Artículo Original / Original Research

Ciencias Matemáticas, Vol. 36, No. Único, 2022-2023, Pag. 35-42
Recibido (*Received*): 25-04-2024, Revisado (*Reviewed*): 22-05-2024
Aceptado (*Accepted*): 27-06-2024, En línea (*Online*): 19-07-2024

Crystal: software para modelos epidemiológicos definidos por ecuaciones diferenciales ordinarias

Crystal: A software for epidemiological models defined by ordinary differential equations

Daniela Rodríguez Cepero [1], Aymée de los Ángeles Marrero Severo[2]*, Wilfredo Morales Lezca[3]

Resumen El objetivo fundamental de este trabajo es presentar el diseño e implementación de una herramienta computacional para la resolución, análisis y graficado de los modelos epidemiológicos, tipo poblacionales. El programa permite resolver modelos matemáticos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias aplicados a las ciencias de la vida. Para el desarrollo del mismo se diseñó un lenguaje de dominio específico, que permite al usuario introducir una amplia clase de modelos matemáticos poblacionales. Dichos modelos se solucionan por métodos numéricos para sistemas de ecuaciones diferenciales. Esta herramienta además, permite el graficado y análisis de los resultados. Para la implementación y desarrollo de esta herramienta se eligieron las tecnologías: *Python*, *FastApi*, *TypeScript* y *Angular*.

Palabras Clave: ecuaciones diferenciales ordinarias, epidemiología, lenguaje de dominio específico, modelos matemáticos poblacionales.

Abstract *The fundamental objective of this paper is to show the design and implementation of the computational tool to for the resolution, analysis and plotting of epidemiological models, population type. This software allows solving population mathematical models defined by systems of ordinary differential equations applied to the life sciences. For its development, a domain-specific language was designed, which allows the user to introduce a wide class of population mathematical models. These models are solved by numerical methods for systems of differential equations. In addition, this tool will allow the plotting and analysis of the results. For the implementation and development of this tool were used the technologies: Python, FastApi, Type-Script y Angular.*

Keywords: *ordinary differential equations, epidemiology, domain specific language, population mathematical models.*

Mathematics Subject Classification: 34-04, 90-04, 92-08.

[1]Departamento Macro y Micro Economía, Facultad de Economía, Universidad de La Habana, Cuba. Email: daroce00@gmail.com
[2]Departamento Matemática Aplicada, Facultad de Matemática y Computación, Universidad de La Habana, Cuba. Email: aymee@matcom.uh.cu, aymeema@gmail.com
[3] Facultad de Matemática y Computación, Universidad de La Habana, Cuba. Email: wilfre@matcom.uh.cu
*__Autor para Correspondencia (*Corresponding Author*)__

Editado y maquetado por (*Edited and layout by*): Damian Valdés Santiago, Facultad de Matemática y Computación, Universidad de La Habana, Cuba.

Citar como: Rodríguez Cepero, D., Marrero Severo, A.A.,& Morales Lezca, W. (2024). CRYSTAL: software para modelos epidemiológicos definidos por ecuaciones diferenciales ordinarias. *Ciencias Matemáticas*, 36(Único), 35–42. DOI: https://doi.org/10.5281/zenodo.13916396. Recuperado a partir de https://revistas.uh.cu/rcm/article/view/9057.

Introducción

Los modelos matemáticos son herramientas poderosas para comprender y predecir la propagación de enfermedades infecciosas en una población. Estos modelos, que pueden expresarse usando ecuaciones diferenciales, ecuaciones en derivadas parciales o modelos probabilísticos y estocásticos, entre otras técnicas, permiten a los investigadores simular diferentes escenarios y tomar decisiones informadas [2, 3, 8]. La modelación mediante ecuaciones diferenciales ordinarias (EDO) es especialmente útil en la investigación epidemiológica. Contar con herramientas computacionales que permitan

analizar la realidad es crucial en un mundo enfrentando constantes epidemias. El Grupo de Modelación Biomatemática de la Universidad de La Habana ha obtenido importantes resultados en el trabajo con modelos epidemiológicos aplicados a diferentes enfermedades, destacándose los obtenidos durante la pandemia de la COVID-19 [1, 4, 5, 7].

El objetivo principal es presentar el diseño e implementación de la herramienta computacional nombrada *Crystal* (por decisión de los autores con vistas a establecer una marca), que contiene una interfaz visual amigable con el usuario para la resolución, análisis y graficación de modelos matemáticos poblacionales definidos por sistemas de EDO aplicados a las ciencias de la vida, específicamente a la epidemiología.

Como parte de la investigación se realizó un estudio sobre las aplicaciones existentes que tratan la resolución de modelos epidemiológicos tanto dentro como fuera del país. Los resultados fueron escasos, se encontró un número pequeño de programas relacionados con esta problemática, la mayoría no son para modelos epidemiológicos y en general, no resuelven numéricamente cualquier modelo, sino que están orientados a algunos modelos en específico. Otros solo permiten el análisis de algunos modelos mediante simulaciones. Además, la mayoría son de pago u obligan a la actualización.

La problemática que resuelve este trabajo es la inexistencia de un programa cubano con interfaz amigable, que permita el tratamiento de los problemas inherentes a la modelación biomatemática, la solución numérica de los modelos, el graficado de resultados, la estimación de los principales parámetros que caracterizan la dinámica de transmisión, el estudio de sensibilidad, el diseño de experimentos, entre otros muchos aspectos. Este resultado brinda, por tanto, soberanía tecnológica.

Para la confección de la aplicación se siguieron los pasos que se muestran a continuación:

- Estudiar el estado del arte sobre los principales programas que resuelven el problema planteado y su accesibilidad al entorno cubano.
- Desarrollar un Lenguaje de Dominio Específico (DSL, por sus siglas en inglés), que permita introducir una amplia clase de modelos matemáticos poblacionales.
- Implementar la funcionalidad de resolver numéricamente modelos matemáticos poblacionales definidos por sistemas de EDO.
- Garantizar el graficado y almacenamiento de los resultados, obtenidos al resolver los modelos definidos por el usuario.
- Diseñar e implementar una aplicación web que contenga una interfaz de usuario agradable.
- Aplicar pruebas de funcionalidad para validar el correcto funcionamiento del programa.

Relevancia del estudio

Esta investigación se corresponde con la necesidad del Grupo de Biomatemática de la Facultad de Matemática y Computación de la UH que trabaja la modelación, solución y análisis de problemas aplicados a las ciencias de la vida.

¿Por qué es importante y necesario contar con un software que automatice la labor de estos investigadores, adecuado al contexto cubano?

La modelación matemática aplicada a la dinámica de transmisión de enfermedades exige el manejo de un volumen importante de datos y resultados, con vistas a su análisis y utilización en la predicción y control de las mismas.

Contar con una herramienta computacional con una interfaz visual amigable con el usuario para la resolución, análisis y graficado de cualquier modelo epidemiológico poblacional definido por EDO amplía las posibilidades del manejo de enfermedades para los decisores de Salud Pública en Cuba.

1. Contenido

El marco teórico-computacional de la solución propuesta, permite abordar las funciones que pueden ejecutar los usuarios y la estructura del programa. La arquitectura, el patrón del diseño y de la visualización del mismo, así como las tecnologías utilizadas, se explican a continuación.

1.1 Diagrama de casos de uso

A continuación, se presentan las diferentes acciones que puede ejecutar un usuario mediante un diagrama de casos de uso[1].

En la aplicación que se presenta y que se ha denominado *Crystal*, solo existe un rol de usuario "Usuario Anónimo", este es un usuario que puede acceder a la aplicación sin introducir ningún dato previo al sistema, teniendo disponibles todas las funcionalidades de la misma.

Como se muestra en la Figura 1, todo usuario de *Crystal* puede:

- Crear un modelo matemático.
- Resolver numéricamente un modelo epidemiológico previamente definido, teniendo la posibilidad, además, de escoger diferentes métodos de solución.
- Obtener los resultados en los diferentes instantes de tiempo.
- Graficar los resultados de un modelo epidemiológico previamente definido y visualizar el mismo.

[1] Un diagrama de casos de uso es una representación gráfica de los requisitos funcionales de un sistema, los actores que se comunican con el sistema y las relaciones entre ellos. Este diagrama es parte del Lenguaje Unificado de Modelado (UML, por sus siglas en inglés). Se utiliza para describir las interacciones entre los usuarios y un sistema.

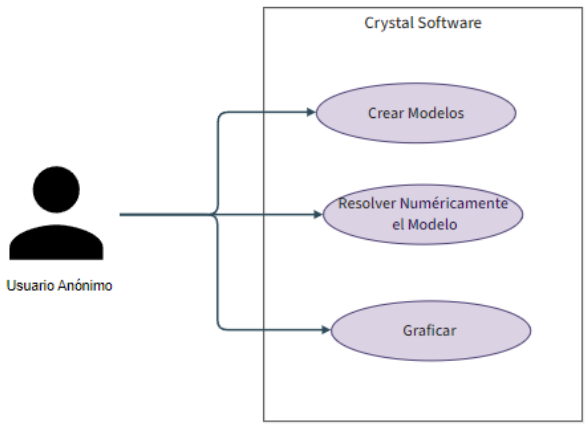

Figura 1. Diagrama de casos de uso de *Crystal* [*The use case diagram for Crystal*].

1.2 Arquitectura de *Crystal*

La arquitectura en capas es un modelo de diseño de software, cuya base es la separación de las diferentes funcionalidades del sistema en capas o niveles. Cada capa se encarga de un conjunto de tareas específicas y se comunica con las capas adyacentes mediante interfaces bien definidas (Code).

El programa *Crystal* implementa una arquitectura de tres capas. En la Figura 2 se puede observar la separación de las capas y la dependencia entre ellas.

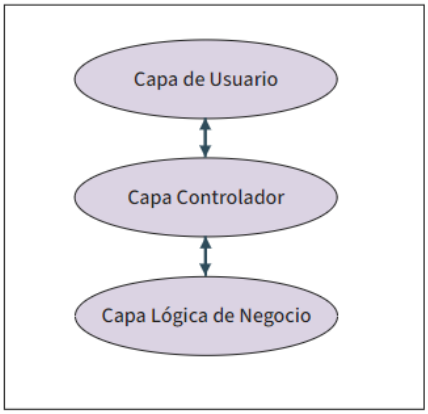

Figura 2. Arquitectura de *Crystal* [*Crystal Architecture*].

La capa Lógica de Negocio es la capa base. No tiene dependencia externa con otras capas. Incluye el análisis sintáctico y lexicográfico del DSL para poder interpretar lo introducido por el usuario en la interfaz visual y ejecutar a partir de ahí acciones de negocio, como la resolución de un modelo.

Para la implementación de esta capa se utilizó el lenguaje *Python*, lenguaje de propósito general que se puede utilizar en diversos campos y es ideal para el desarrollo de programas y admite programación estructurada, funcional y orientada a objetos. Se caracteriza por tener una sintaxis clara y concisa, lo que lo convierte en un lenguaje relativamente fácil de utilizar.

La capa Controlador depende de la capa lógica de negocio. Es visto como un controlador pues recibe una solicitud y da una respuesta, no es más que la interfaz de comunicación donde se conectan el usuario (la persona) con el negocio (la capa de lógica de negocio).

En esta capa se utilizó *FastApi*[2], ambiente web moderno, rápido (de alto rendimiento) para crear API con *Python* 3.8+, basado en sugerencias de tipo estándar de *Python*. Además, no requiere de un servicio de gestión de bases de datos y no cuenta con una arquitectura de capas compleja, brinda gran soporte en los editores con auto completado en todas partes, gasta menos tiempo en el proceso de depuración de errores y minimiza la duplicación de código debido a las múltiples funcionalidades con cada declaración de parámetros.

La capa Usuario depende de las capas anteriores. Contiene la implementación de la interfaz visual y por tanto posee los componentes y servicios encargados de enviar peticiones al *backend* y recibir la respuesta a dichas peticiones.

En esta capa se utilizó el lenguaje de programación *TypeScript* con el ambiente *Angular*, debido a que la arquitectura de *Angular*[3] enlaza *TypeScript*[4] y HTML, el código de ambos ya está sincronizado, ahorrando mucho tiempo a los desarrolladores, admite pruebas unitarias y de integración, permite a los usuarios manipular elementos de una página web sin necesidad de secuencias de comandos complejas. Además, mejora la visualización de páginas con gran cantidad de datos, genera código altamente optimizado, lo que garantiza una carga rápida de aplicaciones y ofrece una división de código automática para cargar solo lo necesario.

1.3 El lenguaje de *Crystal*

Para el desarrollo del lenguaje se implementó un DSL dinámico, cuya sintaxis es parecida a la de *Python*. El lenguaje permite crear modelos epidemiológicos, resolverlos y graficar su resultado; además de la definición de variables y procedmientos como operaciones aritméticas.

1.3.1 Programa y comandos

Para la ejecución de un programa se lee línea a línea de código y cada línea está compuesta por una serie de comandos o declaraciones que indican pasos o acciones a ejecutar:

```
<program> -> <stat-list>
<stat-list> -> <stat>?|
<stat><stat-list>;
```

Algunas de estas acciones pueden ser la declaración de una variable o un modelo, la llamada a ejecución de una función determinada o una simple instrucción de mostrar un resultado:

```
<stat> -> <def-var> |
```

[2] https://fastapi.tiangolo.com/es
[3] https://angular.io
[4] https://www.unir.net/ingenieria/revista/quees-typescript

```
<def-model>|
<func-call>|
<print-stat>|
<def-return>|
<solve-model>|
<def-plot>
```

Ejemplo:

```
a = 5;
print a;
```

En el ejemplo, aparecen dos declaraciones, una que se encarga de definir una variable con valor cinco y otra que se encarga de mostrar en la salida el valor de la variable antes definida.

1.3.2 Declaraciones

Cada declaración tiene su propia estructura que la define. Para mostrar un resultado en específico se utiliza el comando:

```
<print-stat> -> print <expr>
```

Ejemplo:

```
print 3;
```

Esta línea se encarga de mostrar en la salida el número tres en el momento exacto que se procesa la línea.

Para la declaración de los parámetros y variables del sistema se utiliza:

```
<def-var> -> <id> = <expr>
```

Ejemplo:

```
beta = 0.42;
betan = 0 - beta;
```

Esta declaración se encarga de definir las variables `beta` y `betan` con un valor de `0.42` y `-0.42`, respectivamente.

Para la definición de los modelos se utilizan las declaraciones `<def-model>` y `<def-return>`.

Esta última se utiliza con el objetivo de obtener los valores de las variables que caracterizan el sistema de ecuaciones diferenciales, una vez ejecutado el procedimiento que las define:

```
<def-model> -> model<id>
(<arg-list>){<stat-list>}
<def-return> -> return<expr-list>
```

Ejemplo:
El sistema de EDO que define el siguiente modelo:

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI \end{aligned} .$$

Se introduce en el programa según la siguiente secuencia:

```
model name_model (s,i)
{ ds = betan* s * i;
di = beta * s * i;
return ds,di;}
```

El ejemplo anterior muestra la definición del más simple de los modelos tipo SI (considera subpoblaciones de Susceptibles e Infectados) en el lenguaje de *Crystal*.

Si se desea resolver numéricamente un modelo utilizamos el comando:

```
<solve-model> -> <id>solve<expr-list>
```

Ejemplo:

```
name_model.solve('RK45',to,tf,so,io);
```

Esta línea se encarga de resolver numéricamente el modelo llamado `name_model` definido anteriormente.

Es importante resaltar que solo se pueden usar variables que estén previamente definidas. En este modelo `to, tf, so` e `io`, deben estar definidas con anterioridad para poder ser usadas.

También es necesario aclarar que no es obligatorio declarar variables, podrían introducirse los valores de las mismas directamente, pero sí es importante no violar el orden de los argumentos necesarios en esta declaración.

El primer argumento define el método que se usa para la resolución numérica, aprovechando los métodos numéricos que ofrece la librería `scipy.integrate.solve_ivp` de *Python*. A continuación, el tiempo inicial, seguido por el tiempo final y finalmente, los valores iniciales de las variables, en el mismo orden en que se definen en el modelo que se desea resolver.

Ejemplo:

```
name_model.solve('RK45',0,100,10,5);
```

Si se desea graficar los resultados obtenidos al resolver numéricamente el modelo, se utiliza el comando:

```
<def-plot> -> plot(<expr-list>)
```


Rodríguez Cepero, D., Marrero Severo, A.A.,& Morales Lezca, W.
https://doi.org/10.5281/zenodo.13916396

Ejemplo:

```
result = name_model.solve('RK45',to,
tf,so,io);
plot('name_model',result,to,tf,
'S','I');
```

El comando `plot` se encarga de graficar los resultados y guardar en formato de imagen, el modelo.

El primer parámetro representa el nombre con el que se desea guardar la imagen del modelo, el que le sigue, es la denominación del método numérico elegido para resolver el mismo, los dos siguientes definen el período de tiempo, es decir, tiempo inicial y final (`to, tf`), respectivamente, y para finalizar en el mismo orden que se definen las variables del modelo, los símbolos que aparecerán en la leyenda para identificar cada una de las variables que se grafican.

Las siguientes declaraciones se utilizan para representar una serie de parámetros o argumentos que requiere el modelo:

```
<arg-list> -> <id>|<id>,<arg-list>
<expr-list> -> <expr>|<expr>,
   <expr-list>
```

Haciendo uso de la forma normal de Backus (BNF, por sus siglas en inglés), se describen a continuación las expresiones, términos, factores, átomos y definiciones básicas del lenguaje [6].

1.3.3 Expresiones

```
<op> -> +|-|< =|> =|=|<|>|and|or
<expr> -> <solve-model>|
<term>|<expr>
<op><term>
```

1.3.4 Términos

```
<term> -> <factor>|
<term> + <factor>|
<term> - <factor>|
```

1.3.5 Factores

```
<factor> -> <atom>|
<expre>
```

1.3.6 Átomos

```
<atom> -> <string>|
<boolean>| <num>|<id>
```

1.3.7 Definiciones Básicas

```
<num>:[0-9]
<string>:'...'
<boolean>:True|False
<id>:[A-Za-z]
```

2. Pruebas de funcionalidad y resultados

Para demostrar el correcto funcionamiento de la herramienta computacional propuesta, se diseñaron un conjunto de pruebas de correctitud y así satisfacer el cumplimiento de los objetivos planteados. Las mismas consistieron en procesar diferentes modelos matemáticos poblacionales de diversa complejidad, utilizar diferentes métodos de solución numérica y verificar la visualización de las variables que intervienen en los mismos, además de registrar los resultados de cada una de ellas, en cada instante de tiempo.

Las pruebas se realizaron en una laptop con las siguientes características:

- Sistema operativo: Windows 10 Home, 64 bits.
- Procesador: AMD A12-9720P RADEON R7, 12 COMPUTE CORES (4 CPUs), 2.7 GHz.
- Memoria RAM: 16 GB.
- Disco Duro: 1 TB SSD.

Al ejecutar la aplicación web, la primera vista que parece es la página de inicio. La Figura 3 muestra la visualización de la misma.

Figura 3. Página de Bienvenida [*Welcome Page*].

La página contiene cuatro botones principales. El botón comenzar redirige al editor de código, donde se introduce la sintaxis del lenguaje, que permite resolver modelos epidemiológicos y graficar los resultados. Esta tiene dos botones y tres componentes principales.

A modo de ilustración se muestran los resultados para dos modelos. El que se muestra en la Figura 4 es un modelo clásico de tipo SIR considerando muerte natural y por enfermedad. En la Figura 5 se presenta un modelo que simula la dinámica de transmisión de la COVID-19, considerando asintomáticos y expuestos, de la autoría de investigadores del grupo de Biomatemática de la Facultad de Matemática y Computación de la Universidad de La Habana.

En ambas figuras se muestra el modelo, tal y como el usuario lo introduce al programa, así como las condiciones iniciales para las variables y los valores seleccionados para los parámetros y el intervalo de tiempo.

En las líneas finales aparece el método numérico que se seleccionó y el comando para la graficación de los resultados.

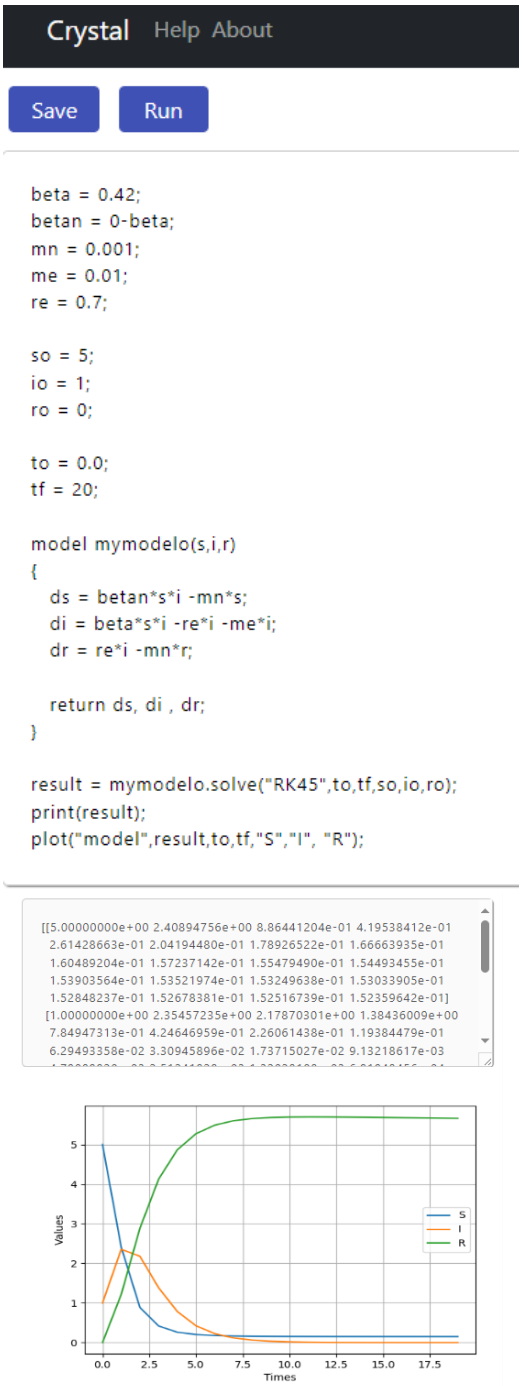

Figura 4. Declaración del modelo SIR por muerte natural y enfermedad en *Crystal* y resultados de su estimación [*Statement of the SIR model for natural death and disease in Crystal and estimation results*].

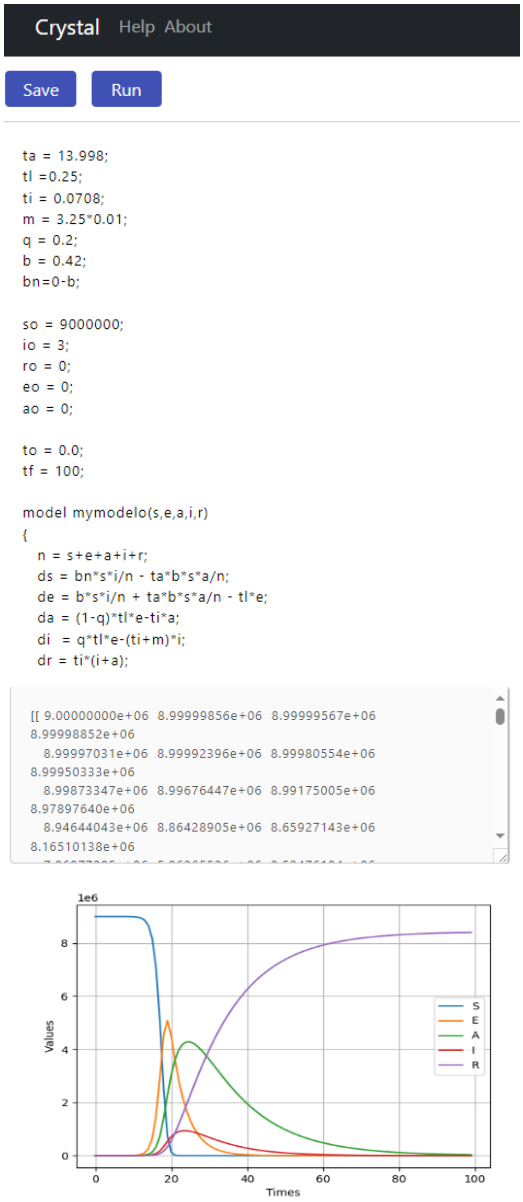

Figura 5. Declaración del modelo que simula la dinámica de transmisión de la COVID-19 en *Crystal* y resultados de su estimación [*Declaration of the model that simulates the transmission dynamics of COVID-19 in Crystal and results of its estimation*].

En la leyenda de las gráficas se presentan las curvas de los Susceptibles ('S'), Infectados ('I') y Recuperados ('R'), para el primer modelo, a los que se incorporan en el segundo modelo, los de Expuestos ('E') y Asintomáticos ('A').

Dichos resultados permiten un análisis visual, en los que, como se espera, la subpoblación de Susceptibles tiende a disminuir y la de Recuperados a crecer a medida que transita la epidemia, mientras los Infectados, Asintomáticos y Expuestos alcanzan su punto máximo (pico de la epidemia) a partir del cual decrecen, lo que significa control de la misma.

Conclusiones

Se diseñó e implementó una primera versión de una herramienta computacional denominada *Crystal*, que contiene una interfaz visual amigable con el usuario para la resolución, análisis y graficación de modelos matemáticos poblacionales definidos por sistemas de EDO aplicados a la epidemiología, objetivo fundamental de este trabajo.

El estudio del estado del arte permitió determinar que no se cuenta con un programa con las características deseadas. Se desarrolló un lenguaje de dominio específico DSL, que permite introducir una amplia clase de modelos matemáticos poblacionales, implementando la funcionalidad de resolver numéricamente modelos matemáticos poblacionales definidos por sistemas de EDO. Además, que permite la graficación de los resultados obtenidos.

Las pruebas de funcionalidad permitieron validar el correcto funcionamiento de la aplicación web. Este primer resultado, permite su extensión para que funcione como un repositorio de modelos con sus potencialidades de solución, manejo de datos y graficación donde se almacenarán importantes resultados del trabajo del Grupo de Biomatemática.

Todos los resultados se muestran de manera amigable haciendo uso de tablas y gráficas. La aplicación está apta para ser utilizada por cualquier investigador, sin necesidad de que tenga conocimientos de programación.

El propio desarrollo de la herramienta y las investigaciones realizadas potencian líneas para continuar el trabajo, específicamente en la mejora de corrección de errores, la posibilidad de incorporar nuevos métodos de solución numérica, trabajar en la solución del problema de estimación de parámetros con algoritmos clásicos de optimización y metaheurísticas. Además, a mediano plazo, integrar una base de datos que permita guardar los resultados y la reutilización de los mismos, incorporar seguridad a la aplicación, implementando un módulo de autenticación y mejorar el diseño gráfico de la interfaz visual.

Agradecimientos

Agradecemos el financiamiento del Proyecto PN223LH010-042 "Nuevas aproximaciones en la modelación dinámica de enfermedades" del Programa Nacional de Ciencias Básicas, Ministerio de Ciencia, Tecnología y Medio Ambiente, Cuba, 2024-2026.

Referencias

[1] Abelló Ugalde, I.A., R. Guinovart Díaz y W. Morales Lezca: *El modelo SIR básico y políticas antiepidémicas de salud pública para la COVID-19 en Cuba*. Revista Cubana de Salud Pública, 46(suppl 1):e2597, 2020. https://scielosp.org/article/rcsp/2020.v46suppl1/e2597/es/.

[2] Ahmad, S. and A. Ambrosetti: *A textbook on ordinary differential equations*, volume 88. Springer, 2015. https://link.springer.com/book/10.1007/978-3-319-16408-3.

[3] Ávila Pozos, R., R.R. Jiménez-Munguía y R. Temoltzi-Ávila: *Modelos estocásticos en epidemiología*. Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI, 6(12):95–101, 2019. doi.org/10.29057/icbi.v6i12.3438.

[4] Guinovart-Diaz, R., W. Morales-Lezca, I.A. Abelló-Ugalde y M.J. Vidal-Ledo: *Un modelo matemático explica la necesidad de la protección para vencer a la COVID-19*. INFODIR, 2020. https://revinfodir.sld.cu/index.php/infodir/article/view/978.

[5] Guinovart-Sanjuán, D., R. Guinovart-Díaz, K. Vajravelu, W. Morales-Lezca, and I. Abelló-Ugalde: *Multi-population analysis of the Cuban SARS-CoV-2 epidemic transmission before and during the vaccination process*. Physics of Fluids, 33(10), 2021. https://pubmed.ncbi.nlm.nih.gov/34737533/.

[6] Janikow, C.Z.: *Adapting representation in genetic programming*. En *Genetic and Evolutionary Computation Conference*, páginas 507–518. Springer, 2004. https://link.springer.com/chapter/10.1007/978-3-540-24855-2_61.

[7] Menció Padrón, D., G. Bayolo Soler y A. Marrero Severo: *Análisis de Modelo Matemático con percepción de riesgo para la CoVid19. Resultados para Cuba*. Revista Cubana de Informática Médica, 12(2), 2020. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1684-18592020000200002.

[8] Teschl, G.: *Ordinary differential equations and dynamical systems*, volumen 140. American Mathematical Society, 2024. https://bookstore.ams.org/gsm-140.

Conflictos de interés

Se declara que no existen conflictos de interés. Ninguno de los autores, ni la Institución hemos recibido pago de terceros para ningún aspecto relacionado con la obra presentada. Hemos presentado documentación que incluye el manual de usuario para el registro del software.

Contribución de autoría

Conceptualización A.A.M.S., D.R.C., W.M.L.

Curación de datos D.R.C, W.M.L.

Análisis formal A.A.M.S., D.R.C., W.M.L.

Adquisición de Financiamiento A.A.M.S., W.M.L.

Investigación D.R.C., A.A.M.S., W.M.L.

Metodología D.R.C., A.A.M.S., W.M.L.

Software D.R.C.

Validación D.R.C., A.A.M.S., W.M.L.

Visualización D.R.C.

Redacción: preparación del borrador original A.A.M.S., D.R.C.

Redacción: revisión y edición A.A.M.S., W.M.L.

Suplementos

Este artículo no contiene información suplementaria.