Enpodishango Na – Tha – Hi
Divulgación Científica en la Región Laja-Bajío

***Volumen** 2, **Número** 2*
Septiembre 2025
***ISSN:** en trámite*

Creación de una librería en Maple para las matrices de Colocación Ortogonal

Mario Alberto Sandoval Hernández
Centro de Bachillerato Tecnológico industrial y de servicios No. 190,
Boca del Río, Veracruz, México

Fernando Iván Molina Herrera
Tecnológico Nacional de México / IT de Celaya,
Celaya, Guanajuato, México

Hugo Jiménez Islas *
Tecnológico Nacional de México / IT de Celaya,
Celaya, Guanajuato, México

Luis Isaí Quemada Villagómez
Tecnológico Nacional de México / IT de Celaya,
Celaya, Guanajuato, México

* Autor de correspondencia: *hugo.jimenez@itcelaya.edu.mx*

Resumen: *Este artículo presenta una guía para crear librerías tipo (.mla) en Maple, dentro de un entorno Linux. La librería desarrollada incluye dos comandos principales: uno que calcula las raíces de los polinomios de Legendre, y otro que genera las matrices utilizadas en el método de colocación ortogonal (A, B, C, D y Q). Este método se emplea para resolver ecuaciones diferenciales con condiciones en la frontera. Además, se incluye un caso de estudio que muestra cómo funciona la librería implementada. Cabe destacar que esta herramienta puede generar matrices y raíces para cualquier grado n, siempre que la memoria del sistema lo permita.*

Palabras clave: *análisis numérico, colocación ortogonal, librerías, software Maple.*

1. Introducción: *lo que debemos saber de inicio*

El programa Maple es una herramienta de software diseñada principalmente para hacer cálculos simbólicos, es decir, manipular expresiones matemáticas de forma exacta, como

derivadas, integrales o ecuaciones. Sin embargo, también permite realizar cálculos numéricos y generar gráficas para visualizar datos o funciones de manera clara y atractiva.

Desde hace varios años, Maple cuenta con una interfaz gráfica intuitiva, lo que facilita su uso, incluso para quienes están comenzando en el manejo de software matemático (Fox, 2011; Fox y Bauldry, 2019; Maplesoft, 2023). Esta interfaz incluye una serie de paletas organizadas por temas. Por ejemplo, si se selecciona la paleta de Cálculo, el usuario podrá acceder fácilmente a operaciones como límites, integrales o derivadas parciales, entre muchas otras. También existen paletas específicas para trabajar con matrices y otros objetos matemáticos. En Maple, los comandos pueden introducirse de forma muy sencilla haciendo clic en los elementos disponibles en las paletas. Esto permite escribir operaciones matemáticas que se ven de manera visual y natural, como si se estuvieran haciendo en una libreta. Por ejemplo, al calcular una derivada, el símbolo aparecerá con la notación usual que se enseña en clase, lo que resulta muy útil en entornos educativos. Esta forma gráfica de usar Maple está pensada especialmente para apoyar la enseñanza y el aprendizaje de las matemáticas. Sin embargo, también es posible escribir los comandos de forma más técnica, usando sintaxis, como en los lenguajes de programación C (Joyanes, 2020) o Fortran 90 (Adams et al., 1992). Maple permite convertir fácilmente entre ambos modos de entrada: desde la vista visual (llamada *entrada matemática 2-D*) a texto plano (conocida como *entrada matemática 1-D*) y viceversa (Fox, 2011; Fox y Bauldry, 2019; Maplesoft, 2023).

Cuando se escribe código en Maple utilizando texto, la notación que se emplea es similar a la que se usa en lenguajes de programación como C o Python (Diaz et al., 2024). Esta forma de trabajo resulta muy útil cuando se desea crear y ejecutar algoritmos de manera más rápida, ya que no requiere del uso de la interfaz gráfica basada en Java, que es la que se utiliza en la entrada matemática en formato 2-D. En Maple, muchas funciones están organizadas en paquetes o librerías, los cuales agrupan comandos por temas específicos. Para poder utilizar esas funciones, es necesario activar el paquete correspondiente usando el comando `with()`. Por ejemplo, si se desea generar una gráfica en tres dimensiones, hay que utilizar el comando plot3d, pero antes se debe activar el paquete `plots` escribiendo `with(plots)`; Maple ofrece una amplia colección de paquetes, cada uno diseñado para resolver distintos tipos de problemas, lo que amplía notablemente las posibilidades del software (Maplesoft, 2023).

En Maple es posible crear paquetes personalizados que agrupan diferentes algoritmos diseñados para cumplir funciones específicas. Una de sus ventajas es que pueden ser llamados fácilmente usando el comando `with`, lo que permite simplificar el código y trabajar con scripts más ordenados y compactos. Además, esta opción es útil cuando se desea mantener privado el contenido de ciertos algoritmos, ya que el código se guarda en archivos con extensión `.mla`, que no muestran su contenido directamente. Este artículo presenta la creación de una librería en Maple que incluye las matrices utilizadas en el método de colocación ortogonal, una técnica numérica que permite encontrar soluciones aproximadas a ecuaciones diferenciales con condiciones en los extremos. El método se basa en evaluar la ecuación en ciertos puntos especiales, llamados puntos de colocación, que corresponden a las raíces de polinomios ortogonales, como los polinomios de Legendre. A partir de esos puntos se construyen matrices que permiten calcular derivadas de forma numérica, lo que transforma el problema original en un sistema de ecuaciones algebraicas que puede resolverse por computadora. La librería también incluye una rutina para determinar las raíces de estos polinomios, basada en métodos desarrollados por Finlayson (1981), Villadsen y Stewart (1967) y Villadsen y Michelensen (1978).

2. Fundamentos Teóricos: *reglas y principios científicos importantes*

Los archivos internos que utiliza Maple para almacenar sus funciones suelen ocupar muy poco espacio. Sin embargo, si se guardaran por separado, uno por uno, se desperdiciaría memoria en el disco, ya que cada archivo necesita una cantidad mínima de espacio asignado. Para evitar esto, Maple organiza sus funciones en archivos tipo biblioteca, con extensión `.mla`, que agrupan muchos elementos en un solo archivo. El archivo principal que contiene las funciones básicas del sistema se llama `maple.mla`. Cuando se desea crear una biblioteca propia en Maple, es necesario utilizar el paquete `LibraryTools`, que incluye herramientas para crear, editar y manejar estas bibliotecas personalizadas. En general, para generar un archivo `.mla`, se deben seguir algunos pasos dentro del entorno de trabajo de Maple, que serán explicados más adelante.

1. Establecer el nombre del archivo con extensión (.mla) que nos interesa generar, por ejemplo, `LibLocation:=cat(currentdir(),"/A_Numericos_V1_0.mla")`. Este comando proporciona el directorio actual o carpeta donde el archivo (.mla) será creado.

Posteriormente dentro del script que se está escribiendo debe utilizarse el comando `march()`. El administrador de archivos de Maple, march, se utiliza para gestionar archivos y variables archivados en una biblioteca de Maple. En este caso se utiliza el argumento `create` para crear un archivo, por ejemplo, la instrucción `march('create', LibLocation)` crea el archivo `A_Numericos_V1_0.mla` en el directorio actual de trabajo. Este varía dependiendo de factores como la instalación del programa y el sistema operativo los cuales pueden ser Linux, MacOs y Windows.

2. En este paso es importante conocer la dirección contenida en la variable `libname`. Al configurar `libname` se puede especificar ubicaciones en las que se buscan rutinas creadas de forma secuencial. Se tienen dos opciones, la primera es garantizar que el archivo (.mla) que se va a generar se encontrara en ese lugar; la segunda, es decirle a Maple que busque ese archivo en la dirección actual de trabajo.
3. Los procedimientos (o funciones con las respectivas tareas) deben estar contenidos en un módulo, declarado por el comando `module()`. Un módulo es una expresión de Maple de primera clase que se crea al evaluar una "definición de módulo". Dentro del módulo deben emplearse los comandos `export`,y `option`. El comando `export` permite que los procedimientos sean accesibles fuera de la definición del módulo. El comando `option` con el atributo `package` permite dar atributos al módulo, en este caso se trata de generar un paquete de Maple. Las exportaciones de paquetes se protegen automáticamente y pueden enlazarse de forma persistente en una sesión interactiva mediante el comando `with`.
4. En el último paso se guarda el nombre del módulo con el comando `savelib('MisRutinas_COL')`.

3. Desarrollo del Trabajo: *aplicando las reglas y principios científicos*

Para ilustrar cómo se crea una librería personalizada en Maple (archivo con extensión `.mla`), se desarrolló un ejemplo que incluye tanto las matrices como las raíces de los polinomios de Legendre utilizadas en el método de colocación ortogonal. En este ejemplo se implementaron dos procedimientos: `M_Col_OR`, encargado de generar las matrices de colocación ortogonal, y `P_Legendre`, que calcula las raíces de los polinomios de Legendre. Ambos procedimientos fueron agrupados en un archivo de biblioteca llamado

A_Numericos_V1_0.mla, el cual contiene un paquete denominado MisRutinas_COL. Este paquete puede activarse fácilmente en cualquier sesión de Maple usando el comando with(MisRutinas_COL); lo que permite acceder de inmediato a todas las funciones implementadas.

```
#Código 1 para Generar librerías utilizando maple 2021
restart;
with(LibraryTools); with(ArrayTools):
with(ListTools):
with(linalg):
LibLocation := cat(currentdir(), "/A_Numericos_V1_0.mla");
march('create', LibLocation);
libname :=libname, LibLocation;

MisRutinas_COL:=module()
   description "Algoritmos numéricos";
   export M_Col_OR, P_Legendre:
   option package;

   # Determina las matrices de colocación ortogonal
   M_Col_OR:=proc(xj, N::integer)
        local jj, ii,i,j,sol, operandos :
        description "Col. O. Calcula las Matrices C, D, Q, A, B, W, F";
        #La Q^(-1) se calcula afuera para  usarla

        #Declaración del espacio para las matrices
        local  A := Matrix(N+2, N+2, []):
        local  B := Matrix(N+2, N+2, []):
        local  Q := Matrix(N+2, N+2, []):
        local  C := Matrix(N+2, N+2, []):
        local DD := Matrix(N+2, N+2, []):
        local dd := Matrix(N+2, N+2, []):
        local Q_Inv,W,F,Qinv :

        #Matriz C
        for jj from 1 by 1 to N+2 do:
           for ii from 1 by 1 to N+2 do:
               if ii>1 then
                   C[jj,ii]:=(ii-1)*(xj[jj])^(ii-2):
               else
                   C[jj,ii]:=(ii-1):
               end if:
           end do:
        end do:

        #Matriz D
        for jj from 1 by 1 to N + 2 do:
           for ii from 1 by 1 to N + 2 do:
               if ii>2 then
                   DD[jj, ii] := (ii - 1)*(ii-2)*xj[jj]^(ii - 3):
               else
                   DD[jj, ii] := (ii - 1)*(ii-2):
               end if;
           end do:
        end do:

        #Matriz Q^(-1)
        for jj from 1 by 1 to N + 2 do :
           for ii from 1 by 1 to N + 2 do :
               Q[jj, ii] := xj[jj]^(ii - 1):
           end do:
        end do:
       Q_Inv:=inverse(Q):

        #Matriz A
        A := C . Q_Inv:

        #Matriz B
        B := DD . Q_Inv:

        #Obtener Matriz W
        for i from 1 by 1 to N+2 do:
           f || i := add(w[j]*xj[j]^(i-1), j = 1 .. N + 2) = 1/i:
        end do:

        sol:=solve({seq(f|| i, i = 1 .. N+2)}, {seq(w[i], i = 1 .. N+2)}):
```

```
        operandos:=seq(rhs(sol[i]), i = 1 .. N+2):
        W := Matrix(1, N + 2, [operandos]):

        #Obtener matriz F
        F := W . Q:
        return [C, DD, Q, A, B, W,F]  ;
    end proc:

    P_Legendre:=proc(n::integer)
      local jj, ii,i,j,sol, operandos,xj,N,legendre_roots :
       description "Calcula las raíces del polinomio de legendre Pn";
       P || n :=expand( 1/(2^n*n!) * diff((X^2-X)^n,[X  $n])) ;
       legendre_roots:=fsolve(P||n):
       xj:=[0,legendre_roots, 1];
       N:=numelems(xj)-2;
       return xj:
    end proc:
end module;
savelib('MisRutinas_COL');


#Código 2 para mostrar el uso de la librería MisRutinas_COL
restart:
with(MisRutinas_COL);
with(linalg):
describe(P_Legendre);
n := 2; xj:=P_Legendre(n);

Mat_C := M_Col_OR(xj, n)[1]; Mat_D := M_Col_OR(xj, n)[2]; Mat_Qinv := inverse(M_Col_OR(xj, n)[3]);
Mat_A := M_Col_OR(xj, n)[4]; Mat_B := M_Col_OR(xj, n)[5]; Mat_W := M_Col_OR(xj, n)[6]; Mat_F := M_Col_OR(xj, n)[7];
```

La Figura 1 se ha obtenido al ejecutar el código 2 para mostrar la generación de las matrices y las raíces del Polinomio de Legendre para n=2 en la metodología de colocación ortogonal utilizando la librería `MisRutinas_COL`. Nótese que el código 2 ha resultado ser compacto. En este código se ha propuesto encontrar las raíces cuando `n:=2`. Al hacer `xj:=P_Legendre(n)` obtenemos las raíces del polinomio de Legendre. Véase que el procedimiento tiene dos argumentos, el grado del polinomio y las raíces. Este procedimiento devuelve 7 matrices. Por ejemplo. para obtener la matriz F se tiene `Mat_F:=M_Col_OR(xj,n)[7]`; la Figura 1 detalla cómo se han obtenido cada una de las matrices.

4. Conclusiones: *lo que podemos aprender de este artículo*

En este artículo se presentó la metodología para crear librerías personalizadas de una forma sencilla y práctica en Maple 2021. A partir de esta metodología, se desarrollaron algoritmos que permiten gene-rar las matrices y los polinomios de Legendre necesarios para aplicar el método de colocación ortogonal, tanto en su forma clásica como en su versión adaptada a elementos finitos. Estas herramientas están pensadas para resolver ecuaciones diferenciales con condiciones en la frontera de manera eficiente.

Durante el trabajo se realizaron diversas pruebas para validar la funcionalidad de la librería, comprobando que puede generar matrices de colocación ortogonal de orden elevado,

siempre que haya suficiente memoria RAM disponible en la computadora. Entre las pruebas realizadas, se incluyó exitosamente el caso de n=100, lo cual demuestra la capacidad y versatilidad de la librería propuesta.

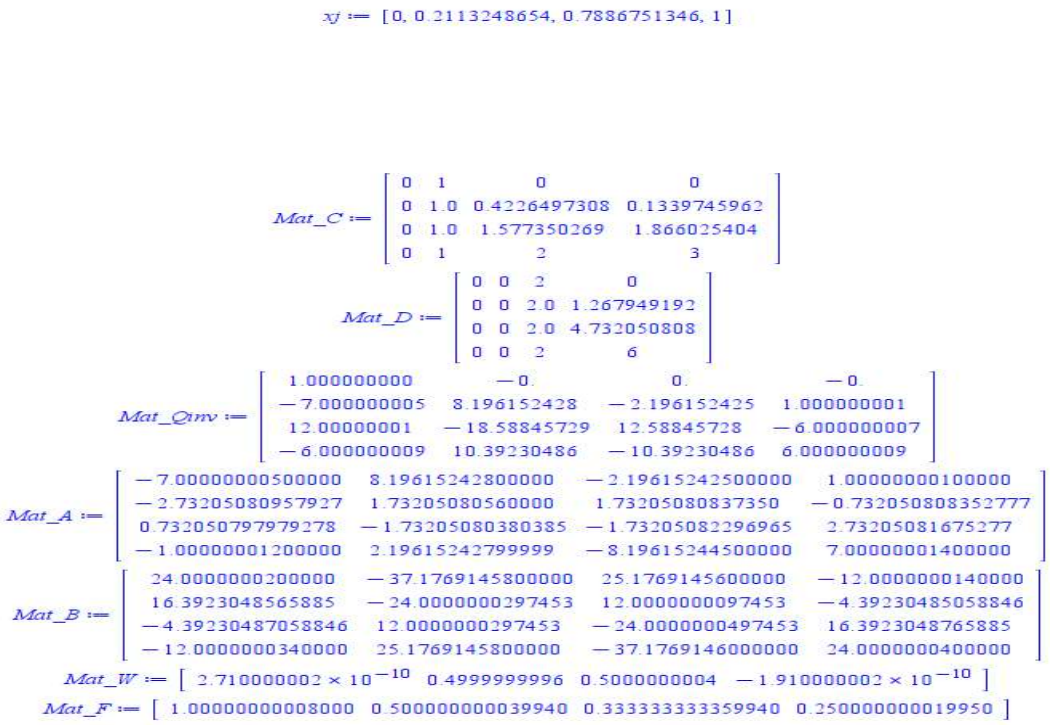

Figura 1. Demostración de la librería creada en Maple 2021.

Obtenida de: elaboración propia.

5. Referencias: *por si quieres seguir conociendo más*

Adams, J. C., Brainerd, W. S., Martin, J. T., Smith, B. T. and Wagener, J. L. (1992). *Fortran 90 handbook*. McGraw-Hill.

Díaz, L. M. C., Díaz, N. A. C. y Álvarez, J. C. (2024). *Iniciando a programar con Python: Guía básica de programación* (Vol. 87). Editorial de la Universidad Pedagógica y Tecnológica de Colombia-UPTC.

Finlayson, B. A. (1981). *Nonlinear analysis in chemical engineering*. McGraw-Hill.

Fox, W. P. (2011). *Mathematical modeling with Maple*. Cengage Learning.

Fox, W. P., and Bauldry, W. C. (2019). *Advanced problem solving with Maple: a first course*. Chapman and Hall/CRC.

Joyanes Aguilar, L. (2020). *Fundamentos de programación* (5.ª ed.). McGraw-Hill.

Maplesoft. (2023). *Maple online help*. https://www.maplesoft.com/support/help/

Villadsen, J. V. and Stewart, W. E. (1967). Solution of boundary-value problems by orthogonal collocation. *Chemical Engineering Science*, 22, 1483–1501.

Villadsen J. V. and Michelsen, M. L, (1978). *Solution of differential equation models by polynomial approximation*. Prentice-Hall International Series.