Índice

Lección 1 del curso - Vídeo 1

Marcos Bujosa

12 de enero de 2024

1. Vectores de $\mathbb{R}^n$ - T01

1.1. Español

1.1.1. Escena 1 - Definición de vector de $\mathbb{R}^n$

L01_V01_E01_VectoresDefinicion

```
<<Carga de la librería Manim y NacAL con AzureService>>

<<copyright>>

class L01_V01_E01_VectoresDefinicion(VoiceoverScene):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )

        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"\usepackage{nacal}")

        # Copyright lateral
        <<copyrightLateral>>

        # Portada
        titulo = Title(r"Definición de vector de \R[n] y notación",
                    tex_template = myTemplate,
                    font_size=70).set_color(BLUE)
        self.play(Write(titulo))
        self.wait(1.5)
        self.play(FadeOut(titulo))

        # Definición de vector
        definicion = Tex("Un vector de ",
                         r"$\R[n]$",
                         r" es una \emph{lista ordenada} de $n$ números",
                         tex_template = myTemplate,
                         ).to_edge(UP).set_width(13)
        with self.voiceover(text=r"Un vector de R n es una lista ordenada de números. ") as tracker:
            self.add(definicion)

        # Aclaración de la notación
        with self.voiceover(text=r"La R indica que los números son reales.") as tracker:
            self.play(Circumscribe(definicion[1][0], fade_out=True), run_time=tracker.duration)

        with self.voiceover(text=r"Y el superíndice n indica que la lista contiene n números.") as tracker:
            self.play(Circumscribe(definicion[1][1], fade_out=True), run_time=tracker.duration)
            self.wait(0.3)

        with self.voiceover(text=r"""que la lista sea ordenada
        significa que importa el orden en el que aparecen sus
        elementos.""") as tracker:
            self.play(Circumscribe(definicion[2][5:18], fade_out=True), run_time=tracker.duration)

```

```
        # Ejemplos
        Ej = Tex(r"\textbf{Ejemplos:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(GREEN).next_to(definicion, DOWN, aligned_edge=LEFT)
        self.add(Ej)

        d = nc.Vector([1,2,3],'fila')
        Ej1 = MathTex( d.latex(), "\\ne", nc.Vector(reversed(d),'fila').latex() )
        with self.voiceover(text=r"Por ejemplo, los vectores, uno dos tres y tres dos uno, son distintos.") as tracker:
            self.play(FadeIn(Ej1))
            self.wait(tracker.duration-0.5)
            self.play(FadeOut(Ej1))

        p = nc.Vector([sp.pi,sp.pi,sp.pi,sp.pi],'fila')
        Ej2 = MathTex(p.latex(), r"\ne", nc.Vector(p[(1,2),'fila').latex() )
        with self.voiceover(text=r"""También son distintos los vectores con distinta cantidad de
        elementos. Por ejemplo, el vector de la izquierda pertenece a R 4 por ser una lista de
        4 números. El de la derecha pertenece a R
        2.""") as tracker:
            self.add(Ej2)
            self.wait(tracker.duration-0.5)
            self.play(FadeOut(Ej2))

        c = nc.Vector([sp.pi,nc.fracc(3,4),0,0.11],'fila')
        Ej3 = MathTex(c.latex(),"=",c.latex())
        with self.voiceover(text=r"""En consecuencia, dos vectores serán iguales si, y solo si, sus correspondientes
        listas son idénticas""") as tracker:
            self.play(FadeIn(Ej3))
            self.wait(tracker.duration)
            self.play(FadeOut(Ej3))
        self.play(FadeOut(Ej))
        self.wait()
```

1.1.2. Escena 2 - Notación vectores de $\mathbb{R}^n$

L01_V01_E02_VectoresNotacion

```
class L01_V01_E02_VectoresNotacion(VoiceoverScene):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )

        #self.set_speech_service(GTTSService(lang="es", tld="com"))
        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"\usepackage{nacal}")

        # Copyright lateral
        <<copyrightLateral>>

        # Definición de vector
        definicion = Tex("Un vector de ",
                         r"$\R[n]$",
                         r" es una \emph{lista ordenada} de $n$ números",
                         tex_template = myTemplate,
                         ).to_edge(UP).set_width(13)
        self.add(definicion)
        d = nc.Vector([1,2,3],'fila')
        p = nc.Vector([sp.pi,sp.pi,sp.pi,sp.pi],'fila')

        # Notacion
        Notac = Tex(r"\textbf{Notación:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(BLUE).next_to(definicion, DOWN, aligned_edge=LEFT)
        self.add(Notac)
        self.wait()

        a    = nc.Vector( [5, 1, 10] , 'fila')
        Not1 = MathTex(a.latex(),        tex_template = myTemplate,)
        Not2 = MathTex(r"=",             tex_template = myTemplate,)
```

```
        Not3 = MathTex(a.copy().latex(), tex_template = myTemplate,)
        grp1 = VGroup(Not1,Not2,Not3).arrange(RIGHT)
        with self.voiceover(text=r"""Para expresar un vector basta indicar la lista de elementos en su
        correspondiente orden.""") as tracker:
            self.add(grp1)
            self.play(Circumscribe(definicion[2][5:18]),run_time=tracker.duration)

        with self.voiceover(text=r"""Por este motivo podemos escribir un mismo vector tanto en
        horizontal como en vertical. Pero tenga en cuenta que la
        mayoría de manuales no respetan este convenio, y consideran,
        al contrario de lo que haremos nosotros, que vectores fila y
        vectores columna son objetos distintos.""") as tracker:
            self.wait(tracker.duration+0.3)

        with self.voiceover(text=r"""Siempre escribiremos la lista de números encerrada entre
        paréntesis; poniendo una coma detrás de cada elemento cuando
        escribamos el vector en horizontal.""") as tracker:
            self.play(Indicate(grp1[0][0][::len(grp1[0][0])-1]),
                      Indicate(grp1[2][0][0:2]), Indicate(grp1[2][0][-2:]),
                      run_time=tracker.duration/2)
            self.play(Flash(grp1[0][0][2]),
                      Flash(grp1[0][0][4]), Flash(grp1[0][0][7]),
                      run_time=tracker.duration/8)
            self.wait(tracker.duration/8)
            self.play(Circumscribe(grp1[0]))
            self.play(FadeOut(grp1))

        VectorNoNumero =  MathTex(r"(3)",r"\ne",(3*nc.V1(1)).latex(),r"\in\R[1]", tex_template = myTemplate,)
        with self.voiceover(text=r"""Así podremos distinguir un número entre paréntesis de un
        vector de R 1;""") as tracker:
            self.add(VectorNoNumero)
            self.play(Indicate(VectorNoNumero[0]),run_time=tracker.duration*2/3)
            self.play(Indicate(VectorNoNumero[2]),run_time=tracker.duration/3)
        with self.voiceover(text=r"""es decir, de una lista con un solo número.""") as tracker:
            self.play(Indicate(VectorNoNumero[3]),
                      Flash(definicion[2][-9]),
                      run_time=tracker.duration)
            self.play(FadeOut(VectorNoNumero))

        Vectores = MathTex(r"\Vect{a}, \Vect{b}, \Vect{c},\ldots\Vect{x}, \Vect{y}, \Vect{z}",
                           tex_template = myTemplate,).move_to( UP )
        Vector1  = MathTex(r"\Vect{a}=",a.copy().latex(),   tex_template = myTemplate,)
        Vector2  = MathTex(r"\Vect{d}=",d.copy().latex(),   tex_template = myTemplate,)
        Vector3  = MathTex(r"\Vect{x}=",p.copy().latex(),   tex_template = myTemplate,)
        grp3 = VGroup(Vector1,Vector2,Vector3).arrange(RIGHT, buff=2).next_to(Vectores, DOWN)
        with self.voiceover(text=r"Para denotar vectores usaremos letras minúsculas en negrita cursiva.") as tracker:
            self.add(Vectores)
            self.add(grp3)
            self.wait(tracker.duration/2)
            self.play(Indicate(Vectores),run_time=tracker.duration/2)
            self.play(FadeOut(Vectores))
            self.play(Indicate(Vector1[0][0],scale_factor=2.),
                      Indicate(Vector2[0][0],scale_factor=2.),
                      Indicate(Vector3[0][0],scale_factor=2.),
                      run_time=1.5)
            self.play(FadeOut(grp3))

        Vnulo = MathTex(r"\Vect{0}", tex_template = myTemplate,)#.move_to( UP )
        with self.voiceover(text=r"Un cero en negrita denota un vector cuyas componentes son todas nulas.") as tracker:
            self.add(Vnulo)
            self.play(Indicate(Vnulo))
            self.wait(tracker.duration/2)
            self.play(FadeOut(Vnulo))

        Vnulo1 = MathTex(r"\Vect{0}=", nc.V0(1).latex(), ",",  tex_template = myTemplate,)
        Vnulo2 = MathTex(r"\Vect{0}=", nc.V0(2).latex(), ",",  tex_template = myTemplate,)
        Vnulo3 = MathTex(r"\Vect{0}=", nc.V0(3).latex(), ",",  tex_template = myTemplate,)
        Vnulo6 = MathTex(r"\Vect{0}=", nc.V0(6).latex(), ",",  tex_template = myTemplate,)
```

```
        VnuloN = MathTex(r"\Vect{0}\in\R[100]",             tex_template = myTemplate,)
        grp2   = VGroup(Vnulo1,Vnulo2,Vnulo3,Vnulo6,VnuloN).arrange(RIGHT, buff=0.7)
        with self.voiceover( text = r"""Fíjese que un cero en negrita
        no indica su número de componentes. Normalmente la cantidad de
        ceros se deduce del contexto.""" ) as tracker:
            self.add(grp2)
            self.wait(tracker.duration)
            self.play(FadeOut(grp2),FadeOut(Notac),FadeOut(definicion))
            self.wait(1.5)
```

1.1.3. Escena 3 - Selección de elementos de un vector de $\mathbb{R}^n$

L01_V01_E03_VectoresElementos

```
class L01_V01_E03_VectoresElementos(VoiceoverScene):
    def construct(self):
        self.set_speech_service( AzureService(voice="es-ES-AlvaroNeural" ) )

        #self.set_speech_service(GTTSService(lang="es", tld="com"))

        myTemplate = TexTemplate()
        myTemplate.add_to_preamble(r"\usepackage{nacal}")

        # Copyright lateral
        <<copyrightLateral>>

        # Notacion
        Notac = Tex(r"\textbf{Notación para los elementos:}",
                 tex_template = myTemplate,
                 font_size=50).set_color(BLUE).to_corner(UL)
        self.wait()
        self.add(Notac)
        self.wait()

        # Elementos de un vector
        v_generico  = nc.Vector(sp.symbols('a:5')[1:],'fila')
        cs = MathTex(r"\Vect{a}=",
                     v_generico.latex(),
                     tex_template = myTemplate,)

        with self.voiceover(text = r"""Lo habitual es denotar cada
        elemento de un vector con la letra de su nombre sin negrita.""" ) as tracker:
            self.wait()
            self.play(FadeIn(cs), run_time=0.5)
            self.play( Circumscribe(cs[1][1]),
                       Circumscribe(cs[1][4]),
                       Circumscribe(cs[1][7]),
                       Circumscribe(cs[1][10]),
                       run_time=tracker.duration/2)

        with self.voiceover(text = r"""indicando con un subíndice su posición en la lista.""" ) as tracker:
            self.play( Flash(cs[1][2]),
                       Flash(cs[1][5]),
                       Flash(cs[1][8]),
                       Flash(cs[1][11]),
                       run_time=tracker.duration)
            self.play(FadeOut(cs))

        c = nc.Vector([sp.pi,nc.fracc(3,4),0,0.11],'fila')
        vector_c = MathTex(r"\Vect{c}=",c.latex(),tex_template = myTemplate,)
        A = VGroup(*[ MathTex("c_"+str(i+1)+"=&"+sp.latex(e))  for i,e in enumerate(c.lista)
                    ]).arrange(DOWN,aligned_edge=LEFT, buff=.5)
        B = Brace(A, LEFT)
        C = VGroup(A,B)
        Elementos_c   = VGroup(vector_c, C).arrange(RIGHT, buff=1)
        with self.voiceover(text = r"""Así, para el vector C """) as tracker:
            self.play(FadeIn(vector_c))
            self.play(GrowFromCenter(B),FadeIn(A))
```

```
55
56          with self.voiceover(text = r"""con c 1 denotamos su primera componente""") as tracker:
57              self.play( Indicate(vector_c[1][1]),    Indicate(A[0]) )
58          with self.voiceover(text = r"""con c 2 la segunda""") as tracker:
59              self.play( Indicate(vector_c[1][3:6]),  Indicate(A[1]) )
60          with self.voiceover(text = r"""y del mismo modo con el resto de componentes""") as tracker:
61              self.play( Indicate(vector_c[1][7],    run_time=tracker.duration/2), Indicate(A[2], run_time=tracker.duration/2) )
62              self.play( Indicate(vector_c[1][9:13], run_time=tracker.duration/2), Indicate(A[3], run_time=tracker.duration/2) )
63              self.wait(0.5)
64              self.play( FadeOut(vector_c), FadeOut(B), FadeOut(A) )
65              self.wait(0.5)
66
67          with self.voiceover(text = r"""El hecho de emplear dos tipos
68          de fuentes:""" ) as tracker:
69              self.add(cs)
70              self.wait(tracker.duration)
71
72          with self.voiceover(text = r"""con negrita los vectores y sin negrita los
73          componentes, dificulta distinguirlos a primera vista""" ) as tracker:
74              self.play( Indicate(cs[0][ 0],scale_factor=2.),
75                         Indicate(cs[0][ 0],scale_factor=2.),
76                         Indicate(cs[1][ 1],scale_factor=2.),
77                         Indicate(cs[1][ 4],scale_factor=2.),
78                         Indicate(cs[1][ 7],scale_factor=2.),
79                         Indicate(cs[1][10],scale_factor=2.), run_time=tracker.duration*2/3)
80
81          MTa = MathTex(r"\eleVR{a}{i}",tex_template = myTemplate).scale(3)
82          MTb = MathTex(r"{a}_{i}=",tex_template = myTemplate).scale(3).next_to(MTa, LEFT)
83          VG  = VGroup(MTb,MTa)
84          with self.voiceover(text = r"""Es más clara y operativa una notación que use un único tipo de fuente,
85          y que denote la selección de elementos con un operador. Por
86          ejemplo con una barra vertical.""" ) as tracker:
87              self.play(cs.animate.to_corner(DL),
88                        run_time=tracker.duration*4/5)
89              self.play(Indicate(VG[1][0][1]))
90              self.wait(0.5)
91
92          def VectorGenerico(s,n):
93              elem = lambda s,i: sp.Symbol(r'\eleVR{'+ s +'}{'+ str(i) + '}')
94              return nc.Vector([elem(s,i) for i in range(1,n+1)], 'fila')
95
96          v_generico2 = VectorGenerico('a',4)
97          cs2 = MathTex(r"=",
98                        v_generico2.latex(),
99                        tex_template = myTemplate,).next_to(cs, RIGHT)
100
101         VGB = VGroup(*[MathTex(sp.latex(e) + "=\; & \eleVR{a}{" + str(i+1) + "}",
102                                tex_template = myTemplate)
103                          for i,e in enumerate(v_generico.lista)
104                          ]).scale(3)
105
106         with self.voiceover( text = r"""Por ello, para denotar una componente, escribiremos un subíndice con una
107         barra que medie entre el vector y el índice de la
108         componente""" ) as tracker:
109             self.play(FadeIn(VG[1]))
110             self.wait(tracker.duration/3)
111             self.play(Indicate(VG[1][0][1:], run_time=tracker.duration/4))
112             #self.wait(tracker.duration/3)
113             self.play(Indicate(VG[1][0][-1], run_time=tracker.duration/5))
114             self.play(Write(VG[0]))
115             self.wait()
116             self.play(VG.animate.move_to([0,0,0]))
117             self.play(Transform(VG[1][0][-1],VGB[0][0][-1]),
118                       Transform(VG[0][0][:2],VGB[0][0][:2]), run_time=1.5)
119             self.play( FadeIn(cs2) )
120             self.play(FadeTransform(VGB[0][0][0:2],cs[1][ 1: 3]),
121                       FadeTransform(VGB[0][0][3:],cs2[1][ 1: 6]), FadeOut(VG), run_time=1.5)
122             self.play(FadeIn(VGB[1]), FadeOut(VGB[0][0][2]))
```

```
            self.play(FadeTransform(VGB[1][0][0:2],cs[1] [4: 6]),
                      FadeTransform(VGB[1][0][3:],cs2[1][ 7:12]), FadeOut(VGB[1][0][2]), run_time=1.5)
            self.play(FadeIn(VGB[2]))
            self.play(FadeTransform(VGB[2][0][0:2],cs[1][ 7: 9]),
                      FadeTransform(VGB[2][0][3:],cs2[1][13:18]), FadeOut(VGB[2][0][2]), run_time=1.5)
            self.play(FadeIn(VGB[3]))
            self.play(FadeTransform(VGB[3][0][0:2],cs[1][10:12]),
                      FadeTransform(VGB[3][0][3:],cs2[1][19:24]), FadeOut(VGB[3][0][2]), run_time=1.5)
            self.play(FadeOut(Notac),FadeOut(cs),FadeOut(cs2))

        MTLR = MathTex(r"\eleVR{a}{i}",r"\;=\eleVL{a}{i}",tex_template = myTemplate).scale(3)
        with self.voiceover( text = r"""Además, admitiremos que el operador selector actúe tanto por la derecha
        como por la izquierda.""" ) as tracker:
            self.play(FadeIn(MTLR[0]), run_time=2*tracker.duration/3)
            self.play(FadeIn(MTLR[1]))
            self.wait(tracker.duration/3+0.5)
            self.play(FadeOut(MTLR))
            self.wait()
```

1.2. Versión en inglés

```
manim_render_translation $fichero.py -s $escena -d $escenaENG -l en -ql
```

2. Trozos comunes de código

2.1. Carga de la librería Manim y NacAL

Carga de la librería Manim y NacAL

```
from manim import *
from manim_voiceover import VoiceoverScene
from manim_voiceover.services.gtts import GTTSService
import nacal as nc
import sympy as sp

# PARA LA TRADUCCIÓN (pero no me ha funcionado)

#from manim_voiceover.translate import get_gettext
# # It is good practice to get the LOCALE and DOMAIN from environment variables
#import os
#LOCALE = os.getenv("LOCALE")
#DOMAIN = os.getenv("DOMAIN")
# The following function uses LOCALE and DOMAIN to set the language, and
# returns a gettext function that is used to insert translations.
#_ = get_gettext()
```

Carga de la librería Manim y NacAL con AzureService

```
from manim import *
from manim_voiceover import VoiceoverScene
from manim_voiceover.services.azure import AzureService
import nacal as nc
import sympy as sp

# PARA LA TRADUCCIÓN (pero no me ha funcionado)

#from manim_voiceover.translate import get_gettext
# # It is good practice to get the LOCALE and DOMAIN from environment variables
#import os
#LOCALE = os.getenv("LOCALE")
#DOMAIN = os.getenv("DOMAIN")
# The following function uses LOCALE and DOMAIN to set the language, and
# returns a gettext function that is used to insert translations.
#_ = get_gettext()
```

2.2. Creditos

copyrightLateral
```
copyright = Tex(r"\textcopyright{\;} 2024\; Marcos Bujosa  ")
CGG  = VGroup(copyright).rotate(PI/2).scale(0.5).to_edge(RIGHT).set_color(GRAY_D)
self.add(CGG)
```

copyright
```
class ZCreditos(Scene):
    def construct(self):
        copyright = Tex(r"\textcopyright{\;} 2024 \; Marcos Bujosa")
        github = Tex(r"\texttt{https://github.com/mbujosab}").next_to(copyright, DOWN)
        CGG  = VGroup(copyright,github).scale(1.1)
        self.add(CGG)
        self.wait(10)
```

3. Rodando: 1,2,3... ¡acción!

1. Generamos un fichero `mpeg` por cada escena

   - Versión de poca calidad

rodando
```
echo $escena | manim -pql $fichero.py --disable_caching
```

   - Versión calidad HD1080

rodandoHD
```
echo $escena | manim -qh $fichero.py --disable_caching
```

2. Concatenamos las escenas en un único fichero `mpeg` y añadimos música de fondo.

   - Montando la versión de baja resolución

montando
```
rm -f $subdir/$video/$calidad/$video.mp4 list.txt
for f in $subdir/$video/$calidad/*.mp4 ; do echo file \'$f\' >> list.txt; done && ffmpeg -f concat -safe 0 -i list.txt -c

mkdir -p tmp

ffmpeg -i $subdir/$video/$calidad/$video.mp4 -i $music.mp3 -filter_complex "[0:a]apad[main]; [1:a]volume=0.04,apad[A]; [m
```

   - Montando la versión de resolución HD1080

montandoHD
```
rm -f $subdir/$video/$calidad/$video.mp4 list.txt
for f in $subdir/$video/$calidad/*.mp4 ; do echo file \'$f\' >> list.txt; done && ffmpeg -f concat -safe 0 -i list.txt -c

mkdir -p tmp

ffmpeg -i $subdir/$video/$calidad/$video.mp4 -i $music.mp3 -filter_complex "[0:a]apad[main]; [1:a]volume=0.04,apad[A]; [m
```

3. Fundimos a negro los últimos segundos del vídeo (y la música).

fade
```
dur=$(ffprobe -loglevel error -show_entries format=duration -of default=nk=1:nw=1 "tmp/$video.mp4") && offset=$(bc -l <<<
```

4. Copiamos el resultado a un lugar público

publicar

```
cp -f $video.mp4 $subdir/$video.mp4
```