Tarea 3: Métodos Numéricos

Rafael Alejandro García Ramírez

10 de septiembre de 2023

1. Genera un programa en C que resuelva el sistema de ecuaciones de la forma $Lx = b$, donde L es una matriz triangular inferior.

■ Proponemos el siguiente programa (llamado dentro de los archivos adjuntos como `problema1.c`). La parte fundamental del problema es la función `Lx_b` que realiza la sustitución hacia adelante:

```c
#include <stdio.h>

void Lx_b(int N, double L[N][N], double b[N] ,double x[N]);

int main(){
    // Ejemplo
    const int N = 3;

    double L[3][3] = {{3,0,0}, {2,4,0}, {1,3,5}};
    double b[3] = {2,5,8};

    double x[N];
    Lx_b(N, L, b, x);
    for (int i = 0; i < N; i++) printf("x[%d] = %f\n", i, x[i]);
    return 0;
}

void Lx_b(int N, double L[N][N], double b[N], double x[N]) {
    /* Resuelve un sistema de ecuaciones lineales triangular inferior Lx = b para x.
     *
     * @ N       El tamano de la matriz triangular inferior L y los vectores b y x.
     * @ L       La matriz triangular inferior de tamano N x N.
     * @ b       El vector de terminos independientes de tamano N.
     * @ x       El vector de solucion, que se actualizara con la solucion despues de la llamada.
     *
     * Esta funcion utiliza sustitucion hacia adelante para resolver el sistema de ecuaciones
    lineales.
     * La matriz L debe ser triangular inferior, y el vector x debe contener una estimacion
    inicial
     * de la solucion, que se actualizara con la solucion final.
     */

    double acumulador;
    for (int i = 0; i < N; i++) {
        acumulador = 0;
        for (int k = 0; k <= i; k++)
            acumulador += L[i][k] * x[k];
        x[i] = (b[i] - acumulador) / L[i][i];
    }
}
```

---

2. Genera un programa en C que resuelva el sistema de ecuaciones de la forma $Ux = b$, donde U es una matriz triangular superior.

■ Realizamos el siguiente programa (cuyo archivo anexo es llamado `problema2.c`). La parte principal de es la función `Ux_b` que realiza la sustitución hacia atrás:

```c
#include <stdio.h>

void Ux_b(int N, double U[N][N], double b[N] ,double x[N]);

int main(){
    // Ejemplo
    const int N = 3;
    double U[3][3] = {{3,1,3}, {0,4,3}, {0,0,5}};
    double b[3] = {2,5,8};

```

```c
    double x[N];
    Ux_b(N, U, b, x);
    for (int i = 0; i < N; i++) printf("x[%d] = %f\n", i, x[i]);
    return 0;
}
void Ux_b(int N, double U[N][N], double b[N] ,double x[N]){
    /* Resuelve un sistema de ecuaciones lineales triangular superior Ux = b para x.
     *
     * @ N       El tamano de la matriz triangular superior U y los vectores b y x.
     * @ U       La matriz triangular superior de tamano N x N.
     * @ b       El vector de terminos independientes de tamano N.
     * @ x       El vector de solucion, que se actualizara con la solucion despues de la llamada.
     *
     * Esta funcion utiliza sustitucion hacia atras para resolver el sistema de ecuaciones lineales.
     * La matriz U debe ser triangular superior, y el vector x debe contener una estimacion inicial
     * de la solucion, que se actualizara con la solucion final.
     */

    double acumulador;
    for(int i = N - 1; i >= 0; i--){
        acumulador = 0;
        for (int k =  N; k > i; k--) acumulador += U[i][k]*x[k];
        x[i] = (b[i] - acumulador) / U[i][i];
    }
}
```

---

3. Genera un programa en C que resuelva un sistema de ecuaciones de la forma $Ax = b$, por el método de eliminación gaussiana. Para ello debes utilizar el algoritmo de construcción de la matriz triangular superior del proceso de eliminación con $A = [a_{i,j}]$ y $b = [b_i]$ para $i, j = 1, \ldots, n$.

Algorithm 1 Algoritmo de construcción de la matríz triangular superior $n \times n$

Require: $A = [a_{i,j}], b = [b_i]$
Ensure: Matriz triangular superior
  for $k = 1, \ldots, n-1$ **do**
    for $i = k+1, \ldots, n$ **do**
      $l_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$
      for $j = k+1, \ldots, n$ **do**
        $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$
      $b_i^{(k+1)} = b_i^{(k)} - l_{ik} b_k^{(k)}$

Una vez que se tenga el sistema equivalente de la forma $Ux = b$, utiliza el código que creaste en el ejercicio 2 para encontrar la solución del sistema de ecuaciones, observa que este paso corresponde a los pasos $8 - 10$ del algoritmo presentado en clase (libro Burden).

■ Utilizado el algoritmo provisto, creamos el siguiente programa (adjunto con el nombre `problema3.c`) que primero construye la matriz triangular inferior mediante la función `construir_matriz_2triangular_superior` y después con el apoyo de la función `Ux_b` resolver este sistema:

```c
#include <stdio.h>

void construir_matriz_2triangular_superior(int N, double A[N][N], double b[N]);
void Ux_b(int N, double U[N][N], double b[N] ,double x[N]);

int main() {
    // Ejemplo
    const int N = 3;

    double A[3][3] = {{3, 1, 3}, {2, 4, 3}, {7, 3, 5}};
    double b[3] = {2, 5, 8};

    double x[N];
    printf("Matriz Original:\n");

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf("%f ", A[i][j]);
        printf("\n");
    }
```

```c

    construir_matriz_2triangular_superior(N, A, b);

    printf("Matriz triangular superior:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf("%f ", A[i][j]);
        printf("\n");
    }

    Ux_b(N, A, b, x);
    printf("Soluciones:\n");
    for (int i = 0; i < N; i++) printf("x[%d] = %f\n", i, x[i]);

    return 0;
}

void construir_matriz_2triangular_superior(int N, double A[N][N], double b[N]) {
    /* Construye una matriz A y un vector b en una forma triangular superior.
     *
     * @ N       El tamano de la matriz cuadrada A y el vector b.
     * @ A       La matriz cuadrada de tamano N x N.
     * @ b       El vector de terminos independientes de tamano N.
     *
     * Esta funcion modifica la matriz A y el vector b para llevarlos a una forma triangular superior.
     * Puede utilizarse como paso intermedio para resolver sistemas de ecuaciones lineales.
     */

    double l;
    for (int k = 0; k < N; k++) {
        for (int i = k + 1; i < N; i++) {
            l = A[i][k] / A[k][k];
            for (int j = k; j < N; j++) A[i][j] = A[i][j] - l * A[k][j];
            b[i] = b[i] - l * b[k];
        }
    }
}

void Ux_b(int N, double U[N][N], double b[N] ,double x[N]){
    /* Resuelve un sistema de ecuaciones lineales triangular superior Ux = b para x.
     *
     * @ N       El tamano de la matriz triangular superior U y los vectores b y x.
     * @ U       La matriz triangular superior de tamano N x N.
     * @ b       El vector de terminos independientes de tamano N.
     * @ x       El vector de solucion, que se actualizara con la solucion despues de la llamada.
     *
     * Esta funcion utiliza sustitucion hacia atras para resolver el sistema de ecuaciones lineales.
     * La matriz U debe ser triangular superior, y el vector x debe contener una estimacion inicial
     * de la solucion, que se actualizara con la solucion final.
     */

    double acumulador;
    for(int i = N - 1; i >= 0; i--){
        acumulador = 0;
        for (int k =  N; k > i; k--) acumulador += U[i][k]*x[k];
        x[i] = (b[i] - acumulador) / U[i][i];
    }
}
```

---

4. Genera un programa en C que resuelva un sistema de ecuaciones de la forma $Ax = b$, con pivoteo parcial. Para ello te sugiero:

   - Crear una función que intercambie 2 filas cualesquiera de una matriz.
   - Crear una función que encuentre el máximo de un vector.
   - Modificar el algoritmo del eliminación gaussiana del ejercicio 3 para hallar la solución de un sistema de ecuaciones con pivoteo parcial, ver algoritmo presentado en clase (libro burden).

■ Siguiendo las recomendaciones, creamos un preprocesador llamado `INTERCAMBIAR(a,b)` que intercambia dos filar (en general cualesquiera dos elementos de tipo `double`) y una función llamado `maximo_vector` que encuentra el máximo de un vector y regresa su índice. Después se crea la función `gauss_pivoteo` que es una modificación a la función `construir_matriz_2triangular_superior` donde primero se intercambiar las filas de la matriz buscando poner en la posición de la diagonal (a_{ii}) el elemento de mayor valor para evitar inestabilidad numérica. Después, se genera la matriz triangular superior. Todo esto para poder utilizar la solución al sistema creado en la función `Ux_b`. Este programa puede ser encuentrado dentro del archivo `problema4.c`.

```c
#include <stdio.h>

#define INTERCAMBIAR(a, b) {double temp; temp = a; a = b; b = temp;}
#define ABS(a) ((a) > 0 ? (a) : -(a))

int maximo_vector(int N, double v[N]);
void Ux_b(int N, double U[N][N], double b[N], double x[N]);
void gauss_pivoteo(int N, double A[N][N], double b[N], double x[N]);

int main() {
    // Ejemplo
    const int N = 3;

    double A[3][3] = {{3, 1, 3}, {2, 4, 3}, {7, 3, 5}};
    double b[3] = {2, 5, 8};

    double x[N];
    printf("Matriz Original:\n");

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf("%f ", A[i][j]);
        printf("\n");
    }
    printf("Vector b:\n");
    for (int i = 0; i < N; i++) printf("%f\n", b[i]);

    gauss_pivoteo(N, A, b, x);
    printf("Soluciones:\n");
    for (int i = 0; i < N; i++) printf("x[%d] = %f\n", i, x[i]);
    return 0;
}

int maximo_vector(int N, double v[N]) {
    // Encuentra el maximo de un vector de tamano N. Retorna el indice del maximo.
    int max = 0;
    for (int i = 1; i < N; i++) max = (v[i] > v[max]) ? i : max;
    return max;
}

void Ux_b(int N, double U[N][N], double b[N], double x[N]) {
    double acumulador;
    for (int i = N - 1; i >= 0; i--) {
        acumulador = 0;
        for (int k = N - 1; k > i; k--) acumulador += U[i][k] * x[k];
        x[i] = (b[i] - acumulador) / U[i][i];
    }
}

void gauss_pivoteo(int N, double A[N][N], double b[N], double x[N]) {
    /* Resuelve un sistema de ecuaciones lineales utilizando el m todo de eliminaci n de Gauss con pivoteo parcial.
     *
     * @ N       El tamano de la matriz cuadrada A, el vector b y el vector de solucion x.
     * @ A       La matriz cuadrada de coeficientes de tamano N x N.
     * @ b       El vector de terminos independientes de tamano N.
     * @ x       El vector de solucion, que se actualizara con la solucion despues de la llamada.
     *
     * Esta funcion resuelve el sistema de ecuaciones lineales Ax = b utilizando eliminacion de Gauss
     * con pivoteo parcial para mejorar la estabilidad numerica.
     */

    double l; int maximo;
    for (int k = 0; k < N - 1; k++) {
        // Intercambia filas para llevar el elemento de mayor magnitud a la diagonal
        maximo = maximo_vector(N - k, &A[k][k]) + k;
        if (maximo != k) {
            INTERCAMBIAR(b[k], b[maximo]);
            for (int j = k; j < N; j++) INTERCAMBIAR(A[k][j], A[maximo][j]);
        }
        // Si el elemento de la diagonal es cero, el sistema no tiene solucion unica
        if (ABS(A[k][k]) < 1e-16) {
            printf("El sistema no tiene solucion unica\n");
            return;
```

```c
        }
        // Eliminacion de Gauss
        for (int i = k + 1; i < N; i++) {
            l = A[i][k] / A[k][k];
            for (int j = k + 1; j < N; j++) A[i][j] = A[i][j] - l * A[k][j];
            b[i] = b[i] - l * b[k];
        }
    }
    Ux_b(N, A, b, x);
}
```

---

5. Con ayuda de los algoritmos creados hallar la solución del sistema $Ax = b$, según corresponda , para

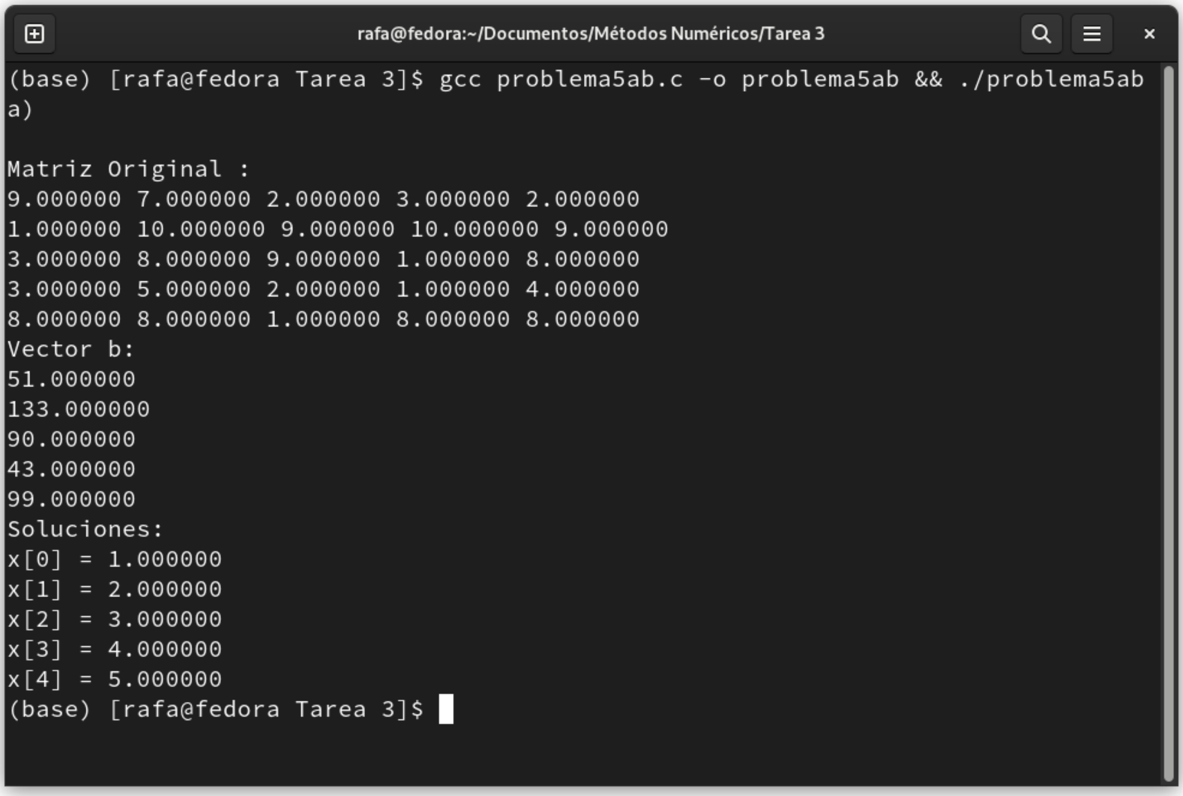
a) $A = \begin{bmatrix} 9 & 7 & 2 & 3 & 2 \\ 1 & 10 & 9 & 10 & 9 \\ 3 & 8 & 9 & 1 & 8 \\ 3 & 5 & 2 & 1 & 4 \\ 8 & 8 & 1 & 8 & 8 \end{bmatrix}, b = \begin{bmatrix} 51 \\ 133 \\ 90 \\ 43 \\ 99 \end{bmatrix}$

b) $A = \begin{bmatrix} 2 & 20 & 18 & 20 & 18 \\ 6 & 16 & 18 & 2 & 16 \\ 18 & 14 & 4 & 6 & 4 \\ 16 & 16 & 2 & 16 & 16 \\ 6 & 10 & 4 & 2 & 8 \end{bmatrix}, b = \begin{bmatrix} 532 \\ 360 \\ 204 \\ 396 \\ 172 \end{bmatrix}$

c) $A = \begin{bmatrix} 1.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.11111 & 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.33333 & 0.61446 & 1.00000 & 0.00000 & 0.00000 \\ 0.33333 & 0.28916 & 0.40984 & 1.00000 & 0.00000 \\ 0.88889 & 0.19277 & 0.84016 & 0.29075 & 1.00000 \end{bmatrix}, b = \begin{bmatrix} 51 \\ 133 \\ 90 \\ 43 \\ 99 \end{bmatrix}$

d) $A = \begin{bmatrix} 9.00000 & 7.00000 & 2.00000 & 3.00000 & 2.00000 \\ 0.00000 & 9.22222 & 8.77778 & 9.66667 & 8.77778 \\ 0.00000 & 0.00000 & 2.93976 & 5.93976 & 1.93976 \\ 0.00000 & 0.00000 & 0.00000 & 5.22951 & 1.59016 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 & 5.69749 \end{bmatrix}, b = \begin{bmatrix} 51.0000 \\ 127.3333 \\ 5.2410 \\ 12.9672 \\ 28.4875 \end{bmatrix}$

Utilizando el programa desarrollado en el problema 4 (archivo llamado `problema4.c`) para el método de gauss con pivoteo podemos resolver el problema del inciso **a** debido a la forma de la matriz A. Tenemos como resultado



```
rafa@fedora:~/Documentos/Métodos Numéricos/Tarea 3
(base) [rafa@fedora Tarea 3]$ gcc problema5ab.c -o problema5ab && ./problema5ab
a)

Matriz Original :
9.000000 7.000000 2.000000 3.000000 2.000000
1.000000 10.000000 9.000000 10.000000 9.000000
3.000000 8.000000 9.000000 1.000000 8.000000
3.000000 5.000000 2.000000 1.000000 4.000000
8.000000 8.000000 1.000000 8.000000 8.000000
Vector b:
51.000000
133.000000
90.000000
43.000000
99.000000
Soluciones:
x[0] = 1.000000
x[1] = 2.000000
x[2] = 3.000000
x[3] = 4.000000
x[4] = 5.000000
(base) [rafa@fedora Tarea 3]$
```

Por lo tanto la solución es el vector $\boxed{\vec{x} = [1, 2, 3, 4, 5]^T}$. Para el inciso **b** utilizaremos, dada la forma de la matriz A, también el mismo método y programa tal que

```
rafa@fedora:~/Documentos/Métodos Numéricos/Tarea 3
(base) [rafa@fedora Tarea 3]$ gcc problema5ab.c -o problema5ab && ./problema5ab
b)

Matriz Original :
2.000000 20.000000 18.000000 20.000000 18.000000
6.000000 16.000000 18.000000 2.000000 16.000000
18.000000 14.000000 4.000000 6.000000 4.000000
16.000000 16.000000 2.000000 16.000000 16.000000
6.000000 10.000000 4.000000 2.000000 8.000000
Vector b:
532.000000
360.000000
204.000000
396.000000
172.000000
Soluciones:
x[0] = 2.000000
x[1] = 4.000000
x[2] = 6.000000
x[3] = 8.000000
x[4] = 10.000000
(base) [rafa@fedora Tarea 3]$
```

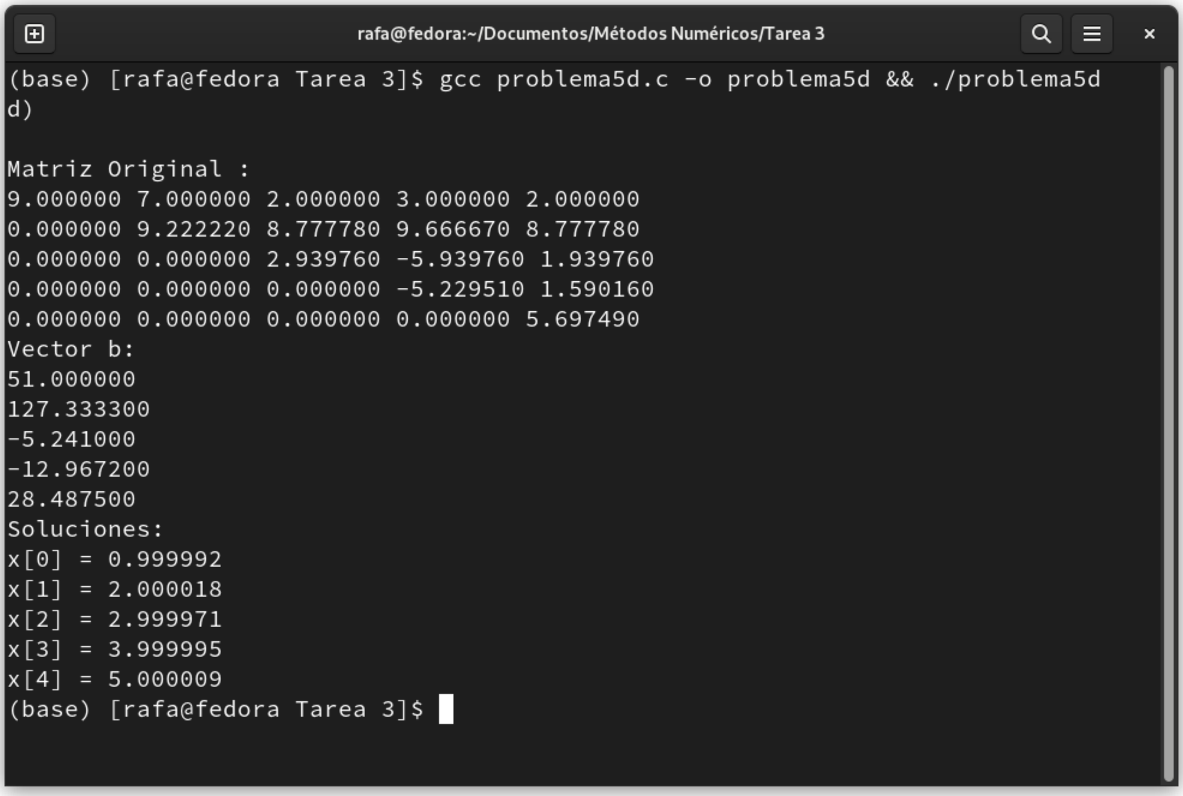
Obtenemos una solución de $\boxed{\vec{x} = [2, 4, 6, 8, 10]^T}$. Para el inciso **c** utilizaremos el programa desarrollado en el problema 1 (llamado `problema1.c`) el cual arroja los siguiente resultados

```
rafa@fedora:~/Documentos/Métodos Numéricos/Tarea 3
(base) [rafa@fedora Tarea 3]$ gcc problema5c.c -o problema5c && ./problema5c
c)

Matriz Original :
1.000000 0.000000 0.000000 0.000000 0.000000
0.111110 1.000000 0.000000 0.000000 0.000000
0.333330 0.614460 1.000000 0.000000 0.000000
0.333330 0.289160 -0.409840 1.000000 0.000000
0.888890 0.192770 -0.840160 0.290750 1.000000
Vector b:
51.000000
133.000000
90.000000
43.000000
99.000000
Soluciones:
x[0] = 51.000000
x[1] = 127.333390
x[2] = -5.241105
x[3] = -12.967567
x[4] = 28.487506
(base) [rafa@fedora Tarea 3]$
```

Tenemos entonces una solución $\boxed{\vec{x} = [51, 127.33339, -5.241105, -12.967567, 28.487506]^T}$. Finalmente para el inciso

d utilizaremos el programa creado en el problema 2 (llamado `problema2.c`) con lo siguientes resultados


```
rafa@fedora:~/Documentos/Métodos Numéricos/Tarea 3
(base) [rafa@fedora Tarea 3]$ gcc problema5d.c -o problema5d && ./problema5d
d)

Matriz Original :
9.000000 7.000000 2.000000 3.000000 2.000000
0.000000 9.222220 8.777780 9.666670 8.777780
0.000000 0.000000 2.939760 -5.939760 1.939760
0.000000 0.000000 0.000000 -5.229510 1.590160
0.000000 0.000000 0.000000 0.000000 5.697490
Vector b:
51.000000
127.333300
-5.241000
-12.967200
28.487500
Soluciones:
x[0] = 0.999992
x[1] = 2.000018
x[2] = 2.999971
x[3] = 3.999995
x[4] = 5.000009
(base) [rafa@fedora Tarea 3]$
```

Así, la solución para este sistema de ecuaciones es $\boxed{\vec{x} = [0.999992, 2.000018, 2.999971, 3.999995, 5.000009]^T}$.

Sobre el uso de los programas

La creación de los códigos en esta tarea se realizaron con un formato de lectura sencillo, sin embargo se puede modificar las respectivas funciones `int main()` en cada uno para leer un archivo con los datos respectivos o generar una entrada con un `scanf`, así mismo se necesitaría optimizar la función `INTERCAMBIAR(a,b)` y `ABS(a)` para evitar desperdicio de memoria. Sin embargo, estos cambios son relativamente sencillos y se tomó la decisión de dejar como tal por propósitos del problema.