Informe I:
Trabajo Práctico 2

Cipolatti Edgardo
edgardocipolatti@hotmail.com

I. EJERCICIO 9

Considere un circuito eléctrico como el que se muestra en la Figura 1. Se pide aplicar las leyes de Kirchhoff para calcular las corrientes i_1, i_2, i_3 e i_4.

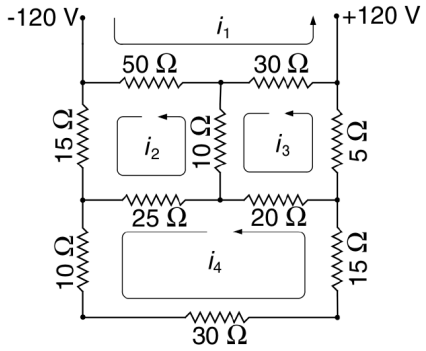

Fig. 1
CIRCUITO EJERCICIO 9.

De acuerdo al circuito planteado obtuve el siguiente sistema de ecuaciones.

$$\begin{array}{rrrrcc} 80i_1 & -50i_2 & -30i_3 & +0i_4 & = & 120 \\ -50i_1 & +100i_2 & -10i_3 & -25i_4 & = & 0 \\ -30i_1 & -10i_2 & +65i_3 & -20i_4 & = & 0 \\ +0i_1 & -25i_2 & -20i_3 & +100i_4 & = & 0 \end{array}$$

Lo obtuve sumando las resistencias siguiendo el sentido de la corriente y restando aquellas que iban en sentido contrario al tramo que consideraba para que la suma nos de un resultado de cero como expresa la segunda ley de Kirchhoff. Siguiendo la corriente i_1 sumo las resistencias de 50Ω y 30Ω pero tengo la corriente i_2 e i_3 en sentido contrario entonces resta 50Ω, y la corriente i_3 resta 30Ω. Ahora siguiendo la corriente i_2 sumo las resistencias de 15Ω, 25Ω, 10Ω y 50Ω pero tengo la corriente i_1, i_3 e i_4 en sentido contrario entonces resta 50Ω, 30Ω y 25Ω respectivamente. Tomando la corriente i_3 sumo las resistencias de 5Ω, 20Ω, 10Ω y 30Ω pero tengo la corriente i_1, i_2 e i_4 en sentido contrario entonces resta 30Ω, 10Ω y 20Ω respectivamente. Con la corriente i_4 sumo las resistencias de 25Ω, 10Ω, 30Ω, 15Ω y 20Ω pero tengo la corriente i_2 e i_3 en sentido contrario entonces resta 25Ω la corriente i_2, y la corriente i_3 resta 20Ω.

Con el sistema ya planteado lo transformo a matrices para la posterior resolución del problema con el código que se muestra a continuación de la matriz

$$\begin{bmatrix} 80 & -50 & -30 & 0 \\ -50 & 100 & -10 & -25 \\ -30 & -10 & 65 & -20 \\ 0 & -25 & -20 & 100 \end{bmatrix} \times \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} 120 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
function [x] = gauss2(A,b)
   [m,n] = size(A);
   tol = 1e-9;
   indx = [1:n];
   for k = 1 : n-1
      aux = [k:n];
   [max_pivot,indxmax] =
       max(abs(A(indx(aux),k)));

      // control de pivot
      if(max_pivot < tol)
         disp("No se puede seguir con el
             algoritmo");
         return;
      end
      // hago intercambio
      if(indx(k) ~= indx(aux(indxmax)))
         m = indx(k);
         indx(k) = indx(aux(indxmax));
         indx(aux(indxmax)) = m;
      end
      // calculo multiplicadores
      A(indx(k+1:n),k) = A(indx(k+1:n),k) /
          A(indx(k),k);
      // eliminación
      for j = k+1:n
         A(indx(k+1:n),j) =
             A(indx(k+1:n),j) -
             A(indx(k+1:n),k) * A(indx(k),j);
      end
      b(indx(k+1:n)) = b(indx(k+1:n)) -
          A(indx(k+1:n),k) * b(indx(k));
   end

   x = backsub2(A,b,indx);
endfunction
```

El algoritmo anterior usa la función backsub2 que es la encargada de hacer la sustitución hacia atrás. el código de esa función se detalla a continuación.

```
function [x] = backsub2(U,b,indx)
   [m,n] = size(U);

   U1 = zeros(n,n);
   b1 = zeros(n,1);

   for i = 1 : n
      U1(i,:) = U(indx(i),:);
      b1(i) = b(indx(i));
   end

   tol = 1e-9;
   x = zeros(n,1); // vector columna de nx1
       ceros

   if(abs(U1(n,n)) < tol) // si Unn es muy
       chico
      disp("No se puede continuar con el
```

```
        algoritmo");
      return;
   end

   x(n) = b1(n) / U1(n,n);
   for i = n-1: -1 : 1 // para i = n-1
       hasta 1 decrementando de a 1
      x(i) = (b1(i) - sum(U1(i,i+1:n) *
          x(i+1:n))) / U1(i,i);
   end
endfunction
```

Con el uso del algoritmo planteado llegamos a la resolución del ejercicio y obtenemos las corrientes i_1, i_2, i_3 e i_4. Las cuales resultan ser:

$i_1 = 4{,}1823949$
$i_2 = 2{,}6645519$
$i_3 = 2{,}7121332$
$i_4 = 1{,}2085646$

II. EJERCICIO 10

Realice la factorización LU de la siguiente matriz siguiendo el orden de Doolittle, con y sin pivoteo parcial (con lo cual, si P es distinta de la identidad, en realidad se tiene PA = LU). Luego, calcule las matrices residuales A-LU y PA-LU y **justifique las diferencias que ocurren.**

$$\begin{bmatrix} 1 & 1+0{,}5e-15 & 3 \\ 2 & 2 & 20 \\ 3 & 6 & 4 \end{bmatrix}$$

Quiero hallar la factorización LU tal que A=LU, entonces utilizando el algoritmo de Doolittle sin pivoteo parcial obtengo las matrices L y U que muestro a continuación

$$U = \begin{bmatrix} 1 & 1 & 3 \\ 0 & -8{,}882D-16 & 14 \\ 0 & 0 & 4{,}729D+16 \end{bmatrix}$$
$$L = \begin{bmatrix} c1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -3{,}378D+15 & 1 \end{bmatrix}$$

El código de Doolittle sin pivoteo ejecutado en Scilab es el siguiente.

```
function [L,U] = doolittle (A)
   //creo las matrices
   L = zeros(size(A));
   U = zeros(size(A));
   [m,n] = size(A);

   for i=1:n
     for j=1:m
   // Estamos arriba de la diagonal
       buscamos U
      if i<=j
          U(i,j) = A(i,j);
          for k=1:i-1
            U(i,j) = U(i,j) - L(i,k)*U(k,j);
          end
      end
   // Estamos abajo de la diagonal buscamos
       L
      if j<=i
         L(i,j) = A(i,j);
       for k=1:j-1
         L(i,j) = L(i,j) - L(i,k)*U(k,j);
        end
        L(i,j) = L(i,j)/U(j,j);
      end
    end
  end
endfunction
```

El resultado de hacer A-LU es la matriz residual y se espera que tenga todas sus entradas en cero. Si no son todas cero como se muestra a continuación, significa que hay errores por utilizar una representación de dígitos finitos para los números y por la aritmética de dígitos finitos empleada en las operaciones, se generan acumulaciones de errores en los redondeos y truncamientos.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Ahora utilizando Gauss con pivoteo parcial obtengo las siguientes matrices P (Permutación), L (Triangular inferior) y U (Triangular superior) que son las matrices de la factorización de Doolittle (PA=LU).

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0{,}6666667 & 1 & 0 \\ 0{,}3333333 & 0{,}5 & 1 \end{bmatrix} \quad U = \begin{bmatrix} c3 & 6 & 4 \\ 0 & -2 & 17{,}333333 \\ 0 & 0 & -7 \end{bmatrix}$$
$$P = \begin{bmatrix} c0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Estas matrices fueron obtenidas por el siguiente algoritmo.

```
      function [x,L,U,P] = Gauss(A,b)

   tol = 1e-9; // tolerancia
   [m,n] = size(A);
   P = eye(n,n);
   L = zeros(n,n);
   U = zeros(n,n);
   m = length(b);

   indx = [1:n];

      for k=1:n-1
            aux = [k:n];
            [max_pivot,indxmax] =
                max(A(indx(aux),k).*A(indx(aux),k));
            if(max_pivot < tol) then
                  disp("No hay solucion.
                      Los posibles pivots
                      son TODOS ceros.");
                  x = [];
                  return;
            end
            if(indx(k) ~=
                indx(aux(indxmax)))
                  disp("Hago cambio de
                      filas");
                  m = indx(k);
                  indx(k) =
                      indx(aux(indxmax));
                  indx(aux(indxmax)) = m;
            end
            A(indx(k+1:n),k) =
                A(indx(k+1:n),k) /
                A(indx(k),k);
            for j=k+1:n
                  A(indx(k+1:n),j) =
                      A(indx(k+1:n),j) -
                      A(indx(k+1:n),k)*A(indx(k),j);
            end
            b(indx(k+1:n)) = b(indx(k+1:n))
                - A(indx(k+1:n),k) *
```

```
            b(indx(k));
      end

      P = P(indx,:);
      for i=1:n
            U(i,i:n) = A(indx(i),i:n);
            L(indx(i),i) = 1;
            L(indx(i),1:i-1) =
                A(indx(i),1:i-1);
      end
      L = P*L;
   [x]=backsub2(A,b,indx);

endfunction
```

Ahora por la igualdad PA=LU puedo obtener el error de cálculo haciendo PA-LU, ya que dispongo de todas las matrices. El resultado de la operación antes mencionada es:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -8-882D-16 \end{bmatrix}$$

Como conclusión se deduce que el algoritmo de Gauss con pivoteo parcial es más conveniente en este caso porque el error es más pequeño. Ambos métodos deberían llegar al mismo resultado pero ya que se trabaja con pivotes muy cercanos a cero el error de cálculo se magnifica y es ahí donde se hace notar la diferencia entre un algoritmo y otro. Estos errores de cálculo se producen por falta de precisión en la representación en punto flotante de los números y la consecuente acumulación de errores de redondeo truncamiento y redondeo en las operaciones aritméticas.

REFERENCIAS

[1] R. L. Burden, *Análisis Numérico*, 7th ed. Thomson Learning, 2002.
[2] V. Sonzogni, "Cálculo numérico - apuntes de cátedra," 2014.
[3] "Leyes de kirchoff," Wikipedia, 2014. [Online]. Disponible: http://es.wikipedia.org/wiki/Leyes_de_Kirchhoff