Espacios vectoriales con producto escalar

1. Objetivo

Generar material educativo audiovisual para apoyar el curso de Álgebra Lineal de la Licenciatura en Física Biomédica de la UNAM, así como a l@s estudiantes de otras licenciaturas de la Facultad de Ciencias, con un enfoque particular hacia sentar las bases del formalismo de la mecánica cuántica.

2. Máximas

- No pensar en qué temas son más "animables", sino en cuáles se podrían beneficiar más de una animación para complementar su comprensión.
- Dentro de los temas anteriores, pensar en cuáles no tienen animaciones todavía, o cuáles tienen animaciones que se puedan mejorar significativamente.
- Dentro de los temas anteriores, dar prioridad a los que típicamente sean más difíciles de entender.
- Procurar presentar las ideas en la secuencia más clara posible, motivando los conceptos adecuadamente siempre que se pueda.
- Procurar que las ideas principales se puedan comprender de forma intuitiva a través de la exposición audiovisual, independientemente del conocimiento previo.
- Procurar pulir el contrapunto entre audio y video (guión, animaciones, colores, sincronización, ...).
- Procurar reducir la "densidad de información visual" en las animaciones[1], así como la "grasita extra" de los guiones.
- En vez de reproducir el material educativo del pasado con "técnicas del futuro", utilizar las técnicas del presente para *crear el material educativo del futuro*.

[1] Para "medir" la densidad de información visual de una escena, podemos imaginar cómo se vería la escena si todo lo que se plasma en ella se quedara "embarrado"; entre menor sea la "mancha" final, mejor (sin exagerar).

3. Ideas

- Siempre dirigirnos a el/la espectadora en primera persona.
- Enfocar al público en problemas concretos para motivar los conceptos y las discusiones.
- Dejar al menos una pregunta y un ejercicio al final de cada video para reforzar y expandir lo aprendido.
- Procurar complementar lo que se anima con referencias en la descripción (pero complementar "de verdad", es decir, con ejemplos que puedan ser muy distintos a lo presentado, pero que se sigan de ello o estén relacionados con el tema expuesto).

4. Videos

1. Producto escalar y bases ortogonales.
2. Norma inducida y bases ortonormales.
3. Ortogonalización y ortonormalización (Teorema de Gram-Schmidt).
4. Complementos ortogonales y sumas directas.
5. Operadores lineales, descomposición espectral y proyecciones ortogonales (introducción al Teorema espectral).
6. Funcionales lineales y operadores adjuntos
7. Operadores autoadjuntos y Teorema espectral.

5. Paleta de colores

Esta es la paleta de colores que utilizo en mis notas del curso, pero podemos cambiar los colores si quieren. Lo importante es que sea un tema consistente que nos sirva como guía. Algo bueno a considerar sería que sea una paleta amigable con personas con daltonismo.

Colores "primarios": Azul (#0087FF) y rojo (#FF0000).

Colores "secundarios": Naranja (#FF7700) y verde (#4FFF00).

En la rara ocasión que requiero un quinto color, usualmente utilizo magenta (#FF00FF) pero, como hemos discutido, esperemos no tener que llegar a eso. Les dejo una página para convertir códigos de colores entre el sistema hexadecimal, rgb con decimales y rgb normalizado con decimales, ya que diferentes programas utilizan distintos sistemas.

6. Convenciones de notación

En donde haya signos de equivalencia ($\equiv$), lo que esté del lado izquierdo será la convención para los videos **antes del de notación bra-ket** y, lo que esté del lado derecho, para los videos **después del de notación bra-ket**.

$\vec{u} \equiv \|u\rangle, \vec{v} \equiv \|v\rangle, ...$	vectores (elementos de un conjunto vectorial V)
$a \equiv \alpha, b \equiv \beta, ...$	escalares (elementos de un campo K que define un espacio vectorial)
$ab \equiv \alpha\beta$	producto entre los escalares a y b
$\overline{a} \equiv \alpha^*$	complejo conjugado del escalar a
$a\vec{u} \equiv \alpha \|u\rangle$	producto del vector $\vec{u}$ por el escalar α
$(x_1, ..., x_n)$	coordenada como n-tupla
$\begin{pmatrix} x_1 & ... & x_n \end{pmatrix}$	vector como n-tupla
$V + W$	suma de los espacios vectoriales V y W
$V \oplus W$	suma directa de los espacios vectoriales V y W
$\langle \mathbf{u}, \mathbf{v} \rangle \equiv \langle v \vert u \rangle$	producto escalar entre los vectores $\mathbf{u}$ y $\mathbf{v}$
$\|\|\mathbf{u}\|\| \equiv \|\| \|u\rangle \|\|$	norma del vector $\mathbf{u}$
$\hat{u}$	vector unitario
$P_{\mathbf{u}}(\mathbf{v}) \equiv P_{\|u\rangle}(\|v\rangle)$	proyección escalar del vector $\mathbf{v}$ sobre el vector $\mathbf{u}$
$\mathbf{u} \perp \mathbf{v} \equiv \|u\rangle \perp \|v\rangle$	ortogonalidad de los vectores $\mathbf{u}$ y $\mathbf{v}$
$\langle G \rangle$	Espacio vectorial generado por el conjunto de vectores G
$l.i.$	Conjunto linealmente independiente
$l.d.$	Conjunto linealmente dependiente
$o.n.$	Conjunto ortonormal
$\dim(V)$	Dimensión del espacio vectorial V

7. Para los *scripts* de Python...

- Hacer un archivo por escena con formato de nombre "A_d_T_E#.py" (Abreviación del Título del video correspondiente) donde # es el número de la escena, de acuerdo al guión; así, se pueden crear más opciones en el "prompt" de Manim para compilar por subescenas en vez de tener que compilar escenas completas cada vez. Las subescenas de cada archivo llevarán por nombre simplemente "SE#" donde # es el número de la subescena. Así aparecerán en el "prompt" de Manim en el mismo orden en que aparecen en la escena.
- Incluir al inicio de cada archivo la versión más actual de Manim en la que se ejecuta correctamente (e.g., "Manim Community V0.6.0").
- Incluir al inicio de cada archivo las "variables globales" que utilicemos en la escena como, por ejemplo, los colores, asumiendo que quienes intentarán reproducir el código **tendrán una instalación fresca de Manim**.
- Cuando en duda, preguntarse, ¿qué sería lo mejor para quienes más adelante quieran experimentar con el código?

8. Acuerdos de trabajo

- Cada sábado a las 12:00, subir los avances de las animaciones de cada escena a su respectivo hilo de Slack y el correspondiente *script* al repositorio, y actualizar los guiones en el repositorio.