|  |
| --- |
| REPLACEMENT MODEL  DUALITY  SENSITIVITY ANALYSIS |
| Assignment #3 / Project  BS (SE-B)  **GROUP MEMBERS**  **MAHA TANWEER 19K-1094**  **MARIAM NEAZ 19K-1086**  **FIZZA ZAKIR 19K-1106** |
| **MAY 30** 2021  **MT303-Operations Research**  **Course Instructor: Dr. Khusro Mian** |



**REPLACEMENT MODEL**

**LANGUAGE: C++**

#include<iostream>

using namespace std;

int main()

{

int no\_of\_years;

cout<<"Enter number of years: ";

cin>>no\_of\_years;

int cost;

cout<<"\nEnter cost of machine: ";

cin>>cost;

int years[no\_of\_years], maintenance\_cost[no\_of\_years], cumulative\_maintenance\_cost[no\_of\_years]={0}, total\_cost[no\_of\_years];

float average\_cost[no\_of\_years];

for(int i=0; i<no\_of\_years; i++)

{

years[i]=i+1;

}

int option;

cout<<"\n1. Depreciation\n2. Salvage\n";

cin>>option;

switch(option)

{

case 1:

{

int depreciation[no\_of\_years];

cout<<"\nEnter maintenace cost for every year\n";

int maintenance;

for(int i=0; i<no\_of\_years; i++)

{

cin>>maintenance;

maintenance\_cost[i]=maintenance;

}

cout<<"\nEnter Depreciation for every year\n";

int dep;

for(int i=0; i<no\_of\_years; i++)

{

cin>>dep;

depreciation[i]=dep;

}

cumulative\_maintenance\_cost[0]=maintenance\_cost[0];

for(int i=1; i<no\_of\_years; i++)

{

cumulative\_maintenance\_cost[i]+=maintenance\_cost[i]+cumulative\_maintenance\_cost[i-1];

}

for(int i=0; i<no\_of\_years; i++)

{

total\_cost[i]=depreciation[i]+cumulative\_maintenance\_cost[i];

}

for(int i=0; i<no\_of\_years; i++)

{

average\_cost[i]=total\_cost[i]/years[i];

}

int replace=-1;

for(int i=0; i<no\_of\_years-1; i++)

{

if(average\_cost[i]<average\_cost[i+1])

{

replace=years[i];

break;

}

}

for(int i=0; i<no\_of\_years; i++)

{

cout<<years[i]<<" | "<<maintenance\_cost[i]<<" | "<<cumulative\_maintenance\_cost[i]<<" | "<<total\_cost[i]<<" | "<<average\_cost[i];

cout<<endl;

}

if(replace==-1)

{

cout<<"\nThere is no need to replace the machine\n";

}

else

{

cout<<"The machine should be replcae at the end "<<replace<<"th year or at the beginning of "<<replace+1<<"th year";

}

}

case 2:

{

int opt;

cout<<"\n1. Salvage Value(one value)\n2. Resale Value(for every year)\n";

cin>>opt;

if(opt==1)

{

int salvage;

cout<<"\nEnter Salvage Value";

cin>>salvage;

cout<<"\nEnter maintenance cost for every year\n";

int maintenance;

for(int i=0; i<no\_of\_years; i++)

{

cin>>maintenance;

maintenance\_cost[i]=maintenance;

}

cumulative\_maintenance\_cost[0]=maintenance\_cost[0];

for(int i=1; i<no\_of\_years; i++)

{

cumulative\_maintenance\_cost[i]+=maintenance\_cost[i]+cumulative\_maintenance\_cost[i-1];

}

for(int i=0; i<no\_of\_years; i++)

{

total\_cost[i]=cost-salvage+cumulative\_maintenance\_cost[i];

}

for(int i=0; i<no\_of\_years; i++)

{

average\_cost[i]=total\_cost[i]/years[i];

}

int replace=-1;

for(int i=0; i<no\_of\_years-1; i++)

{

if(average\_cost[i]<average\_cost[i+1])

{

replace=years[i];

break;

}

}

for(int i=0; i<no\_of\_years; i++)

{

cout<<years[i]<<" | "<<maintenance\_cost[i]<<" | "<<cumulative\_maintenance\_cost[i]<<" | "<<total\_cost[i]<<" | "<<average\_cost[i];

cout<<endl;

}

if(replace==-1)

{

cout<<"\nThere is no need to replace the machine\n";

}

else

{

cout<<"The machine should be replcae at the end "<<replace<<"th year or at the beginning of "<<replace+1<<"th year";

}

}

else if(opt==2)

{

int resale\_value[no\_of\_years];

cout<<"\nEnter maintenace cost for every year\n";

int maintenance;

for(int i=0; i<no\_of\_years; i++)

{

cin>>maintenance;

maintenance\_cost[i]=maintenance;

}

cout<<"\nEnter resale value for every year\n";

int resale;

for(int i=0; i<no\_of\_years; i++)

{

cin>>resale;

resale\_value[i]=resale;

}

cumulative\_maintenance\_cost[0]=maintenance\_cost[0];

for(int i=1; i<no\_of\_years; i++)

{

cumulative\_maintenance\_cost[i]+=maintenance\_cost[i]+cumulative\_maintenance\_cost[i-1];

}

for(int i=0; i<no\_of\_years; i++)

{

total\_cost[i]=cost-resale\_value[i]+cumulative\_maintenance\_cost[i];

}

for(int i=0; i<no\_of\_years; i++)

{

average\_cost[i]=total\_cost[i]/years[i];

}

int replace=-1;

for(int i=0; i<no\_of\_years-1; i++)

{

if(average\_cost[i]<average\_cost[i+1])

{

replace=years[i];

break;

}

}

for(int i=0; i<no\_of\_years; i++)

{

cout<<years[i]<<" | "<<maintenance\_cost[i]<<" | "<<cumulative\_maintenance\_cost[i]<<" | "<<total\_cost[i]<<" | "<<average\_cost[i];

cout<<endl;

}

if(replace==-1)

{

cout<<"\nThere is no need to replace the machine\n";

}

else

{

cout<<"The machine should be replcae at the end "<<replace<<"th year or at the beginning of "<<replace+1<<"th year";

}

}

}

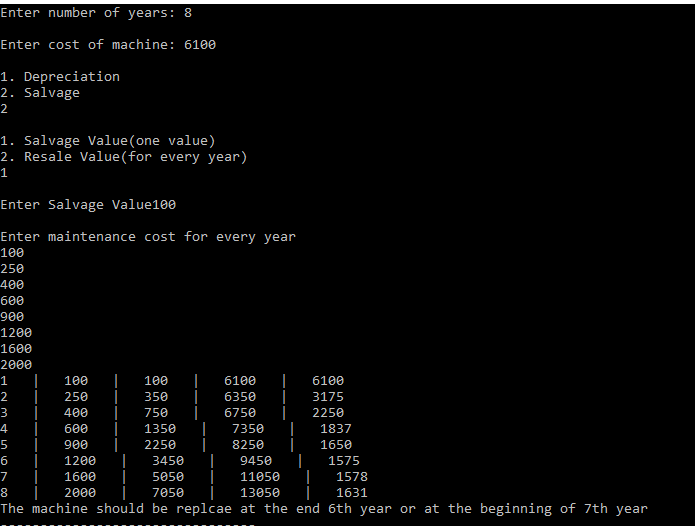
}

return 0;

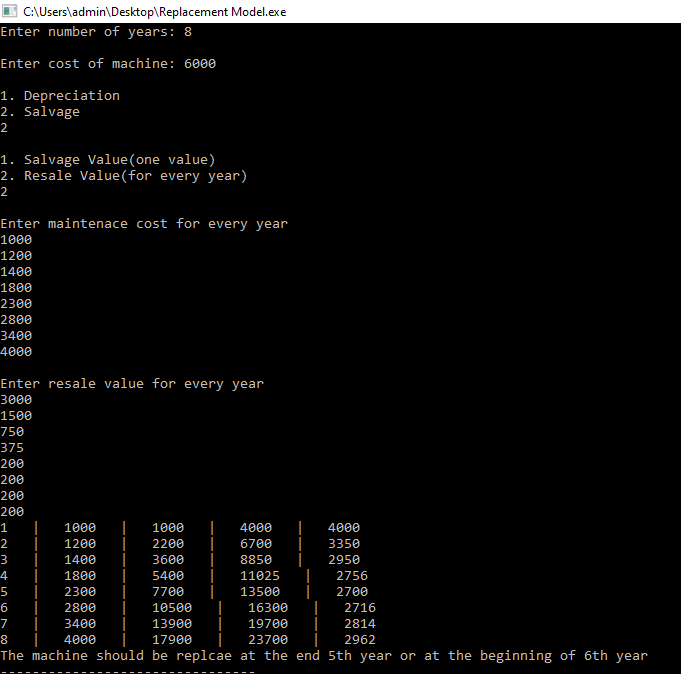
}

**OUTPUT**

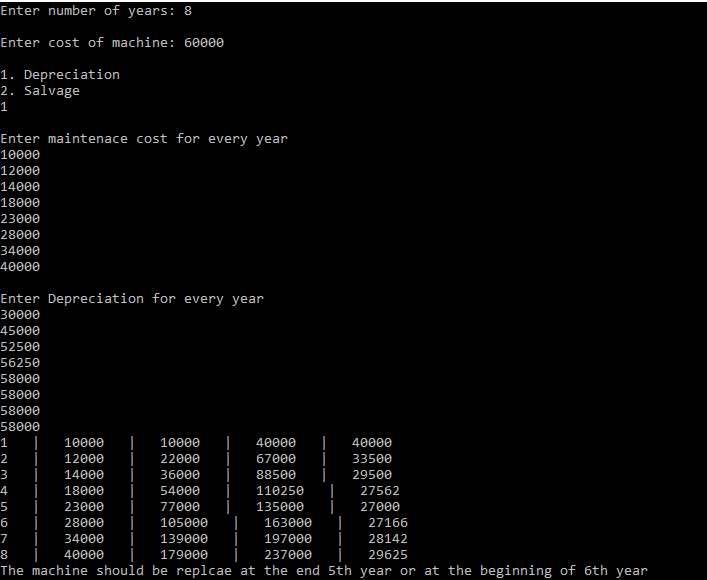
**CASE 1:**



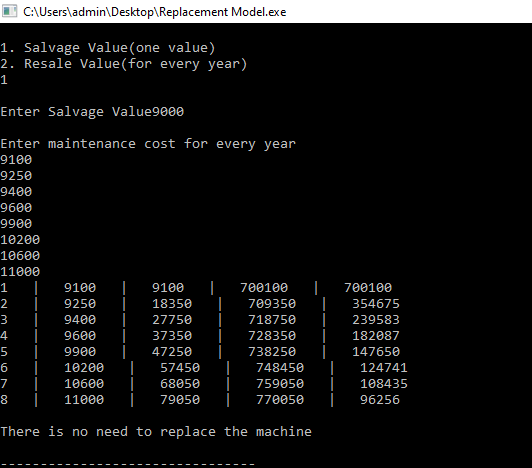
**CASE 2:**



**CASE 3:**



**CASE 4:**



**DUALITY**

**LANGUAGE: C++**

#include<iostream>

using namespace std;

void matrix(double \*\*array2,int con,int var)

{

for(int i=0;i<con;i++)

{

for(int j=0;j<var;j++)

cout<<array2[i][j]<<"\t";

cout<<"\n";

}

}

void display(double \*\*array2,double \*array1,int con,int var)

{

cout<<"VALUES:>>>\n\tZ = "<<array2[con][var];

cout<<"\nAt:\n";

for(int i=0;i<con;i++)

{

cout<<"\tX"<<array1[i+var]<<" = "<<array2[i][var]<<"\n";

}

for(int i=0;i<var;i++)

{

cout<<"\tX"<<array1[i]<<"=0\n";

}

}

void transpose (double \*\*array2,double \*obj,int con,int var)

{

cout<<"Enter Constraints: \n";

char sign,g='>';

for(int i=0;i<con;i++)

{

for(int j=0;j<var;j++)

{

cout<<"cons["<<i<<"]"<<"["<<j<<"]:";

cin>>array2[i][j];

}

cout<<"SIGN ( > , < ):";

cin>>sign;

cin>>array2[i][var];

if(sign==g)

{

for(int j=0;j<=var;j++)

{

array2[i][j]\*=-1;

}

}

}

for(int i=0;i<var;i++)

{

array2[con][i]=-1\*obj[i];

}

array2[con][var]=0;

}

void ReduceForm(double\*\* array2,int con,int var,int a,int b) {

double p;

double \*\*array3;

for(int i=0;i<var+1;i++)

{

array3[i]= new double [con+1];

}

for(int i=0;i<=con;i++)

{

for(int j=0;j<=var;j++)

{

array3[i][j]=array2[i][j];

}

}

p=array3[a][b];

array2[a][b]=1/p;

for(int i=0;i<=con;i++)

{

if(i!=a)

{

array2[i][b]=-1\*array3[i][b]/p;

}

}

for(int j=0;j<=var;j++)

{

if(j!=b)

{

array2[a][j]=array3[a][j]/p;

}

for(int i=0;i<=con;i++)

{

for(int j=0;j<=var;j++)

{

if((i!=a)&&(j!=b))

{

array2[i][j]= ((array3[a][b]\*array3[i][j])-(array3[a][j]\*array3[i][b]))/p;

}

}

}

}

}

double dual(double \*\*array2,double \*array1,int con,int var,int check)

{

int I=-1,J;

double min=0,DI;

for(int i=0;i<con;i++)

{

if(array2[i][var]<min)

{

min=array2[i][var];

I=i;

}

}

double sum=-1,ratio;

for(int j=0;j<con;j++)

{

if(array2[I][j]<0)

{

ratio=array2[con][j]/array2[I][j];

if((sum==-1)||(sum>ratio))

{

sum=ratio;

J=j;

}

}

}

if(check==1)

{

DI=array2[I][var]\*array2[con][J]/array2[I][J];

if(DI<0) DI\*=-1; cout<<"DI="<<DI<<"\n";

return DI;

}

ReduceForm(array2,con,var,I,J);

int temp;

temp=array1[I+var];

array1[I+var]=array1[J];

array1[J]=temp;

return 0;

}

double prime(double \*\*array2,double \*array1,int con,int var,int check) {

int J=-1,I;

double min=0,PI;

for(int j=0;j<var;j++)

{

if(array2[con][j]<min)

{

min=array2[con][j];

J=j;

}

}

double sum=-1,ratio;

for(int i=0;i<con;i++)

{

if(array2[i][J]>0)

{

ratio=array2[i][var]/array2[i][J];

if((sum==-1)||(sum>ratio))

{

sum=ratio; I=i;

}

}

}

if(check==1)

{

PI=array2[I][var]\*array2[con][J]/array2[I][J];

if(PI<0)

{

PI\*=-1;

cout<<"PI="<<PI<<"\n";

}

return PI;

}

ReduceForm(array2,con,var,I,J);

int temp;

temp=array1[I+var];

array1[I+var]=array1[J];

array1[J]=temp;

return 0;

}

void simplexMethod(double\*\* array2,double \*array1,int con,int var) {

int c1=0,c2=0;

for(int i=0;i<con;i++)

{ if(array2[i][var]<0)

{

c1=1;

}

}

for(int j=0;j<var;j++)

{ if(array2[con][j]<0)

{

c2=1;

}

}

if((c1==0)&&(c2==0)) return;

else if(c1==0)

{

prime(array2,array1,con,var,0);

}

else if(c2==0)

{

dual(array2,array1,con,var,0);

}

else {

double PI,DI;

PI=prime(array2,array1,con,var,1);

DI=dual(array2,array1,con,var,1);

if(PI>=DI)

{

prime(array2,array1,con,var,0);

}

else

{

dual(array2,array1,con,var,0);

}

}

simplexMethod(array2,array1,con,var);

}

int main()

{

int var,con; //variable, constraints

int x=0;

cout<< "\n ENTER 0 FOR MAXIMIZE or ENTER 1 FOR MINIMIZE\n";

cin>>x;

cout<< "\nENTER NO OF VARAIBLE:";

cin>> var;

cout<< "\nENTER NO OF CONSTRAINTS:";

cin>> con;

double \*obj=NULL;

double \*Array1=NULL;

double \*\*Array2=NULL; //innitiaizing memory

obj=new double[var];

Array1=new double[var+con];

Array2= new double\*[var];

for(int i=0;i<var+1;i++)

{

Array2[i]= new double [con+1];

}

cout<< "ENTER COEFFICIENTS OF OBJECTIVE FUNCTION:\n";

for(int i=0; i<var; i++)

{

cout<<"coff["<<i<<"]:";

cin>> obj[i];

}

if(x==1)

{

for(int i=0; i<var; i++)

{

obj[i]=-1\*obj[i];

}

}

transpose(Array2,obj,var,con);

simplexMethod(Array2,Array1,var,con);

if(x==1)

{

Array2[con][var]\*=-1;

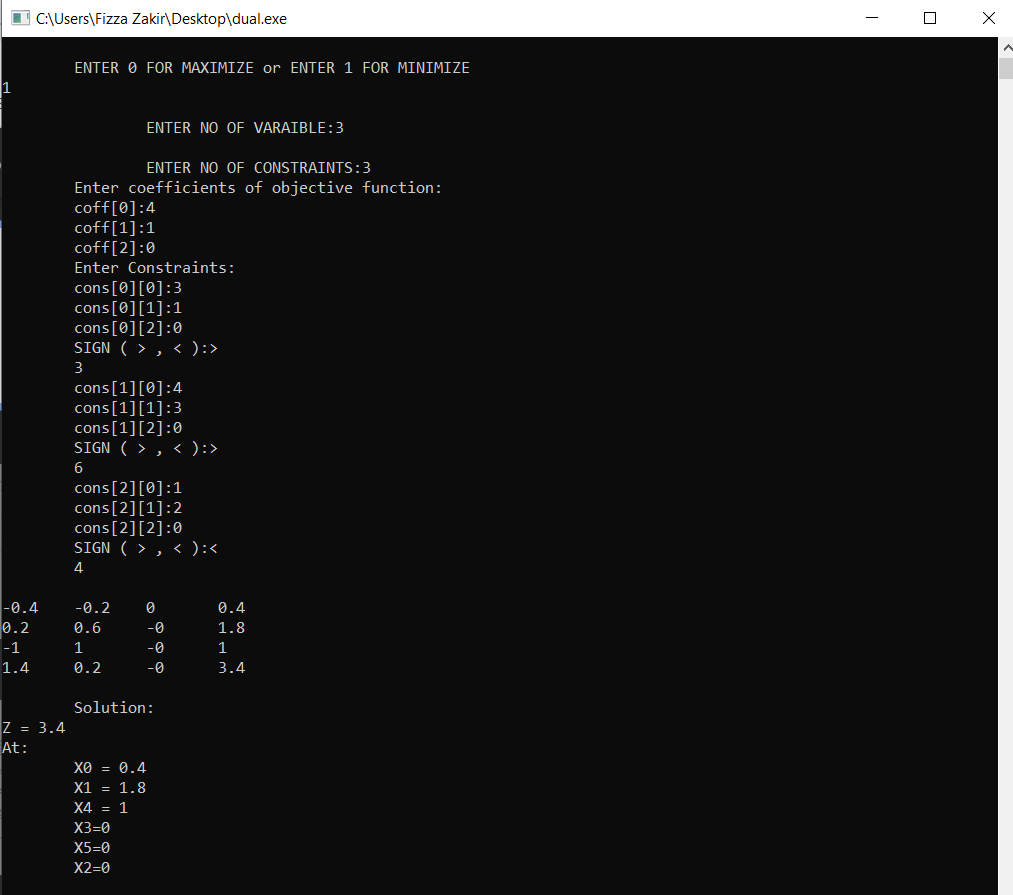
}

matrix(Array2,con+1,var+1);

display(Array2,Array1,con,var);

}

**OUTPUT**



**SENSITIVITY ANALYSIS**

**LANGUAGE: C**

#include <stdio.h>

#include <unistd.h>

double findZ(double x, double y, double coef1, double coef2){

double Z = ((x\*coef1) + (y\*coef2));

return Z;

}

double findLimit(double coefx, double coefy,double x, double y){

double result = (coefx\*x) + (coefy\*y);

return result;

}

double simultaneousEq(double a, double b, double c, double d, double e, double f, double coef1, double coef2){

/\* we solve the linear system

\* ax+by=e

\* cx+dy=f

\*/

double x,y;

double determinant = a\*d - b\*c;

if(determinant != 0) {

x = (e\*d - b\*f)/determinant;

y = (a\*f - e\*c)/determinant;

printf("\nIntersection Point (%f, %f)\n", x, y);

} else {

printf("there are either no solutions or many solutions exist.\n");

}

double Z = findZ(x,y,coef1,coef2);

return Z;

}

int main(int argc, char\*\* argv) {

printf("\t\t\tSENSITIVITY ANALYSIS\n\n");

int flag = 0; char choice;

double coef1,coef2,rate,Z1,newE,newF,limit1,limit2;

double a,b,e;

double c,d,f;

printf("Enter coefficients of the objective function:\n");

scanf("%lf",&coef1); fflush(stdin);

scanf("%lf",&coef2); fflush(stdin);

printf("\nWhere, \nax+by=e\ncx+dy=f are constraints\nEnter values of a,b,e:\n");

scanf("%lf",&a); fflush(stdin);

scanf("%lf",&b); fflush(stdin);

scanf("%lf",&e); fflush(stdin);

printf("\nPress y to change constraint, else press n\n");

scanf("%c", &choice); fflush(stdin);

if(choice == 'Y' || choice == 'y'){

flag = 1;

printf("\nEnter new value to replace %d in the constraint: ", (int)e);

scanf("%lf", &newE); fflush(stdin);

}

printf("\nConstraint 2, \n%dx+%dy=%d\ncx+dy=f \nEnter values of c,d,f:\n", (int)a,(int)b, (int)e);

scanf("%lf",&c); fflush(stdin);

scanf("%lf",&d); fflush(stdin);

scanf("%lf",&f); fflush(stdin);

printf("\nPress y for change in constraint, else press n\n");

scanf("%c", &choice); fflush(stdin);

if( flag == 0 && choice == 'Y' || choice == 'y'){

flag = 2;

printf("\nEnter new value to replace, %d in constraint: ", (int)f);

scanf("%lf", &newF);

}

double Z = simultaneousEq(a,b,c,d,e,f,coef1,coef2);

if(flag == 1){

Z1 = simultaneousEq(a,b,c,d,newE,f,coef1,coef2);

rate = (Z1 - Z)/(newE-e);

limit1 = findLimit(a,b,f/c, 0);

limit2 = findLimit(a,b,0 , f/d);

}

if(flag == 2){

Z1 = simultaneousEq(a,b,c,d,e,newF,coef1,coef2);

rate = (Z1 - Z)/(newF-f);

limit1 = findLimit(a,b,f/c, 0);

limit2 = findLimit(a,b,0 , f/d);

}

printf("\nRate of Revenue: %lf", rate);

printf("\nRange of Capacity: %lf <= var <= %lf ", limit1, limit2);

return 0;

}

**OUTPUT**

