** UNIVERSIDAD DE BUENOS AIRES  
 FACULTAD DE INGENIERÍA  
 <75.12> ANÁLISIS NUMÉRICO**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATOS DEL TRABAJO PRÁCTICO** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 1 | | | | | | | 2 | | 0 | | | 1 | | 7 | | | Cálculo de rentabilidad | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AÑO | | | | | | | | | | de proyecto de | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | inversión energía solar | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TP NRO | | | | | | | CUAT | | | | | | | | | | TEMA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **INTEGRANTES DEL GRUPO** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 29 | | | | | | | César Ezequiel Herrera | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 97429 | | | | | | | | | |
| GRUPO | | | | | | | APELLIDO Y NOMBRE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | PADRÓN | | | | | | | | | |
| **DATOS DE LA ENTREGA** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | | |  | |  | | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| C | T | 2 | | 9 | | . | | T | | X | | | T | | | 5 | | 4 | | - | | 6 | | 7 | | 6 | | 2 | | 0 | | 1 | | | 3 | | - | 1 | | 0 | | - | | 1 | | 7 | | 1 | | 3 | | - | | 1 | | 0 | | - | | 1 | | 7 |
| ARCHIVO | | | | | | | | | | | | | | | | NRO CONTROL | | | | | | | | | | | | | | | | FECHA VENC | | | | | | | | | | | | | | | | FECHA ENTR | | | | | | | | | | | | | | |
| **CORRECCIONES** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | | |  | |  |  | | | |  | |  | |  | |  | |  | |  | |  | |  | | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| FECHA | | | | | | | | | | | NOTA | | | | | | | | | | | | | OBSERVACIONES | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | |  | | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | |  | | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DOCENTE | | | | | | | | | | | FIRMA | | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | |  | | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

# Introducción

El objetivo del TP consiste en poder encontrar la Tasa Interna de Retorno de un proyecto de energía solar mediante el uso de varios métodos numéricos.

La Tasa Interna de Retorno es la tasa de interés para la cual el flujo de fondos actualizado (VAN) es igual a la inversión.

Con:

* Io = Inversión Inicial.
* N = Vida útil del proyecto.
* FCFn = Flujo de caja del proyecto en el año n.

Con:

* + Ahorrosn = Ahorros de electricidad más ahorros de potencia.
  + Costosn = Costos de mantenimiento.
  + α = Impuesto a la ganancia.

# Desarrollo del trabajo

## Orden de convergencia

ρ se puede aproximar por:

* Bisección:
* Punto fijo (NR):
* Secante:

## Distintos escenarios

Tabla que muestra las distintas TIRs que se obtienen con el método de la secante si se cambian los datos.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inversión | FCF | TIR |
| Control | -945000 | 71218.45 | 0.04 ± 0.01 |
| a) CU \* 0.7 | -661500 | 71218.45 | 0.09 ± 0.01 |
| b) α = 0 | -945000 | 109381 | 0.10 ± 0.01 |
| c) CELEC \* 2 | -945000 | 127895.3 | 0.12 ± 0.01 |
| d) FU \* 2 | -945000 | 71218.45 | 0.6 ± 0.01 |
| e) α = 0 ⬄ N <= 5 | -945000 | N <= 5 → 109381  N > 5 → 71218.45 | 0.7 ± 0.01 |

# Conclusiones

Al ser un modelo muy simplificado no se puede sacar conclusiones acerca de la rentabilidad del proyecto, sin embargo es muy útil para comparar distintos métodos numéricos en forma académica.

El método que logro encontrar una TIR con un error menor al 1% fue el de Newton-Raphson (caso particular de punto fijo), solo necesitando 4 iteraciones. Elegí usar NR porque la derivada en la raíz es mucho mayor a 1, lo que impedía usar a g(x) = x – f(x).

Finalmente, si quisiera obtener una menor propagación de errores, necesitaría cambiar la formula VAN(i), por ejemplo, fijando FCF y sacándolo de la sumatoria.

# Anexos

Código fuente (con pruebas unitarias) disponible en: <https://github.com/keze87/CT29>

## CT29.c

|  |
| --- |
| // César Ezequiel Herrera 97429  // Numero de grupo: 29  //  // Repo con pruebas unitarias:  // https://github.com/keze87/CT29  //  // Make:  // gcc -lm -std=c11 -Wall -pedantic -pedantic-errors -o CT29 CT29.c -I.  //#include "tests.h" // Pruebas Unitarias  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <math.h>  #define NUMERODEPADRON 97429  #define TRUE 0  #define FALSE 1  #define MAXITERACIONES 32000  #define FRACASO -300000  #define N 20  #define h 0.000001  #define MINDIVISOR 0.0000000001  #define TAMMATRIZ 6  #define MESES 12  #define HORASPORANIO 8760  #define FACTORREDUCCIONPOTENCIA 0.3  #define DOLARAPESO 17.5  struct vectorDatos **{**  // Inversión inicial  int potencia**;**  double costoUnitarioPotencia**;**  // FCFn  int costos**;**  float ganancias**[**N**+**1**];**  // Ahorros  double factorUso**;**  float costoElec**;**  int costoPot**;**  **};**  void imprimirEnunciado **(**short enunciado**)** **{**  **switch** **(**enunciado**)** **{**  **case** 1**:**  printf**(**"\n1) Calcular la inversión requerida, los ahorros y el resultado del flujo de fondos para cada año.\n"**);**  printf**(**" Mostrar los valores en una tabla:\n\n"**);**  **break;**  **case** 2**:**  printf**(**"\n2) Aplicar el método de bisección para calcular la TIR del proyecto.\n"**);**  printf**(**" Para ello, encuentre un intervalo que contenga a la raíz buscada e itere hasta lograr una precisión de 1%%.\n"**);**  printf**(**" Exprese el resultado correctamente.\n\n"**);**  **break;**  **case** 3**:**  printf**(**"\n3) Aplicar el método de punto fijo para calcular la TIR del proyecto utilizando la función de iteración g(x) = x - f(x).\n"**);**  printf**(**" Para ello, utilice como semilla, un valor similar al encontrado mediante bisección.\n"**);**  printf**(**" Exprese el resultado correctamente.\n\n"**);**  **break;**  **case** 4**:**  printf**(**"\n4) Evalúe experimentalmente las condiciones de convergencia del método anterior mediante pruebas con diferentes semillas.\n"**);**  printf**(**" Reduzca el valor de la semilla hasta encontrar el límite inferior del intervalo de convergencia. \n"**);**  printf**(**" Luego aumente el valor de la semilla hasta encontrar el límite superior del intervalo de convergencia.\n"**);**  printf**(**" Muestre el intervalo de convergencia obtenido.\n\n"**);**  **break;**  **case** 5**:**  printf**(**"\n5) Repita los puntos 3) y 4) aplicando el método de la secante.\n\n"**);**  **break;**  **case** 7**:**  printf**(**"\n7) Calcule la TIR del proyecto utilizando el método de la secante para los siguientes escenarios:\n"**);**  printf**(**" a) El costo unitario de la instalación fotovoltaica se reduce un 30%%.\n"**);**  printf**(**" b) Los proyectos solares quedan exentos del pago del impuesto a las ganancias (α=0).\n"**);**  printf**(**" c) El costo de la electricidad se duplica.\n"**);**  printf**(**" d) La constante del Factor de Uso aumenta a 0,2 (mejora la eficiencia de los paneles)\n"**);**  printf**(**" e) Los proyectos solares quedan exentos del pago del impuesto a las ganancias por los primeros 5 años.\n\n"**);**  **break;**  **default** **:**  printf**(**"Error\n"**);**  **}**  **}**  /\*  \*  \* Cálculos auxiliares  \*  \*/  // Devuelve un string del número redondeado  char **\*** redondear **(**double numero**)** **{**  char **\*** aux **=** malloc**(sizeof(**char**)** **\*** 31**);**  char **\*** auxInteger **=** malloc**(sizeof(**char**)** **\*** 31**);**  char auxModulo**[**30**];**  long entero **=** round**(**numero**);**  snprintf**(**auxInteger**,** 30**,** "%li"**,** entero**);**  snprintf**(**auxModulo**,** 30**,** "%li"**,** labs**(**entero**));**  // Si es mayor o igual (en modulo) a 100 devuelvo solo la parte entera  // Sino devuelvo un decimal o 2  **if** **(**strlen**(**auxModulo**)** **>=** 3**)** **{**  free**(**aux**);**  **return** auxInteger**;**  **}** **else** **if** **(**strlen**(**auxModulo**)** **>=** 2**)** **{**  snprintf**(**aux**,** 30**,** "%.1f"**,** numero**);**  **}** **else** **{**  snprintf**(**aux**,** 30**,** "%.2f"**,** numero**);**  **}**  free**(**auxInteger**);**  **return** aux**;**  **}**  double potencia **(**double x**,** int n**)** **{**  **if** **(**n **<** 0**)**  **return** FRACASO**;**  double aux **=** 1**;**  **for** **(**int i **=** 1**;** i **<=** n**;** i**++)** **{**  aux **=** aux **\*** x**;**  **}**  **return** aux**;**  **}**  // Devuelve el error relativo  double error **(**double Xk1**,** double Xk**)** **{**  **return** fabs**(**Xk1 **-** Xk**)** **/** fabs**(**Xk1**)** **\*** 100**;**  **}**  // PRE: decimales después del punto  char **\*** incerteza **(**char **\*** raiz**)** **{**  char **\*** punteroADecimalDespuesDePunto **=** **NULL;**  char **\*** retorno **=** malloc**(sizeof(**char**)** **\*** 30**);**  int cantidadDeDecimales**;**  // el puntero es igual a la posición de memoria del caracter después del punto  // "123.456" -> "456"  **for** **(**int i **=** 0**;** i **<** strlen**(**raiz**);** i**++)** **{**  **if** **(**raiz**[**i**]** **==** '.'**)** **{**  punteroADecimalDespuesDePunto **=** raiz **+** i **+** 1**;**  **break;**  **}**  **}**  **if** **(**punteroADecimalDespuesDePunto **==** **NULL)** **{**  strcpy**(**retorno**,** "1"**);**  **return** retorno**;**  **}**  cantidadDeDecimales **=** strlen**(**punteroADecimalDespuesDePunto**);**  strcpy**(**retorno**,** "0."**);**  **for** **(**int j **=** 1**;** j **<** cantidadDeDecimales**;** j**++)**  strcat**(**retorno**,**"0"**);**  strcat**(**retorno**,**"1"**);**  **return** retorno**;**  **}**  /\*  \*  \* Imprimir a pantalla  \*  \*/  // Llena una matriz (para imprimir) con los títulos y los números redondeados  void cargarMatrizRedondada **(**char **\*** matriz**[**TAMMATRIZ**][**TAMMATRIZ**],** double matrizDatos**[**5**][**N**+**1**])** **{**  // Pido memoria para los títulos  matriz**[**0**][**0**]** **=** malloc**(sizeof(**char**)** **\*** 2**);**  **for** **(**int i **=** 1**;** i **<** TAMMATRIZ**;** i**++)** **{**  matriz**[**i**][**0**]** **=** malloc**(sizeof(**char**)** **\*** 22**);**  matriz**[**i**][**4**]** **=** malloc**(sizeof(**char**)** **\*** 4**);**  matriz**[**0**][**i**]** **=** malloc**(sizeof(**char**)** **\*** 22**);**  **}**  // Escribo los títulos  strcpy**(**matriz**[**0**][**0**],** " "**);** strcpy**(**matriz**[**0**][**1**],** "Anio 0"**);** strcpy**(**matriz**[**0**][**2**],** "Anio 1"**);**  strcpy**(**matriz**[**0**][**3**],** "Anio 2"**);** strcpy**(**matriz**[**0**][**4**],** "..."**);** strcpy**(**matriz**[**0**][**5**],** "Anio N"**);**  strcpy**(**matriz**[**1**][**0**],** "Inversion"**);** strcpy**(**matriz**[**1**][**4**],** "..."**);**  strcpy**(**matriz**[**2**][**0**],** "Ahorro electricidad"**);** strcpy**(**matriz**[**2**][**4**],** "..."**);**  strcpy**(**matriz**[**3**][**0**],** "Ahorro potencia"**);** strcpy**(**matriz**[**3**][**4**],** "..."**);**  strcpy**(**matriz**[**4**][**0**],** "Costos"**);** strcpy**(**matriz**[**4**][**4**],** "..."**);**  strcpy**(**matriz**[**5**][**0**],** "FCF"**);** strcpy**(**matriz**[**5**][**4**],** "..."**);**  // Agrego los datos  **for** **(**int i **=** 1**;** i **<=** 3**;** i**++)** **{**  // Inversiones  matriz**[**1**][**i**]** **=** redondear**(**matrizDatos**[**0**][**i **-** 1**]);** strcat**(**matriz**[**1**][**i**],** " $"**);**  // Ahorros E  matriz**[**2**][**i**]** **=** redondear**(**matrizDatos**[**1**][**i **-** 1**]);** strcat**(**matriz**[**2**][**i**],** " $"**);**  // Ahorros P  matriz**[**3**][**i**]** **=** redondear**(**matrizDatos**[**2**][**i **-** 1**]);** strcat**(**matriz**[**3**][**i**],** " $"**);**  // Costos  matriz**[**4**][**i**]** **=** redondear**(**matrizDatos**[**3**][**i **-** 1**]);** strcat**(**matriz**[**4**][**i**],** " $"**);**  // FCFs  matriz**[**5**][**i**]** **=** redondear**(**matrizDatos**[**4**][**i **-** 1**]);** strcat**(**matriz**[**5**][**i**],** " $"**);**  **}**  // Año N  // Inversiones  matriz**[**1**][**5**]** **=** redondear**(**matrizDatos**[**0**][**N**]);** strcat**(**matriz**[**1**][**5**],** " $"**);**  // Ahorros E  matriz**[**2**][**5**]** **=** redondear**(**matrizDatos**[**1**][**N**]);** strcat**(**matriz**[**2**][**5**],** " $"**);**  // Ahorros P  matriz**[**3**][**5**]** **=** redondear**(**matrizDatos**[**2**][**N**]);** strcat**(**matriz**[**3**][**5**],** " $"**);**  // Costos  matriz**[**4**][**5**]** **=** redondear**(**matrizDatos**[**3**][**N**]);** strcat**(**matriz**[**4**][**5**],** " $"**);**  // FCFs  matriz**[**5**][**5**]** **=** redondear**(**matrizDatos**[**4**][**N**]);** strcat**(**matriz**[**5**][**5**],** " $"**);**  **}**  void liberarMemoriaMatriz **(**char **\*** matriz**[**TAMMATRIZ**][**TAMMATRIZ**])** **{**  **for** **(**int i **=** 0**;** i **<** TAMMATRIZ**;** i**++)**  **for** **(**int j **=** 0**;** j **<** TAMMATRIZ**;** j**++)**  free**(**matriz**[**i**][**j**]);**  **}**  // Busca el largo del elemento mas largo de cada columna  void calcularAnchoColumnas **(**int anchos**[**TAMMATRIZ**],** char **\*** matriz**[**TAMMATRIZ**][**TAMMATRIZ**])** **{**  **for** **(**int i **=** 0**;** i **<** TAMMATRIZ**;** i**++)**  anchos**[**i**]** **=** 0**;**  **for** **(**int i **=** 0**;** i **<** TAMMATRIZ**;** i**++)** **{**  **for** **(**int j **=** 0**;** j **<** TAMMATRIZ**;** j**++)** **{**  **if** **(**strlen**(**matriz**[**j**][**i**])** **>** anchos**[**i**])**  anchos**[**i**]** **=** strlen**(**matriz**[**j**][**i**]);**  **}**  **}**  **}**  // Imprime línea entre filas  void imprimirLineaSeparadora **(**int anchos**[**TAMMATRIZ**])** **{**  int i **=** 0**;**  int anchoTotal **=** 0**;**  **for** **(**int j **=** 0**;** j **<** TAMMATRIZ**;** j**++)**  anchoTotal **=** anchoTotal **+** anchos**[**j**];**  anchoTotal **=** anchoTotal **+** 5 **\*** 3**;** // Tamaño separador  printf**(**"\n"**);**  **while** **(**i **<=** anchoTotal**)** **{**  printf**(**"-"**);**  i**++;**  **}**  printf**(**"\n"**);**  **}**  // Imprime espacios y un separador  void imprimirSeparador **(**int anchoElemento**,** int anchoColumna**)** **{**  int aux **=** anchoElemento**;**  **while** **(**aux **<** anchoColumna**)** **{**  printf**(**" "**);**  aux**++;**  **}**  printf**(**" | "**);**  **}**  // Imprime matriz de strings agregando separadores  void imprimirMatriz **(**char **\*** matriz**[**TAMMATRIZ**][**TAMMATRIZ**])** **{**  int anchos**[**TAMMATRIZ**];**  calcularAnchoColumnas**(**anchos**,** matriz**);**  **for** **(**int i **=** 0**;** i **<** TAMMATRIZ**;** i**++)** **{**  **for** **(**int j **=** 0**;** j **<** TAMMATRIZ**;** j**++)** **{**  printf**(**"%s"**,** matriz**[**i**][**j**]);**  **if** **(**j **!=** TAMMATRIZ **-** 1**)**  imprimirSeparador**(**strlen**(**matriz**[**i**][**j**]),** anchos**[**j**]);**  **}**  **if** **(**i **!=** TAMMATRIZ **-** 1**)**  imprimirLineaSeparadora**(**anchos**);**  **}**  printf**(**"\n"**);**  **}**  // Calcula datos y manda a imprimir  int imprimirTabla **(**double matriz**[**5**][**N**+**1**])** **{**  char **\*** matrizRedondeada**[**TAMMATRIZ**][**TAMMATRIZ**];**  cargarMatrizRedondada**(**matrizRedondeada**,** matriz**);**  imprimirMatriz**(**matrizRedondeada**);**  liberarMemoriaMatriz**(**matrizRedondeada**);**  **return** TRUE**;**  **}**  void imprimirRaiz **(**double raiz**,** char **\*** metodo**)** **{**  //printf("%.16F\n", raiz);  **if** **(**raiz **!=** FRACASO**)** **{**  printf**(**"Raíz por %s: "**,** metodo**);**  char **\*** auxRaiz **=** redondear**(**raiz**);**  char **\*** auxIncerteza **=** incerteza**(**auxRaiz**);**  printf**(**"%s +/- %s\n"**,** auxRaiz**,** auxIncerteza**);**  free**(**auxRaiz**);** free**(**auxIncerteza**);**  **}** **else** **{**  printf**(**"No se puede resolver por %s.\n"**,** metodo**);**  **}**  **}**  /\*  \*  \* Cálculos de datos  \*  \*/  struct vectorDatos cargarDatos **()** **{**  struct vectorDatos aux**;**  aux**.**potencia **=** 30**;**  aux**.**costoUnitarioPotencia **=** 1800**;**  aux**.**costos **=** 5000**;**  aux**.**factorUso **=** 0.18 **\*** NUMERODEPADRON **/** 100000**;**  aux**.**costoElec **=** 1.9**;**  aux**.**costoPot **=** 250**;**  **for** **(**int i **=** 0**;** i **<=** N**;** i**++)** **{**  aux**.**ganancias**[**i**]** **=** 0.35**;**  **}**  **return** aux**;**  **}**  double ahorroElectricidad **(**int potencia**,** double factorUso**,** float costoElec**)** **{**  **return** **(**potencia **\*** HORASPORANIO **\*** factorUso **\*** costoElec**);**  **}**  double ahorroPotencia **(**int potencia**,** int costoPot**)** **{**  **return** **(**potencia **\*** FACTORREDUCCIONPOTENCIA **\*** costoPot **\*** MESES**);**  **}**  double fcf **(**double ahorroElectricidad**,** double ahorroPotencia**,** int costos**,** float ganancias**)** **{**  **return** **(**ahorroElectricidad **+** ahorroPotencia **-** costos**)** **\*** **(**1 **-** ganancias**);**  **}**  // Llena la matriz con los datos  void cargarMatriz **(**double matriz**[**5**][**N**+**1**],** struct vectorDatos datos**)** **{**  // Inversión inicial  matriz**[**0**][**0**]** **=** **-** datos**.**potencia **\*** datos**.**costoUnitarioPotencia **\*** DOLARAPESO**;**  // En los demás años no hay inversión  **for** **(**int i **=** 1**;** i **<=** N**;** i**++)**  matriz**[**0**][**i**]** **=** 0**;**  **for** **(**int i **=** 0**;** i **<=** N**;** i**++)** **{**  matriz**[**1**][**i**]** **=** ahorroElectricidad**(**datos**.**potencia**,** datos**.**factorUso**,** datos**.**costoElec**);**  matriz**[**2**][**i**]** **=** ahorroPotencia**(**datos**.**potencia**,** datos**.**costoPot**);**  matriz**[**3**][**i**]** **=** datos**.**costos**;**  matriz**[**4**][**i**]** **=** fcf**(**matriz**[**1**][**i**],** matriz**[**2**][**i**],** matriz**[**3**][**i**],** datos**.**ganancias**[**i**]);**  **}**  **}**  double sumatoriaVan **(**double i**,** double arrayFCF**[**N**+**1**])** **{**  double aux **=** 0**;**  **for** **(**int n **=** 1**;** n **<=** N**;** n**++)** **{**  aux **=** aux **+** arrayFCF**[**n**]** **/** potencia**(**1 **+** i**,** n**);**  **}**  **return** aux**;**  **}**  // Io+Sum[FCFn/(i+1)^n, {n, 20}]  double van **(**double i**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  **if** **(**i **!=** **-**1**)**  **return** inversion **+** sumatoriaVan**(**i**,** arrayFCF**);**  **return** FRACASO**;**  **}**  // (f(x + h) - f(x)) / h, h -> 0  double vanDerivada **(**double i**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  **if** **(**i **!=** **-**1**)**  **return** **(**van**(**i **+** h**,** inversion**,** arrayFCF**)** **-** van**(**i**,** inversion**,** arrayFCF**))** **/** h**;**  **return** FRACASO**;**  **}**  /\*  \*  \* Métodos numéricos  \*  \*/  double biseccion **(**int inversion**,** double arrayFCF**[**N**+**1**],** double intervaloMin**,** double intervaloMax**)** **{**  int i **=** 1**;**  double puntoMedio **=** intervaloMin**;**  **if** **(** **!** **((**intervaloMin **<** intervaloMax**)** **&&** \  **(**van**(**intervaloMin**,** inversion**,** arrayFCF**)** **\*** van**(**intervaloMax**,** inversion**,** arrayFCF**)** **<** 0**))** **)** **{**  **return** FRACASO**;**  **}**  **while** **(**i **<** MAXITERACIONES**)** **{**  double puntoMedioAnterior **=** puntoMedio**;**  puntoMedio **=** **(**intervaloMin **+** intervaloMax**)** **/** 2**;**  **if** **((**van**(**puntoMedio**,** inversion**,** arrayFCF**)** **==** 0**)** **||** **(**error**(**puntoMedio**,** puntoMedioAnterior**)** **<** 1 /\* % \*/**))** **{**  **break;** // Encontré solución  **}**  **if** **(**van**(**intervaloMin**,** inversion**,** arrayFCF**)** **\*** van**(**puntoMedio**,** inversion**,** arrayFCF**)** **<** 0**)** **{**  intervaloMax **=** puntoMedio**;**  **}** **else** **{**  intervaloMin **=** puntoMedio**;**  **}**  //printf("b %d -> %F\n", i, puntoMedio); //TODO  i**++;**  **}**  **return** puntoMedio**;**  **}**  // g(x) = f(x) / f'(x)  double puntoFijo **(**int inversion**,** double arrayFCF**[**N**+**1**],** double semilla**)** **{**  int i **=** 1**;**  double XiMas1**;**  double Xi **=** semilla**;**  **while** **(**i **<** MAXITERACIONES**)** **{**  double resultadoDerivada **=** vanDerivada**(**Xi**,** inversion**,** arrayFCF**);**  **if** **(**fabs**(**resultadoDerivada**)** **<** MINDIVISOR**)** **{**  **return** FRACASO**;**  **}**  double fXi **=** van**(**Xi**,** inversion**,** arrayFCF**);**  XiMas1 **=** Xi **-** fXi **/** resultadoDerivada**;**  **if** **(**fabs**(**XiMas1 **-** Xi**)** **>** 1**)** **{**  **return** FRACASO**;**  **}** **else** **if** **(**error**(**XiMas1**,** Xi**)** **<** 1 /\* % \*/**)** **{**  **break;**  **}**  Xi **=** XiMas1**;**  //printf("p %d -> %F\n", i, XiMas1); //TODO  i**++;**  **}**  **return** XiMas1**;**  **}**  double secante **(**int inversion**,** double arrayFCF**[**N**+**1**],** double intervaloMin**,** double intervaloMax**)** **{**  int i **=** 1**;**  double XiMas1**;**  double Xi **=** intervaloMax**;**  double XiMenos1 **=** intervaloMin**;**  **while** **(**i **<** MAXITERACIONES**)** **{**  double fXi **=** van**(**Xi**,** inversion**,** arrayFCF**);**  double fXiMenos1 **=** van**(**XiMenos1**,** inversion**,** arrayFCF**);**  XiMas1 **=** Xi **-** **(**Xi **-** XiMenos1**)** **\*** fXi **/** **(**fXi **-** fXiMenos1**);**  **if** **(**fabs**(**XiMas1 **-** XiMenos1**)** **>** 1**)** **{**  **return** FRACASO**;**  **}** **else** **if** **(**error**(**XiMas1**,** Xi**)** **<** 1 /\* % \*/**)** **{**  **break;**  **}**  **if** **(**fabs**(**XiMas1 **-** XiMenos1**)** **<** fabs**(**Xi **-** XiMas1**))** **{**  XiMenos1 **=** XiMas1**;**  **}** **else** **{**  Xi **=** XiMas1**;**  **}**  //printf("s %d -> %F\n", i, XiMas1); //TODO  i**++;**  **}**  **return** XiMas1**;**  **}**  // El signo es para saber si quiero el límite de convergencia por derecha o izquierda  /\*  \*  \* Ej: si quiero el mínimo, resto x hasta que el método diverja, después achico delta y sumo hasta que converja.  \* Itero y corto cuando el error relativo sea menor a 1.  \*  \*/  double convPFijo **(**short signo**,** double raizBiseccion**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  **if** **(** **!(**signo **==** 1 **||** signo **==** **-**1**)** **)**  **return** FRACASO**;**  double x **=** raizBiseccion**;**  double delta **=** 0.01**;**  int i **=** 1**;**  double Xi**;**  double XiMas1 **=** raizBiseccion**;**  **while** **(**i **<** MAXITERACIONES**)** **{**  // La única raíz que me importa es menor (en módulo) que 0.05  **while** **(**fabs**(**puntoFijo**(**inversion**,** arrayFCF**,** x**))** **<** 0.05**)** **{**  x **=** x **+** signo **\*** delta**;**  **}**  Xi **=** XiMas1**;**  XiMas1 **=** x**;**  signo **=** signo **\*** **-**1**;**  delta **=** delta **/** 10**;**  **while** **(**fabs**(**puntoFijo**(**inversion**,** arrayFCF**,** x**))** **>** 0.05**)** **{**  x **=** x **+** signo **\*** delta**;**  **}**  Xi **=** XiMas1**;**  XiMas1 **=** x**;**  signo **=** signo **\*** **-**1**;**  delta **=** delta **/** 10**;**  **if** **(**error**(**XiMas1**,** Xi**)** **<** 1 /\* % \*/**)**  **break;**  i**++;**  **}**  **if** **(**puntoFijo**(**inversion**,** arrayFCF**,** XiMas1**)** **==** FRACASO**)**  **return** Xi**;**  **return** XiMas1**;**  **}**  double convSecante **(**short minOMax**,** double raizBiseccion**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  **if** **(** **!(**minOMax **==** 1 **||** minOMax **==** **-**1**)** **)**  **return** FRACASO**;**  short signo**;**  double x**;**  double xmin **=** raizBiseccion **-** 0.1**;**  double xmax **=** raizBiseccion **+** 0.1**;**  double delta **=** 0.1**;**  int i **=** 1**;**  double Xi**;**  double XiMas1 **=** raizBiseccion**;**  **if** **(**minOMax **==** **-**1**)** **{**  signo **=** **-**1**;**  **}** **else** **{**  signo **=** 1**;**  **}**  **while** **(**i **<** MAXITERACIONES**)** **{**  // La única raíz que me importa es menor (en modulo) que 0.1  **while** **(**fabs**(**secante**(**inversion**,** arrayFCF**,** xmin**,** xmax**))** **<** 0.05**)** **{**  // busco mínimo  **if** **(**minOMax **==** **-**1**)** **{**  xmin **=** xmin **+** signo **\*** delta**;**  x **=** xmin**;**  **}** **else** **{** // máximo  xmax **=** xmax **+** signo **\*** delta**;**  x **=** xmax**;**  **}**  **}**  Xi **=** XiMas1**;**  XiMas1 **=** x**;**  signo **=** signo **\*** **-**1**;**  delta **=** delta **/** 10**;**  **while** **(**fabs**(**secante**(**inversion**,** arrayFCF**,** xmin**,** xmax**))** **>** 0.05**)** **{**  // busco mínimo  **if** **(**minOMax **==** **-**1**)** **{**  xmin **=** xmin **+** signo **\*** delta**;**  x **=** xmin**;**  **}** **else** **{** // máximo  xmax **=** xmax **+** signo **\*** delta**;**  x **=** xmax**;**  **}**  **}**  Xi **=** XiMas1**;**  XiMas1 **=** x**;**  signo **=** signo **\*** **-**1**;**  delta **=** delta **/** 10**;**  **if** **(**error**(**XiMas1**,** Xi**)** **<** 1 /\* % \*/**)**  **break;**  i**++;**  **}**  **if** **(**puntoFijo**(**inversion**,** arrayFCF**,** XiMas1**)** **==** FRACASO**)**  **return** Xi**;**  **return** XiMas1**;**  **}**  /\*  \*  \* Main  \*  \*/  double buscarTIRBiseccion **(**int inversion**,** double arrayFCF**[**N**+**1**])** **{**  // Busco en un intervalo [0.02, 0.06]  double raiz **=** biseccion**(**inversion**,** arrayFCF**,** 0.02**,** 0.06**);**  imprimirRaiz**(**raiz**,** "bisección"**);**  **return** raiz**;**  **}**  void buscarTIRPuntoFijo **(**double raizBiseccion**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  double raiz **=** puntoFijo**(**inversion**,** arrayFCF**,** raizBiseccion **-** 0.1**);**  imprimirRaiz**(**raiz**,** "punto fijo"**);**  **}**  void buscarTIRSecante **(**double raizBiseccion**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  double raiz **=** secante**(**inversion**,** arrayFCF**,** raizBiseccion **-** 0.1**,** raizBiseccion **+** 0.1**);**  imprimirRaiz**(**raiz**,** "secante"**);**  **}**  void buscarTIREscenarios **()** **{**  double matriz**[**5**][**N**+**1**];**  struct vectorDatos datos **=** cargarDatos**();**  // a)  datos**.**costoUnitarioPotencia **=** datos**.**costoUnitarioPotencia **\*** 0.7**;**  cargarMatriz**(**matriz**,**datos**);**  printf**(**"a)\n"**);**  imprimirRaiz**(**secante**(**matriz**[**0**][**0**],** matriz**[**4**],** 0**,** 0.5**),** "secante"**);**  // b)  datos **=** cargarDatos**();**  **for** **(**int i **=** 0**;** i **<=** N**;** i**++)**  datos**.**ganancias**[**i**]** **=** 0**;**  cargarMatriz**(**matriz**,**datos**);**  printf**(**"b)\n"**);**  imprimirRaiz**(**secante**(**matriz**[**0**][**0**],** matriz**[**4**],** 0**,** 0.5**),** "secante"**);**  // c)  datos **=** cargarDatos**();**  datos**.**costoElec **=** datos**.**costoElec **\*** 2**;**  cargarMatriz**(**matriz**,**datos**);**  printf**(**"c)\n"**);**  imprimirRaiz**(**secante**(**matriz**[**0**][**0**],** matriz**[**4**],** 0**,** 0.5**),** "secante"**);**  // d)  datos **=** cargarDatos**();**  datos**.**factorUso **=** 0.2**;**  cargarMatriz**(**matriz**,**datos**);**  printf**(**"d)\n"**);**  imprimirRaiz**(**secante**(**matriz**[**0**][**0**],** matriz**[**4**],** 0**,** 0.09**),** "secante"**);**  // e)  datos **=** cargarDatos**();**  **if** **(**N **>** 5**)**  **for** **(**int i **=** 0**;** i **<=** 5**;** i**++)**  datos**.**ganancias**[**i**]** **=** 0**;**  cargarMatriz**(**matriz**,**datos**);**  printf**(**"e)\n"**);**  imprimirRaiz**(**secante**(**matriz**[**0**][**0**],** matriz**[**4**],** 0**,** 0.1**),** "secante"**);**  **}**  void buscarIntervaloConvergenciaPF **(**double raizBiseccion**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  double min **=** convPFijo**(-**1**,** raizBiseccion**,** inversion**,** arrayFCF**);**  double max **=** convPFijo**(+**1**,** raizBiseccion**,** inversion**,** arrayFCF**);**  printf**(**"Convergencia de punto fijo: %.2F -> %.2F\n"**,** min**,** max**);**  **}**  void buscarIntervaloConvergenciaSec **(**double raizBiseccion**,** int inversion**,** double arrayFCF**[**N**+**1**])** **{**  double min **=** convSecante**(-**1**,** raizBiseccion**,** inversion**,** arrayFCF**);**  double max **=** convSecante**(+**1**,** raizBiseccion**,** inversion**,** arrayFCF**);**  printf**(**"Convergencia de secante:\n"**);**  printf**(**"\tMínimo con intervalo máximo = %.2F: %.2F\n"**,** raizBiseccion **+** 0.1**,** min**);**  printf**(**"\tMáximo con intervalo mínimo = %.2F: %.2F\n"**,** raizBiseccion **-** 0.1**,** max**);**  **}**  int proceso **()** **{**  struct vectorDatos datos **=** cargarDatos**();**  double matriz**[**5**][**N**+**1**];**  double raizBiseccion**;**  int inversion**;**  cargarMatriz**(**matriz**,** datos**);**  inversion **=** matriz**[**0**][**0**];**  imprimirEnunciado**(**1**);**  imprimirTabla**(**matriz**);**  imprimirEnunciado**(**2**);**  raizBiseccion **=** buscarTIRBiseccion**(**inversion**,** /\* FCF \*/matriz**[**4**]);**  imprimirEnunciado**(**3**);**  buscarTIRPuntoFijo**(**raizBiseccion**,** inversion**,** /\* FCF \*/matriz**[**4**]);**  imprimirEnunciado**(**4**);**  buscarIntervaloConvergenciaPF**(**raizBiseccion**,** inversion**,** /\* FCF \*/matriz**[**4**]);**  imprimirEnunciado**(**5**);**  buscarTIRSecante**(**raizBiseccion**,** inversion**,** /\* FCF \*/matriz**[**4**]);**  buscarIntervaloConvergenciaSec**(**raizBiseccion**,** inversion**,** /\* FCF \*/matriz**[**4**]);**  imprimirEnunciado**(**7**);**  buscarTIREscenarios**();**  printf**(**"\n"**);**  **return** TRUE**;**  **}**  int main **()** **{**  printf**(**"\nCesar Ezequiel Herrera 97429\n"**);**  //correrTests(); // Pruebas Unitarias  **return** proceso**();**  **}** |

## Salida

|  |
| --- |
| Cesar Ezequiel Herrera 97429  1) Calcular la inversión requerida, los ahorros y el resultado del flujo de fondos para cada año.  Mostrar los valores en una tabla:  | Anio 0 | Anio 1 | Anio 2 | ... | Anio N  --------------------------------------------------------------------  Inversion | -945000 $ | 0.00 $ | 0.00 $ | ... | 0.00 $  --------------------------------------------------------------------  Ahorro electricidad | 87567 $ | 87567 $ | 87567 $ | ... | 87567 $  --------------------------------------------------------------------  Ahorro potencia | 27000 $ | 27000 $ | 27000 $ | ... | 27000 $  --------------------------------------------------------------------  Costos | 5000 $ | 5000 $ | 5000 $ | ... | 5000 $  --------------------------------------------------------------------  FCF | 71218 $ | 71218 $ | 71218 $ | ... | 71218 $  2) Aplicar el método de bisección para calcular la TIR del proyecto.  Para ello, encuentre un intervalo que contenga a la raíz buscada e itere hasta lograr una precisión de 1%.  Exprese el resultado correctamente.  Raíz por bisección: 0.04 +/- 0.01  3) Aplicar el método de punto fijo para calcular la TIR del proyecto utilizando la función de iteración g(x) = x - f(x).  Para ello, utilice como semilla, un valor similar al encontrado mediante bisección.  Exprese el resultado correctamente.  Raíz por punto fijo: 0.04 +/- 0.01  4) Evalúe experimentalmente las condiciones de convergencia del método anterior mediante pruebas con diferentes semillas.  Reduzca el valor de la semilla hasta encontrar el límite inferior del intervalo de convergencia.  Luego aumente el valor de la semilla hasta encontrar el límite superior del intervalo de convergencia.  Muestre el intervalo de convergencia obtenido.  Convergencia de punto fijo: -0.84 -> 0.31  5) Repita los puntos 3) y 4) aplicando el método de la secante.  Raíz por secante: 0.04 +/- 0.01  Convergencia de secante:  Mínimo con intervalo máximo = 0.14: -0.20  Máximo con intervalo mínimo = -0.06: 1.43  7) Calcule la TIR del proyecto utilizando el método de la secante para los siguientes escenarios:  a) El costo unitario de la instalación fotovoltaica se reduce un 30%.  b) Los proyectos solares quedan exentos del pago del impuesto a las ganancias (α=0).  c) El costo de la electricidad se duplica.  d) La constante del Factor de Uso aumenta a 0,2 (mejora la eficiencia de los paneles)  e) Los proyectos solares quedan exentos del pago del impuesto a las ganancias por los primeros 5 años.  a)  Raíz por secante: 0.09 +/- 0.01  b)  Raíz por secante: 0.10 +/- 0.01  c)  Raíz por secante: 0.12 +/- 0.01  d)  Raíz por secante: 0.06 +/- 0.01  e)  Raíz por secante: 0.07 +/- 0.01 |