Implantación de Estrategias de Control del Factor de Potencia en el Parque Eólico Experimental de Sotavento

Descripción del algoritmo de contro

Versión, Mayo 2007l

Peticionario: Antonio Dorado Díaz
Sotavento Eólica
986 541519

Vigo, 11 de mayo de 2007

Grupo de Electrotecnia y Redes Eléctricas
Departamento de Ingeniería Eléctrica
Universidad de Vigo

Dirección:
E.T.S. de Ingenieros Industriales y Minas
Lagoas Marcosende, 9 36280 Vigo
Tel: 986 812221 Fax: 986 812173
Página WEB: http://www.le.uvigo.es
e-mail: jcidras@uvigo.es

ÍNDICE

1 Introducción

En este informe se va a describir el código necesario para programar la **versión de Mayo de 2007** del algoritmo de control de sistema central de la subestación.

El lenguaje que se empleará es el del código de MATLAB.

Las modificaciones con respecto a la versión anterior se marcarán con color amarillo.

Las principales modificaciones con respecto a la versión anterior son:

- Se establece una duración de segmento de 60s.
- Se lee el contador de las operaciones de los pasos de cada regulador de reactiva (variable **BatNOp**).
- El algorimo central de regulación de reactiva tiene en cuenta el número de operaciones de los pasos, minimizando el número de operaciones para alcanzar un determinado valor de reactiva. Además equilibra el número de operaciones entre los pasos iguales de cada aero. Para ello se introducen las funciones nuevas "**RAeroRankF" y "RAeroRank"**, además de modificarse la función "**VectorAcuBaterias"** pasándose a llamar "**VAcuBatRank".**
- Ahora todas las consignas para los reguladores son de estado de los pasos, igual que ocurría en la versión anterior con el regulador de la subestación.
- Se han corregido errores de programación con respecto a la versión anterior.

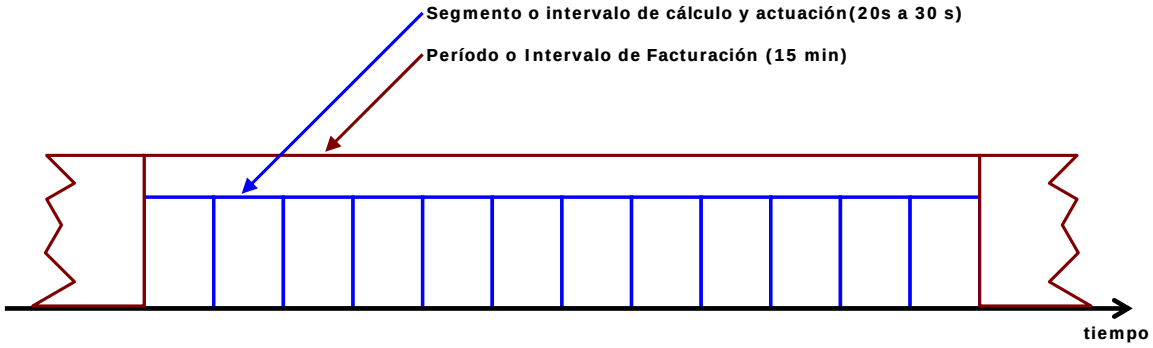
Además de lo anterior, deben tenerse en cuenta los comentarios realizados en el informe "**Evaluación del comportamiento del regulador central. Datos: 24/03/2007 – 26/03/2007. (REVISADO 25/04/2007**)" en lo referente a los ficheros de configuración, alarmas,... Añadir, que en cuanto a los ficheros de configuración, además de las constantes "mcm" y "kxo", debería incluirse la constante "kparabola", todas ellas de la función "**VAcuBatRank".**

2 Sobre los intervalos de tiempo

La escala de tiempos del sistema de control se divide en tres tipos de intervalo:

- Intervalo del lectura, durante el cual se leen todos los parámetros de los equipos de medida y compensación a nivel de aerogenerador y de máquina.
- Intervalo de cálculo y actuación, donde a partir del valor medio de las variables del intervalo de lectura se generarán las correspondientes consignas de aerogenerador y subestación.

El valor recomendado es de 60 s



3 Nomenclatura

A continuación se describe la nomenclatura más relevante empleada en la programación.

aero, es un vector de estructuras, donde cada elemento representa a un aerogenerador (24 x 1). Por cada aerogenerador se ha de disponer de la siguiente información:

Nombre: Nombre del aerogenerador

- Pnom: Potencia nominal en MW
- NBAT: Numero de baterias
- BAT: Pasos o escalones de reactiva instalados
- QcombF: Reactiva TOTAL de todas las combinaciones DISTINTAS de las baterias (MVAr)
- RankF: Incremento MINIMO del numero de operaciones para cada combinacion QcombF
- Regula: 0 (NO) y 1 (SI). Indica si el aero participa en la regulacion central
- AlarmaExt: 1 (aero fuera de servicio) y 0 (aero en operacion)
- P: Potencia activa (MW) media para el segmento
- Qcomp: Potencia reactiva compensada media para el segmento (MVAr)
- ~~Pu: Ultima potencia activa medida (MW)~~ **(obsoleta)**
- EstBat: Estado ACTUAL de baterias (0:Off, 1:On)
- EstBatobj: Estado Objetivo de baterias (0:Off, 1:On)
- BatNOp: Numero de operaciones de cada paso o escalón de reactiva
- Qc: Potencia reactiva generara por las baterias (MVAr)
- ~~FPobj: Factor de potencia OBJETIVO para el regulador (positivo = inductivo, negativo = capacitivo)~~ **(obsoleta)**
- vQcombF: Reactiva OBjetivo (MVAr)
- FPact: Factor de potencia alcanzado (positivo = inductivo, negativo = capacitivo)

Toda la información de los aerogeneradores se almacena en un vector de 24 elementos, donde cada uno de ellos es una estructura con los campos indicados arriba. De modo que la potencia nominal del aerogenerador de la torre nº 5 es:

aero(5). Pnom

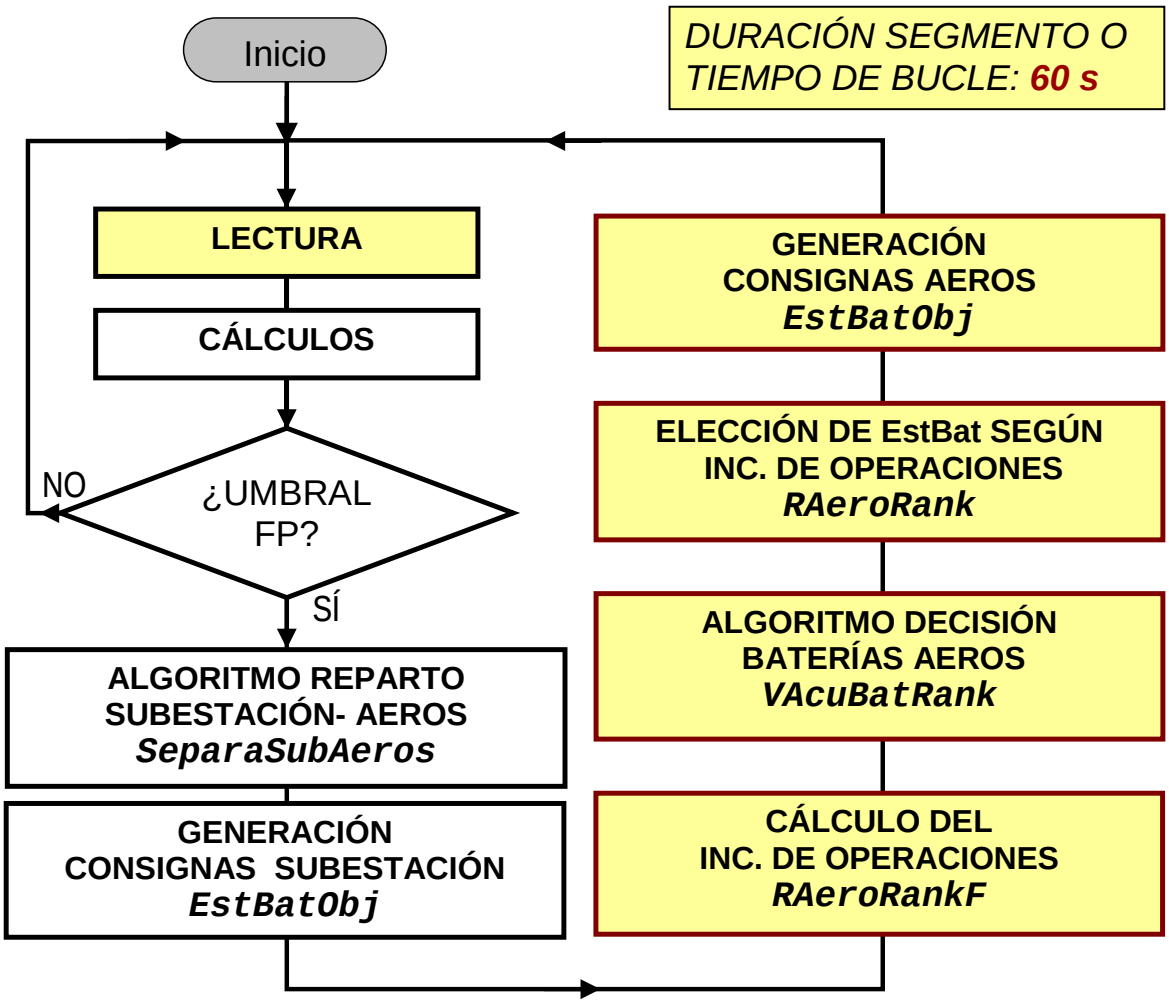
subAT, es una estructura similar a la aero, con los siguientes campos:

- NBAT: Numero de baterias
- BAT: Pasos o escalones de reactiva instalados en MVAr
- QcombF: Reactiva TOTAL de todas las combinaciones DISTINTAS de las baterias (MVAr)
- P: Potencia activa (MW) media para el segmento
- Qcomp: Potencia reactiva compensada media para el segmento (en MVAr)
- EstBat: Estado ACTUAL de baterias (0:Off, 1:On)
- EstBatOld: Estado ANTERIOR de baterias (0:Off, 1:On)
- EstBatobj: Estado Objetivo de baterias (0:Off, 1:On)
- BatNOp: Numero de operaciones de cada paso o escalón de reactiva
- Qc: Potencia reactiva generada por las baterias (MVAr)
- Pfact: Potencia activa promediada (en MW) en el período de 15 minutos de la facturacion
- Qfact: Potencia reactiva promediada (en MVAr) en el período de 15 minutos de la facturacion
- FPfact: Factor de potencia promediado en el período de 15 minutos de la facturacion
- Bonfica: Porcentaje de bonificación obtenido en el período de 15 minutos
- Qtesp: Reactiva esperada para todas las baterias (para el algoritmo de regulación)
- Trdesc: Tiempo restante para la descarga en s (para el algoritmo de regulación)

4 Diagrama de bloques del algoritmo

El algoritmo del regulador central se puede dividir en cuatro bloques:

- LECTURAS, donde se leerían las variables del CIRCUTOR CVMK y LOVATO DCRJ
- CÁLCULOS, donde se calcularían variables necesarias para el algoritmo
- ALGORITMO REPARTO SUBESTACIÓN AEROS (**SeparaSubAeros**), donde se decidiría que escalones o pasos se pondrían en funcionamiento en la subestación, es decir, se calcula el vector `EstBatObj`
- GENERACIÓN CONSIGNAS SUBESTACIÓN (**EstBatObj**), el valor calculado para el estado de baterías se envía al regulador de reactiva de la subestación.
- CÁLCULO DEL INC. DE OPERACIONES (**RAeroRankF**), donde en cada aero se calcula para cada valor de reactiva de reactiva alcanzable el número mínimo de operaciones necesarias teniendo en cuenta el estado actual de los pasos.
- ALGORITMO DECISIÓN BATERÍAS AEROS (**VAcuBatRank**), donde para cada aero decide cual es valor óptimo de la reactiva a generar en los condensadores. Para ello se tiene en cuneta el número de operaciones calculado en `RAeroRankF`.
- ELECCIÓN DE EstBat SEGÚN INC. DE OPERACIONES (**RAeroRank**), donde para cada aero decide cual es estado de los pasos más adecuado (`EstBatObj`) para generar la reactiva calculada en `VAcuBatRank`.
- GENERACIÓN CONSIGNAS AEROS (**EstBatObj**), el valor calculado para el estado de baterías se envía a cada uno de los reguladores de los aerogeneradores.


Inicio
DURACIÓN SEGMENTO O TIEMPO DE BUCLE: 60 s
LECTURA
CÁLCULOS
¿UMBRAL FP?
NO
SÍ
ALGORITMO REPARTO SUBESTACIÓN- AEROS
SeparaSubAeros
GENERACIÓN CONSIGNAS SUBESTACIÓN
EstBatObj
GENERACIÓN CONSIGNAS AEROS
EstBatObj
ELECCIÓN DE EstBat SEGÚN INC. DE OPERACIONES
RAeroRank
ALGORITMO DECISIÓN BATERÍAS AEROS
VAcuBatRank
CÁLCULO DEL INC. DE OPERACIONES
RAeroRankF

5 Descripción de las lecturas

Cabe diferenciar las medidas PERIÓDICA que han de realizarse cada 60s (período del segmento) y las medidas OFF-LINE, que habría que realizar únicamente cuando se realice alguna modificación en las baterías del parque.

MEDIDAS PERIÓDICAS:

- MEDIDOR CIRCUTOR CVMK
  - Potencia activa media (P) en MW
  - Potencia reactiva media (Qcomp) en MVAr
  - Potencia activa instántanea (Pu) en MW (sólo para la subestación)

Los valores de potencia activa media e instantánea deberían obtener de los valores de energía medidos por el CIRCUTOR en cada segmento.

- REGULADOR LOVATO
  - EstBat, estado de las baterías
  - Alarma, estado de funcionamiento del LOVATO y AEROGENERADOR
  - BatNOp, numero de operaciones de conexión de los escalones de reactiva

- MEDIDOR ENERGÍA FACTURACIÓN
  - EnergiaP, energía activa del período quince-minutal (MW-h
  - EnergiaQ, energía reactiva del período quince-minutal (MVAr-h)
  - Período Horario (esta variable podría generarse por software)

MEDIDAS OFFLINE

- BAT, baterías del aerogenerador y subestación (en MVAr)
- subAT.tdesc, tiempo de descarga de las baterías de la subestación (en s)

6 Posiciones de memoria del regulador LOVATO DCRJ12

En este apartado se describen los parámetros que se pueden leer del regulador LOVATO y su relación con las variables del algoritmo.

Variable: BAT

Vector con la potencia de los pasos o escalones de reactiva en MVAR, siendo su numero de elementos el numero de pasos instalados en cada aerogenerador. Está ordenado de mayor (primer elemento) a menor (último elemento) potencia del paso de reactiva.

Variables LOVATO relacionadas:

- 300h a 316h: En estas posiciones de consulta la potencia reactiva de las baterías en kVAr/100. Es necesario ordenarlas de mayor a menor una vez leidas.

Variable: EstBat

Vector con el estado de los pasos o escalones de reactiva, siendo su numero de elementos el numero de pasos instalados en cada aerogenerador.

Variables LOVATO relacionadas:

- 10h (bits: 0 a 11): Bits con 0 o 1 en función del estado del paso. El resultado es necesario ordenarlo siguiendo el mismo orden que el establecido en BAT. De moto que el primer elemento de EstBat se corresponda con el primer elemento de BAT.

Variable: AlarmaExt

Alarma que indica si el aerogenerador está o no en funcionamiento.

Variables LOVATO relacionadas:

- 16h (bit: 11): Bit relacionado con la alarma exterior del LOVATO.

Variable: BatNOp

Número acumulado de operaciones de conexión de los pasos o escalones de reacriva.

Variables LOVATO relacionadas:

- 100h a 116h: En estas posiciones de consulta el contador de operaciones.

Variable: tdesc

Indica el tiempo de descarga máximo de las baterías de condensadores.

Variables LOVATO relacionadas:

- 1603h: Tiempo de reconexión o descarga (en s)

Variable: FPobj

Indica la consigna de factor de potencia a indicarle al regulador.

Variables LOVATO relacionadas:

- 3000h: Cambio de modalidad operativa. Ha de estar en automático (=2)
- 200Dh: Ajuste del factor de potencia

Variable: EstBatobj

Indica el estado de baterías que ha de tener el regulador.

Variables LOVATO relacionadas:

- 3000h: Cambio de modalidad operativa. Ha de estar en manual (=1)
- 3005h: Para la conexión de un paso
- 3006h: Para la desconexión de un paso

Tabla 1: Funciones y direcciones de memoria del regulador LOVATO DCRJ12

Función	Dirección	Palabra	Descripción	Rango	Tipo Datos
4	10h	1	Estado de las salidas		unsigned long
4	16h	1	Estado de las alarmas		unsigned long
4	100h a 116h	2	Contador de operaciones de los pasos		unsigned long
4	300h a 316h	2	Potencias del paso 1 al paso 12 (en kVAr/100)		unsigned long
6	200Dh	1	Ajuste del factor de potencia 80…99 = 0.80… 0.9 ind 100 = 1.00 101…120 = 0.99…0.80 cap	80 a 120	unsigned long
6	3000h	1	Cambio de modalidad operativa (0 paso de manual a automático y viceversa; 1 paso a manual; 2 paso a automático)		unsigned integer
6	3004h	1	Almacenamiento de parámetros en la EEPROM del regulador. 0: Almacenamiento en la EEPROM de los valores de la memoria 1: Almacenamiento en la EEPROM de los valores de la memoria + Restablecimiento de la potencia original de los escalones		unsigned integer
6	3005h	1	Conexión de un paso. En esta posición de memoria se escribe el número del paso que se activa. *Si se intenta conectar un escalón mientras se está descargando esta orden se ignora.*		unsigned integer
6	3006h	1	Desconexión de un paso En esta posición de memoria se escribe el número del paso que se desconecta		unsigned integer
4	1601h	1	Valor de reactiva del paso más pequeño (en kVAr)	10 a 30000	unsigned integer
4	1603h	1	Tiempo de reconexión o descarga (en s)	5 a 240	unsigned integer
4	1605h	1	Coeficiente del paso 1	0 a 16	unsigned integer
4	1606h	1	Coeficiente del paso 2	0 a 16	unsigned integer
…	…	…	…	…	…
4	1605h + Ah	1	Coeficiente del paso 11	0 a 19	Unsigned integer
4	1605h + Bh	1	Coeficiente del paso 12	0 a 19	Unsigned integer

7 Modificaciones en el código

Programa principal del regulador central

PROPUESTO

```
 %Lectura de datos
% Potencia activas y reactivas medias de cada aero (aero(kae).P y aero(kae).Q
% Potencia activas instantanea de cada aero (aero(kae).Pu
% Potencia activas y reactivas medias de la subestacion (subAT.P y subAT.Q)
% Estado de baterias de los reguladores (aero(kae).EstBat)
% Estado de baterias del regulador de la subestacion (subAT.EstBat)
% Estado de la alarma exterior de los reguladores (aero(kae).AlarmaExt)
% Periodo horario (PHorario)
% Numero de operaciones de cada paso

[aero,subAT,PHorario] = LecturaDatos;
for kae = AerosRegula
    if aero(kae).AlarmaExt == 1
        aero(kae).Regula = 0;
    else
        aero(kae).Regula = 1;
    end
    aero(kae).Qc = aero(kae).EstBat*aero(kae).BAT';
    FP=cos(atan2(aero(kae).Qcomp,aero(kae).P));
    SIGNO = sign(aero(kae).Qcomp); if SIGNO== 0, SIGNO = 1; end
    aero(kae).FPact = -1*FP*SIGNO;
end
subAT.Qc = subAT.EstBat*subAT.BAT';

FP=cos(atan2(subAT.Qcomp,subAT.P));
SIGNO = sign(subAT.Qcomp); if SIGNO== 0, SIGNO = 1; end
subAT.FPact = -1*FP*SIGNO;

 %Calculo del tiempo restante para la descarga de los condesadores de la subestacion
 %Control del tiempo de descarga de los condensadores para el algoritmo central
for k=1:subAT.NBAT
   if subAT.EstBat(k) == 1
       subAT.trdesc(k) = 0;
   elseif subAT.EstBat(k)==0 & subAT.EstBatOld(k)==1  %En la desconexion empieza a contar el
tiempo de descarga
       subAT.trdesc(k) = subAT.tdesc;
   elseif subAT.EstBat(k)==0 & subAT.EstBatOld(k)==0 & subAT.trdesc(k)>0 %Durante la
desconexion cuenta el tiempo de descarga
       subAT.trdesc(k) = subAT.trdesc(k) - Ts;
   end
   if subAT.trdesc(k) < 0
       subAT.trdesc(k) = 0;
   end
end

 %Calculo de la bonificacion en cada periodo
[subAT.Bonifica,subAT.FPfact,subAT.Pfact,subAT.Qfact] = CalcBonifica(subAT,kT,Ts,PHorario);

 %El FPObj del parque depende del horario
if PHorario == 1      % Punta <<<>>>
    subAT.FPobj = -0.9;
    FPlim = [-0.88 -0.92];
elseif PHorario == -1 % Valle <<<>>>
    subAT.FPobj = 0.9;
    FPlim = [+0.88 +0.92];
else                  % Llano <<<>>>
```

ANTERIOR

```
%Lectura de datos
    % Potencia activas y reactivas medias de cada aero (aero(kae).P y aero(kae).Q
    % Potencia activas instantanea de cada aero (aero(kae).Pu
    % Potencia activas y reactivas medias de la subestacion (subAT.P y subAT.Q)
    % Estado de baterias de los reguladores (aero(kae).EstBat)
    % Estado de baterias del regulador de la subestacion (subAT.EstBat)
    % Estado de la alarma exterior de los reguladores (aero(kae).AlarmaExt)
    % Periodo horario (PHorario)


    [aero,subAT,PHorario] = LecturaDatos;
    for kae = AerosRegula
        if aero(kae).AlarmaExt == 1
            aero(kae).Regula = 0;
        else
            aero(kae).Regula = 1;
        end
        aero(kae).Qc = aero(kae).EstBat*aero(kae).BAT';
FP=cos(atan2(aero(kae).Qcomp,aero(kae).P));
        SIGNO = sign(aero(kae).Qcomp); if SIGNO== 0, SIGNO = 1; end
        aero(kae).FPact = -1*FP*SIGNO;
    end
    subAT.Qc = subAT.EstBat*subAT.BAT';

    FP=cos(atan2(subAT.Qcomp,subAT.P));
    SIGNO = sign(subAT.Qcomp); if SIGNO== 0, SIGNO = 1; end
    subAT.FPact = -1*FP*SIGNO;

    %Calculo del tiempo restante para la descarga de los condesadores de la subestacion
    %Control del tiempo de descarga de los condensadores para el algoritmo central
    for k=1:subAT.NBAT
       if subAT.EstBat(k) == 1
           subAT.trdesc(k) = 0;
       elseif subAT.EstBat(k)==0 & subAT.EstBatOld(k)==1  %En la desconexion empieza a contar
el tiempo de descarga
           subAT.trdesc(k) = subAT.tdesc;
       elseif subAT.EstBat(k)==0 & subAT.EstBatOld(k)==0 & subAT.trdesc(k)>0 %Durante la
desconexion cuenta el tiempo de descarga
           subAT.trdesc(k) = subAT.trdesc(k) - Ts;
       end
       if subAT.trdesc(k) < 0
           subAT.trdesc(k) = 0;
       end
    end

    %Calculo de la bonificacion en cada periodo
    [subAT.Bonifica,subAT.FPfact,subAT.Pfact,subAT.Qfact] = CalcBonifica(subAT,kT,Ts,PHorario);

    %El FPObj del parque depende del horario
    if PHorario == 1      % Punta <<<>>>
        subAT.FPobj = -0.9;
        FPlim = [-0.88 -0.92];
    elseif PHorario == -1 % Valle <<<>>>
        subAT.FPobj = 0.9;
        FPlim = [+0.88 +0.92];
    else                  % Llano <<<>>>
```

```
        subAT.FPobj = 1;
        FPlim = [-0.998 +0.998];
    end

    %Si el factor de potencia del parque esta dentro los limites de FPlim
    % lo damos por valido y el regulador central no actua.
    ActuaRegulador = 1;
    if     (PHorario ==  1) & (subAT.FPact < FPlim(1)) & (subAT.FPact > FPlim(2))
           ActuaRegulador = 0;
    elseif (PHorario == -1) & (subAT.FPact > FPlim(1)) & (subAT.FPact < FPlim(2))
           ActuaRegulador = 0;
    elseif (PHorario ==  0) & ((subAT.FPact < FPlim(1)) | (subAT.FPact > FPlim(2)))
           ActuaRegulador = 0;
    end

    %ACTUACION DEL REGULADOR CENTRAL DEL PARQUE
    if ActuaRegulador == 1
        %Calculo de la reactiva de deberan generar todos las baterias del parque
        Qobj = - subAT.P*tan(acos(subAT.FPobj));
        DQ = Qobj - subAT.Qcomp;
        subAT.QTesp = DQ + subAT.Qc;
        for kae=1:NAeros
            if aero(kae).Regula == 1
                subAT.QTesp = subAT.QTesp + aero(kae).Qc;
            end
        end

        %!!!! MODIFICACION 14/03/07:
        %Calculamos el numero de operaciones minimo para alcanzar todos
        %los posibles estados de baterias en todos los aeros.
        for kae=1:NAeros
            if aero(kae).Regula == 1
                [regula(kae).Rank,aero(kae).RankF] = RAeroRankF(aero(kae),regula(kae));
            else
               regula(kae).Rank = 0; aero(kae).RankF = 0;
            end
        end

        %CALCULO DE CONSIGNAS >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
        % Calculo de la reactiva que debe aportar las baterias de la
        % subestacion (Qsub) y los aerogeneraores (Qesp)
        % EstBatobj es el estado de baterias de la subestacion
        [Qesp,Qsub,EstBatobj]=SeparaSubAeros(aero,subAT,subAT.QTesp);
        for i=1:1:NAeros
           Psegm(i)=aero(i).P;
        end

        %Calculo de la reactiva a aportar por las baterias de los aeros
        [eQcombF,vQcombF]= VAcuBatRank(aero,Psegm,Qesp);

        % Establecimiento del objetivo para la subestacion
        subAT.EstBatobj=EstBatobj;
        subAT.EstBatOld = subAT.EstBat;

        % Establecimiento del objetivo para los aeros
        for kae=1:1:NAeros
            if aero(kae).Regula == 1
                %Actuacion por estado de baterias
                %!!!! MODIFICACION 14/03/07:
                %Calculamos la combinacion de pasos optima con
                %objeto de equilibrar el numero de operaciones a
                %realizar en cada uno de los pasos de los aeros.
                aero(kae).EstBatobj = RAeroRank(aero(kae),regula(kae),vQcombF(kae));
            elseif aero(kae).AlarmaExt == 1
                aero(kae).EstBatobj = zeros(1,aero(kae).NBAT);
```

```
        subAT.FPobj = 1;
        FPlim = [-0.998 +0.998];
    end

    %Si el factor de potencia del parque esta dentro los limites de FPlim
    % lo damos por valido y el regulador central no actua.
    ActuaRegulador = 1;
    if     (PHorario ==  1) & (subAT.FPact < FPlim(1)) & (subAT.FPact > FPlim(2))
           ActuaRegulador = 0;
    elseif (PHorario == -1) & (subAT.FPact > FPlim(1)) & (subAT.FPact < FPlim(2))
           ActuaRegulador = 0;
    elseif (PHorario ==  0) & ((subAT.FPact < FPlim(1)) | (subAT.FPact > FPlim(2)))
           ActuaRegulador = 0;
    end

    %ACTUACION DEL REGULADOR CENTRAL DEL PARQUE
    if ActuaRegulador == 1
        %Calculo de la reactiva de deberan generar todos las baterias del parque
        Qobj = - subAT.P*tan(acos(subAT.FPobj));
        DQ = Qobj - subAT.Qcomp;
        subAT.QTesp = DQ + subAT.Qc;
        for kae=1:NAeros
            if aero(kae).Regula == 1
                subAT.QTesp = subAT.QTesp + aero(kae).Qc;
            end
        end

        %CALCULO DE CONSIGNAS >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
        % Calculo de la reactiva que debe aportar las baterias de la
        % subestacion (Qsub) y los aerogeneraores (Qesp)
        % EstBatobj es el estado de baterias de la subestacion
        [Qesp,Qsub,EstBatobj]=SeparaSubAeros(aero,subAT,subAT.QTesp);
        for i=1:1:NAeros
           Psegm(i)=aero(i).P;
        end

        %Calculo de la reactiva a aportar por las baterias de los aeros
        [eQcombF,vQcombF]=VectorAcuBaterias(aero,Psegm,Qesp);

        % Establecimiento del objetivo para la subestacion
        subAT.EstBatobj=EstBatobj;
        subAT.EstBatOld = subAT.EstBat;

        % Establecimiento del objetivo para los aeros
        for kae=1:1:NAeros
            if aero(kae).Regula == 1
                Qaux = aero(kae).Qcomp-sum(aero(kae).EstBat.*aero(kae).BAT)+vQcombF(kae);
                FPaux=cos(atan2( Qaux , aero(kae).Pu ));
                %El regulador LOVATO no admite consignas inferiores a 0.8
                if FPaux < 0.8
                    FPaux = 0.8;
                end
                %FP > 0 Inductivo; FP < 0 Capacitivo
                SIGNO = sign(Qaux); if SIGNO==0; SIGNO=1; end
```

```
            end
        end

    end %ActuaRegulador

    %ESCRITURA DE CONSIGNAS
    %_____________________________________
%SIMULACION DEL COMPORTAMIENTO DEL PARQUE
```

```
                aero(kae).FPobj = -SIGNO*FPaux;
            end
        end

    end %ActuaRegulador

    %ESCRITURA DE CONSIGNAS
    %_____________________________________
%SIMULACION DEL COMPORTAMIENTO DEL PARQUE
```

SeparaSubAeros (SIN CAMBIOS)

```
function [Qesp,Qsub,EstBatobj]=SeparaSubAeros(aero,subAT,QTesp)  %CALCULA Qesp(dan aeros) y Qsub
pminhist=0.1; %coef multiplicador en limite inferior de histeresis
pmaxhist=0.9; %coef multiplicador en limite superior de histeresis

Qesp=0;
naeros=size(aero,1);

%maxQae: suma toda la Q de C aeros que pueden participar en la regulacion
maxQae=0;
for i=1:1:naeros
    if aero(i).NBAT>0 & aero(i).Regula>0
        maxQae = maxQae + aero(i).QcombF(size(aero(i).QcombF,1));
    end
end

%se determina los limites de histeresis para las bat de sub conectadas en ese instante
if sum(subAT.EstBat)>0
    minhist=pminhist*maxQae;
else
    minhist=0;
end
if sum(subAT.EstBat)<subAT.NBAT
    maxhist=pmaxhist*maxQae;   % falta limitar su valor para Psub pequeña
else
    maxhist=maxQae;
end

%QsubBat: suma toda la Q de las baterias conectadas en la sub
% operador '.*': producto 1 a 1 de elementos de 2 vectores
QsubBat=sum(subAT.EstBat.*subAT.BAT);

%si la QTesp se puede dar sin cambiar las baterias de sub o es menos y estan todas desconectadas, o es mas y estan todas conectadas, se queda como esta
if (QTesp-QsubBat>minhist & QTesp-QsubBat<maxhist) | (QTesp-QsubBat<=minhist & sum(subAT.EstBat)==0) | (QTesp-QsubBat>=maxhist & sum(subAT.EstBat)==subAT.NBAT)
    Qesp=QTesp-QsubBat;
    Qsub=QsubBat;
    EstBatobj=subAT.EstBat;
    return;
end

% se calcula la combinacion de Q de baterias de subestacion M, que baterias la dan B y que Q total es cada combinacion Ms
M=[];
B=[];
for k=1:1:subAT.NBAT
    aux=nchoosek(subAT.BAT,k);
    aux2=nchoosek([1:1:subAT.NBAT],k);
    sM=size(M);
    M=[M zeros(sM(1),size(aux,2)-sM(2));aux];  %matriz de combinaciones Q
    B=[B zeros(sM(1),size(aux,2)-sM(2));aux2];  %matriz de indices combinaciones
end
M=[zeros(1,size(M,2)); M];
B=[zeros(1,size(B,2)); B];
Ms=sum(M')';  %pot suma para cada combinacion

%Se determina que combinaciones es o no posible emplear en este instante portiempos de descarga
posible=ones(size(Ms,1),1);
[im,jm]=size(M);
for i=1:1:im
    for j=1:1:jm
        if B(i,j)>0 & subAT.trdesc(B(i,j))>0
            posible(i)=0;
        end
    end
end
```

```
% si no hay ninguna posible, se queda como esta
if sum(posible)==0
    Qesp=QTesp-QsubBat;
    Qsub=QsubBat;
    EstBatobj=subAT.EstBat;
    return;
end

%si hay alguna posible, busca la que haria trabajar a mitad de Q en aeros
%aun no tiene en cuenta si es punta, llano o valle, para decidir la mejor
mejor=1e30;
imejor=0;
for i=1:1:im
    if posible(i)
        desvmedio=(QTesp-Ms(i))-maxQae/2;
        if abs(desvmedio)<mejor
            mejor=abs(desvmedio);
            imejor=i;
        end
    end
end
Qesp=QTesp-Ms(imejor);
Qsub=Ms(imejor);
EstBatobj=zeros(size(subAT.EstBat));
for j=1:1:size(B,2)
    if B(imejor,j)>0
        EstBatobj(B(imejor,j))=1;
    end
end
return;
```

VAcuBatRank & VectorAcuBaterias

PROPUESTO

```
function [eQcombF,vQcombF]=VAcuBatRank(aero,Psegm,Qesp)
mcm=0.0125; % paso del escalon
kxo=.05;   % factor de funcion de coste
kparabola=1e-3;

naeros=size(aero,1);

%si la Q<=0 todo desconectado
if Qesp<=0
    eQcombF=zeros(1,naeros);
    vQcombF=zeros(1,naeros);
    return
end

%maxQae: suma toda la Q de C aeros que pueden participar en la regulacion
maxQae=0;
for i=1:1:naeros
    if aero(i).NBAT>0 & aero(i).Regula>0
        maxQae = maxQae + aero(i).QcombF(size(aero(i).QcombF,1));
    end
end
nmaxQae=1+round(maxQae/mcm);  %tamaño maximo de la matriz para el escalon seleccionado
%inicializacion de matriz de vectores de acumulacion. Cada fila es un vector de acumulacion
MVacu.M=zeros(size(aero,1) , nmaxQae);

% calculo de matriz de vectores de acumulacion.
for i=1:1:naeros
   if i==1    %para el primer vector, son los datos del primer aero
      if aero(i).Regula>0      %si aero regula, tiene todos los escalones y sino solo Qc=0
(indice=1)
         jmax=size(aero(i).QcombF,1);
      else
         jmax=1;
      end
      for j=1:1:jmax
         %calcula la funcion de coste del aero i si quiere dar la Q de la combinacion j
         x=aero(i).QcombF(j);
         xo=kxo*(Psegm(i)-aero(i).Pnom)/aero(i).Pnom;
         cx=(x-xo)^2;                                                    %coste
         %!!!! MODIFICACION 16/03/07 ELO. Inlcuimos el numero de
         %operaciones en la funcion de coste
         MVacu.M(i,1+ round(aero(i).QcombF(j)/mcm))=aero(i).RankF(j)+kparabola*cx;
         MVacu.N(i,1+ round(aero(i).QcombF(j)/mcm)).E=j;
         MVacu.N(i,1+ round(aero(i).QcombF(j)/mcm)).V=aero(i).QcombF(j);
      end
   else  %para los demas vectores, son los datos del aero i mas el vector del acumulado en  i-1
      for k=1:1:nmaxQae
         if k==1 | MVacu.M(i-1,k)~=0
            if aero(i).Regula>0    %si aero regula, tiene todos los escalones y sino solo
Qc=0 (indice=1)
               jmax=size(aero(i).QcombF,1);
            else
               jmax=1;
            end
            for j=1:1:jmax
               %calcula la funcion de coste del aero i si quiere dar la Q de la combinacion j
                x=aero(i).QcombF(j);
                xo=kxo*(Psegm(i)-aero(i).Pnom)/aero(i).Pnom;
                cx=(x-xo)^2;                                    %coste
                %!!!! MODIFICACION 16/03/07 ELO. Inlcuimos el numero de
                %operaciones en la funcion de coste
```

ANTERIOR

```
function [eQcombF,vQcombF]=VectorAcuBaterias(aero,Psegm,Qesp)
mcm=0.0125; % paso del escalon
kxo=.05;   % factor de funcion de coste


naeros=size(aero,1);

%si la Q<=0 todo desconectado
if Qesp<=0
    eQcombF=zeros(1,naeros);
    vQcombF=zeros(1,naeros);
    return
end

%maxQae: suma toda la Q de C aeros que pueden participar en la regulacion
maxQae=0;
for i=1:1:naeros
    if aero(i).NBAT>0 & aero(i).Regula>0
        maxQae = maxQae + aero(i).QcombF(size(aero(i).QcombF,1));
    end
end
nmaxQae=1+round(maxQae/mcm);  %tamaño maximo de la matriz para el escalon seleccionado
%inicializacion de matriz de vectores de acumulacion. Cada fila es un vector de acumulacion
MVacu.M=zeros(size(aero,1) , nmaxQae);

% calculo de matriz de vectores de acumulacion.
for i=1:1:naeros
   if i==1    %para el primer vector, son los datos del primer aero
      if aero(i).Regula>0      %si aero regula, tiene todos los escalones y sino solo Qc=0
(indice=1)
         jmax=size(aero(i).QcombF,1);
      else
         jmax=1;
      end
      for j=1:1:jmax
         %calcula la funcion de coste del aero i si quiere dar la Q de la combinacion j
         x=aero(i).QcombF(j);
         xo=kxo*(Psegm(i)-aero(i).Pnom)/aero(i).Pnom;
         cx=(x-xo)^2;                                                    %coste


         MVacu.M(i,1+ round(aero(i).QcombF(j)/mcm))=cx;
         MVacu.N(i,1+ round(aero(i).QcombF(j)/mcm)).E=j;
         MVacu.N(i,1+ round(aero(i).QcombF(j)/mcm)).V=aero(i).QcombF(j);
      end
   else  %para los demas vectores, son los datos del aero i mas el vector del acumulado en  i-1
      for k=1:1:nmaxQae
         if k==1 | MVacu.M(i-1,k)~=0
            if aero(i).Regula>0    %si aero regula, tiene todos los escalones y sino solo
Qc=0 (indice=1)
               jmax=size(aero(i).QcombF,1);
            else
               jmax=1;
            end
            for j=1:1:jmax
               %calcula la funcion de coste del aero i si quiere dar la Q de la combinacion j
                x=aero(i).QcombF(j);
                xo=kxo*(Psegm(i)-aero(i).Pnom)/aero(i).Pnom;
                cx=(x-xo)^2;                                    %coste
```

```
                    %se queda con el de menor coste
                    if MVacu.M(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm))==0 | MVacu.M(i,1+ (k-1)
+ round(aero(i).QcombF(j)/mcm)) >   MVacu.M(i-1,k) + aero(i).RankF(j)+kparabola*cx
                        MVacu.M(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm))=  MVacu.M(i-1,k) +
aero(i).RankF(j)+kparabola*cx   ;
                        MVacu.N(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm)).E=[MVacu.N(i-1,k).E
,   j   ];
                        MVacu.N(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm)).V=[MVacu.N(i-1,k).V
,   aero(i).QcombF(j)  ];
                    end
                end
            end
        end
    end
end


nQesp=1+round(Qesp/mcm);  % indice del vector reactiva/baterías ( ultima fila de MVacu.M(:,:)
) para la Q especificada

if nQesp>nmaxQae                        % no puede dar mas del maximo
    nQesp=nmaxQae;
end
%si el elemento del vector correspondiente a la Qesp es nulo, busca el mas proximo
%esta busqueda deberia ser en diferente sentido, en funcion de si es punta o valle, y asi en
llano
if nQesp~=1 & MVacu.M(naeros,nQesp)==0
    j=1;
    while (nQesp+j<=nmaxQae & MVacu.M(naeros,nQesp+j)==0) | (nQesp-j>=1 & MVacu.M(naeros,nQesp-
j)==0)
        j=j+1;
    end
    if nQesp-j>=1
        nQesp=nQesp-j;
    else
       if nQesp+j<=nmaxQae
          nQesp=nQesp+j;
       else
          nQesp=1; %MODIFICACION
       end
    end
end
eQcombF=MVacu.N(naeros,nQesp).E;  %vector con el indice del escalon de "aero(i).QcombF[1..]"
conectado en cada aero
vQcombF=MVacu.N(naeros,nQesp).V;  %vector con el valor de reactiva conectada en cada aero
```

```
                    %se queda con el de menor coste
                    if MVacu.M(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm))==0 | MVacu.M(i,1+ (k-1)
+ round(aero(i).QcombF(j)/mcm)) >   MVacu.M(i-1,k) + cx
                        MVacu.M(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm))=  MVacu.M(i-1,k) + cx
;
                        MVacu.N(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm)).E=[MVacu.N(i-1,k).E
,   j   ];
                        MVacu.N(i,1+ (k-1) + round(aero(i).QcombF(j)/mcm)).V=[MVacu.N(i-1,k).V
,   aero(i).QcombF(j)  ];
                    end
                end
            end
        end
    end
end


nQesp=1+round(Qesp/mcm);  % indice del vector reactiva/baterías ( ultima fila de MVacu.M(:,:)
) para la Q especificada

if nQesp>nmaxQae                        % no puede dar mas del maximo
    nQesp=nmaxQae;
end
%si el elemento del vector correspondiente a la Qesp es nulo, busca el mas proximo
%esta busqueda deberia ser en diferente sentido, en funcion de si es punta o valle, y asi en
llano
if nQesp~=1 & MVacu.M(naeros,nQesp)==0
    j=1;
    while (nQesp+j<=nmaxQae & MVacu.M(naeros,nQesp+j)==0) | (nQesp-j>=1 & MVacu.M(naeros,nQesp-
j)==0)
        j=j+1;
    end
    if nQesp-j>=1
        nQesp=nQesp-j;
    else
       if nQesp+j<=nmaxQae
          nQesp=nQesp+j;
       else
          nQesp=0;
       end
    end
end
eQcombF=MVacu.N(naeros,nQesp).E;  %vector con el indice del escalon de "aero(i).QcombF[1..]"
conectado en cada aero
vQcombF=MVacu.N(naeros,nQesp).V;  %vector con el valor de reactiva conectada en cada aero
```

RegulaAERO_Rank (nueva)

```
function EstBatCombF = RegulaAERO_Rank(aero,regula,Qbat)
%
% function EstBatCombF = RegulaAERO_Rank(aero,regula,Qbat)
% En esta funcion se calcula la combinacion de pasos que aporta la reactiva
% especificada por Qbat atendiendo a varios criterios.
%
% ENTRADA
%    aero
%    regula
%    Qbat: reactiva capacitiva demandada a cada aero.
%
% SALIDA
%   EstBatCombF: Estado de baterias seleccionado
%   uno de los elementos de Qcomb o EstBatComb.
%

%Comprobamos si la reactiva actual es igual a la objetivo
QAct = sum(aero.EstBat.*aero.BAT);
if abs(Qbat-QAct)<1e-6
    EstBatCombF = regula.EstBat;
    return
end

% Buscamos las combinaciones que den la reactiva especificada por Qbat
IndQ = (regula.Qcomb<=Qbat+1e-6)&(regula.Qcomb>=Qbat-1e-6);

% Buscamos las combinaciones que impliquen el menor número de operaciones
IndF = find((aero.QcombF<=Qbat+1e-6)&(aero.QcombF>=Qbat-1e-6));
RankMin = min(aero.RankF(IndF));
IndR =  regula.Rank==RankMin;

% Buscamos las combinaciones que cumplan los dos criterios anteiores
IndSel = find(IndQ.*IndR);

%Si sólo hay combinacion seleccionada: FIN
if length(IndSel)==1
    EstBatCombF = regula.EstBatComb(IndSel,:);
    return
end

EstBatAct = aero.EstBat; %Estado actual de las baterias

%Seleccion de las combinaciones posibles de baterías.
EstBatComb = regula.EstBatComb(IndSel,:);
NComb = size(EstBatComb,1);

%Creamos una matriz de indicadores de operaciones
%según los criterios indicados a continuación.
for k=1:NComb
    EstBat = EstBatComb(k,:);
    Crit = 0;

    %Criterio nº1: Numero de operaciones ACUMULADAS
    Crit = Crit + 1;
    EstBatA = xor(EstBat,EstBatAct);
    Ind = find(EstBatA);
    if isempty(Ind)==1
        NOpA = 0;
    else
        NOpA = sum(aero.BatNOp(Ind));
    end
    Rank(k,Crit) = NOpA;
```

```
 %Criterio nº2: Numero de operaciones ACUMULADAS en PASOS A SACAR
 Crit = Crit + 1;
 EstBatA = not(EstBat)&EstBatAct;
 Ind = find(EstBatA);
 if isempty(Ind)==1
     NOpA = 0;
 else
     NOpA = sum(aero.BatNOp(Ind));
 end
 Rank(k,Crit) = NOpA;

%Criterio nº3: Numero de operaciones ACUMULADAS en PASOS A METER
 Crit = Crit + 1;
 EstBatA = EstBat & not(EstBatAct);
 Ind = find(EstBatA);
 if isempty(Ind)==1
     NOpA = 0;
 else
     NOpA = sum(aero.BatNOp(Ind));
 end
 Rank(k,Crit) = NOpA;

%Criterio nº4: Operaciones de la bateria a meter menos utilizada
 Crit = Crit + 1;
 EstBatA = EstBat & not(EstBatAct);
 Ind = find(EstBatA);
 if isempty(Ind)==1
     NOpA = 0;
 else
     NOpA = min(aero.BatNOp(Ind));
 end
 Rank(k,Crit) = NOpA;
end

%Buscamos la mejor combinacion de baterias segun TODOS los criterios
CritM = Crit;
Crit = 1; Sigue = 1;
while Sigue
      RankMin = min(Rank(:,Crit));
      Ind = find(Rank(:,Crit)==RankMin);
      Rank = Rank(Ind,:);
      EstBatComb = EstBatComb(Ind,:);
      if length(Ind)>1 & Crit < CritM
           Crit = Crit + 1;
      else
           Sigue = 0;
      end
end

NComb = size(EstBatComb,1);
if NComb>1
    Ind = round(rand*(NComb-1)) + 1;
    EstBatCombF = EstBatComb(Ind,:);
else
    EstBatCombF = EstBatComb;
end
```

RAeroRankF (nueva)

```
function [Rank,RankF] = RAeroRankF(aero,regula)
%
% function [Rank,RankF] = RAeroRankF(aero,regula)
% En esta funcion se calcula el numero de operaciones de CONEXION y
% DESCONEXION necesarias para pasar desde el estado actual de baterias (aero.EstBat) a
% alguno de los estados posibles (regula.Qcomb).
%
% ENTRADA
%    aero
%    regula
%
% SALIDA
%   Rank: Vector con el numero de operaciones necesarias para pasar a cada
%   uno de los elementos de Qcomb o EstBatComb.
%   RankF: Vector con el minimo numero de operaciones necesarias para pasar a cada
%   uno de los elementos de QcombF
%

EstBatAct = aero.EstBat; %Estado actual de las baterias
[NComb,NBAT] = size(regula.EstBatComb);

%Establecemos un RANKING segun el numero de operaciones entre la bateria
%actual y la posible
Rank = zeros(NComb,1);
for k=1:NComb
    EstBat = regula.EstBatComb(k,:);
    %Criterio nº1: Numero de operaciones Con/Desc para pasar
    % de EstBatAct a EstBat
    NOp = sum(xor(EstBat,EstBatAct));
    Rank(k,1) = NOp;
end

%Filtramos el resultado de modo que para cada valor posible de reactiva se
%le asigna el minimo numero de pasos para conseguirlo.
NF = length(aero.QcombF);
RankF = zeros(NF,1);
NOpTotal = sum(aero.BatNOp);
for k=1:NF
    Q = aero.QcombF(k);
    Ind = find((regula.Qcomb<=Q+1e-6)&(regula.Qcomb>=Q-1e-6));
    %Buscamos la mejor combinacion de baterias segun TODOS los criterios
    RankMin = min(Rank(Ind,1));
    RankF(k) = RankMin;
end
```