ANEXO III: PROGRAMACIÓN EN MATLAB 6.1.

- **Función general:**

```
% 6-sept-2003
% funcion que controla todo el proceso
% recibe como argumentos:
% N, alfa_bus, alfa_metro, ruta_metro, L,fichflu, n_entre, nombrefich,
% n_max_nodos, fichpriv, beta_dir, beta_b_m, beta_b_b, existeprivada
% Devuelve la matriz de rutas y los vectores de carga y de longitudes
function[ruta,carga,Longitud_rutas]=zgeneral10(N, alfa_bus, alfa_metro, ruta_metro, L, fichflu, n_entre, nombrefich, n_max_nodos, fichpriv, beta_dir, beta_b_m, beta_b_b, existeprivada)

Longitud_rutas=zeros(1,N)

% creaci´on de la matriz sim´etrica de transporte p´ublico
fij=load(fichflu)
Fij=fij+fij';

% creaci´on de la matriz sim´etrica de transporte privado
if existeprivada==0
   n=size(Fij,1)
   Fpriv=zeros(n,n)
else
fpriv=load(fichpriv)
Fpriv=fpriv+fpriv';
end

 lij=load(nombrefich)
 Lij=(lij+lij')/2;

% llamada a la funcion maximos para calcular los nodos generadores de las rutas
 [nodo_origen,nodo_destino,ruta,pasajeros]=maximos(Fij, ruta_metro, Lij, N, L, n_entre, Fpriv)
```

```
estado=[0 ones(1,N)]

% mientras haya alguna ruta "abierta" se llamará a la función calculo
while(any(estado~=0))
[ruta,estado,Longitud_rutas]=calculo(Fij,estado,nodo_origen,nodo_destino,ruta,Lij,N,L,
alfa_bus,alfa_metro,Longitud_rutas,n_entre,n_max_nodos,Fpriv,beta_dir,beta_b_m,bet
a_b_b)
end

% terminado el calculo de rutas, se llama a la función que me calcula su carga.
[carga]=carga_rutas(ruta,Fij,N,alfa_bus,alfa_metro,beta_dir,beta_b_m,beta_b_b,Fpriv)
```

- **Función maximos:**

```
% esta funcion calcula cuales son los N pares de nodos entre los que hay un mayor flujo
% de viajeros
% se aceptara un par de nodos, siempre que la distancia entre ellos sea menor que la
%  longitud maxima de ruta permitida y ambos no pertenezcan a la linea de metro


function[nodo_origen,nodo_destino,ruta,pasajeros]=maximos(Fij,ruta_metro,Lij,N,L,n_
entre,Fpriv,beta_dir)

% creacion de la matriz conjunta de transporte publico y privado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fpriv afectado por beta_dir   %
Ftotal=Fij+Fpriv*beta_dir


[f,c]=size(Fij)  % calcula la dimension de Fij
ruta{1}=[ruta_metro]
nodo_origen=0
nodo_destino=0
j=1
i=0
n_posible_rutas=f*(f-1)/2  % nº posible de rutas que se podrian formar en principio,
                           % dado el nº de nodos

while(i<=n_posible_rutas & j<=N) % mientras no tenga las N rutas principales, y no
                                 %haya llegado al limite i de rutas que teóricamente
                                 %podrian ser creadas
    i=i+1
    [maximos,posicion]=max(Ftotal);
    [xm,pm]=max(maximos);
    maxi(i)=xm;
    columna=pm
    fila=posicion(pm)
    ruta_posible=[];
    Lruta_posible=[];

    % comprobar si ambas paradas pertenecen a la ruta de metro y que no se encuentran
    % a una distancia superior a L

    if ~(any(ruta_metro==fila)&any(ruta_metro==columna))&Lij(fila,columna)<=L
      if j>1
        for h=2:j
          ruta_h=ruta{h}
          nodo=fila
          [ruta_nueva,L_ruta]=Longitud_ruta3(h, ruta_h, nodo, Lij, nodo_origen,
                                             nodo_destino, n_entre)
```

```
                x=find(ruta_nueva==nodo)
                n_x=length(x)
                if n_x>1
                  ruta_nueva=[ruta_nueva(1:x(1)-1) ruta_nueva(x(1)+1:end)]
                end
            if L_ruta<L
              nodo=columna

[ruta_nueva,L_ruta]=Longitud_ruta3(h,ruta_nueva,nodo,Lij,nodo_origen,nodo_destino,
n_entre)
                  y=find(ruta_nueva==nodo)
                  n_y=length(y)
                  if n_y>1
                    ruta_nueva=[ruta_nueva(1:y(1)-1) ruta_nueva(y(1)+1:end)]
                  end
              if L_ruta<L

                ruta_posible{h}=[ruta_nueva]
                Lruta_posible(h)=L_ruta

              end


          end

        end
        n_rp=length(ruta_posible)
        if n_rp>0

          ceros=find(Lruta_posible==0)
          n_ceros=length(ceros)
          Lru_pos=Lruta_posible
          for d=n_ceros:-1:1
            c=ceros(d) % ir tomando los indices de los elementos nulos
            Lru_pos=[Lru_pos(1:c-1) Lru_pos(c+1:end)]
          end

            [min,pos]=min(Lru_pos)

            pmin=find(Lruta_posible==min)

            if length(pmin)==1
            ruta{pmin}=[ruta_posible{pmin}]
            else
            ruta{pmin(1)}=[ruta_posible{pmin(1)}]
            end

        else
          nodo_origen(j)=fila
```

```
            nodo_destino(j)=columna

            pasajeros(j)=xm

            j=j+1
            ruta{j}=[fila columna]

          end
            Ftotal(fila,columna)=0;
            Ftotal(columna,fila)=0;


        else

            nodo_origen(j)=fila
            nodo_destino(j)=columna

              pasajeros(j)=xm

            j=j+1
            Ftotal(fila,columna)=0;
            Ftotal(columna,fila)=0;
            ruta{j}=[fila columna]


        end

    end

    Ftotal(fila,columna)=0;
    Ftotal(columna,fila)=0;

end
```

- **Función cálculo:**

```
% funcion que halla que nodo unido a que ruta,
% proporciona un mayor incremento en la funcion objetivo


function
[ruta,estado,Longitud_rutas]=calculogordo(Fij,estado,nodo_origen,nodo_destino,ruta,Li
j,N,L,alfa_bus,alfa_metro,Longitud_rutas,n_entre,n_max_nodos,Fpriv,beta_dir,beta_b_
m,beta_b_b)


f=size(Fij,1);

for i=2:N+1  % Siendo N el numero de rutas principales. La 1 es de metro
   if estado(i)==1  % si la ruta(i) aun no esta cerrada

      ruta_i=ruta{i};

      lista=1:f;   % se crea lista con todos los nodos candidatos.

      nN=length(lista);
      %eliminar los nodos de la lista que ya
      %pertenecen a la ruta en estudio.
      [c]=intersect(lista,ruta_i);
      n=length(c);

      lista(c(1:n))=0; % hace 0 todos los nodos que ya pertenecen a la ruta en estudio
                  % y que por tanto no son nodos candidatos a unir a esa ruta

    if(any(lista~=0))

      for k=1:nN  % se van tomando cada elemento de la lista
                % para ser analizado
         v7=0;
         contador=0;
         suma1=0;
         suma1_priv=0;
         suma1_total=0;
         suma3=0;
         suma3_priv=0;
         suma3_total=0;
         su3=0;
         su3_priv=0;
         nx=0;

         restadir=0;
         restadir_priv=0;
```

```
        nodo=lista(k); % estudiamos el valor de la F.O. al unir `nodo´ a la ruta i

      if nodo~=0 %  que si nodo es cero, es decir, no es un candidato
                 % pasa al siguiente elemento.


  [ruta_nueva,L_ruta]=Longitud_ruta3(i,ruta_i,nodo,Lij,nodo_origen,nodo_destino,
                                     n_entre)

% comprobar al principio si L(nodo,i)>L, en tal caso no es necesario seguir con la
% funcion, pasandose al siguiente nodo de la lista

  Long_ruta(nodo,i)=L_ruta;
  if L_ruta<=L

     for h=1:N  % comprobar si `nodo´ pertenece
           % a alguna de las rutas principales / metro
           sux=0;
           sux_priv=0;
           sux2=0;
           sux2_priv=0;
           suma_com1=0;
           suma_com1_priv=0;

           ruta_h=ruta{h};

        if h~=i
          s=any(ruta_h==nodo)
          % el valor de alfa sera diferente
          % si `nodo´ pertenece a la ruta de metro o a una de bus

              if h==1
                 alfa=alfa_metro;

                 beta=beta_b_m;

              else
                 alfa=alfa_bus;

                 beta=beta_b_b;

              end
          if s==1  % si nodo pertenece a la ruta h.

            % si la ruta en estudio i, y la ruta h
            % a la que ya pertenece `nodo´
            % ya tenian paradas en comun,
            % no sumo el nuevo posible transbordo.
            [v]=intersect(ruta_i,ruta_h)
```

```
                if length(v)==0 % si ruta i y ruta h
                        % no se interseccionan anteriormente

                   % Para sumar los transbordos entre ruta i y ruta h
                   % tengo que eliminar `nodo´ de la ruta h
                   nv=ruta_h;
                   z=find(nv==nodo);
                   nv=[nv(1:z-1) nv(z+1:end)];
                   suma1=suma1+sum((sum(Fij(ruta_i,nv))))*alfa;

                   suma1_priv=suma1_priv+sum((sum(Fpriv(ruta_i,nv))))*beta;


                   for t=1:N  % para no sumar los viajes entre una parada de i
                           % y otra de h que ya se unian por otra ruta t

                      ruta_t=ruta{t};

                      if t~=i
                         if t~=h
                            v5=intersect(ruta_i,ruta_t);
                            if v5>0
                               v6=intersect(ruta_h,ruta_t);
                               if v6>0
                               sum_com1(t)=sum(sum(Fij(v5,v6)));

sum_com1(t)=sum(sum(Fij(v5,nv)))+sum(sum(Fij(v6,ruta_i)))-sum(sum(Fij(v5,v6)));

                               sum_com1_priv(t)=sum(sum(Fpriv(v5,v6)));

sum_com1_priv(t)=sum(sum(Fpriv(v5,nv)))+sum(sum(Fpriv(v6,ruta_i)))-
sum(sum(Fpriv(v5,v6)));

                                  % elimino de nx los nodos de v6,
                                  % para evitar problemas de restar dos veces
                                  % posibles viajes, una vez en sum_com1
                                  % y otra en sux2.
                                  if intersect(nx,v6)>0
                                     n_v6=length(v6);
                                     for s=n_v6:-1:1
                                        z=find(nx==v6(s));
                                        nx=[nx(1:z-1) nx(z+1:end)];
                                     end
                                  end

                               end
                            end
                         end
                      end
```

```
            end

        end
        com=intersect(ruta_h,nx);
            if com>0
                sux2=sux2+sum(Fij(ruta_i,com(s)))*alfa;

                sux2_priv=sux2_priv+sum(Fpriv(ruta_i,com(s)))*beta;

            end

        suma_com1=sum(sum_com1)*alfa;

        suma_com1_priv=sum(sum_com1_priv)*beta;


        suma1=(suma1-suma_com1-sux2);

        suma1_priv=(suma1_priv-suma_com1_priv-sux2_priv);

        suma1_total=suma1+suma1_priv;


    end
    nx=[ruta_h nx];


else % si nodo No pertenece a la ruta h
    [v4]=intersect(ruta_i,ruta_h);
    if length(v4)>0

        for t=1:N

            ruta_t=ruta{t};


            if t~=i
                if t~=h
                    [v2]=intersect(ruta_h,ruta_t);
                    if length(v2)>0
                        if any(ruta_t==nodo)
                            contador=contador+1;
                        end
                    end
                end
            end
        end
```

```
                if contador==0

                  nnv4=length(v4);
                  nv4=ruta_h;
                  for j=1:nnv4
                     x=v4(j);
                     z=find(nv4==x);
                     nv4=[nv4(1:z-1) nv4(z+1:end)];
                  end
                  com=intersect(ruta_h,nx);
                  if com>0 % para no contar dos veces los transbordos
                        % entre `nodo´ y una parada.
                    sux=sum(Fij(com,nodo));

                    sux_priv=sum(Fpriv(com,nodo));

                  end

                  su3=su3+(sum(Fij(nv4,nodo))-sux)*alfa;

                  su3_priv=su3_priv+(sum(Fpriv(nv4,nodo))-sux_priv)*beta;
                 end
                  nx=[ruta_h nx];
               end

            end

          end

        end

       suma3=sum(su3);

       suma3_priv=sum(su3_priv);

       suma3_total=suma3+suma3_priv;

        trans(nodo,i)=suma1_total+suma3_total;



    % ahora al añadir el flujo directo, tengo que sumar el privado afectado por beta_dir

       % a la hora de calcular el flujo directo, voy a arreglar el error que habia.
       % soluciono el problema de sumar viajes entre nodos de la ruta y 'nodo', cuando
estos, ya se encontraban
       % unidos por una ruta de bus o metro, y por lo tanto no debiera sumarlos.

       for n=1:N
```

```
            if i~=n
               ruta_n=ruta{n};
               z=any(ruta_n==nodo);
               if z==1
                  q=intersect(ruta_i,ruta_n);
                  if q>0
                     restadir=restadir+sum(Fij(q,nodo));
                     restadir_priv=restadir_priv+sum(Fpriv(q,nodo));
                  end
               end
            end
         end

         directo(nodo,i)=sum(Fij(ruta_i,nodo))-restadir; % ojo!, esto me mide el
%incremento de viajeros. NO el flujo de ellos en esta ruta
         directo_priv(nodo,i)=(sum(Fpriv(ruta_i,nodo))-restadir_priv)*beta_dir;

         directo_total(nodo,i)=directo(nodo,i)+directo_priv(nodo,i);

         total(nodo,i)=directo_total(nodo,i)+trans(nodo,i);


         rut=ruta_i;

          FO(nodo,i)=total(nodo,i)/Long_ruta(nodo,i)-Longitud_rutas(i);


       else %del if L(nodo,i)>L
          FO(nodo,i)=0;
       end

         end
       end
    end % fin del if(lista~=0)
   end
end
  tam=size(FO,2);
  for i=2:tam       % cuando una columna de esta matriz sea 0, la ruta
                % correspondiente quedara finalmente cerrada
     if FO(:,i)<=0
        estado(i)=0;
     end

  end


  kl=all(estado==0) ;% si no todos los elementos de FO son nulos:
  if kl==0
    [maxim,pos]=max(Heu_2);
```

```
    [m,p]=max(maxim);
    maximo=m;    % valor maximo de la funcion objetivo
    ruta_mod=p;  % ruta que se va a modificar

    ruta_p=ruta{p};


    nodo=pos(p); % nodo que se va a unir a la ruta

[ruta_nueva,L_ruta]=Longitud_ruta(p,ruta_p,nodo,Lij,nodo_origen,nodo_destino,n_entr
e)

    ruta{p}=[ruta_nueva];

   Longitud_rutas(p)=L_ruta;

    end

% Si la ruta i ha alcanzado el n_max_nodos se debe cerrar: estado(i)=0

for d=1:N
   n_nodos=length(ruta{d+1});
   if n_nodos==n_max_nodos
      estado(d+1)=0;
   end
end
```

- **Función Longitud de Ruta:**

```
% funcion Longitud_ruta
% indicara en que forma deben ir unidos los nodos que componen la ruta, de modo que
% la longitud de la misma sea lo menor posible.
% Por lo tanto, tambien calculara la longitud de la ruta, lo que permitira
% aceptar o no el ultimo nodo añadido, dependiendo si la longitud de la ruta
% supera o no el limite L fijado.
% ademas, esta funcion comprobara que el numero de paradas entre los dos nodos
% iniciales entre los que existe un flujo mayor, no sea superior al numero establecido de
% antemano

function[ruta_nueva,L_ruta]=Longitud_ruta3(i,rut,nodo,Lij,nodo_origen,nodo_destino,
n_entre)

L1=0
L2=0
L3=0
L4=0
L5=0
L6=0
L7=0

x=find(rut==nodo_origen(i-1))
y=find(rut==nodo_destino(i-1))
inc=abs(x-y)-1

if inc<n_entre % si todavia no se ha alcanzado el nº de paradas max permitidas entre
                % entre ni y nj
   l=Lij(rut,nodo)  % se calculan todas las longitudes entre todos los nodos de la ruta y
                    % el nodo que se desea añadir a la misma
else % no puedo unir `nodo´ a nodos entre el nodo_origen y el nodo_destino de la ruta

   nodo_origen=nodo_origen(i-1)
   nodo_destino=nodo_destino(i-1)
   p_n_or=find(rut==nodo_origen)
   p_n_ds=find(rut==nodo_destino)
   rut_mod=[rut(1:p_n_or) rut(p_n_ds:end)]
   nodos_entre=[rut(p_n_or+1:p_n_ds-1)]
   n_nodos_entre=length(nodos_entre)
   for w=1:n_nodos_entre-1
      L_int(w)=Lij(nodos_entre(w),nodos_entre(w+1))
   end
   L_intermedia=sum(L_int)
   l=Lij(rut_mod,nodo)
   rut_prin=rut
   rut=rut_mod
```

```
end

  n=length(rut) % se calcula el numero de nodos que componen la ruta
  %se calcula cual es el nodo de la ruta: `nodo_de_ruta´ que se encuentra mas cerca del
% nodo que se desea unir
  [mi,pmin]=min(l)
  nodo_de_ruta=rut(pmin)
  if inc==n_entre
     rut=rut_prin
     pmin=find(rut_prin==nodo_de_ruta)
  end
  n=length(rut)
  if pmin==n % si `nodo´ debe ser unido al ultimo nodo de la ruta

   if (inc==n_entre & nodo_de_ruta==nodo_destino)

    rut(n+1)=nodo
    ruta_nueva=rut
     for i=1:n
        L_tramos(i)=Lij(ruta_nueva(i),ruta_nueva(i+1))
     end
    L_ruta=sum(L_tramos)
   else
      for i=1:n-2
       L1=L1+Lij(rut(i),rut(i+1))
      end
      L4=Lij(rut(n-1),rut(n))
      L6=Lij(rut(n-1),nodo)
      L3=Lij(nodo_de_ruta,nodo)
      Lcaso1=L1+L4+L3
      Lcaso2=L1+L6+L3
      if Lcaso1<=Lcaso2
         rut(n+1)=nodo
         ruta_nueva=rut
         L_ruta=Lcaso1
      else
         ruta_nueva=rut(1:n-1)
         ruta_nueva(n)=nodo
         ruta_nueva(n+1)=rut(n)
         L_ruta=Lcaso2
      end
   end

  elseif pmin==1 % si nodo debe ser unido al primer nodo de la ruta

   if (inc==n_entre & nodo_de_ruta==nodo_origen)

    ruta_nueva(1)=nodo
    for i=2:n+1
```

```
      ruta_nueva(i)=rut(i-1)
    end
    for i=1:n
      L_tramos(i)=Lij(ruta_nueva(i),ruta_nueva(i+1))
    end
    L_ruta=sum(L_tramos)
   else  % en el resto de casos ver si conviene ser el primer o el segundo nodo de la ruta
     for i=2:n-1
      L1=L1+Lij(rut(i),rut(i+1))
     end
     L3=Lij(nodo_de_ruta,nodo)
     L5=Lij(nodo,rut(2))
     L7=Lij(rut(1),rut(2))
     Lcaso1=L3+L7+L1
     Lcaso2=L3+L5+L1
     if Lcaso1<=Lcaso2
       ruta_nueva(1)=nodo
       for i=2:n+1
          ruta_nueva(i)=rut(i-1)
       end
       L_ruta=Lcaso1

     else
        ruta_nueva(1)=rut(1)
        ruta_nueva(2)=nodo
        for i=3:n+1
        ruta_nueva(i)=rut(i-1)
        end
        L_ruta=Lcaso2
     end
   end

  else % si nodo debe ser unido a un nodo intermedio de la ruta


    k=find(rut==nodo_de_ruta)
    L3=Lij(nodo_de_ruta,nodo)

    for i=1:k-2
      L1=L1+Lij(rut(i),rut(i+1))
    end
    for i=1:n-k-1
      L2=L2+Lij(rut(k+i),rut(k+1+i))
    end
    L4=Lij(rut(k-1),rut(k))
    L5=Lij(nodo,rut(k+1))
    L6=Lij(rut(k-1),nodo)
    L7=Lij(rut(k+1),rut(k))
    Lfijo=L1+L2+L3
```

```
 Lcaso1=Lfijo+L4+L5
 Lcaso2=Lfijo+L6+L7

if inc<n_entre

 if Lcaso1<Lcaso2
   L_ruta=Lcaso1
   for i=1:k
      ruta_nueva(i)=rut(i)
   end
   ruta_nueva(k+1)=nodo
   for i=1:n-k
       ruta_nueva(k+1+i)=rut(k+i)
   end
 else
   L_ruta=Lcaso2
   for i=1:k-1
      ruta_nueva(i)=rut(i)
   end
   ruta_nueva(k)=nodo
   for i=1:n-k+1
      ruta_nueva(k+i)=rut(k+i-1)
   end
 end

elseif nodo_de_ruta==nodo_origen
   L_ruta=Lcaso2

   for i=1:k
      ruta_nueva(i)=rut_prin(i)
   end
   ruta_nueva(k+1)=nodo
   for i=k+2:n
      ruta_nueva(i)=rut_prin(i)
   end


elseif nodo_de_ruta==nodo_destino
   L_ruta=Lcaso1

   for i=1:k
      ruta_nueva(i)=rut_prin(i)
   end
   ruta_nueva(k+1)=nodo
   for i=k+2:n+1
      ruta_nueva(i)=rut_prin(i-1
   end

elseif Lcaso1<Lcaso2
```

```
    L_ruta=Lcaso1
    for i=1:k
       ruta_nueva(i)=rut_prin(i)
    end
    ruta_nueva(k+1)=nodo
    for i=1:n-k
        ruta_nueva(k+1+i)=rut_prin(k+i)
    end
  elseif Lcaso1>=Lcaso2
    L_ruta=Lcaso2

    for i=1:k-1
       ruta_nueva(i)=rut_prin(i)
    end
    ruta_nueva(k)=nodo
    for i=1:n-k+1
       ruta_nueva(k+i)=rut_prin(k+i-1)
    end
   end


end
```

- **Función carga_rutas:**

```
% esta funcion halla la carga de pasajeros de cada una de las rutas
% Contabilizara el flujo directo entre cada una de las paradas de la ruta,
% asi como el flujo de viajeros que soporta, al querer viajar mediante un transbordo
% a otras paradas que pertenecen a otra ruta que corta a la ruta en estudio.
% Debe recibir como argumentos:
% la matriz de rutas: ruta
% Fij, Fpriv
% N, nº total de rutas principales
% alfa, porcentaje de pasajeros que transbordan a otro bus, y al metro.
% parámetros beta


function[carga]=zcarga_rutas9(ruta,Fij,N,alfa_bus,alfa_metro,beta_dir,beta_b_m,beta_b
_b,Fpriv)

% a partir de la matriz de rutas, voy a crear una matriz que me indique por cuantas
% rutas estan unidos, de manera directa, dos nodos.
% Si dos nodos de la ruta en estudio, se encuentran unidos por mas rutas, la carga que
% aporta el viaje entre esos dos nodos, sera menor, en funcion del numero de rutas
% que unan esos dos nodos.
%29-abril-03
% Si dos nodos de la ruta i, se encuentran tambien unidos por la ruta de metro,
% el aporte de viajeros a la ruta i entre esos dos nodos no se contabiliza, pues todos los
% viajeros lo haran a traves de la ruta de metro.

f=size(Fij,1)
Dire=zeros(f) % matriz dire inicialmente nula

for i=1:N+1

   ruta_i=ruta{i}


    n_ri=length(ruta_i)
    for k=1:n_ri
       for h=k+1:n_ri
          if i==1
             Dire(ruta_i(k),ruta_i(h))=Dire(ruta_i(k),ruta_i(h))+N+1
             Dire(ruta_i(h),ruta_i(k))=Dire(ruta_i(h),ruta_i(k))+N+1
          else
             Dire(ruta_i(k),ruta_i(h))=Dire(ruta_i(k),ruta_i(h))+1
             Dire(ruta_i(h),ruta_i(k))=Dire(ruta_i(h),ruta_i(k))+1
          end
       end
    end
end
```

```
% Ahora, se analizaran todas las rutas, una a una.
for i=2:N+1
 if i~=1
      ruta_i=ruta{i}
      suma1=0
      suma2=0
      suma3=0
      suma4=0
      suma5=0
      int=0
      cont=1
      directo=0
      %%%%%%%%%%%%%%%%%%
      suma1_priv=0
      suma2_priv=0
      suma3_priv=0
      suma4_priv=0
      suma5_priv=0

      %%%%%%%%%%%%%%%%%%
      suma1_total=0
      suma2_total=0
      suma3_total=0
      suma4_total=0
      suma5_total=0


    % ir tomando el resto de rutas para ver si se cortan con la ruta_i
    for h=1:N+1
      if h~=i
              if h==1
                alfa=alfa_metro

                beta=beta_b_m

              else
                alfa=alfa_bus

                beta=beta_b_b

              end

        ruta_h=ruta{h}
        vx=intersect(ruta_i,ruta_h)
        nvx=length(vx)
        if nvx>0 % si la ruta_i y la ruta_h, interseccionan, se producen transbordos.
```

```
int=[int h] % vector de rutas que intersecionan con i


    % para sumar los transbordos entre ruta_i y ruta_h,
    % eliminar los nodos de interseccion.
    nvx=length(vx)
    nv_h=ruta_h
    for j=nvx:-1:1
       x=vx(j)
       z=find(nv_h==x)
       nv_h=[nv_h(1:z-1) nv_h(z+1:end)]
    end

    nv_i=ruta_i
    for j=nvx:-1:1
       g=vx(j)
       y=find(nv_i==g)
       nv_i=[nv_i(1:y-1) nv_i(y+1:end)]
    end

    suma1=suma1+sum(sum(Fij(nv_i,nv_h)))*alfa

    suma1_priv=suma1_priv+sum(sum(Fpriv(nv_i,nv_h)))*beta
    suma1=suma1+suma1_priv



    % restar los transbordos que ya fueron sumados anteriormente
    % comprobar si la ruta h, se corta con alguna ruta anteriormente analizada
    % que tambien se cortaba con la ruta i.
    % habra que eliminar los transbordos entre la ruta i y esos nodos de corte
    % entre las rutas h.

    n_int=length(int)
    if n_int>2 %si ya se han contabilizado mas de una ruta que corta a la ruta_i
      for k=2:n_int-1
       t=int(k)
       ruta_t=ruta{t}


          v1=intersect(ruta_h,ruta_t)
          if length(v1)>0
             if h==1
                alfa=alfa_metro

                beta=beta_b_m

             else
                alfa=alfa_bus
```

```
      beta=beta_b_b
   end

% Analizar si (vx=v1) o (v1 C vx) o (vx C v1) o vx~=v1
% si (vx=v1) o (v1 C vx) no hay que restar nada.
% si (vx C v1) solo hay que hacer suma3
% si vx~=v1, hay que restar suma2, suma3, suma4, suma5

int_vx_v1=intersect(vx,v1)


if length(int_vx_v1)==0 % si vx y v1 no tienen elementos comunes

   suma2=suma2+sum(sum(Fij(nv_i,v1)))*alfa
   % restar el flujo entre la ruta i (excepto el nodo de corte con h)
   % y el pto de intersect. de ruta h con la anterior que cortaba a i, y
   %    ahora se corta con la ruta h.


   suma2_priv=suma2_priv+sum(sum(Fpriv(nv_i,v1)))*beta
   suma2_total=suma2+suma2_priv

end

   if t==1
      alfa=alfa_metro

      beta=beta_b_m

   else
      alfa=alfa_bus

      beta=beta_b_b

   end

if length(int_vx_v1)==0 % si vx y v1 no tienen elementos comunes

   % restar flujos entre los ptos de interseccion entre i y h, h y t
   suma3=suma3+sum(sum(Fij(vx,v1)))*alfa

   suma3_priv=suma3_priv+sum(sum(Fpriv(vx,v1)))*beta
   suma3_total=suma3+suma3_priv

elseif  length(vx)<length(v1) % o si vx esta contenido en v1
   suma3=suma3+sum(sum(Fij(vx,v1)))*alfa

   suma3_priv=suma3_priv+sum(sum(Fpriv(vx,v1)))*beta
```

```
  suma3_total=suma3+suma3_priv

end

if length(int_vx_v1)==0 % si vx y v1 no tienen elementos comunes
  % tambien habra que restar flujo, entre el nodo de inters. de i con t,
  %  y los nodos de h puesto que este flujo se puede compartir si la
  % ruta t no es de metro.

  % quitamos los nodos de v1 a la ruta nv_h
  n_v1=length(v1)
  nv2_h=nv_h
  for p=1:n_v1
     inn=find(nv2_h==v1(p))    %inn: vector con los indices de los
                               % nodos de v1
     n_inn=length(inn)
     for d=n_inn:-1:1
       j=inn(d) % ir tomando los indices de los elementos de v1
       nv2_h=[nv2_h(1:j-1) nv2_h(j+1:end)]
     end
  end

  v2=intersect(ruta_i,ruta_t)
  if h==1
     alfa=alfa_metro

     beta=beta_b_m

  else
     alfa=alfa_bus

     beta=beta_b_b

  end
  if t==1
    metro=1
    suma4=sum(sum(Fij(v2,nv2_h)))*alfa+suma4

    suma4_priv=sum(sum(Fpriv(v2,nv2_h)))*beta+suma4_priv
    suma4_total=suma4+suma4_priv

  else
    cont=cont+1
    suma4=sum(sum(Fij(v2,nv2_h)))*alfa/2+suma4 %se reparte a
                                               %partes iguales el flujo

    suma4_priv=sum(sum(Fpriv(v2,nv2_h)))*beta/2+suma4_priv
    suma4_total=suma4+suma4_priv
```

```
end


% quitar v1 y v2 de la ruta t
n_v1=length(v1)
n_v2=length(v2)
nv2_t=ruta_t
for p=1:n_v1
   inn=find(nv2_t==v1(p))    %inn: vector con los indices de los
                             %nodos de v1
   n_inn=length(inn)
   for d=n_inn:-1:1
    j=inn(d) % ir tomando los indices de los elementos de v1
    nv2_t=[nv2_t(1:j-1) nv2_t(j+1:end)]
   end
end

for p=1:n_v2
   inn=find(nv2_t==v2(p))    % inn: vector con los indices de los
                             % nodos de v2
   n_inn=length(inn)
  for d=n_inn:-1:1
    j=inn(d) % ir tomando los indices de los elementos de v2
    nv2_t=[nv2_t(1:j-1) nv2_t(j+1:end)]
  end
end
if t==1
  alfa=alfa_metro

  beta=beta_b_m

  suma5=sum(sum(Fij(vx,nv2_t)))*alfa/2+suma5

  suma5_priv=sum(sum(Fpriv(vx,nv2_t)))*beta/2+suma5_priv
  suma5_total=suma5+suma5_priv


else
   alfa=alfa_bus

   beta=beta_b_b

  suma5=sum(sum(Fij(vx,nv2_t)))*alfa/2+suma5

  suma5_priv=sum(sum(Fpriv(vx,nv2_t)))*beta/2+suma5_priv
  suma5_total=suma5+suma5_priv

 end
end   % end del if v1~=vx
```

```
                    end
                end
              end


          end
        end

    end

    trans=suma1_total-suma2_total-suma3_total-suma4_total-suma5_total

    n_ri=length(ruta_i)
    for k=1:n_ri
      for h=k+1:n_ri
        if Dire(ruta_i(k),ruta_i(h))>N
          directo=directo+0
        else

directo=directo+(sum(Fij(ruta_i(k),ruta_i(h)))+sum(Fpriv(ruta_i(k),ruta_i(h))*beta_dir)
)/Dire(ruta_i(k),ruta_i(h))
        end
      end
    end

    carga(i)=trans+directo
  end
end
```