




## Parametrizando ejercicios con Python y LaTeX: una novedosa estrategia para generación de materiales de enseñanza y evaluación en matemáticas

| Parameterizing exercises with Python and LaTeX: a novel strategy for generating teaching and assessment materials in mathematics |

 **Jorge Arroyo Hernández<sup>1</sup>**  
[jorge.arroyo.hernandez@una.ac.cr](mailto:jorge.arroyo.hernandez@una.ac.cr)  
Universidad Nacional  
Heredia, Costa Rica

 **Federico Mora Mora<sup>2</sup>**  
[federico.mora.mora@una.ac.cr](mailto:federico.mora.mora@una.ac.cr)  
Universidad Nacional  
Heredia, Costa Rica

 **Eithel Trigueros Rodríguez<sup>3</sup>**  
[eithel.trigueros.rodriguez@una.ac.cr](mailto:eithel.trigueros.rodriguez@una.ac.cr)  
Universidad Nacional  
Sarapiquí, Costa Rica

 **Karen Porras Lizano<sup>4</sup>**  
[karen.porras.lizano@una.ac.cr](mailto:karen.porras.lizano@una.ac.cr)  
Universidad Nacional  
Heredia, Costa Rica

Recibido: 1 noviembre de 2022

Aceptado: 30 setiembre 2023

**Resumen:** En este artículo se presenta una propuesta novedosa para su uso en docencia, que consiste en la construcción de ejercicios generados a partir de una estructura base con parámetros pseudoaleatorios, de manera que al sustituirlos con valores particulares en rangos predeterminados se obtengan múltiples versiones. La elaboración y digitalización de los ítems se apoya en el lenguaje de programación Python y sus bibliotecas Sympy y Numpy; los cuales permiten automatizar la generación de los ejercicios, sus soluciones y los archivos LaTeX. Estos últimos compilados de manera masiva. Entre los principales aportes de este desarrollo se destacan: los listados de ítems de dificultad similar personalizados con retroalimentación para cada estudiante y ahorro de tiempo en horas de docencia.

**Palabras Clave:** Python, LaTeX, Ejercicios parametrizados, Evaluación y enseñanza de las matemáticas.

<sup>1</sup>Jorge Arroyo Hernández. Académico-Universidad Nacional Campus Omar Dengo. Heredia, Costa Rica. Código Postal: 86-3000. Correo electrónico: [jorge.arroyo.hernandez@una.ac.cr](mailto:jorge.arroyo.hernandez@una.ac.cr)

<sup>2</sup>Federico Mora Mora. Académico-Universidad Nacional Campus Omar Dengo. Heredia, Costa Rica. Código Postal: 86-3000. Correo electrónico: [federico.mora.mora@una.ac.cr](mailto:federico.mora.mora@una.ac.cr)

<sup>3</sup>Eithel Eduardo Trigueros Rodríguez. Académico-Universidad Nacional Campus Sarapiquí. Heredia, Costa Rica. Código Postal: 41003. Correo electrónico: [eithel.trigueros.rodriguez@una.ac.cr](mailto:eithel.trigueros.rodriguez@una.ac.cr)

<sup>4</sup>Karen Porras Lizano. Académica-Universidad Nacional Campus Omar Dengo. Heredia, Costa Rica. Código Postal: 10110. Correo electrónico: [karen.porras.lizano@una.ac.cr](mailto:karen.porras.lizano@una.ac.cr)

**Abstract:** This article presents a novel approach to teaching by generating exercises from a structured base with pseudorandom parameters. By substituting values with individuals in predetermined ranges, multiple versions of exercises can be obtained. The elements are elaborated and digitized using the Python programming language and its Sympy and Numpy libraries, which automate the generation of exercises, their solutions, and the Latex files. These files can then be compiled massively. One of the main contributions of this development is the creation of personalized lists of elements of similar difficulty, each with feedback for individual students. This approach also saves time in teaching hours.

**Keywords:** Python, Latex, Parameterizing exercises, Evaluation and teaching of the Mathematics.

## 1. Introducción

---

El continuo avance de la tecnología repercute en todas las actividades del ser humano, entre ellas, la educación formal. Lo anterior provoca nuevos retos y desafíos para el personal académico en su quehacer diario. Por tanto, existe una constante pero imperante necesidad de actualización a su currículum que le permita integrar nuevos elementos a las tareas metodológicas que favorezcan y promuevan nuevas oportunidades de mejora en procesos de mediación y evaluación del conocimiento.

En este sentido, los sistemas informáticos proveen una excelente oportunidad. Esto lo reafirman los autores (Aguayo et al., 2021) quienes indican que la tecnología aplicada a la educación es de gran ayuda para que los estudiantes lleven a cabo un proceso de enseñanza y aprendizaje de una manera favorable, en particular, en el área de educación matemática.

En la actividad sustancial de esta disciplina, existen diversos tipos de ejercicios por medio de los cuales se puede ejemplificar un concepto o resultado. Por ejemplo, los ejercicios aritméticos o algebraicos que, por lo general son muy utilizados en la práctica docente, tanto para la enseñanza como en las evaluaciones.

Actualmente existen programas informáticos orientados a la solución de ejercicios matemáticos en línea como Wolfram o Symbolab. Estos son capaces de realizar el cálculo de sus resultados e inclusive desplegar procedimientos de su resolución (Jankvist & Misfeldt, 2015). Este aspecto dificulta la evaluación de los contenidos, debido a que los estudiantes en ocasiones recurren a estas plataformas tecnológicas para resolver las pruebas y compartir los resultados. Esto se agudiza en los casos en los que se aplican evaluaciones sumativas fuera del salón de clase.

Una estrategia de solución que viene a mitigar esta problemática es la evaluación individual a cada estudiante con diferentes ejercicios. Sin embargo, estas acciones presentan dos dificultades centrales: el tiempo docente de elaboración y validación de los ejercicios y, la homogenización en la dificultad de los mismos. Ambas son enormes desafíos por resolver.

Debido a lo anterior, en este trabajo se presenta una propuesta novedosa que abre una ventana de nuevas oportunidades de mejora en los procesos educativos, particularmente, en actividades evaluativas formativas y las sumativas. La propuesta consiste en la construcción de evaluaciones que incluyan ejercicios en diferentes versiones generados a partir de valores parametrizables. Estas evaluaciones se construyen mediante el uso lenguajes de programación con acceso a librerías de cálculo simbólico y de edición de texto.

El objetivo principal en este trabajo es dar a conocer una nueva forma de construcción de materiales interactivos compuestos con listados de ejercicios, donde cada uno de estos es generado con valores aleatorios sustituidos en variables auxiliares denominadas *parámetros*, provocando que de un mismo ejercicio se obtengan múltiples versiones con el mismo nivel de dificultad. Cabe resaltar que los valo-

res aleatorios son obtenidos de la librería *Numpy* que provienen de algoritmos para la generación de números pseudoaleatorios.

Esta propuesta se desarrolla con software de libre acceso, por lo que, puede ser explorada por cualquier persona con interés en el aprendizaje de estos entornos. Con esta formulación, se garantiza múltiples ventajas, entre las que se encuentran: la implementación en diversos sistemas operativos, las facilidades de actualización y, acceso a las comunidades virtuales y foros de información en línea (Díaz et al., 2005).

Además, este trabajo está dirigido a un público meta afín a la enseñanza de la matemática, tanto a nivel de educación secundaria como universitaria, en ambientes en los que la opción de la elaboración de prácticas con ejercicios parametrizables pueda ser aplicado. Por otro lado, en el desarrollo teórico y práctico del presente trabajo se utilizan ejercicios parametrizados aplicados como tareas evaluativas en el curso Cálculo Diferencial e Integral de la Universidad Nacional de Costa Rica.

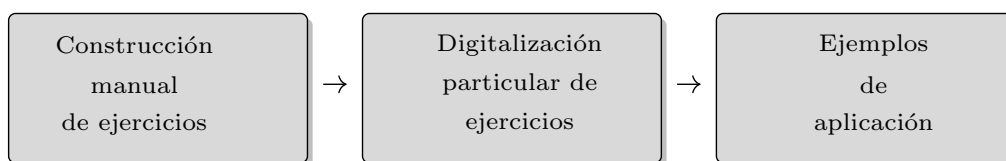
Se aclara al lector que las competencias deseables para el acceso correcto al conocimiento que se brinda en este documento, es el dominio básico de los lenguajes de edición de texto LaTeX y de programación Python; aunque no es imprescindible. Además, aunque el nivel de complejidad de los ejemplos presentados es básico, este se puede adaptar según las necesidades del lector.

El presente trabajo se estructura de la siguiente manera. En la sección 2.1, se realiza la descripción de la propuesta. En el apartado 2.2 se explica el proceso de digitalización, sintaxis y los procedimientos previos de parametrización. La sección 2.3 se muestra aplicaciones de esta propuesta y el proceso de compilación masiva de los archivos. Finalmente, se cierra este trabajo con la sección 3 que incluye las conclusiones y futuras líneas de investigación.

## 2. Elaboración de ejercicios

---

La elaboración de prácticas con ejercicios parametrizables conlleva un proceso que considera aspectos iniciales en la selección de ejercicios y su digitalización. En las secciones siguientes se abordan en detalle estos elementos. La figura 1 esquematiza el orden de estas secciones correspondientes a la secuencia de los procesos en la ejecución de la propuesta.



**Figura 1:** Diagrama de proceso de elaboración de prácticas parametrizadas. Elaboración propia.

### 2.1. Construcción manual de ejercicios

En la selección de los ejercicios a parametrizar es indispensable tomar en cuenta consideraciones generales básicas. En primera instancia, es necesario plantear un ejercicio con una estructura base definida y parámetros que puedan ser sustituidos por diferentes valores con el objetivo de conseguir casos particulares del mismo. El análisis de los parámetros y los valores a reemplazar es medular en este proceso pues con ello se evita problemas de indefiniciones y complejidad.

Por ejemplo, el siguiente es un límite en una variable con una estructura base apoyada del parámetro  $a$ :

$$\lim_{x \rightarrow a^2} \frac{x - a^2}{\sqrt{x} - a} = 2a. \quad (1)$$

En este se debe considerar las restricciones para  $a$  de modo que el límite esté siempre bien definido. En particular, el valor  $a$  al que tiende el límite debe ser diferente de 0 para darle dificultad al ejercicio y para asegurar que la solución se realice a través de racionalización o cambio de variable.

Otra consideración a tomar en cuenta es la asignación de valores aleatorios en los parámetros. Al sustituir el valor en un rango determinado se obtienen diferentes versiones numéricas. Sin embargo, por ejemplo, si el rango es seleccionado con valores enteros positivos "grandes" se generaría ejercicios con operaciones aritméticas muy complejas de realizar perdiendo parcialmente el objetivo de resolución del mismo.

También es importante considerar al número de parámetros a utilizar. Entre más parámetros se incorporen, mayor variabilidad de versiones del ejercicio se obtienen, pero a su vez, dichas combinaciones pueden provocar indefiniciones más complejas de analizar o expresiones no útiles en los ejercicios.

Por ejemplo, para una función definida con criterio:

$$f(x) = \frac{ax - a^2}{x^2 + (b - a)x - ab} \quad (2)$$

se utilizan dos parámetros  $a$  y  $b$ . Independientemente de lo que se desee preguntar, el análisis debe enfocarse en los valores asignados a los parámetros. Así por ejemplo, si  $a = 0$  y  $b \in \mathbb{R}$  entonces el criterio resultante de la función sería el nulo lo cual carecería de utilidad.

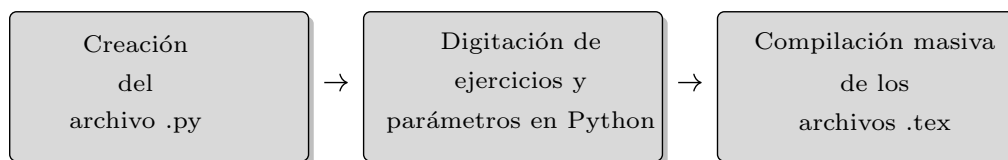
Otra consideración al utilizar varios parámetros es la forma de parametrización: una opción sería sustituir todos los parámetros con valores numéricos en un rango determinado, y la otra es cambiar únicamente un parámetro con valores numéricos, y mantener el otro como un valor fijo no numérico para obtener versiones diferentes y más complejas del ejercicio. En el primer caso la respuesta sería un valor numérico, y en el segundo caso, la respuesta quedaría en términos del parámetro fijo. Una alternativa posible para generar mayor diversidad en los ejercicios, es intercambiar los parámetros fijos y de asignación numérica. En este escrito, los ejemplos (1) y (2) serán tomados como base para desarrollar las siguientes secciones.

## 2.2. Digitalización de ejercicios

Después del análisis y la resolución manual de los ejercicios, se continua con el proceso de su digitalización. Para lo anterior, es menester cubrir las siguientes etapas:

1. Creación del archivo base generador y su posterior creación de los archivos .tex;
2. Transcripción a lenguaje de programación de los ejercicios algebraicos elegidos junto con la declaración de los parámetros.
3. Proceso de compilación masiva en archivos para generar los documentos en formato pdf.

Este orden de trabajo permite reutilizar los archivos iniciales en futuras asignaciones. En la figura 2 se muestra las etapas.



**Figura 2:** Proceso de creación de archivos con ejercicios parametrizados: el montaje del archivo generador y proceso de compilación masiva. Elaboración propia.

### 2.2.1. Creación del archivo .py

En este apartado se explica la creación del archivo genérico reutilizable básico para la construcción de prácticas y evaluaciones. Estos archivos son creados a partir de scripts de Python y apoyados de las librerías de código abierto *Sympy* (Lamy, 2013) y *Numpy* (Bressert, 2012). La librería *Sympy* es un motor de cálculo algebraico (CAS por sus siglas en inglés) que provee los resultados de operaciones que involucran expresiones algebraicas y vectoriales, expresadas en formato simbólico. En la tabla 1 se hace una descripción de las áreas temáticas en las que se potencia el uso de la librería *Sympy*. La *Numpy* se enfoca en estructuras de vectores y álgebra lineal.

El inicio es por medio de la invocación de las librerías *NumPy* y *SymPy*. Esto se realiza a través de la líneas de código:

```
import numpy as np
import sympy as sp
```

donde *sp* y *np* son los “etiquetas” para su llamado.

**Tabla 1:** Extracto de las áreas temáticas de aplicación de la librería *Sympy*.

Área	Descripción
Cálculo simbólico	Expresiones simbólicas
	Sustitución y evaluaciones en expresiones simbólicas
	Conversiones de strings a expresiones SymPy
	Conversión de expresiones simbólicas a formato LaTeX
Matemática básicas	Simplificación
	Operaciones algebraicas básicas
	Potenciación
	Logaritmos y exponenciales
	Conjuntos
	Sistemas de ecuaciones
	Simplificación
Matemáticas avanzadas	Derivadas
	Integrales
	Límites
	Series
	Diferencias finitas
	Matrices
	Ecuaciones diferenciales

La construcción de las líneas de código en el archivo Python son una mezcla de strings que contienen

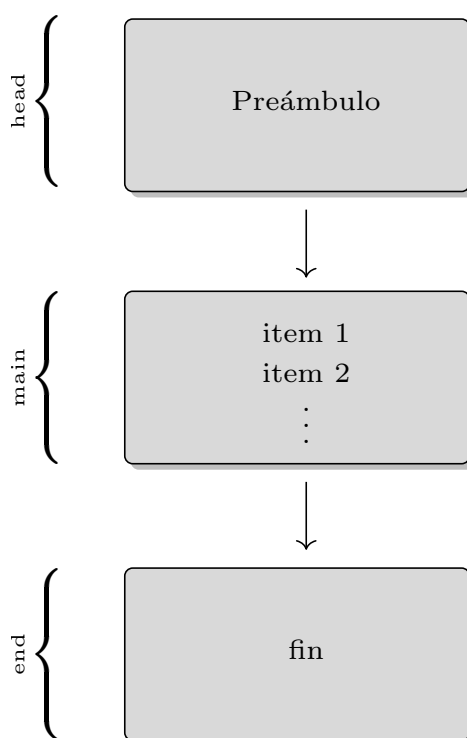
empotrados de código LaTeX. Estas se constituyen con partes que son resultado de los enunciados parametrizables y partes fijas (o constantes).

Los archivos `.tex` a generar, deben ser almacenados en una carpeta localizada en la misma ubicación del archivo generador. En este trabajo, la carpeta de guardado de los mismos se denomina la `tex_pre`. Lo anterior permite separar los archivos generados de los fuentes.

El preámbulo y el final del archivo `.tex` a generar se importan a partir de dos archivos de texto plano fijos denominados `head` y `end` respectivamente que contiene los paquetes necesarios para la compilación de los archivos LaTeX. Son agregados en bloques al inicio y final del script de Python. En la sección C del apéndice se muestra el código de ambos archivos. El siguiente código lo muestra:

```
file.write(str(head))
file.write(str(end))
```

En la figura 3 se exhibe los bloques de constitución del archivo de generación de ejercicios. Cabe resaltar que cada item contiene un ejercicio parametrizado contenidos en el bloque `main`. El listado total de los ejercicios provoca una lista personalizada para cada estudiante.



**Figura 3:** Diagrama de la constitución del archivo generador de ejercicios.

Luego, el nombre de cada uno de los archivos `.tex` generados debe ser distinto. Para ejecutar la acción anterior, se concatena un nombre fijo de raíz denominado `ejerc_` a un string convertido de la variable `i` proveniente de una iteración de un ciclo de ejecución, unido al final con la extensión del archivo. Se debe agregar los permisos de escritura sobre dichos archivos. Ambas acciones se resumen en el siguiente código:

```
namefile = "./tex_pre/ejerc_" + str(i) + ".tex"
file = open(namefile, 'w', encoding='utf-8')
```

Lo anterior genera el listado de los archivos `.tex`. Se les puede incluir los datos de cada persona estudiante si se desea personalizar.

## 2.3. Transcripción de ejercicios a lenguaje Python

En las siguientes secciones se presentan ejemplos de parametrizaciones tomando los ejercicios mencionados anteriormente. Para la elaboración de los archivos, primeramente se declara las variables y parámetros a utilizar. Esto se explica en el sección 2.3.1. Posteriormente, en la apartado 2.3.2 se desarrolla un ejemplo con un parámetro. Y en sección 2.3.3, se detalla el proceso a partir de la aleatorización de dos parámetros.

### 2.3.1. Declaración de variables y parámetros en el archivo .py

La declaración de variables y parámetros a utilizar se da después del análisis de las restricciones explicado en la sección 2.1 y la elaboración de un archivo generador con base a la figura 3. En este apartado, es importante tener en cuenta que existen dos tipos de variables: las simbólicas y las que se utilicen para sustituir valores numéricos (parámetros). La variable simbólica se declara al inicio del script .py y corresponde a la variable propia de los ejercicios algebraicos. En los ejemplos (1) y (2), la variable simbólica es  $x$ . En lenguaje Python, su declaración corresponde a:

```
x = sp.Symbol('x')
```

En el caso que la expresión requiera más variables simbólicas, por ejemplo  $x$ ,  $y$  y  $z$ , se declaran de la forma:

```
x, y, z = sp.symbols('x y z')
```

En el caso de los parámetros, la aleatorización de valores se guarda en una variable o un vector. En este trabajo por simpleza y comodidad, se elige los valores enteros en rangos almacenados en un vector. Sin embargo, se deja claro que existen formas alternativas para su declaración. Se hace énfasis que el rango numérico escogido debe hacer asequible el ejercicio y elegirse de manera que no presente inconsistencias en la sustitución de los valores en los parámetros. En la siguiente línea se ejemplifica la declaración del vector:

```
np.random.randint(p, q, n)
```

donde  $p$  y  $q$  son los límites del rango y  $n$  el tamaño del vector. En el ejemplo (1), el parámetro  $a$  un valor es un entero aleatorio seleccionado entre  $p = 1$  y  $q = 4$ , pues de lo contrario, la asignación entraría en valores inusuales. De igual manera, en el ejercicio (2) se debe cuidar que el parámetro  $a$  no sea cero por lo indicado en la sección 2.1. En ambos casos, los errores se evitan declarando los vectores adecuadamente según el análisis previo de restricciones.

### 2.3.2. Ejemplo con un parámetro

Luego de la creación del archivo generador, se lleva a cabo el proceso de digitalización. Este consiste en la transcripción del ejercicio escrito en papel a lenguaje Python. En esta sección se realiza dicho proceso utilizando el ejercicio (1), un parámetro  $a$  y la variable algebraica  $x$ .

Cada comando o función de SymPy recibe parámetros predeterminados necesarios para su funcionamiento; en el caso de este ejercicio, se utiliza el comando *limit* que recibe cuatro: la función real del límite, la variable simbólica, el valor hacia donde tiende el límite y, el tipo de evaluación: si es por izquierda o derecha, o ambos. La transcripción del ejercicio queda con la siguiente sintaxis:

```
a = np.random.randint(1, 4, 2)
exprFun = sp.Limit( ( x-a[0]**2 ) / ( sp.sqrt(x) - a[0] ), x,
    a[0]**2, '+-')
```



donde *expFun* es el nombre de la variable que almacena la cadena de *strings*.

Posteriormente, la salida almacenada en *exp* se concatena con texto fijo formado por una cadena de *strings* con el objetivo de imprimir el resultado así como su solución. Los comandos *latex* y *str* son necesarios pues convierten a código látex las líneas de *strings* de los resultados. La sintaxis del código resultante es:

```
file.write('\textit{Respuesta}: $' + str(sp.latex(exprFun)) + '$')
```

Se hace hincapié en la correcta y oportuna consulta de los parámetros de cada comando en referencias bibliográficas. En este ejercicio, si se desea la respuesta del ejercicio, se puede calcular con el comando *limit*. En la figura 4 se muestra el resultado del proceso.

$$\lim_{x \rightarrow 36} \left( \frac{x - 36}{\sqrt{x} - 6} \right)$$

(a)  $a = 6$

$$\lim_{x \rightarrow 25} \left( \frac{x - 25}{\sqrt{x} - 5} \right)$$

(b)  $a = 5$

$$\lim_{x \rightarrow 1} \left( \frac{x - 1}{\sqrt{x} - 1} \right)$$

(c)  $a = 1$

**Figura 4:** Salida de tres ejercicios de parametrización el dado en (1) usando distinto valores del parámetro  $a$ .

Cabe resaltar que los tres ejercicios de la figura 4, siendo todos diferentes entre sí, conllevan el mismo proceso de resolución, lo que garantiza los aspectos de equidad en la dificultad y complejidad, y además, en la cantidad de pasos de resolución para efectos de evaluación sumativa.

### 2.3.3. Ejemplo con dos parámetros

Otra alternativa que ofrece esta metodología es el uso de más de un parámetro. En esta sección se utiliza el ejercicio (2) y un tema clásico del curso de cálculo diferencial: análisis de la continuidad en un punto  $(x_0, f(x_0))$  de funciones reales de variable real, con la aleatorización de dos parámetros.

El objetivo de este tema se centra en analizar la continuidad de la función dado el criterio. Previamente, el proceso de análisis conlleva la factorización del denominador de la forma  $(x-a)(x+b)$ . para obtener su dominio  $\mathbb{R} - \{a, -b\}$ . A partir de este, se conoce los puntos de discontinuidad por se una función racional. Luego, se calculan los límites laterales para conocer a priori el tipo de discontinuidad en cada punto: la existencia de una evitable en  $x = a$  y de una no evitable  $x = -b$ .

Una elección adecuada de los parámetros reales son  $a$  y  $b$  tomados aleatoriamente como números enteros en los intervalos  $[3, 7]$  y  $[-2, 3]$  respectivamente se basa según lo analizado previamente y en la sección 2.1. En la figura siguiente se muestra la sintaxis del código de parametrización y el criterio de la función.

```
a = np.random.randint(3, 7, 1)
b = np.random.randint(-2, 3, 1)

# funcion f
fun = (a[0] * x - a[0]**2) / (x**2 + (b[0] - a[0])*x - a[0]*b[0])
file.write('Analice la continuidad de la función $f$ con criterio de
asociación:\n')
file.write('\begin{center} \n')
file.write('$\displaystyle f(x) := ' + str(sp.latex(fun)) + '$')
file.write('\end{center}\n')
```

La salida del código anterior usando dos iteraciones se muestra en la figura 5 que incluye la realimentación para el estudiante. Se deja claro que la especificación de la solución puede ser extendida



con mayor detalle. En las secciones A y B del apéndice se muestran los códigos de los ejemplos con la especificación de la solución como realimentación para los estudiantes.

Analice la continuidad de la función  $f$  dada por el criterio:

$$f(x) := \frac{-36 + 6x}{12 - 8x + x^2}$$

**Respuesta**

1. El dominio de la función es  $\mathbb{R} - \{6, 2\}$
2.  $f$  posee una discontinuidad evitable en  $x = 6$  pues:

$$\lim_{x \rightarrow 6} \left( \frac{-36 + 6x}{12 - 8x + x^2} \right) = \frac{3}{2}.$$

3.  $f$  posee una discontinuidad inevitable en  $x = 2$  pues:

$$\lim_{x \rightarrow 2^-} \left( \frac{-36 + 6x}{12 - 8x + x^2} \right) = -\infty \quad \text{y} \quad \lim_{x \rightarrow 2^+} \left( \frac{-36 + 6x}{12 - 8x + x^2} \right) = \infty.$$

(a) Los valores parametrizados son  $a = 6$  y  $b = -2$ .

Analice la continuidad de la función  $f$  dada por el criterio:

$$f(x) := \frac{-9 + 3x}{-3 - 2x + x^2}$$

**Respuesta**

1. El dominio de la función es  $\mathbb{R} - \{3, -1\}$
2.  $f$  posee una discontinuidad evitable en  $x = 3$  pues:

$$\lim_{x \rightarrow 3} \left( \frac{-9 + 3x}{-3 - 2x + x^2} \right) = \frac{3}{4}.$$

3.  $f$  posee una discontinuidad inevitable en  $x = -1$  pues:

$$\lim_{x \rightarrow -1^-} \left( \frac{-9 + 3x}{-3 - 2x + x^2} \right) = -\infty \quad \text{y} \quad \lim_{x \rightarrow -1^+} \left( \frac{-9 + 3x}{-3 - 2x + x^2} \right) = \infty.$$

(b) Los valores parametrizados son  $a = 3$  y  $b = 1$ .

**Figura 5:** Salida de dos ejercicios parametrizados para los valores de  $a$  y  $b$  de la función dada en (2).

Al igual que ocurre en el ejemplo de la sección 2.3.2, ambos ejercicios posee igual nivel de dificultad y es la misma la cantidad de pasos necesarios para obtener la respuesta correcta.

Una posible variación del ejercicio (2) parametrizando  $a$  y conservar el otro en forma simbólica  $b$ . Esta variación hace que el ejercicio incremente su nivel de complejidad pues el análisis del comportamiento  $b$  está en correspondencia del valor numérico de  $a$ ; por ejemplo, si  $a$  toma valores negativos o positivos el comportamiento de la continuidad varía. En la figura 6 se muestra dos iteraciones de esta variación.

Analice la continuidad de la función  $f$  dada por el criterio:

$$f(x) := \frac{-36 + 6x}{-6b - 6x + bx + x^2}$$

(a) Los valores parametrizados  $a = 3$  y  $b$  simbólico.

Analice la continuidad de la función  $f$  dada por el criterio:

$$f(x) := \frac{-16 + 4x}{-4b - 4x + bx + x^2}$$

(b) Los valores parametrizados  $a = 6$  y  $b$  simbólico.

**Figura 6:** Salida de dos ejercicios de parametrización el dado en (2) usando  $b$  de forma simbólico.

### 2.3.4. Compilado masivo de archivos

Los documentos de tarea o exámenes que son generados de formato **tex** se compilan en formato **pdf** en un proceso masivo usando el comando **pdflatex**. El proceso se puede ejecutar en diferentes sistemas operativos. Por ejemplo, en el ambiente **GNU-Linux** se emplea la siguiente línea de código en la terminal de comandos *shell*:

```
for i in *.tex; do pdflatex $i; done
```

o alternativamente, en caso que se desee borrar los archivos auxiliares, puede utilizarse la línea:

```
for i in *.tex; do pdflatex $i; done && rm *.log *.aux *.out
```

En ambientes asociados a sistema operativo Microsoft Windows, el compilado masivo se realiza mediante la ejecución de la línea de código en la *Ventana del símbolo del sistema*:

```
FOR %i IN (*.tex) DO pdflatex %i
```

o alternativamente, si se desea borrar los archivos auxiliares:

```
FOR %i IN (*.tex) DO pdflatex %i && DEL *.log *.aux *.out
```

## 3. Conclusiones

En este trabajo se propuso el uso de una nueva estrategia tecnológica en la educación matemática, brindando oportunidades de enriquecimiento de los procesos de enseñanza y aprendizaje. Específicamente, en la construcción de materiales con items parametrizables que sean utilizados en ambientes de aula y, en evaluaciones formativas y sumativas usando las herramientas tecnológicas LaTeX y Python.

En este sentido, el proceso se podría pensarse como complejo y tedioso debido al uso de estas tecnologías, sin embargo, una vez realizado se garantiza un producto semilla con cuantiosos beneficios en los procesos de evaluación en el desempeño estudiantil. En particular, brinda la oportunidad de la

generación de listados de ejercicios personalizados para cada estudiante, incluso, con sus respectivos procedimientos de sus resolución como medio de realimentación.

Respecto a la construcción de los ejercicios, la revisión exhaustiva que se obliga en todo el proceso permite la determinación correcta de las restricciones y a la vez se vuelve un proceso fundamental que evita inconsistencias en los resultados. Igualmente, un buen manejo de los parámetros puede dar un mejor aprovechamiento a las prácticas. Lo anterior garantiza la equidad en la dificultad, la variabilidad en las versiones de los ítems y disminuye los intentos de fraude por parte de los estudiantes.

Asimismo, una vez determinado el ejercicio con los parámetros la implementación en Python y la generación de los archivos .tex son procesos más “mecánicos”. Se puede adaptar la digitalización a otros software conocidos y tomar de referencia este trabajo como base para otras aplicaciones más complejas.

Por otra parte, es importante dejar claro que para la estrategia presentada en este artículo es necesario tener conocimiento matemático, pues se requiere de ingenio y creatividad para la elaboración de ítems adecuados. Se recomienda digitalizar cada ejercicio posterior a su resolución manual, en términos de los parámetros, para asegurar su solución general. También, es deseable contar con conocimientos básicos en programación para la digitalización y elaboración de los archivos.

Otro aspecto a resaltar es que en este escrito se presenta un desarrollo enfocado en documentos con formato pdf; sin embargo, los desarrollos pueden ser migrados a sistemas interactivos web como por ejemplo los Colab de Google.

En futuras líneas de investigación, este tipo de iniciativas podrían ser implementadas en la enseñanza de otras áreas afines de conocimiento, como por ejemplo álgebra lineal, lógica y teoría de conjuntos. Además, investigar la percepción de los estudiantes ante las prácticas/tareas/pruebas que se construyen de esta forma. Eso con el fin de mejorar el producto final. Esperamos lograr mejorar las experiencias tecnológicas que los profesores y estudiantes tengan.

## 4. Bibliografía

---

Aguayo, R., Lizarraga, C., & Quiñonez, Y. (2021). Evaluación del desempeño académico en entornos virtuales utilizando el modelo PNL. *RISTI: Revista Ibérica de Sistemas e Tecnologías de Información*, (41), 34-49. <https://doi.org/10.17013/risti>

Bressert, E. (2012). *SciPy and NumPy: an overview for developers* (1.<sup>a</sup> ed.). O'Reilly Media, Inc.

Díaz, F. J., Harari, V., & Banchoff Tzancoff, C. M. (2005). Ventajas del software libre en las escuelas. *I Jornadas de Educación en Informática y TICs en Argentina*, 55-59.

Jankvist, U. T., & Misfeldt, M. (2015). CAS-induced difficulties in learning mathematics? *For the Learning of Mathematics*, 35(1), 15-20.

Lamy, R. (2013). *Instant SymPy Starter* (1.<sup>a</sup> ed.). Packt Publishing Ltd.

## 5. Apéndice

---

Los ejemplos pueden ser descargados de [https://github.com/jorgearr23/python\\_latex](https://github.com/jorgearr23/python_latex)

O bien, clonados:

```
git branch -m main ejemplo
git fetch origin
git branch -u origin/ejemplo ejemplo
git remote set-head origin -a
```

## A. Código completo de ejercicio parametrizado dado en (1).

```
import sympy as sp
import numpy as np
import tools as t

head = t.read_file("admin.header")
tail = t.read_file("admin.tail")

sp.init_printing(use_unicode=True)

def ejer1(m):
    for i in range(0,m):

        #-- Creación del archivo .tex --

        # Nombre del archivo
        namefile = "./tex/ejerc1-" + str(i)+".tex"

        # -- Apertura del archivo --
        file = open(namefile, 'w')

        # -- Escritura del preámbulo en el archivo .tex --
        file.write(str(head))

        # -- Digitación del límite --

        # -- Vector aleatorio de números enteros --
        a = np.random.randint(1,20,10)

        # -- Declaración de la variable simbólica --
        x = sp.Symbol('x')

        # -- Uso del comando Limit para la escritura del límite --
        expr = sp.Limit( ( x-a[0]**2 ) / ( sp.sqrt(x) - a[0]), x,
            a[0]**2,'+-')

        # -- Escritura del límite en el archivo .tex --
        file.write('$ \displaystyle' + str(sp.latex(expr)) + '$ \\\n')

        # -- Respuesta al límite --

        # -- Uso del comando limit para el cálculo del resultado --
        resp = sp.limit( ( x-a[0]**2 ) / ( sp.sqrt(x) - a[0]), x,
            a[0]**2,'+-')
```

```

# -- Escritura del resultado límite en el archivo .tex --
file.write('\textit{Respuesta}: $' + str(sp.latex(resp)) + '$
\\\\ \n')

# -- Escritura final en el archivo .tex --
file.write(str(tail))

# -- Cierre del archivo .tex --
file.close()

# -- Generación de 10 ejercicios .tex --
m=10
ejer1(m)

# -- Ejecutar en la terminal para general los archivos .pdf: --

# GNU-Linux
# for i in *.tex; do pdflatex $i; done && rm *.aux *.log *.tex

# MS Windows
# FOR %i IN (*.tex) DO pdflatex %i && DEL *.log *.aux *.out

```

## B. Código completo de ejercicio parametrizado dado en (2).

---

```

import sympy as sp
import numpy as np
import tools as t

head = t.read_file("admin_header")
tail = t.read_file("admin_tail")

sp.init_printing(use_unicode=True)

def ejer2(m):

    for i in range(0,m):

        #-- Creación del archivo .tex --

        # Nombre del archivo
        namefile = "./tex/ejerc2-" + str(i)+".tex"

        # -- Apertura del archivo --
        file = open(namefile, 'w')

        # -- Escritura del preámbulo en el archivo .tex --
        file.write(str(head))

        # -- Declaración de los vectores pseudo aleatorios de números
        enteros --

```

```

a = np.random.randint(3,7,1)
b = np.random.randint(-2,3,1)

# -- Declaración de la variable simbólica --
x = sp.Symbol('x')

# Declaración de la función f
fun = (a[0] * x - a[0]**2) / (x**2 + (b[0] - a[0])*x - a[0]*b[0])

# Encabezados del ejercicio
file.write('Analice la continuidad de la función $$$ con criterio
de asociación:\n')
file.write('\begin{center} \n')
file.write('$\displaystyle f(x) := ' + str(sp.latex(fun)) + '$')
file.write('\end{center}\n')

# Respuesta al ejercicio
file.write('\begin{center}\n')
file.write('\textbf{Respuesta}\n')
file.write('\end{center}\n')

file.write('\begin{enumerate}\n')

# -- Dominio de la función --
file.write('\item El dominio de la función es $\{\rm I\!R\} - \{')
file.write(str(a[0]) + ', ' + str(-1*b[0]) + '\}\n')

# -- Cálculo de los límites --
simb = str(sp.latex(sp.Limit(fun, x, a[0], '+-')))
res = str(sp.latex(sp.limit(fun, x, a[0], '+-')))

# -- Respuesta para x=a --
file.write('\item $$$ posee una discontinuidad evitable en ')
file.write('$ x = ' + str(a[0]) + '$ pues: \n')
file.write('\begin{center}')
file.write('$ \displaystyle' + simb + ' $ = ' + '$ \displaystyle' +
res + '$.\n')
file.write('\end{center}\n')

# -- Respuesta para x=b --
simb = str(sp.latex(sp.Limit(fun, x, -1*b[0], '-')))
file.write('\item $$$ posee una discontinuidad inevitable en ')
file.write('$ x = ' + str(-1*b[0]) + '$ pues: \n')
file.write('\begin{center}')

res = str(sp.latex(sp.limit(fun, x, -1*b[0], '-')))
file.write('$ \displaystyle' + simb + ' $ = ' + '$ \displaystyle' +
res + '$ \n')

simb = str(sp.latex(sp.Limit(fun, x, -1*b[0], '+')))
res = str(sp.latex(sp.limit(fun, x, -1*b[0], '+')))

```

```

file.write(' y $ \displaystyle'+ simb + ' $ =' + '$ \displaystyle'
+ res + '$. \n' )
file.write('\end{center}')

file.write('\end{enumerate}')

# -- Escritura final en el archivo .tex --
file.write(str(tail))

# -- Cierre del archivo .tex --
file.close()

# -- Generación de 10 ejercicios .tex y correr el archivo --
m=10
ejer2(m)

# -- Ejecutar en la terminal para general los archivos .pdf: --

# GNU-Linux
# for i in *.tex; do pdflatex $i; done && rm *.aux *.log *.tex

# MS Windows
# FOR %i IN (*.tex) DO pdflatex %i && DEL *.log *.aux *.out

```

## C. Código de los archivos de texto plano head y tail.

---

### head

```

\documentclass[varwidth]{standalone}
\usepackage{amssymb}
\begin{document}

```

### tail

```

\vspace{0.2cm}
\end{document}

```