DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 12 No. Especial 4 (2024) 303–309

Interfaz gráfica para la simulación del modelo mínimo de Bergman
Graphical interface for the simulation of the bergman minimum model

Fernando Torres-Velazco [a,*], Paul J. Campos-Hernández [a], Catherine Ramírez-Ubieta [a], José R. Cárdenas-Valdez [a]

[a] *Instituto Tecnológico de Tijuana, Tecnológico Nacional de México, 22435, Tijuana, Baja California, México.*

Resumen

En este trabajo de investigación, se presenta el desarrollo de una interfaz gráfica en Python para la simulación del modelo mínimo de Bergman, el cual describe la interacción entre la glucosa y la insulina en pacientes con diabetes tipo 1. El modelo fue implementado en una interfaz que permite la simulación de diferentes escenarios de la enfermedad, facilitando el análisis de cómo diversos parámetros influyen en las dinámicas glucosa-insulina. La validación del modelo se llevó a cabo mediante comparaciones con estudios previos, confirmando su precisión en la representación de estas interacciones. Como resultado, se logró crear una herramienta que facilita la exploración y el estudio de las dinámicas de glucosa e insulina en diferentes condiciones, proporcionando una base para futuras investigaciones en el control de la glucosa en diabetes tipo 1.

Palabras Clave: Diabetes, Glucosa, Insulina, Interfaz gráfica de usuario, Modelo mínimo de Bergman, Python.

Abstract

In this research work, the development of a graphical interface in Python for the simulation of the Bergman minimal model, which describes the interaction between glucose and insulin in patients with type 1 diabetes, is presented. The model was implemented in an interface that allows the simulation of different disease scenarios, facilitating the analysis of how various parameters influence glucose-insulin dynamics. The model was validated through comparisons with previous studies, confirming its accuracy in representing these interactions. As a result, a tool was created that facilitates the exploration and study of glucose and insulin dynamics under different conditions, providing a foundation for future research in glucose control in type 1 diabetes.

Keywords: Diabetes, Extended minimal model of Bergman, Glucose, Graphical user interface, Insulin, Python.

1. Introducción

La diabetes mellitus tipo 1 (T1DM) es una enfermedad crónica caracterizada por la destrucción autoinmune de las células beta del páncreas, resultando en una deficiencia de insulina. Tradicionalmente, el manejo de la diabetes se ha realizado mediante la monitorización intermitente de la glucosa en sangre y la administración de dosis apropiadas de insulina (Lambert y Bingley, 2002). Sin embargo, lograr un control óptimo de la glucemia sigue siendo un desafío debido a la variabilidad en las necesidades de insulina durante el día y la complejidad del manejo de la diabetes. En este contexto, la creación de una interfaz gráfica de usuario (GUI), accesible y eficaz, es fundamental en el diseño de sistemas de control automático para regular los niveles de glucosa en individuos con T1DM.

El progreso en la tecnología del páncreas artificial ha sido significativo en las últimas décadas, culminando en el desarrollo de sistemas híbridos comercialmente disponibles en Norteamérica y Europa, que han mejorado notablemente el control de la glucemia en personas con T1DM (Ramli *et al.*, 2019). Estos sistemas, también conocidos como sistemas de circuito cerrado o de administración automatizada de insulina, buscan imitar la función endocrina de un páncreas sano para la homeostasis de la glucosa, incorporando un sensor para monitoreo continuo de glucosa, una bomba de insulina y un algoritmo que conecta ambos dispositivos (Peyser *et al.*, 2014).

El uso de simuladores aprobados por la Administración de Alimentos y Medicamentos (FDA), como el UVA/Padova, permite la validación y el entrenamiento de algoritmos de control en pacientes virtuales, lo cual es crucial para la evaluación de nuevos métodos de control automático de la glucosa (Lee *et al.*, 2020). Este enfoque ha sido utilizado por investigadores para

*Autor para correspondencia: fernando.torres19@tectijuana.edu.mx
Correo electrónico: fernando.torres19@tectijuana.edu.mx (Fernando Torres-Velazco), paul.campos@tectijuana.edu.mx (Paul Javier Campos-Hernández), catherine.ramirez193@tectijuana.edu.mx (Catherine Ramírez-Ubieta), jose.cardenas@tectijuana.edu.mx (José Ricardo Cárdenas-Valdez)

Historial del manuscrito: recibido el 29/06/2024, última versión-revisada recibida el 03/10/2024, aceptado el 20/09/2024, publicado el 30/11/2024. **DOI:** https://doi.org/10.29057/icbi.v12iEspecial4.13328

desarrollar y evaluar elementos de retroalimentación auxiliar de seguridad en el páncreas artificial. En particular, (Revert *et al.*, 2013) emplearon el simulador UVA/Padova para demostrar que su método reduce significativamente los eventos de hipoglucemia, una condición en la que los niveles de glucosa en sangre caen por debajo de 70 mg/dL, lo que puede causar síntomas como mareos, confusión o incluso pérdida de conciencia. La reducción de estos episodios mejora la seguridad y eficacia del control de glucosa en pacientes con diabetes tipo 1. Además, el desarrollo de modelos in silico y simuladores de la diabetes ha permitido reemplazar los ensayos en animales, acelerando el progreso de los algoritmos de páncreas artificial y permitiendo pruebas preclínicas eficientes y rentables (Kovatchev, 2019).

Los ensayos clínicos de páncreas artificial combinan el uso de una bomba de insulina inteligente con un monitor continuo de glucosa (CGM) y un algoritmo de control para mejorar la regulación de la glucemia en personas con T1DM. Estos ensayos han demostrado que la implementación de restricciones de insulina en el cuerpo (IOB) mediante algoritmos de control como el elemento de retroalimentación auxiliar de seguridad (SAFE) reduce los eventos hipoglucémicos y mejora el control glucémico tanto in silico como in vivo (Fushimi *et al.*, 2018). Además, la evaluación de algoritmos de control híbrido y de lazo cerrado completo es esencial para garantizar la viabilidad de las estrategias propuestas como métodos efectivos para tratar a pacientes con T1DM.

Este artículo está organizado de la siguiente manera: en la Sección 2 se presenta el modelo mínimo glucosa-insulina de Bergman, en la Sección 3 se presentan las librerías de Python para el desarrollo de la GUI, en la Sección 4 se presenta la metodología para el desarrollo de la GUI; en la Sección 5 se presentan los resultados. Finalmente, en la Sección 6 se muestran las conclusiones y trabajos a futuro.

2. Modelo Mínimo Glucosa-Insulina de Bergman

El enfoque del modelo mínimo glucosa-insulina de Bergman se dirige hacia la interacción entre la glucosa y la insulina en el organismo, explicando cómo la insulina controla los niveles de glucosa en la sangre y viceversa. El modelo está definido por las siguientes ecuaciones diferenciales:

$$\frac{dG}{dt} = -S_G \cdot G(t) - X(t) \cdot G(t) + G_b, \tag{1}$$

$$\frac{dI}{dt} = -n \cdot (I(t) - I_b), \tag{2}$$

$$\frac{dX}{dt} = -p \cdot X(t) + p \cdot S_I \cdot (I(t) - I_b); \tag{3}$$

donde $G(t)$ representa la concentración de glucosa en la sangre en el instante de tiempo t, la sensibilidad a la insulina en relación con la glucosa se representa como S_G; mientras, la acción de la insulina en un momento dado t se expresa como $X(t)$. La variable G_b representa la tasa de producción basal de glucosa; $I(t)$ indica la concentración de insulina en la sangre en un momento dado t. n es el factor de degradación de la insulina; I_b representa el nivel basal de insulina en la sangre. El parámetro p representa la velocidad a la que la acción de la insulina se disipa; S_I indica la sensibilidad a la insulina.

Adicionalmente en (Fisher, 1991) se realiza una extensión al Modelo Mínimo Glucosa-Insulina de Bergman en la cual se agrega una cuarta ecuación, la cual representa la perturbación causada por la ingesta de alimentos durante el tratamiento. El modelo está dado por las siguientes ecuaciones:

$$\dot{x_1} = -p_1(x_1 - G_b) - x_1 x_2 + x_4, \tag{4}$$

$$\dot{x_2} = -p_2 x_2 + p_3(x_3 - I_b), \tag{5}$$

$$\dot{x_3} = -p_4(x_3 - I_b) + u(t), \tag{6}$$

$$\dot{x_4} = -p_5 x_4; \tag{7}$$

donde x_1 es la concentración de glucosa en la sangre; x_2, es la concentración de insulina remota; x_3 es la concentración plasmática de insulina; x_4 es la perturbación de comida de Fisher; G_b es la glucosa plasmática basal; p_1 el factor de efectividad de glucosa; p_2 el retraso de acción de insulina; p_3 es la tasa de transferencia de insulina; p_4 es la degradación de insulina; p_5 es la perturbación de comida; y $u(t)$ es la infusión externa de insulina.

3. Python

Python es un lenguaje de programación flexible que se ha vuelto indispensable y, debido a sus bondades de código abierto se ha utilizado en el campo de la investigación científica y el avance tecnológico. La claridad y simplicidad de su estructura, junto con una amplia variedad de bibliotecas, hacen que sea adecuado para actividades exigentes como la simulación de modelos matemáticos y la creación de GUI (Chongchitnan, 2023), (Optimization, 2023).

La utilización de Python en este proyecto se centra en dos áreas fundamentales: la simulación del modelo mínimo de Bergman y el desarrollo de una GUI que facilite la interacción con el modelo. Para alcanzar este objetivo, se utilizan diversas bibliotecas fundamentales de Python que se muestran a continuación.

3.1. Tkinter

Tkinter es la biblioteca estándar para la creación de GUI para el lenguaje de programación Python. Este lenguaje, al ser utilizado junto con la biblioteca Tkinter, ofrece una manera eficiente y sencilla de desarrollar aplicaciones con una GUI. Tkinter proporciona una interfaz robusta y basada en objetos para el conjunto de herramientas GUI de TK (Lundh, 1999). Algunos de los módulos empleados para la creación de la GUI son:

- **tkinter**: el cual es el módulo principal para la creación de GUI.
- **tkinter.ttk**: proporciona widgets temáticos mejorados.
- **tkinter.messagebox**: el cual es utilizado para crear cuadro de diálogos para mensajes.

3.2. SciPy

SciPy es un conjunto de algoritmos matemáticos y funciones de utilidad desarrolladas sobre la base de NumPy. Python adquiere una relevancia notable al ofrecer al usuario comandos y clases de alto nivel que permiten la manipulación y visualización de datos. SciPy se estructura en subpaquetes que abarcan distintos campos de la informática científica (Virtanen *et al.*, 2020).

Para llevar a cabo este proyecto, se emplea la función **odeint** de la biblioteca **scipy.integrate**, la cual utiliza el algoritmo de integración basado en el solucionador de *livermore* para ecuaciones diferenciales ordinarias con conmutación automática de métodos para problemas rígidos y no rígidos (LSODA), el cual tiene la capacidad de alternar de manera automática entre métodos de integración de ecuaciones diferenciales rígidas y no rígidas, según lo requiera la situación (SciPy, 2020). Un sistema de ecuaciones diferenciales ordinarias se clasifica como rígido cuando existe una marcada disparidad en las velocidades de cambio de sus soluciones, lo cual se evidencia mediante sus valores propios con una notable diferencia en sus magnitudes (Press *et al.*, 2007). Esta disparidad en las tasas de cambio provoca que ciertas variables del sistema se desarrollen a velocidades considerablemente diferentes entre si.

Dentro del contexto del modelo mínimo de Bergman, la evaluación de la rigidez del sistema se realiza mediante el cálculo de los valores propios de la matriz Jacobiana. Estos están influenciados por los parámetros del sistema p_2, p_3, p_4, p_5, y la variable de estado x_2. Según los valores iniciales empleados en este estudio, el sistema no se clasifica como rígido, ya que no se observa una diferencia significativa en las tasas de cambio entre los valores propios. No obstante, la rigidez del sistema puede variar en función de diferentes valores de p_2, p_3, p_4 y x_2, lo que podría modificar el comportamiento del sistema en diversas condiciones clínicas.

Los procedimientos empleados cuando el sistema de ecuaciones no es rígido incluyen el método de Adams-Bashforth-Moulton, el cual es un método de múltiples pasos explícito. Por otro lado, cuando se identifica que el sistema es rígido, se cambia automáticamente a los métodos de fórmulas de diferenciación hacia atrás (BDF), los cuales son métodos implícitos y más apropiados para problemas rígidos.

3.3. NumPy

NumPy es una biblioteca ampliamente reconocida en el lenguaje de programación Python, la cual se emplea principalmente para llevar a cabo operaciones matemáticas y científicas (Harris *et al.*, 2020). Las operaciones que se utilizarán en este proyecto son:

- **np.linspace**: la cual se emplea para crear un conjunto de puntos de tiempo equidistantes que se utilizan como base en la simulación e integración del sistema de ecuaciones diferenciales.
- **np.argmax**: esta función se emplea para hallar el índice del valor máximo en un arreglo. En el contexto de este proyecto, se empleó para determinar el momento exacto en el que la concentración de glucosa alcanza su nivel máximo.
- **np.max**: la cual se emplea para determinar el valor máximo dentro de un arreglo. En este proyecto se empleó para determinar el valor máximo de concentración de glucosa durante la simulación.

3.4. Matplotlib

Matplotlib es una biblioteca en Python que ofrece una manera eficaz de crear representaciones gráficas y visuales. Es extensamente empleada en la comunidad por su habilidad para generar gráficos complejos con un alto grado de personalización. Uno de los elementos principales de la biblioteca es **matplotlib.pyplot**, que consiste en un conjunto de funciones que permiten que *Matplotlib* se comporte de manera similar a *MATLAB*. Cada función de *pyplot* realiza diversas acciones en una figura, como crearla, generar un área de trazado en la misma, dibujar líneas en el área de trazado y añadir etiquetas decorativas al gráfico, entre otras funcionalidades (Hunter, 2007).

4. Metodología

En esta sección se describe la metodología empleada para crear la GUI utilizando Tkinter en el lenguaje de programación Python. En primer lugar, se lleva a cabo la simulación y validación del modelo mínimo de Bergman. Posteriormente, se procede con el diseño de la GUI y finalmente se realiza la integración del modelo.

4.1. Validación y simulación

Se lleva a cabo la simulación del modelo mínimo de Bergman, descrito por las ecuaciones (4)-(7), empleando el lenguaje de programación Python. Se utilizaron las bibliotecas SciPy para la resolución del sistema de ecuaciones y Matplotlib en combinación con NumPy para la representación gráfica del modelo. Los valores empleados para la simulación del sistema están detallados en la Tabla 1(Anirudh Nath, 2019); los rangos y valores típicos son establecidos a través de estudios experimentales y clínicos (Furler *et al.*, 1985), (Bergman *et al.*, 1981).
.

Tabla 1: Parámetros y valores del sistema.

Parámetro	Valor
Glucosa basal en plasma (Gb)	4.5 $mMolL^{-1}$
Factor de efectividad de glucosa (p1)	0 min^{-1}
Retardo en la acción de insulina (p2)	0.015 min^{-1}
Tasa de transferencia de insulina (p3)	0.000002 $mUL^{-1}min^{-2}$
Tasa de degradación de insulina (p4)	0.021 min^{-1}
Disturbio por la comida (p5)	0.05 min^{-1}
Insulina basal en plasma (Ib)	4.5 mUL^{-1}

En la Figura 1, se muestra la representación visual de la simulación del modelo utilizando la herramienta Matplotlib. Además, se ha incluido un indicador en forma de flecha que señala el momento y el valor en el que se alcanzó el máximo nivel de glucosa. Las condiciones iniciales empleadas son

idénticas a las descritas en (Anirudh Nath, 2019), lo cual resulta en la obtención de resultados similares en el gráfico.

Inicialmente el modelo es simulado utilizando MATLAB para establecer una referencia base de los resultados, posteriormente, es simulado utilizando la biblioteca Matplolib, para validar la exactitud del modelo, los resultados de las dos simulaciones fueron comparados con los reportados por (Ali y Padhi, 2011), (Anirudh Nath, 2019) y (Acharya y Das, 2021). Dado que los resultados coinciden en todos los casos, se considera que el modelo está correctamente validado.

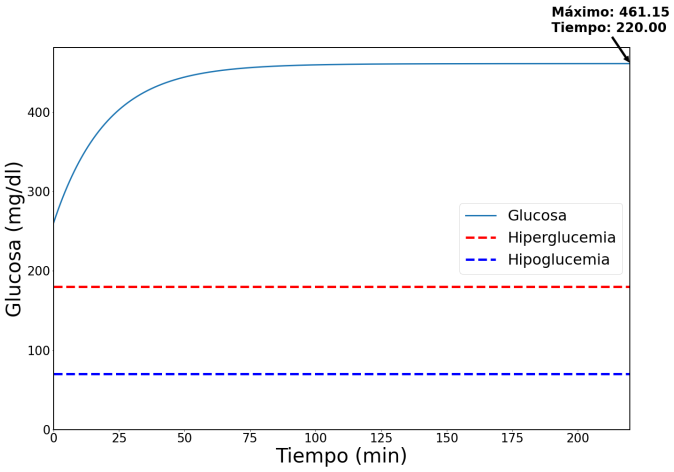

Figura 1: Simulación del modelo en *Matplotlib*.

4.2. *Diseño de la GUI*

Con el fin de facilitar la comprensión del proceso de desarrollo de la GUI, se incluye un diagrama de flujo en la Figura 2, el cual muestra las etapas fundamentales desde la configuración inicial hasta la obtención de los resultados de la simulación. Este diagrama de flujo proporciona una visión general del flujo de trabajo y facilita la identificación de las interacciones entre los distintos componentes. El diagrama de flujo se encuentra dividido en diversas secciones principales, las cuales se detallan a continuación.

1. **Inicio y configuración del sistema**:
   - Importar librerías: el proceso comienza con la importación de las librerías necesarias para el funcionamiento de la GUI y la simulación.
   - Crear ventana principal de la GUI: se configura la ventana principal donde se ubica toda la GUI.
   - Configurar paneles y secciones: se organizan los paneles y las secciones dentro de la ventana principal para estructurar adecuadamente la GUI.
2. **Configuración de la interfaz de usuario**:
   - Añadir etiquetas y campos de entrada: se añaden etiquetas descriptivas y campos de entrada para que el usuario pueda introducir los datos necesarios para la simulación.
   - Crear botón para ejecutar la simulación: se incorpora un botón que permite al usuario iniciar la simulación.
   - Crear etiquetas para mostrar resultados: se establecen etiquetas que muestran los resultados de la simulación una vez completada.
   - Configurar área para gráfica: se define un área dentro de la GUI donde se muestra la gráfica resultante de la simulación.
3. **Ejecución de la simulación**:
   - Leer datos de entrada: se verifican los datos introducidos por el usuario para asegurarse de que son valores numéricos válidos.
   - Definir y resolver ecuaciones diferenciales: se definen y resuelven las ecuaciones diferenciales del modelo mínimo de Bergman.
   - Calcular glucosa máxima y tiempo de hiperglucemia: se calculan parámetros importantes como la glucosa máxima y el tiempo de hiperglucemia.
   - Dibujar gráfica y anotar puntos importantes: se genera la gráfica que muestra los resultados de la simulación, destacando los puntos clave.
   - Mostrar resultados: finalmente, se presentan los resultados calculados y la gráfica en la GUI.
4. **Gestión de errores**:
   - Mostrar mensaje de error: si los datos de entrada no son válidos, se muestra un mensaje de error indicando al usuario que debe corregir los valores introducidos.

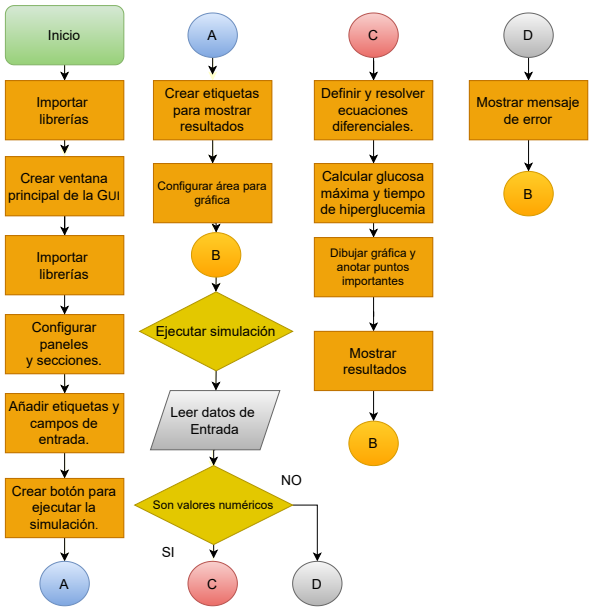

Figura 2: Diagrama de flujo de la GUI.

En la Figura 3, se muestra la primera fase de la GUI, en la que se visualizan dos columnas. En esta etapa, se han empleado dos tipos de elementos de interfaz proporcionados por la librería Tkinter: el widget Label para mostrar texto y el widget

Entry para ingresar valores numéricos. Se ha utilizado el método **entry.insert** para configurar los valores predeterminados al iniciar la GUI.

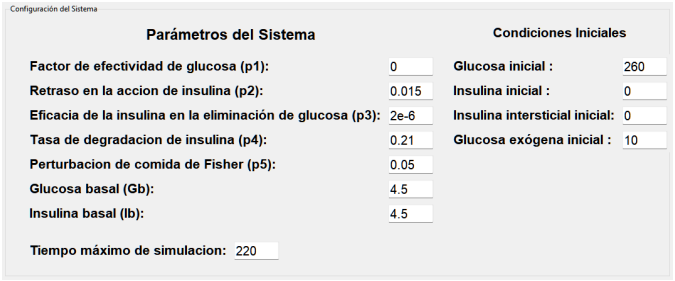

Figura 3: Primera fase del diseño de la GUI.

Posteriormente, se implementa un botón para ejecutar la simulación,dos etiquetas que muestran el valor máximo de glucosa y el momento en que el paciente entra en hiperglucemia, además de cuatro botones para seleccionar la variable a visualizar. Con la biblioteca **matplotlib.backend** (Matplotlib, 2020), se crea un elemento denominado "canvas" que permite visualizar gráficos de Matplotlib en tkinter.

Finalmente, el modelo se modifica para que al ejecutar la simulación esta se lleve a cabo con los datos proporcionados por el usuario. Además, se implementa una restricción para aceptar únicamente valores numéricos. En caso de que el usuario introduzca algún carácter que no sea un número, se muestra una ventana de error.

5. Resultados

En esta sección se exponen los resultados logrados tras la ejecución de la GUI para la simulación del modelo mínimo de bergman, en las cuales se incluye:

- Primer implementación.
- Prueba de excepción numérica.
- Simulación cambiando parámetros predeterminados.

Donde se evalúa la correcta funcionalidad de los widgets: label, entry y el botón, los cuales son previamente detallados en el diseño de la GUI.

5.1. Primer implementación

Al iniciar la GUI, los parámetros del sistema especificados en la Tabla 1, se introducen automáticamente. En la Figura 4 se observa que al ejecutar la simulación, esta se lleva a cabo durante un período máximo de 220 minutos, que en este caso es el tiempo especificado. La GUI contiene indicadores que representan los valores máximos de glucosa durante el proceso de simulación. Es importante señalar que al tener una concentración inicial de glucosa en el sistema de 260 mg/dl (hiperglucemia), la etiqueta que indica el tiempo de hiperglucemia comienza desde el minuto 0.

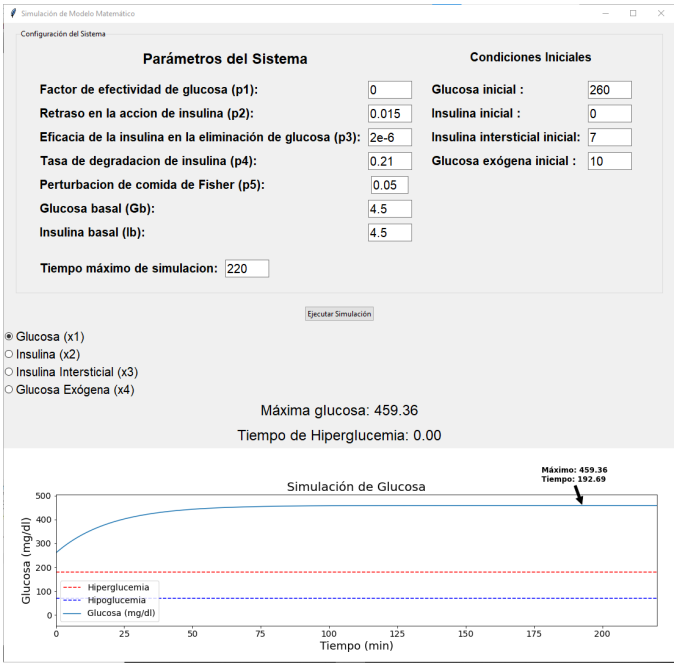

Figura 4: Primer implementación de la GUI (Glucosa).

De igual manera al seleccionar el botón de "Insulina", la gráfica del "canvas" se actualiza por la variable de insulina. En la Figura 5 se observa como el valor inicial de insulina es cero y este aumenta hasta alcanzar un valor de $2e{-}5\mu U/ml$, para después estabilizarse en cero de nuevo.

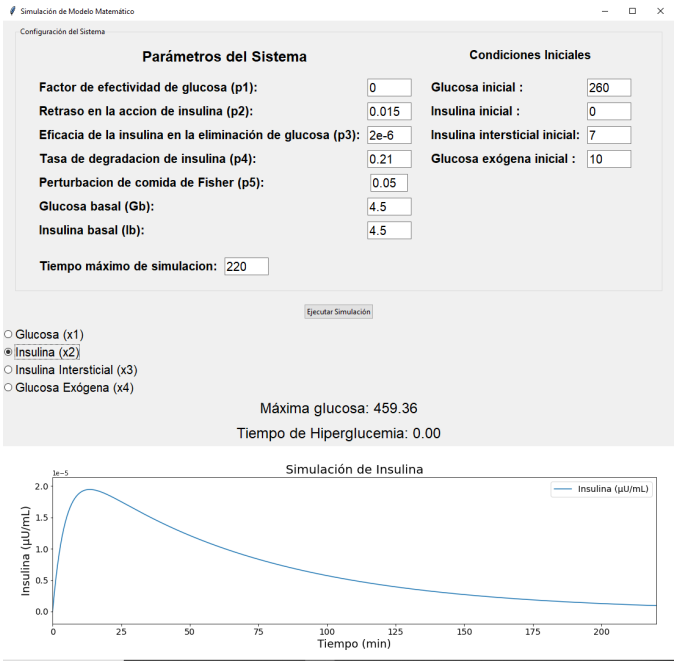

Figura 5: Primer implementación de la GUI (Insulina).

Al seleccionar la variable de insulina intersticial se observa en la Figura 6 que la simulación comienza en un valor inicial dado por el usuario de $7\mu U/ml$, para después estabilizarse en la insulina basal la cual corresponde a $4.5\mu U/ml$.

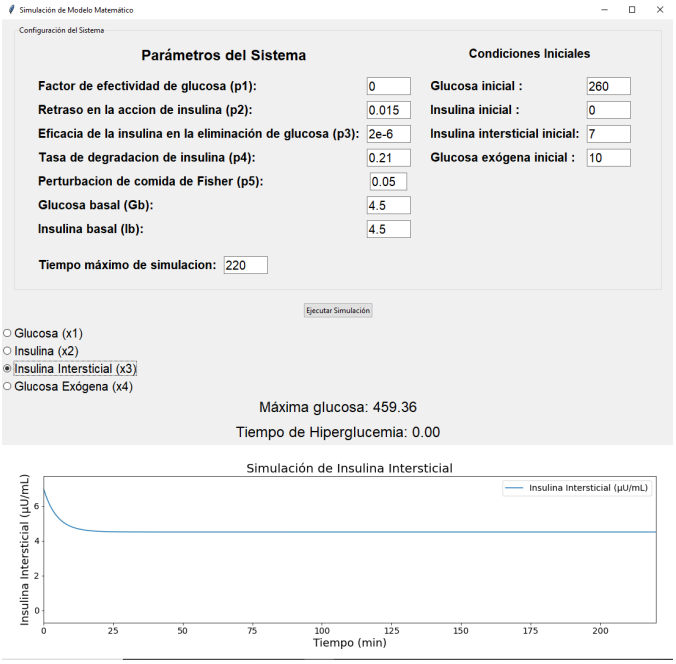

Figura 6: Primer implementación de la GUI (Insulina intersticial).

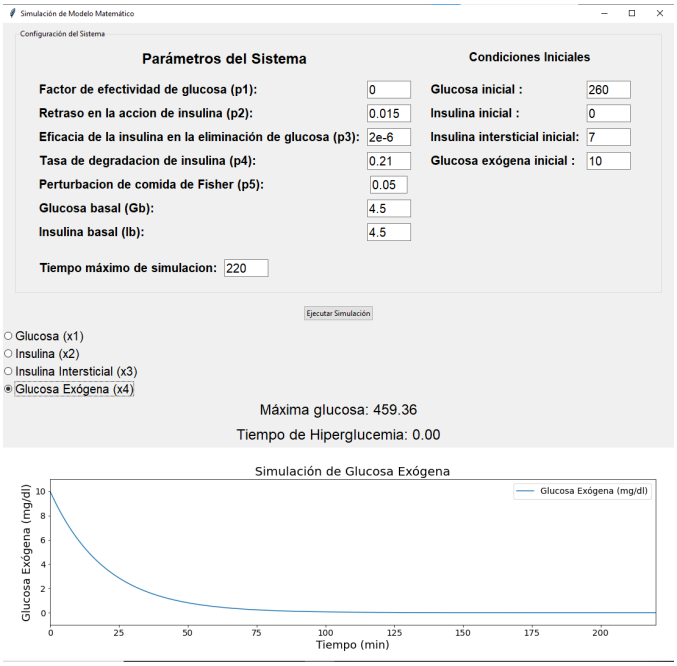

Figura 7: Primer implementación de la GUI (Glucosa exógena).

5.2. *Prueba de excepción numérica*

Posteriormente, se lleva a cabo la prueba de la excepción añadida, en la que, al ingresar un carácter que no sea un número, se muestra una ventana con un mensaje pidiendo al usuario que introduzca los valores de forma correcta como se observa en la Figura 8.

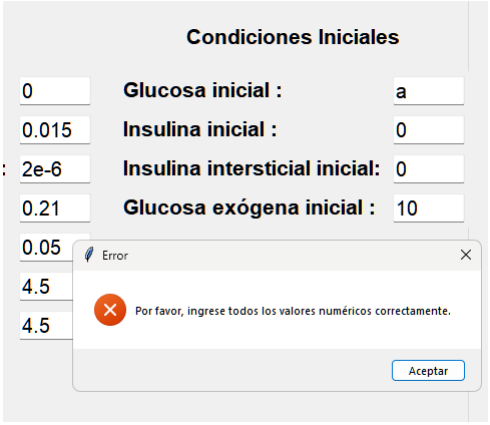

Figura 8: Prueba de excepción numérica.

5.3. *Simulación cambiando parámetros predeterminados.*

Finalmente, en la Figura 9 se lleva a cabo la simulación con parámetros y tiempo de simulación diferentes a los establecidos por defecto. Se observa la flecha que señala el punto máximo y el valor de la glucosa en el gráfico, así como el momento en el que el paciente entra en hiperglucemia.

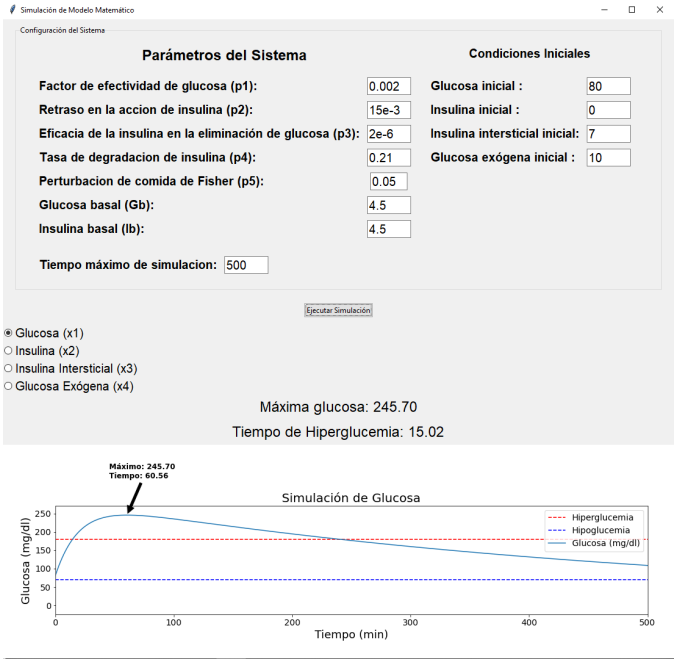

Figura 9: Simulación cambiando parámetros predeterminados.

Conclusiones

A través del lenguaje de programación Python, se crea una herramienta para llevar a cabo la simulación del modelo mínimo de Bergman. Mediante la GUI, los usuarios pueden modificar los parámetros del modelo, establecer las condiciones iniciales y visualizar las simulaciones de manera clara, el sistema está acotado a valores fisiológicos establecidos en estudios experimentales y clínicos.

Se lleva a cabo diversas simulaciones partiendo de niveles elevados de glucosa en sangre y niveles normales de glucosa. Asimismo, se realiza simulaciones modificando parámetros

que podrían variar en las diferentes etapas de desarrollo de la enfermedad, como por ejemplo p_1, que representa el factor de eficacia de la glucosa. Los resultados señalan que la GUI es efectiva en la representación del comportamiento del estado x_1, que se refiere a la concentración de glucosa en la sangre. Esta representación es crucial para la integración y verificación de señales de control.

En futuros proyectos, se buscará la integración de diversos modelos matemáticos de diabetes a través de un menú desplegable para su selección. Además, se espera integrar uno o varios controladores al sistema. Además, en cuanto a la transmisión y recepción de datos, se propone dividir el programa en dos secciones: una para simular modelos de diabetes y otra para analizar los datos de los pacientes recibidos.

Agradecimientos

Esta investigación fue financiada por los proyectos de investigación del Tecnológico Nacional de México (TecNM), número 20586.24-P y 20136.24-P.

Referencias

Acharya, D. y Das, D. K. (2021). Non linear back stepping based sliding mode controller design with real-time simulator for regulating glucose in type-1 diabetic patient. pp. 1–5.

Ali, S. F. y Padhi, R. (2011). Optimal blood glucose regulation of diabetic patients using single network adaptive critics. *Optimal Control Applications and Methods*, 32(2):196–214.

Anirudh Nath, Rajeeb Dey, C. A.-A. (2019). Observer based nonlinear control design for glucose regulation in type 1 diabetic patients: An LMI approach. *Biomedical Signal Processing and Control*, 47:7–15.

Bergman, R. N., Phillips, L. S., y Cobelli, C. (1981). Physiologic evaluation of factors controlling glucose tolerance in man: measurement of insulin sensitivity and beta-cell glucose sensitivity from the response to intravenous glucose. *Journal of Clinical Investigation*, 68(6):1456–1467.

Chongchitnan, S. (2023). *Exploring University Mathematics with Python*. Springer Nature Switzerland AG.

Fisher, M. E. (1991). A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *IEEE Transactions on Biomedical Engineering*, 38(1):57–61.

Furler, S. M., Biomed E, M., Kraegen, E. W., Smallwood, R. H., y Chisholm, D. J. (1985). Blood glucose control by intermittent loop closure in the basal mode: Computer simulation studies with a diabetic model. *Diabetes Care*, 8(6):553–561.

Fushimi, D., Nakamura, Y., y Tanaka, H. (2018). Clinical trials and validation of control algorithms in artificial pancreas systems for type 1 diabetes. *Journal of Diabetes and Its Complications*, 32(8):789–796.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., y Oliphant, T. E. (2020). Array programming with numpy. *Nature*, 585:357–362.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Kovatchev, B. (2019). In silico diabetes models and simulators for accelerated progress of artificial pancreas systems. *Diabetes Technology & Therapeutics*, 21(6):289–298.

Lambert, P. y Bingley, P. J. (2002). What is type 1 diabetes? *Medicine*, 30(1):1–5. Diabetes 1.

Lee, J., Smith, A., y Doe, J. (2020). Use of FDA-approved simulators for validation and training of control algorithms in virtual patients with diabetes. *Journal of Diabetes Science and Technology*, 14(2):123–130.

Lundh, F. (1999). An introduction to tkinter. `http://www.pythonware.com/library/tkinter/introduction/`.

Matplotlib (2020). Matplotlib backends. `https://matplotlib.org/stable/users/explain/backends.html`.

Optimization, G. (2023). Getting started with mathematical optimization in python. `https://www.gurobi.com/resource/intro-to-modeling-with-python/`.

Peyser, T., Dassau, E., Breton, M., y Skyler, J. S. (2014). The artificial pancreas: current status and future prospects in the management of diabetes. *Annals of the New York Academy of Sciences*, 1311(1):102–123.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., y Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing*. Cambridge University Press, Cambridge, 3rd edición.

Ramli, R., Reddy, M., y Oliver, N. (2019). Artificial pancreas: Current progress and future outlook in the treatment of type 1 diabetes. *Drugs*, 79(11):1089–1101.

Revert, A., Garelli, F., Picó, J., De Battista, H., Rossetti, P., Vehi, J., y Bondia, J. (2013). Safety auxiliary feedback element for the artificial pancreas in type 1 diabetes. *IEEE Transactions on Biomedical Engineering*, 60(8):2113–2122.

SciPy (2020). Integration and odes (scipy.integrate). `https://docs.scipy.org/doc/scipy/reference/integrate.html`.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., y Contributors, S. . (2020). Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272.

Zadka, M. (2019). Prioritizing simplicity in your python code. *Opensource.com*.