



UNC

Universidad  
Nacional  
de Córdoba

FAMAF

Facultad de Matemática,  
Astronomía, Física y  
Computación

# Proyecto: Modelo de Klausmeier

Claudio Armas

28 de noviembre de 2024

## Resumen

En el presente informe se plantea, desarrolla e implementa un simulador simple para el modelo de Klausmeier.

## 1. Introducción

La desertificación es la degradación de las tierras de zonas áridas y semiáridas resultante de diversos factores, tales como las variaciones climáticas y las actividades humanas. A finales de la década de 1960 comenzó aemerger como un problema relevante. Debido a la falta de lluvias, la región del Sahara en África occidental experimentó una grave sequía, causando serios daños a los ecosistemas locales, los medios de vida de las personas y la ecología global. En la actualidad, es un problema que relaciona la economía, la sociedad y la ecología. Por lo tanto, prevenir la desertificación se ha convertido en un tema central y prioritario en la investigación global, donde la vegetación y el agua desempeñan un papel crucial [1]. Aquí es donde entra el modelo de Klausmeier, el cual se centra en la dinámica de vegetación en áreas semidesérticas. El modelo se basa en la hipótesis de redistribución del agua, centrada en la idea de que el agua de lluvia en regiones áridas apenas se infiltra en el suelo. En su lugar, el agua corre principalmente cuesta abajo hacia una área de vegetación.

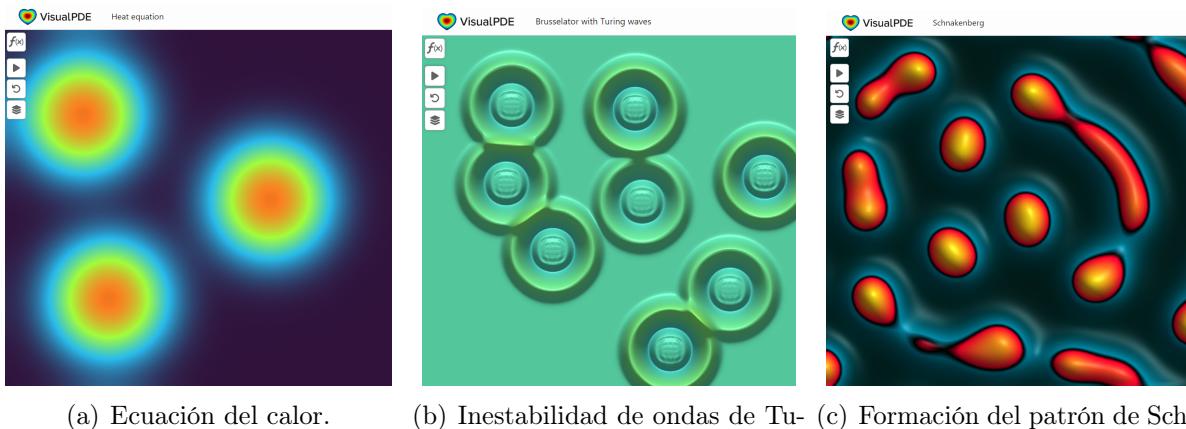
Muchos investigadores han intentado resolver el modelo de Klausmeier pues obtener una solución precisa o exacta no es tarea fácil ya que se trata de una ecuación diferencial parcial no lineal (EDPNL). Por el contrario, las soluciones numéricas son relativamente más fáciles de obtener. En este informe, se busca simular una aproximación de la solución del modelo de Klausmeier. Para este fin, se utilizará una herramienta llamada VisualPDE, una web que permite realizar simulaciones interactivas de ecuaciones diferenciales parciales (EDP) con respecto del tiempo en una y dos dimensiones espaciales [2]. Por otra parte, también se plantea y desarrolla una implementación del modelo de Klausmeier usando esquemas de diferencias finitas con el objetivo de replicar las simulaciones de VisualPDE.

## 2. VisualPDE y el simulador del modelo Klausmeier

En esta sección, se ofrecerá una breve introducción sobre qué es VisualPDE y cómo se puede aplicar esta herramienta al modelo de Klausmeier.

### 2.1. Breve introducción a VisualPDE

El acceso generalizado a la computación ha tenido un profundo impacto en la ciencia y en la sociedad en general. En particular, los modelos matemáticos relativamente avanzados se han vuelto ampliamente utilizados en numerosos campos de la ciencia. Esto a llevado a una alta demanda de herramientas en línea por parte de estudiantes, docentes e investigadores, y al desarrollo de entornos de aprendizaje más dinámicos e interactivos. Por ejemplo, existe una creciente literatura sobre el uso de software de visualización matemática basado en la web, aplicado a una variedad de conceptos matemáticos. Estas herramientas interactivas y accesibles permiten a los estudiantes explorar ideas de forma autónoma. Algunos ejemplos de estas herramientas pueden ser GeoGebra, Desmos, y WolframAlpha, entre otras. En particular, los creadores de VisualPDE buscaban crear una plataforma que abstraiga las complejidades de los métodos numéricos y su implementación, de modo que un estudiante tenga la libertad de explorar modelos a los que normalmente solo se puede acceder mediante simulación. VisualPDE está diseñado para ser un solucionador de EDP flexible y para ser ejecutado en un navegador web en el dispositivo de un usuario [2].



(a) Ecuación del calor. (b) Inestabilidad de ondas de Turing del Brusselator. (c) Formación del patrón de Schrödinger e hiperbólico de Brusselator. nakanberg.

Figura 1: Ilustración de tres modelos de EDP del sitio wed VisualPDE.

### 2.2. Probando VisualPDE para el modelo Klausmeier

El modelo de Klausmeier es un sistema de difusión-advección-reacción. La EDP es de la forma,

$$\begin{cases} \frac{\partial w}{\partial t} = a - w - wn^2 + v \frac{\partial w}{\partial x} + \nabla^2 w \\ \frac{\partial n}{\partial t} = wn^2 - mn + \nabla^2 n \end{cases} \quad (1)$$

donde  $w(x, y, t)$  es la densidad del agua,  $n(x, y, t)$  es la densidad de vegetación,  $t$  es el tiempo y  $(x, y)$  es el espacio 2D. Los parámetros  $a$ ,  $m$  y  $v$  representan la tasa de lluvia, la pérdida de vegetación y la advección descendente de  $w$ , respectivamente. Note que como es una ecuación

de difusión los coeficientes de difusión  $D_w$  y  $D_n$  valen uno para este modelo.

La Figura 2 muestra una simulación del modelo de Klausmeier (1). Observe que la vegetación genera relieves (como si fueran colinas) de color verde sobre la superficie desértica. En la parte superior a la izquierda se puede observar la EDP (1). Debajo de esta se encuentran distintas pestañas que nos permiten editar los distintos parámetros del modelo, como por ejemplo, las condiciones de borde o las condiciones iniciales. Entre las opciones que se pueden elegir se encuentran las condiciones periódicas, de Dirichlet o de Neumann.

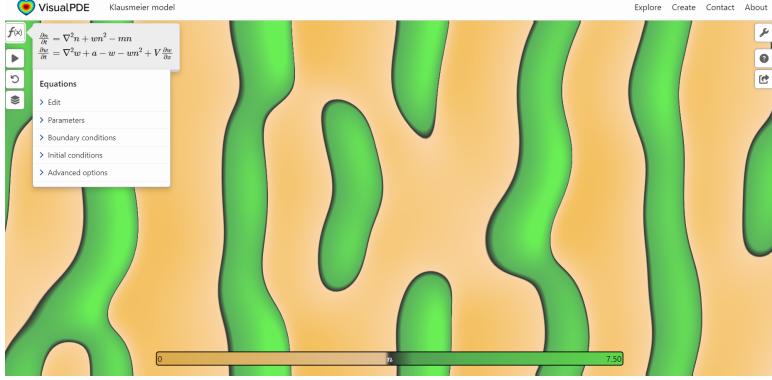


Figura 2: Captura pausada del simulador del sitio web visualPDE.

En las Figuras 3 y 4 se puede observar ajustes de los parámetros  $a$  y  $m$ . En la Figura 3(a) se ajusta la tasa de lluvia en un valor muy bajo y esto produce poca vegetación sobre la superficie desértica. En cambio en la 3(b), donde se selecciona un mayor valor del parámetro  $a$ , la vegetación crece rápidamente. Por otra parte, ajustar el parámetro  $m$  que es la pérdida de vegetación tiene un efecto contrario al anterior parámetro. Cuando menor sea  $m$  hay una menor perdida de vegetación y cuando mayor sea  $m$  mayor será la perdida de vegetación. Esto mismo se muestran en las Figuras 4(a) y 4(b).

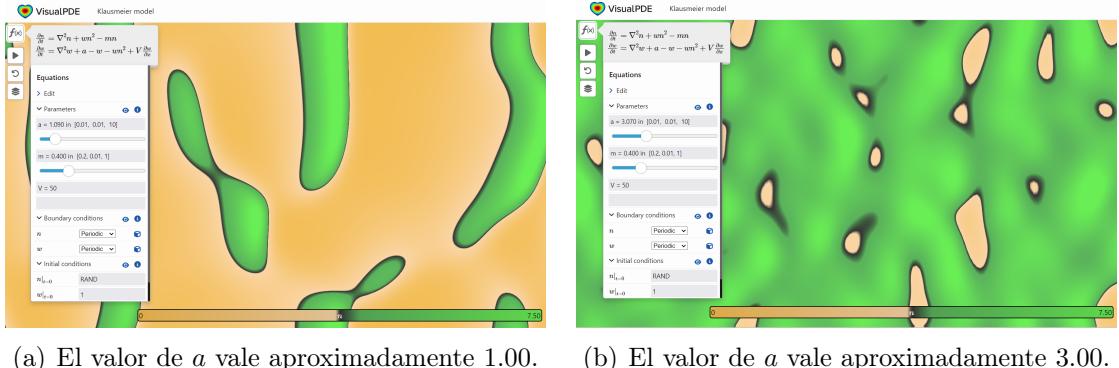
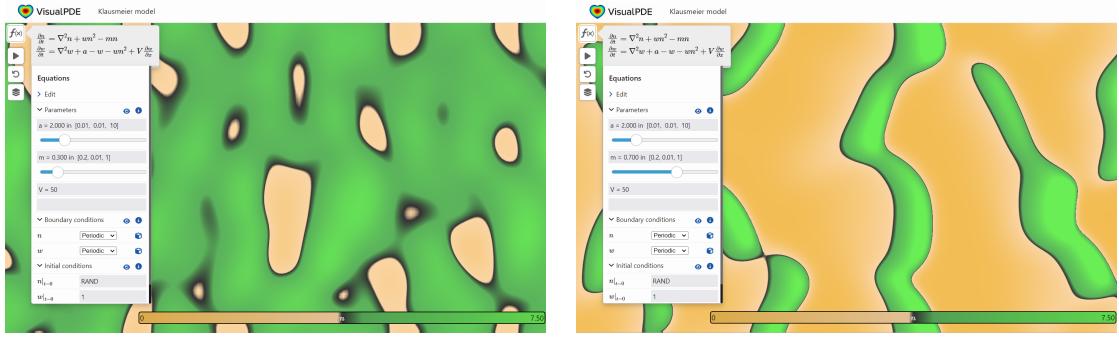


Figura 3: Distintos valores del parámetro  $a$  (tasa de lluvia).

Estos resultados no son los únicos que se pueden obtener de VisualPDE. Todos los parámetros y constantes que intervienen en el modelo de Klausmeier, se pueden ajustar o variar como uno lo deseé y obtener una infinidad de simulaciones.



(a) El valor de  $m$  vale aproximadamente 1.00.

(b) El valor de  $m$  vale aproximadamente 3.00.

Figura 4: Distintos valores del parámetro  $m$  (pérdida de vegetación).

### 3. Esquema del Modelo de Klausmeier

Para resolver ecuaciones diferenciales parciales, VisualPDE utiliza un esquema de diferencias finitas centradas para todas las derivadas espaciales, junto con discretizaciones no centradas con el fin de incorporar métodos de actualización [2]. En esta sección, se desarrolla el esquema para el modelo (1) que se utilizará para poder implementar un simulador del modelo de Klausmeier, con el objetivo de generar imágenes similares al de VisualPDE.

En un primer paso, se reordenar la ecuación (1) de la siguiente forma,

$$\begin{cases} \frac{\partial w}{\partial t} = \nabla^2 W + v \frac{\partial w}{\partial x} + a - w - wn^2 \\ \frac{\partial n}{\partial t} = \nabla^2 n + wn^2 - mn \end{cases} \quad (2)$$

Como se puede observar en (2) tenemos las derivadas parciales  $\partial w / \partial t$ ,  $\partial / \partial x$  y el operador  $\nabla^2$  para  $w$  y,  $\partial / \partial t$  y  $\nabla^2$  para  $n$ . La idea aquí es aplicar los esquemas de diferencias finitas para aproximar tanto a las derivadas parciales como al operador  $\nabla^2$ . Para ello, seguimos lo presentado en [2]. Pero primero es necesario definir algunos preliminares para el problema (2).

Sean un dominio cuadrado  $\Omega = [0, L] \times [0, L]$  (la cuadrícula),  $\Delta x$  y  $\Delta y$  pasos espaciales entre los nodos y, un entero  $R_{nd}$  para la cantidad de nodos. Para el paso del tiempo definimos un  $\Delta t$ . Dentro de la cuadrícula, cada nodo espacial se define como  $x_i = i\Delta x$  e  $y_j = j\Delta y$ , para  $i, j = 0, 1, 2, \dots, R_{nd}$ , y lo mismo para la variable de tiempo  $t_k = k\Delta t$  para  $k = 0, 1, 2, \dots, T$ , donde  $T$  es el tiempo máximo. Entonces los esquemas que aproximan la derivada primera de  $w$  con respecto a la variable espacial  $x$  y al tiempo  $t$  son,

$$\frac{\partial w}{\partial t}(t, x, y) \approx \frac{W_{i,j}^{k+1} - W_{i,j}^k}{\Delta t}, \quad \frac{\partial w}{\partial x}(t, x, y) \approx \frac{W_{i,j+1}^k - W_{i,j}^k}{\Delta x}, \quad (3)$$

y la derivada de  $n$  con respecto al tiempo  $t$  es,

$$\frac{\partial n}{\partial t}(t, x, y) \approx \frac{N_{i,j}^{k+1} - N_{i,j}^k}{\Delta t}. \quad (4)$$

Estos son los esquemas de diferencias adelantada. Para el operador  $\nabla^2$  con respecto a  $w$  es,

$$\nabla^2 w(t, x, y) \approx \frac{W_{i+1,j}^k - 2W_{i,j}^k + W_{i-1,j}^k}{\Delta x^2} + \frac{W_{i,j+1}^k - 2W_{i,j}^k + W_{i,j-1}^k}{\Delta y^2}, \quad (5)$$

y para el operador  $\nabla^2$  con respecto a  $n$  es,

$$\nabla^2 n(t, x, y) \approx \frac{N_{i+1,j}^k - 2N_{i,j}^k + N_{i-1,j}^k}{\Delta x^2} + \frac{N_{i,j+1}^k - 2N_{i,j}^k + N_{i,j-1}^k}{\Delta y^2}. \quad (6)$$

Estos son los esquemas de diferencias centrada. Utilizando (3), (4), (5) y (6) para aproximar (2), se obtiene el siguiente esquema para el modelo de Klausmeier,

$$\left\{ \begin{array}{l} \frac{W_{i,j}^{k+1} - W_{i,j}^k}{\Delta t} = \frac{W_{i+1,j}^k - 2W_{i,j}^k + W_{i-1,j}^k}{\Delta x^2} + \frac{W_{i,j+1}^k - 2W_{i,j}^k + W_{i,j-1}^k}{\Delta y^2} \\ \quad + v \frac{W_{i+1,j}^k - W_{i,j}^k}{\Delta x} + a - W_{i,j}^k - W_{i,j}^k N_{i,j}^k \\ \\ \frac{N_{i,j}^{k+1} - N_{i,j}^k}{\Delta t} = \frac{N_{i+1,j}^k - 2N_{i,j}^k + N_{i-1,j}^k}{\Delta x^2} + \frac{N_{i,j+1}^k - 2N_{i,j}^k + N_{i,j-1}^k}{\Delta y^2} \\ \quad + W_{i,j}^k (N_{i-1,j}^k)^2 - m N_{i,j}^k. \end{array} \right. \quad (7)$$

El sistema 7 se puede resolver utilizando uno de los cuatro esquemas de diferencias finitas explícitos de intervalo de tiempo fijo: método de Euler, Adams-Bashforth, el método de Euler modificado y el método de Runge-Kutta (RK4). En particular, en este trabajo se emplea el método de Euler, entonces el sistema 7 queda de la siguiente forma,

$$\left\{ \begin{array}{l} W_{i,j}^{k+1} = W_{i,j}^k + \Delta t \left( \frac{W_{i+1,j}^k - 2W_{i,j}^k + W_{i-1,j}^k}{\Delta x^2} + \frac{W_{i,j+1}^k - 2W_{i,j}^k + W_{i,j-1}^k}{\Delta y^2} \right) \\ \quad + \Delta t \left( v \frac{W_{i+1,j}^k - W_{i,j}^k}{\Delta x} + a - W_{i,j}^k - W_{i,j}^k N_{i,j}^k \right) \\ \\ N_{i,j}^{k+1} = N_{i,j}^k + \Delta t \left( \frac{N_{i+1,j}^k - 2N_{i,j}^k + N_{i-1,j}^k}{\Delta x^2} + \frac{N_{i,j+1}^k - 2N_{i,j}^k + N_{i,j-1}^k}{\Delta y^2} \right) \\ \quad + \Delta t (W_{i,j}^k (N_{i-1,j}^k)^2 - m N_{i,j}^k) \end{array} \right. \quad (8)$$

El metodo de RK4 otorga la mayor estabilidad y precisión de orden. Sin embargo, posee el mayor costo computacional. Mientras que el método de Euler representa la opción computacionalmente más simple y con un menor costo [2].

## 4. Resultados Numéricos

En esta sección, presentamos los resultados de implementar el esquema (8). El principal objetivo es generar simulaciones similares a las de visualPDE. Para tal efecto, se utilizó Python.

Se implementó un simulador llamado `simulador_klausmeier` donde las entradas son los parámetros  $a$ ,  $m$   $v$  y la salida son 16 frames para tiempos distintos con un máximo de 50s. Dentro del simulador se ejecuta el método de Euler de forma iterativa, es decir, se va actualizando a medida que avanza el tiempo  $t$ . Se define un `step_plot` que va a permitir plotear los frames de las aproximaciones luego de un numero finito de iteraciones.

Los resultados de aplicar de aplicar el `simulador_klausmeier` se puede observar en la figura 5. Como se puede observar las imágenes son similares a las que presentaron en la sección (2.2). Son 16 frames que muestran la evolución del sistema 1 con un tiempo máximo de 50s. Con el método de Euler se alcanza a capturar la naturaleza de la solución del modelo de klausmeier, tiene sus desventajas con respecto al tiempo, pero nos permite una implementación simple y un menor costo computacional.

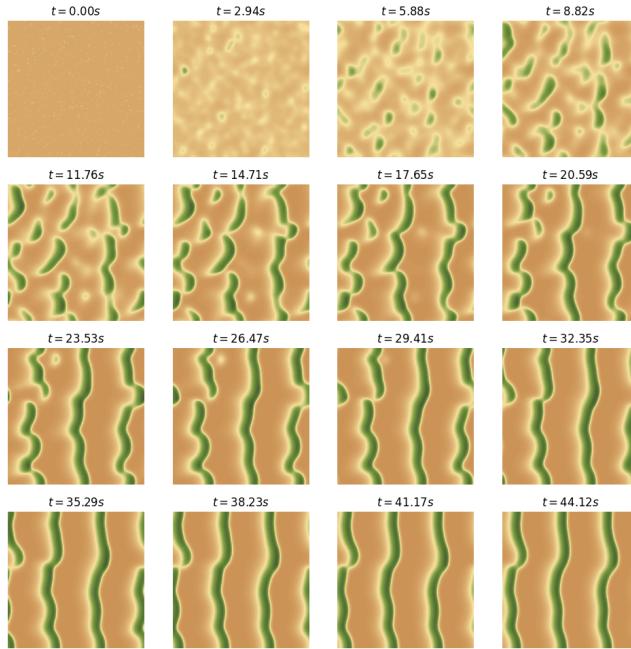


Figura 5: Una muestra del algoritmo ejecutado para un tiempo maximo 50s.

## 5. Conclusiones

No siempre es necesario recurrir a métodos numéricos que ofrezcan una solución extremadamente precisa; en muchos casos, basta con capturar la esencia del modelo. VisualPDE entiende esta idea y se enfoca en la simplicidad, permitiendo que tanto estudiantes como investigadores puedan aprender e implementar modelos de EDP de forma práctica y accesible.

En lo personal, haber conocido y explorado VisualPDE me sirvió un montón para implementar el simulador. Este tipo de herramienta son necesarias hoy en la actualidad para estudiantes o investigadores que no tienen un estudio profundo en métodos numéricos.

## A. Simulador de Klausmeier

```

1 def simulador_klausmeier(a, m, v):
2
3     num_nodos = 100 # Cantidad de nodos para x e y (malla en 2D)
4     long_xy = 100.0 # Longitud del dominio (cuadrado)
5     delta_x = long_xy/num_nodos # paso espacial
6     delta_y = long_xy/num_nodos # paso espacial
7
8     # Condiciones iniciales
9     W_0 = np.zeros((num_nodos, num_nodos))+1.0
10    N_0 = (np.random.rand(num_nodos, num_nodos)<0.05)*1.0 + 1.0
11
12    time_max = 50.0 # Tiempo maximo
13    delta_t = 0.001 # paso de tiempo
14    num_plot = 17 # numeros de plots a graficar
15
16    # Generamos los frame para plotear las simulaciones
17    fig, axes = plt.subplots(4, 4, figsize=(12, 12))
18

```

```

19 # Numero maximo de iteraciones
20 iter_max = int(time_max / delta_t)
21
22 # Vamos a plotear algunos frame entonces definimos un paso de plot
23 step_plot = iter_max // num_plot
24
25 # Empezamos a iterar
26 for idx in range(int(time_max / delta_t)):
27
28     # Metodo de Euler
29     W = W_0 + delta_t * ( a - W_0 - W_0*N_0*N_0 + v*Dx(W_0, delta_x)
30                         + dw*DDxy(W_0, delta_x, delta_y))
31     N = N_0 + delta_t * ( W_0*N_0*N_0 - m*N_0 + dn * DDxy(N_0, delta_x, delta_y))
32
33     # Rescribimos para la siguiente iteracion
34     W_0 = W
35     N_0 = N
36     # Si se satisface las condiciones ploteamos
37     if idx % step_plot == 0 and idx < ((num_plot-1) * step_plot):
38         ax = axes.flat[idx // step_plot]
39         ax.imshow(N, cmap=VegMap, interpolation='bilinear', extent=[0, 1, 0, 1])
40         ax.set_axis_off()
41         ax.set_title(f'$t={idx * delta_t:.2f}$')

```

Para ver del código completo entre al siguiente enlace: [simulador klausmeier](#)

## Referencias

- [1] Lv, Jianping and Li, Chunguang and Dong, Jianqiang. *A high accuracy compact difference scheme and numerical simulation for a type of diffusive plant-water model in an arid flat environment.* AIMS Mathematics 2024.
- [2] Walker, Benjamin J and Townsend, Adam K and Chudasama, Alexander K and Krause, Andrew L. *VisualPDE: rapid interactive simulations of partial differential equations.* Bulletin of Mathematical Biology. Springer 2023.