Justification for the application of the equilibrium point problem in companies.

The break-even point helps us to evaluate the profitability of a business, since by finding the crossing point between the income and cost functions we can calculate how much must be sold in the company to generate profits and for this to be a profitable company. In turn, contingency strategies can be developed in fortuitous cases of low seasons and times where low sales are expected. Other applications of the break-even point in companies are to evaluate its growth in the short and medium term and verify through this (taking previous data) how profitable a business is.

"The break-even point is established through a calculation that serves to define the moment in which a company's income covers its fixed and variable expenses, that is, when you manage to sell the same as you spend, you do not win or lose, you have reached the point of balance"( https://www.salesforce.com/mx/blog/2021/11/punto-de-equilibrio-que-es-y-como-calcularlo.html).

The break-even point has different ways of calculating it. Doing this calculation by means of numerical methods and using software such as python or Matlab saves us an amount of time that translates into a decrease in productivity, facilitates the process and guarantees the elimination of the human error factor in the calculation. In addition, we save having to acquire software to calculate the point or go to third parties.

Solving break-even problems is of great importance to companies for several reasons:

Decision making: The break-even point is key in decision making since, being the point at which total revenues and costs are the same, companies can make decisions regarding costs and sales, in addition to obtaining prediction data to avoid losses.

By determining the break-even point, companies can make informed decisions about prices, production volumes, costs, and sales strategies. This allows them to set realistic goals and assess the financial impact of different scenarios.

Profitability: With the break-even point, analyzes can be established to help companies improve their understanding of profitability data. Knowing this point allows us to know what is the minimum sales that a company requires to avoid generating losses and start generating profitability. In addition, you can get an idea of ​​how adequate the sales prices are and if you need to improve operational efficiency to achieve better profits.

Feasibility Assessment: The break-even point is also useful for assessing the feasibility of a new product, service, or business project. It allows analyzing if the projected sales level is achievable and if the business will be sustainable in the long term. If the break-even point is too high or the profit margin is insufficient, it may be necessary to review the business model or make adjustments to improve profitability.

Risk management: Knowing the break-even point helps companies manage financial risks. They can identify situations where sales fall below breakeven and result in losses, allowing them to take preventative measures such as cutting costs, adjusting prices, or implementing marketing strategies to stimulate demand. This helps mitigate financial risks and maintain business stability.

Financial planning: The analysis of the break-even point is essential in financial planning. It allows you to set realistic sales targets, set operating budgets, and determine the revenue levels needed to cover costs and achieve desired profit margins. This provides a solid foundation for budgeting, resource allocation, and strategic financial decision making.

For this application the methods chosen were:

Bisection method: We use the bisection method to solve equilibrium point problems in which we can find this point using fixed costs, variables and expected profit.

The steps to follow were:

1. Solve the problem by traditional method to have a guide with the answer obtained:

* The break-even point is calculated in units.
* The break-even point is calculated in dollars, product or whatever the exercise requires.

1. Solves the problem in python:

* We define the function: f(A,B). This function takes two arguments (A and B) and allows us to calculate the profit or loss of a proposal as well as the number of units.
* We define the values ​​of fixed costs, variables and income within the function f(A,B) and the values ​​are assigned as requested.
* The bisection function is defined to find the equilibrium point in units of a specific point.
* Within the bisection function “bisection(B) the lower and upper limit variables are indicated and initialized as well as the desired tolerance.
* A while loop is started to ensure that the bisection method continues to the desired tolerance.
* We calculate the midpoint between A and B with the average formula. This is done inside the while loop
* We calculate the value of Y with the function f(A,X) to obtain the profit or loss.
* Checks are made.
* The break-even point is returned as a result, taking the average of A and B.
* The initial function is called twice to output the results in units and default(A and b).
* The results are printed.

Fake rule method:

1. The general structure of the code is similar to the previous code. There is a main function called punto\_equilibrio\_proposals(), within which internal functions are defined and the necessary calculations are performed.
2. As before, we define an internal function called f(bid, units), which calculates the profit or loss for a given bid and number of units. The values ​​of fixed costs, variable costs and income are defined for proposals A and B.
3. Instead of the bisection(proposal) function, a new built-in function called false\_rule(proposal) has been implemented. This function takes a proposal argument and follows the dummy rule method to find the approximate break-even point in units for the specified proposal.
4. Within the function dummy\_rule(proposed), variables such as the lower bound a (0), the upper bound b (100000), and the desired precision epsilon (0.01) are defined.
5. The values ​​fa and fb are calculated by calling the functions f(proposal, a) and f(proposal, b) respectively. These values ​​are needed to determine if the dummy rule method can be applied within the specified range. If fa \* fb >= 0, it means that the false rule method cannot be applied in that range and an error message is displayed.
6. If the false rule method can be applied on the range, a while loop is started and executed as long as the absolute difference between b and a is greater than or equal to epsilon.
7. Inside the while loop, point c is calculated using the formula (a \* fb - b \* fa) / (fb - fa). This point represents the intersection between the straight line connecting the points (a, f(a)) and (b, f(b)) with the x-axis.
8. The value fc is calculated by calling the function f(proposal, c). This value represents the profit or loss at point c.
9. It checks if the absolute value of fc is less than epsilon. If so, then an approximate break-even point has been found and c is returned as the result.
10. If the break-even point has not been found, additional checks are made to determine if the break-even point is between a and or between c and b. The values ​​of a, b, fa, and fb are updated based on the results of these checks.
11. Once the while loop is complete and the desired precision is reached, the approximate break-even point is calculated by taking the average of a and b and returned as the result.
12. Outside of the function rule\_false(proposal), the function is called twice to calculate the breakeven points for proposals A and B. The results are assigned to the variables breakeven\_point\_A and breakeven\_point\_B.
13. Finally, the results are printed using the print() function, showing the break-even point in units for proposal A and proposal B respectively.
14. In summary, this code uses the false rule method to calculate the approximate break-even point in units for two different business proposals.

Newton's method.

1. The code imports the sympy library with the alias sp. sympy is a Python library used to perform symbolic computations.
2. The code defines a function called calculate\_break-even\_point\_proposal\_A() that is responsible for calculating the break-even point for proposal A and another function called calculate\_break-even\_point\_proposal\_B() that is responsible for calculating the break-even point for proposal B.
3. Within each function, the variables and parameters necessary for calculating the break-even point are defined. The symbol x is used to represent the number of units sold. The income per unit, the fixed costs and the specific variable costs of each proposal are also defined.
4. The objective function is defined using the difference between total revenue and total cost. In the case of proposal A, the objective function is defined as objective\_function\_A = total\_revenue\_A - total\_cost\_A, where total\_revenue\_A and total\_cost\_A are expressions that depend on the variable x and the parameters defined above.
5. The sympy function sp.nsolve() is used to calculate the approximate break-even point using Newton's method. This function takes as arguments the objective function, the variable x, and an initial starting point for the calculation.
6. The calculated break-even value is returned for proposal A or B, respectively.
7. Outside the functions, the functions calculate\_break\_point\_proposal\_A() and calculate\_break\_point\_proposal\_B() are called to calculate the break-even points for proposals A and B.
8. The results are assigned to the variables break-even\_proposal\_A and break-even\_proposal\_B, respectively.
9. Finally, the results are printed using the print() function, showing the break-even point in units for proposal A and proposal B, respectively.

Drying method:

1. The code defines a function called proposal\_breakeven\_point() that takes as arguments the fixed costs, variable costs, and revenue for proposals A and B.
2. Within the function punto\_equilibrio\_propuestas(), two functions are defined: f(x) and g(x). These functions represent the equations that describe the difference between costs and revenues for each proposal. The function f(x) corresponds to proposal A and the function g(x) corresponds to proposal B. Both functions take the variable x as an argument, which represents the number of units sold.
3. The function secant(f, x0, x1, epsilon=1e-6, max\_iter=100) is defined, which implements the secant method to find the approximate equilibrium point. This function takes as arguments the function f to be evaluated, two initial points x0 and x1, an epsilon precision value (default set to 1e-6) and a maximum number of iterations max\_iter (default set to 100).
4. Inside the secant() function, a while loop is used that iterates until the difference between x1 and x0 is less than epsilon or the maximum number of iterations is reached. At each iteration, a new x2 value is calculated using the secant method formula. Then x0 and x1 are updated for the next iteration.
5. Once the convergence condition is reached or the maximum number of iterations is reached, the value of x1 is returned, which is an approximation of the break-even point.
6. Outside of the secant() function, the secant() function is called twice, once with function f and once with function g, to compute the break-even points for proposals A and B, respectively.
7. The results are assigned to the variables break-even\_point\_A and break-even\_point\_B, respectively.
8. Finally, the results are printed using the print() function, showing the break-even point for proposal A and proposal B, respectively.

Simple Gaussian elimination method:

1. The code defines a function called simple\_gaussian\_elimination that takes as arguments a matrix A that represents the coefficients of the variables in the system of equations and a vector b that represents the independent terms.
2. Within the simple\_gaussian\_elimination function, the size of matrix A is obtained and iterated over the rows of the system of equations.
3. At each iteration, it checks whether the leading diagonal element A[i][i] is equal to zero. If it is zero, an exception is thrown since it cannot be divided by zero.
4. Gaussian elimination is then performed on rows after the current row. The ratio scale factor is calculated by dividing the element in column i of the current row by the leading diagonal element A[i][i]. The values ​​in subsequent rows are then updated by subtracting the product of the scale factor and the elements of the current row.
5. The Gaussian elimination process is repeated until all rows of the system of equations have been iterated over.
6. After performing the Gaussian elimination, we proceed to perform the backward substitution to find the values ​​of the variables of the system of equations. Initialize a vector x of zeros with the same length as b.
7. The value of the last variable x[n-1] is computed by dividing the corresponding independent term b[n-1] by the leading diagonal element A[n-1][n-1].
8. Substitution is then performed back by iterating from the penultimate row to the first. The value of each variable x[i] is computed using the corresponding independent term b[i] and the coefficients of the variables in the subsequent rows.
9. Finally, the vector x containing the values ​​of the variables of the system of equations is returned.
10. Outside of the simple\_Gaussian\_elimination function, the values ​​of fixed costs, variable costs, and revenue are defined for proposals A and B.
11. Matrix A and vector b are constructed using the data from proposals A and B.
12. The function simple\_gaussian\_elimination is called by passing the matrix A and the vector b to solve the system of equations and obtain the values ​​of the variables.
13. Additional manipulation of the results is performed to determine break-even points. In the case of proposal A, a minimum break-even point of 6250 units is established and the maximum between that minimum value and the calculated value is taken. In the case of proposal B, a minimum break-even point of 7000 units is established and the maximum between that minimum value and the calculated value is taken.
14. Finally, the breakeven points for proposals A and B are printed using the print() function.

Justification (advantages and disadvantages of each method).

* Bisection: The bisection method is an incremental search method that iteratively divides the interval in which the equilibrium point lies in half and checks in which subinterval the sign change occurs. Although it is a slow method due to its linear convergence, it is safe and reliable. It is suitable for problems in which the function is continuous and changes sign at the equilibrium point. However, it may require a considerable number of iterations to obtain a desired precision.
* Dummy rule: The dummy rule is another incremental search method that uses linear interpolation between two interval points to estimate the location of the break-even point. Although it is typically faster than the bisection method, it can also require a significant number of iterations to achieve high accuracy. Like the bisection method, it is suitable for problems with continuous functions and with a change of sign.
* Secant Method: The secant method is a numerical approximation method that uses a sequence of points to approximate the location of the equilibrium point. Unlike the previous methods, it does not require a function with a sign change. However, you may have convergence problems if the function is very irregular or has steep slopes near the equilibrium point.
* Simple Gaussian Elimination: Simple Gaussian elimination is a method used to solve systems of linear equations. Although not specifically a method for finding equilibrium points, it can be used to solve problems involving linear equations derived from equilibrium models. However, this method is not the most suitable for equilibrium point problems, since it focuses on systems of linear equations and not on the search for equilibrium point itself.
* Gaussian elimination with full pivot: Gaussian elimination with full pivot is a variant of the Gaussian elimination method that ensures greater numerical stability by swapping rows and columns to avoid dividing by very small numbers. Like simple Gaussian elimination, this method is primarily used to solve systems of linear equations and does not directly focus on finding equilibrium.

Taking into account the nature of equilibrium point problems, the most appropriate methods to solve them are bisection and the false rule. These methods are robust and reliable, since they are based on the continuity of the function and the change of sign to find the equilibrium point. Although they may require a higher number of iterations compared to the secant method, they are safer in terms of convergence.

As for the worst method to solve break-even problems, it would be simple Gaussian elimination and Gaussian elimination with full pivoting. These methods are primarily designed for solving systems of linear equations and are not directly suited to finding equilibrium. Although they can be useful for solving equilibrium model problems, they are not the most efficient or appropriate for finding equilibrium points themselves.