Department of Mathematics and Computer Science
University of Southern Denmark, Odense

April 17, 2018
Marco Chiarandini

DM559 – Linear and integer programming

Obligatory Assignment 0.2, Spring 2018 [pdf format]

Solution:

Contains Solutions!

Deadline: Wednesday, April 4 at 18:00.

In red the modifications after publication.
This is the second obligatory assignment in DM559. The number 0.2 indicates that this is Assignment 2 of the part 0 of the course (the part on Linear Algebra). The part 1 will start in week 12.
The Assignment has to be carried out individually. You can consult with peers, if you are unable to proceed, but without exchanging full solutions.
The submission is electronic via:

`http://valkyrien.imada.sdu.dk/milpApp/`.

The deliverable is a PDF document, as it will be at the written exam. Hence, experiment and get acquainted with the tools and forms you want to use for producing this document in a similar setting as the exam. In any case, you have to put your answers in this template You can handwrite your answers and add them as picture in the template. Be aware that at the exam you can use digital pen or hand scanner (that is, a silent scanner) but you cannot bring handycameras. You can of course also typset your answers in LaTeX. You can use either Danish or English. Keep the newpage separation after each exercise present in the template. Write your name and CPR number where indicated in the template.

In some parts, you are asked to write Python code. You have to include the code in the PDF document *together with the output of the execution of the code*. Use the LaTeX environment `lstlisting` as shown in the template for doing this. Your code must work correctly if fully copied and pasted from your report.

In your answers, you have to justify the steps that you are doing. If theorems and definitions from the slides of the course or from the text book (specify which edition) are used, give reference.

The tasks are all about linear algebra. Your goal is to answer correctly to as many subtasks as you can. Tasks and subtasks are presented in increasing order of difficulty. You are welcome to ask in class for explanations that could put you on the right path for solving the tasks.

Exercises 1-3b must be completed to pass the assignment and I expect that they can be carried out in less than one working day if you are up-to-date with the subject. Exercises 3c-5 are real-life applications of linear algebra.

Exercise 1* Change of basis

Carry out the computations for the following exercise in Python (remember to provide the code as well as the result from execution):
Set:

```
import numpy as np

U = np.random.randint(-10,10,size=(4,4))
V = np.random.randint(0,10,size=(4,4))
b = np.ones(4)
```

(a) Use the Numpy function `numpy.linalg.matrix_rank` to determine whether the column vectors of a matrix form a basis of $\mathbb{R}^4$. That is, compute the rank of U and the rank of V and explain how from this result you can deduce whether U and V form a basis for $\mathbb{R}^4$.

Solution:

```
In [54]: import numpy as np
    ...:
    ...: U = np.random.randint(-10,10,size=(4,4))
    ...: V = np.random.randint(0,10,size=(4,4))
    ...: b = np.ones(4)

In [55]: print( np.linalg.matrix_rank(U) )
    ...: print( np.linalg.matrix_rank(V) )
4
4
```

(b) Use Python to compute the transition matrix from the standard basis for $\mathbb{R}^4$ to the ordered basis $E = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\}$. Use this transition matrix to compute the coordinate vector $\mathbf{c}$ of $\mathbf{b}$ with respect to E. Verify that

$$\mathbf{b} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + c_3\mathbf{u}_3 + c_4\mathbf{u}_4 = U\mathbf{c}.$$

[Recall that in Python a column vector $\mathbf{u}_{\cdot,j}$, or simply $\mathbf{u}_j$, of a matrix U is determined by `U[:,j]`. Use the function `numpy.allclose` with relative tollerance 10^{-5} and absolute tollerance 10^{-8} to verify that the computed vector $\mathbf{b}$ corresponds to the original one.]

Solution:

```
In [56]: P_E=U
    ...: P_E_inv = np.linalg.inv(P_E)
    ...:
    ...: c=np.dot(P_E_inv,b)
    ...:
    ...: b1=np.dot(U,c)
    ...:
    ...: print( np.allclose(b1, b, rtol=1e-05, atol=1e-08, equal_nan=False) )
True
```

(c) Compute the transition matrix from the standard basis to the basis $F = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$, and use this transition matrix to find the coordinate vector $\mathbf{d}$of $\mathbf{b}$ with respect to F. Verify that

$$\mathbf{b} = d_1\mathbf{v}_1 + d_2\mathbf{v}_2 + d_3\mathbf{v}_3 + d_4\mathbf{v}_4 = V\mathbf{d}.$$

Solution:

```
In [57]: P_F=V
    ...: P_F_inv = np.linalg.inv(P_F)
    ...:
    ...: d=np.dot(P_F_inv,b)
    ...:
    ...: b1=np.dot(V,d)
    ...:
    ...: print( np.allclose(b1, b, rtol=1e-05, atol=1e-08, equal_nan=False) )
True
```

(d) Use Python to compute the transition matrix S from E to F and the transition matrix T from F to E. How are S and T related? Verify that $S\mathbf{c} = \mathbf{d}$ and $T\mathbf{d} = \mathbf{c}$.

Solution:

```
In [58]: S = np.dot(P_F_inv, P_E)
    ...: T = np.dot(P_E_inv, P_F)
    ...:
    ...: print( np.allclose(np.dot(S,T), np.eye(4)) )
    ...:
    ...: d1 = np.dot(S,c)
    ...: print( np.allclose(d1, d, rtol=1e-05, atol=1e-08, equal_nan=False) )
    ...:
    ...:
    ...: c1 = np.dot(T,d)
    ...: print( np.allclose(c1, c, rtol=1e-05, atol=1e-08, equal_nan=False) )
True
True
True
```

Exercise 2* Eigenvalues and eigenvectors

(a) Find the eigenvalues and the corresponding eigenspaces for the following matrices:

$$\begin{bmatrix} 3 & -1 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Carry out the operations for the first matrix by hand and those for the second with Python.

(b) For each of the two matrices, state whether the matrix can be diagonalized. (To answer this last part it may help you to consider the algebraic and the geometric multiplicity of the eigenvalues).

Solution:

- $\lambda_1 = \lambda_2 = 2$, the eigenspace is spanned by $[1, 1]$. The algebraic multiplicity of the eigenvalue 2 is 2 and the geometric multiplicity is 1. Hence, the matrix is not diagonalizable.
- $\lambda_1 = 2$ and the eigenspace is spanned by $[1, 1, 0]$. $\lambda_2 = \lambda_3 = 1$, the eigenspace is spanned by $[1, 0, 0], [0, 1, -1]$. The algebraic multiplicity of the eigenvalue 1 is 2 and the geometric multiplicity is also 2. Hence, the matrix is diagonalizable.

Exercise 3* Linear transformations and computer graphics

Let

$$R = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

The columns of R represent the coordinates of points in the plane.

a) Draw the figure whose vertices correspond to the column vectors of R and whose sides are the segments connecting the points. What type of figure is it?

b) Recall from the lectures that we can transform a figure by changing the positions of the vertices and then re-drawing the figure. If the transformation L is linear, it can be carried out as a matrix multiplication, that is, $L(\mathbf{x}) = A\mathbf{x}$. Then, viewing a succession of such drawings will produce the effect of animation.

The four primary geometric transformations that are used in computer graphics are as follows:

i. Dilations and contractions. A dilation increases the size of the figure by a factor $c > 1$, and a contraction shrinks the figure by a factor $c < 1$. Provide the matrix A for these transformations by a generic factor c.

Solution:

$$cI$$

ii. Reflections about an axis. Provide the matrix A for the reflection around, for example, the x-axis.

Solution:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

iii. Rotations around the origin by an angle θ in the counterclockwise direction. Provide the matrix A for a generic rotation of an angle θ.

iv. Translations. Write this transformation in matrix and vector notation and argue that this transformation is not linear.

c) The way to perform translations as linear transformation is to introduce a new system of coordinates called *homogeneous coordinates*. The homogeneous coordinate system is formed by equating each vector in $\mathbb{R}^2$ with a vector in $\mathbb{R}^3$ having the same first two coordinates and having 1 as its third coordinate.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \iff \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

When we want to plot a point represented by the homogeneous coordinate vector $[x_1, x_2, 1]$, we simply ignore the third coordinate and plot the ordered pair $[x_1, x_2]$. The linear transformations discussed earlier must now be represented by 3×3 matrices.

Rewrite the matrix R in homogeneous coordinates and provide the corresponding transformation matrices for the dilations, reflections, and rotations introduced above.

Solution:

$$\begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

d) Show how a translation by a vector $\mathbf{a} = [a_1, a_2, a_3]$ can be performed as a linear transformation and hence as a matrix multiplication in the homogeneous systems.

Exercise 4 Shadow projection

Assume we have a plane that goes through the origin with the normal vector $\mathbf{n} = [0, 0, 1]$. Furthermore, assume the sun is infinitely far away from the origin, positioned in the direction $\mathbf{r} = [0.6, 0.3, 1]$. We have a lot of points above the plane (for example the corners of a cube), and would like to know their shadows on the plane. Assume also that we have been reassured that this particular type of shadow projection can be expressed as a linear mapping or transformation.

It would be possible to calculate the shadow point by point, but it is more convenient if we create a linear transformation $\mathbf{y} = A\mathbf{x}$ so that we can directly get the projected point $\mathbf{y}$ by simply multiplying any point $\mathbf{x}$ by the matrix $\mathbf{A}$. This should be possible, given that the problem is linear.

In this case, we take advantage of one of the results seen in the lectures that to describe completely the transformation matrix A we only need to know what happens to the three unit vectors $\mathbf{e}_1 = [1, 0, 0]$, $\mathbf{e}_2 = [0, 1, 0]$ and $\mathbf{e}_3 = [0, 0, 1]$. The first column vector of the transformation matrix is simply the image of the first unit vector $\mathbf{e}_1$, and so forth. Hence, if we find out where on the plane the shadow of the point $[1, 0, 0]$ lands, we have the first column of A.

1. Do this. Construct the matrix A projecting the three points determined by the vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. To find the projected (transformed) points you have to find the intersection between the ray of sun passing through the points and the plane. Hence, write the equation of the plane and for each point the vector equation of the line passing through it, and then find the intersection point with the plane. The intersection points will give the columns of the matrix A. (The procedure is depicted in the first three plots of Figure 1.)

   Solution:

   We can write the plane on the form $ax + by + cz = d$, and since we know the normal is $[0, 0, 1]$ this simplifies to $cz = d$. Furthermore, since the origin is contained in the plane, the equation further simplifies to $z = 0$. We can now take the line that goes through the point $[p_x, p_y, p_z]$ and follow it in the direction of the sun, $[r_x, r_y, r_z]$,

   $$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \lambda \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \lambda \begin{pmatrix} 0.6 \\ 0.3 \\ 1.0 \end{pmatrix}. \tag{1}$$

   Inserting this line equation into the plane equation should give us the distance λ that we need to travel from P to get to the intersection. When we insert it into the plane equation $z = 0$ we get $p_z + 1.0\lambda = 0$.

   For the first unit vector $e_1 = [1, 0, 0]$, we have $p_z = 0$, giving $0 + 0.6\lambda = 0$, which means that $\lambda = 0$. The intersection point is therefore at $[1, 0, 0] + 0\mathbf{r} = [1, 0, 0]$. This is therefore the first column of A. For the second unit vector $e_2 = [0, 1, 0]$, we have $p_z = 0$, giving the equation $0 + 0.3\lambda = 0$, or $\lambda = 0$. Therefore the second column of A is $[0, 1, 0] + 0\mathbf{r} = [0, 1, 0]$. The third unit vector gets $p_z = 1$ and hence $1 + 1.0\lambda = 0$ and $\lambda = -1$ the last column of is $[0, 0, 1] - 1[0.6, 0.3, 1] = [-0.6, -0.3, 0]$.

   In summary, we have now created a linear mapping $\mathbf{y} = A\mathbf{x}$ that takes a point $\mathbf{x}$ and maps it to its shadow y. The matrix equals

   $$A = \begin{pmatrix} 1 & 0 & -0.6 \\ 0 & 1 & -0.3 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2}$$

2. The matrix found at the previous point is used in the last two plots of Figure 1. Each point of the cube is just projected to the plane using $\mathbf{y} = A\mathbf{x}$ with the matrix A from above. By drawing each face of the cube using the projected coordinates instead of the original coordinates, it is possible to draw the shadow of the cube.

   Find the shadow of the face of the cube whose vertices are the columns of this matrix:

   $$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 & 2 \end{bmatrix}$$

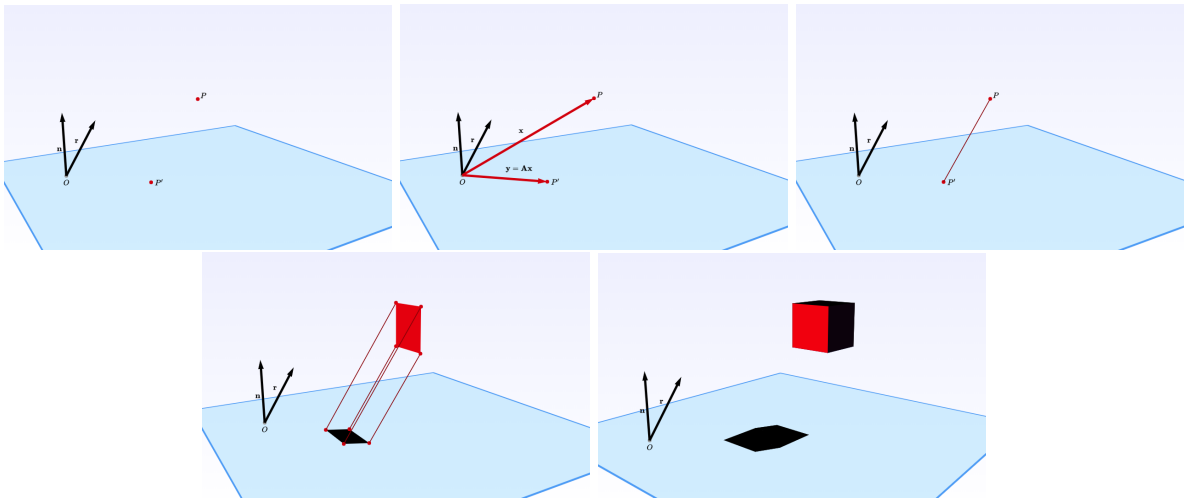


Figure 1: A perspective view of the situation in Exercise .

(Use Python for the calculations.)

Solution:

```
: A = np.array([[ 1,0,-0.6],[ 0,1,-0.3],[0, 0,0]])
: C=np.array([[ 1,1,1,1,1],[2,2,1,1,2],[2,1,1,2,2]])
:
: print( np.dot(A,C) )

[[-0.2 0.4 0.4 -0.2 -0.2]
 [ 1.4 1.7 0.7 0.4 1.4]
 [ 0. 0. 0. 0. 0. ]]
```

Exercise 5 Modeling a web surfer: Google's PageRank

[This exercise is an adaptation from Philip Klein's course: "Coding the Matrix"]
PageRank is the algorithm that Google originally used to determine the "importance" (or rank) of a web page.

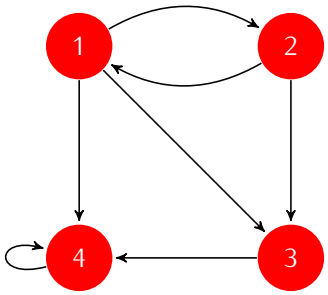
> The idea for PageRank is this: Define a Markov chain that describes the behavior of a random web-surfer. Consider the stationary distribution of this Markov chain. Define the weight of a page to be the probability of that page in the stationary distribution. Higher-probability pages are considered better. So when the user submits a query consisting of a set of words, present the web pages containing these words, in descending order of probability.
>
> Conveniently, the PageRank vector (the stationary distribution) does not depend on any particular query, so it can be computed once and then used for all subsequent queries. (Of course, Google periodically recomputes it to take into account changes in the web.)

We start with a rudimentary Markov chain and we discover why it needs to be improved.

a) In each iteration, the random surfer selects an outgoing link from his current web page, and follows that link. (If the current web page has no outgoing link, the surfer stays stand.) We consider the small Web network of the picture, consisting of only four sites linked together as shown.

Let $k(j)$ denote the number of links from page j to other pages in the network. If $k(j) \neq 0$ and no other knowledge about the relevance of the pages is given, then it is reasonable to assume that the surfer follows links from the current web page with probability:

$$a_{ij} = \begin{cases} 1/k(j) & \text{if there is a link from page } j \text{ to page } i \\ 0 & \text{otherwise} \end{cases}$$



Write the transition matrix A_1 of the Markov chain.

According to this Markov chain, how likely is that the random surfer is at each page after many iterations? What are the most likely pages?

Use the power method and carry out the operations in Python for the following cases:

- if it starts at page 2 and takes an even number of iterations
- if it starts at page 2 and takes an odd number of iterations
- if it starts at page 3 and takes an odd number of iterations

You should find that the answer depends on where the surfer starts and how many steps he takes.

Solution:

```
A=np.matrix([[f(0),f(1,2),f(0),f(0)],[f(1,3),f(0),f(0),f(0)],[f(1,3),f(1,2),f(0),f(0)
    ],[f(1,3),f(0),f(1,1),f(1)]])
np.linalg.det(A)
x_0=np.array([0,1,0,0])
print A
A_n=A
for i in range(21):
    A_n = np.dot(A_n,A)
x_n = np.dot(A_n,x_0)
x_n
```

The probabilities are almost the same except that the probabilities of 1 and 2 are swapped.

b) From the point of view of computing definitive pageranks using the power method, there are two things wrong with this Markov chain:

i. There are multiple clusters in which the surfer gets stuck. One cluster is page 1 and page 2, and the other cluster is page 4.

ii. There is a part of the Markov chain that induces periodic behavior: once the surfer enters the cluster page 1, page 2, the probability distribution changes in each iteration.

The first property implies that there are multiple stationary distributions. The second property means that the power method might not converge. We want a Markov chain with a unique stationary distribution so we can use the stationary distribution as an assignment of importance weights to web pages. We also want to be able to compute it with the power method. We apparently cannot work with the Markov chain in which the surfer simply chooses a random outgoing link in each step.

Consider then an alternative very simple Markov chain: the surfer jumps from whatever page he's on to a page chosen uniformly at random.

Write the transition matrix A_2 for this Markov chain. Is there a unique stationary distribution? Which one?

c) As you may have discovered the latter Markov chain does not in any way reflect the structure of the Web network. Using the stationary distribution to assign weights would provide no information at all.

Instead, we will use a mixture of these two Markov chains. That is, we will use the Markov chain whose transition matrix is

$$A = 0.85A_1 + 0.15A_2$$

Show that the matrix A is a *stochastic matrix*, that is, the sum of the entries in all columns is 1.

Solution:

Since every column of A_1 sums to 1, every column of $0.85A_1$ sums to 0.85, and since every column of A_2 sums to 1, every column of $0.15A_2$ sums to 0.15, so (finally) every column of $0.85A_1 + 0.15A_2$ sums to 1.

The Markov chain corresponding to the matrix A describes a surfer obeying the following rule:

- With probability 0.85, select one of the links from the current web page, and follow it.
- With probability 0.15, jump to a web page chosen uniformly at random. (This is called teleporting in the context of PageRank.)

You can think of the second item as modeling the fact that sometimes the surfer gets bored with where he is. However, it plays a mathematically important role. The matrix A is a positive matrix (every entry is positive). A theorem ensures that there is a unique stationary distribution, and that the power method will converge to it.

Calculate how likely it is that the random surfer is at each page after many iterations and rank consequently the pages. You can use the theory of diagonalization or the power method and carry out the calculation with Python.

Exercise 6* Linear transformations and change of basis

Let L be the linear operator mapping $\mathbb{R}^3$ into $\mathbb{R}^3$ defined by

$$L(\mathbf{x}) = L\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} 3x_1 - x_2 - 2x_3 \\ 2x_1 - 2x_3 \\ 2x_1 - x_2 - x_3 \end{bmatrix}$$

and let

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{v}_2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \quad \mathbf{v}_3 = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}.$$

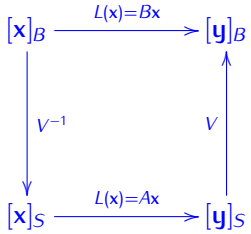
Find the transition matrix V corresponding to a change of basis from $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ to $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and use it to determine the matrix B representing L with respect to $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$. Explain by words what the transformation does in this coordinate system.

Solution:

Let $S = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and $B = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$. Recalling that $[\mathbf{x}]_S = V[\mathbf{x}]_B$ the transition matrix V from B to S is given by the representation of the basis B with respect to S, hence:

$$V = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & -2 \\ 1 & 0 & 1 \end{bmatrix}$$

To find the matrix B that represents L with respect to B we need to find the composition of the following transformations: first transform $[\mathbf{x}]_B$ to $[\mathbf{x}]_S$, then apply the transformation L to obtain $[\mathbf{y}]_S = L([\mathbf{x}]_S)$ and then transform $[\mathbf{y}]_S$ to $[\mathbf{y}]_B$:



Thus $B = V^{-1}AV$. We use Python to calculate the inverse of V:

```
import numpy as np

V=np.array([[1,1,0],[1,2,-2],[1,0,1]])
V_1 = np.linalg.inv(V)
% run bmatrix
print bmatrix(V_1)

# Alternative way:
import sympy as sy
I=np.eye(3)
V1=sy.Matrix(np.hstack([V,I]))
V1.rref()
```

$$V^{-1} = \begin{bmatrix} -2. & 1. & 2. \\ 3. & -1. & -2. \\ 2. & -1. & -1. \end{bmatrix}$$

Thus,

$$B = V^{-1}AV = \begin{bmatrix} -2 & 1 & 2 \\ 3 & -1 & -2 \\ 2 & -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & -1 & -2 \\ 2 & 0 & -2 \\ 2 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & -2 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0. & 0. & 0. \\ 0. & 1. & 0. \\ 0. & 0. & 1. \end{bmatrix}$$

```
A=np.array([[3,-1,-2],[2,0,-2],[2,-1,-1]])

print bmatrix(np.dot(np.dot(V_1,A),V))
```

The transformation with respect to the basis V is a projection on the plane $\mathbf{v}_2, \mathbf{v}_3$ since every $\mathbf{v}_1$ coordinate is set to zero while the other two coordinates are kept as they are.