+1
    return B
def get_U():
    return np.array(itertools.chain(*generate_slate(T))) - K
```