Model Name: **h6b**

Description: H6b hydro turbine governor model

Prerequisite: h6bd data management model ahead of this model in the input data file

generator model ahead of this model in the input data file

Inputs: Shaft speed
Generator electrical power

Parameters:

Name	Description	Value
tw	water inertia time constant, sec	2.00
ptp	user-defined gate boost input, pu	0.02
ftp	frequency deviation for transient boost to pick up, pu	0.005
ttp	definite time delay for transient boost pickup	1.00
tpw	washout time constant for gate boost	30.00
vtp	velocity limit on feedforward signal	0.002
kfp	feedforward gain for speed-load reference	0.00
tff	feedforward filter time constant	1.00
fbus	bus number of 'family' model (hyg6d)	

Notes:

a) Per unit parameters are on base of turbine MW capability. The base power for the must be stated by means of the "mwcap=" entry in this record. The turbine base power entered in the associated h6bd data record is NOT USED. If no value is entered for "mwcap", the generator MVA base is used.

b) The turbine and the feedback functions of the governor are described by the parameters entered and maintained by the h6bd model. See the h6bd model data page for notes on the turbine and basic governor modeling.

c) Two separate feedforward paths can be used to maneuver the turbine power output by addition of a signal to the gate position command of the governor.

Changes in the speed-load reference, genbc[].pref, are fed forward through the gain, Kfp. This feedforward gives a quick open-loop response to changes in the speed-load reference of the governor.

The externally provided signal, Ptp, is fed forward to provide an open-loop adjustment of gate opening upon command. The application of this feedforward signal is initiated when frequency measured at the generator terminals deviates from 1.0 per unit by more than <ftp> per unit for longer than <ttp> seconds.

d) The load controller model, lcfb1, should have its first parameter, type, set to zero when it is applied to h6b.

Output Channels:

Record Level	Name	Description
1	gate	Turbine gate position, pu
1	pm	Turbine power, MW
1	bld	Turbine blade position, pu

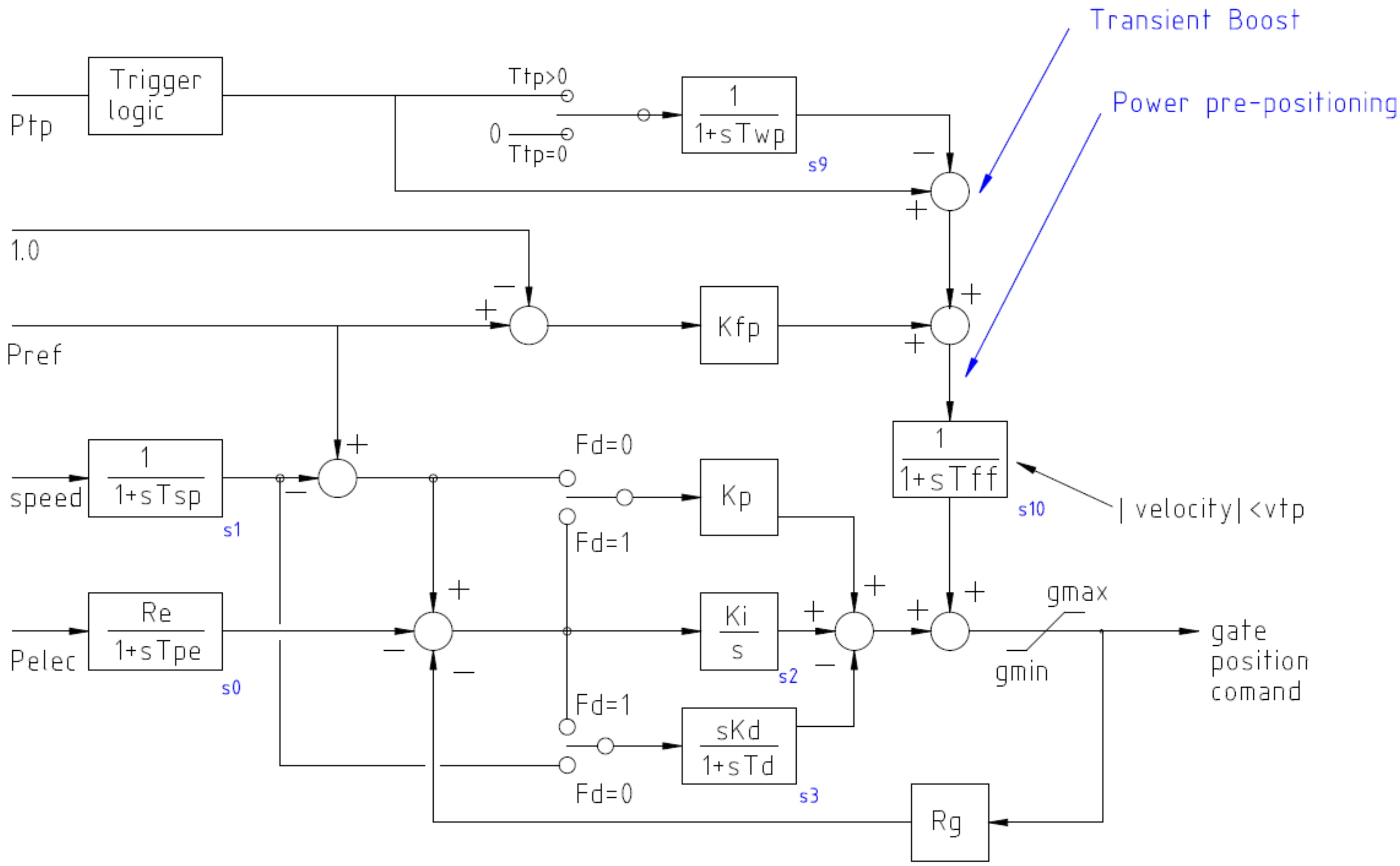

Figure 1 Governor Schematic Diagram

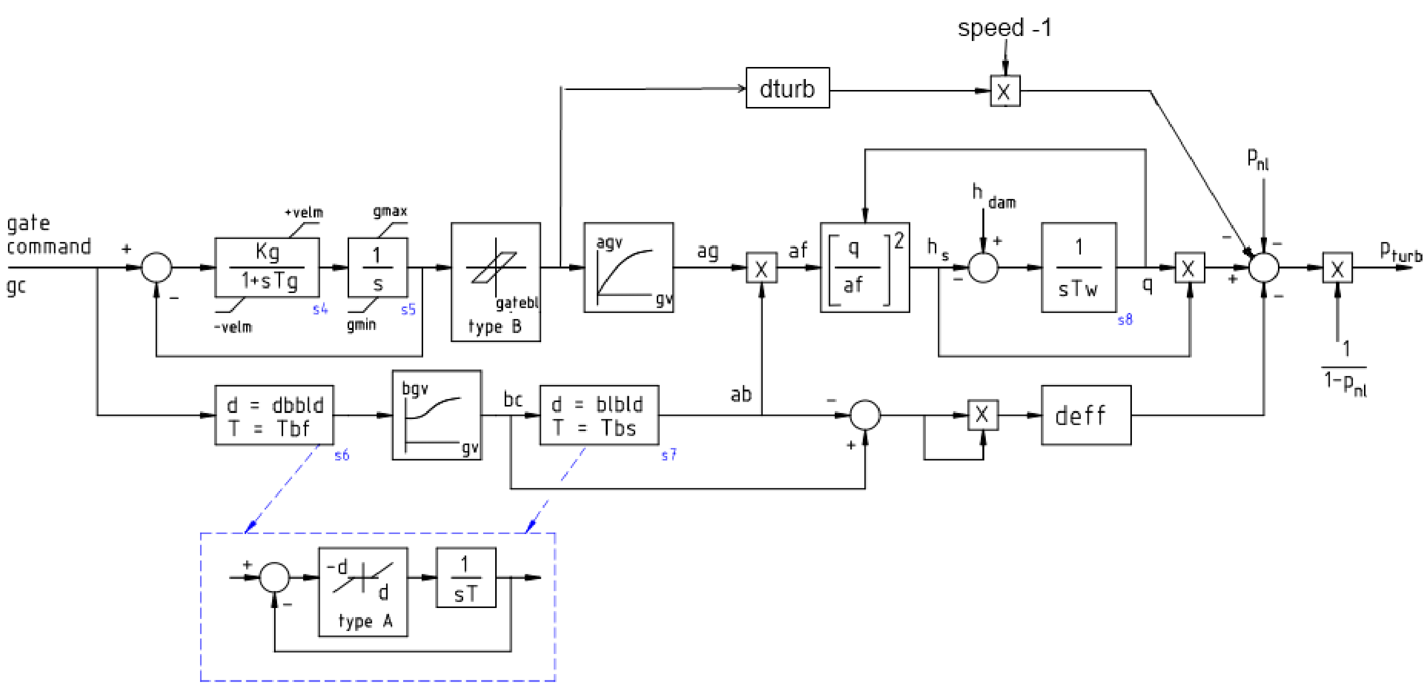

Figure 2 Turbine Schematic Diagram

Model Name: **h6bd**

Description: h6bd model to manage parameter data
for the h6b hydro turbine governor model

Inputs: n/a

Parameters:

Name	Description	Value
re	Permanent droop for electrical power feedback, pu	0.0
rg	Permanent droop for gate position feedback, pu	0.05
tpe	Electric power feedback transducer time const, sec	0.025
tsp	Shaft speed transducer time const, sec	-0.50
fd	Flag for proportional elec power signal	0.00
	1 Elec power is used in proportional path	
	0 Elec power not used in prop path	
kp	Governor proportional gain	***
ki	Governor integral gain	***
kd	Governor derivative gain	***
td	Derivative gain filter time constant, sec	0.05
kg	Pilot servovalve gain	5.00
tg	Pilot servovalve time constant, sec	0.05
velm	Maximum gate actuator velocity, pu/sec	0.20
gmax	Maximum gate actuator stroke, pu	1.00
gmin	Minimum gate actuator stroke, pu	0.00
dturb	Turbine speed sensitivity constant	0.50
pnl	Hydraulic power needed at full speed no load	0.12
blgate	Gate linkage backlash, pu	0.00
tbf	Kaplan turbine blade angle filter time const, pu	2.50
tbs	Kaplan turbine blade angle servo time const, pu	0.50
dbbld	Blade intentional deadband, pu	0.0025
blbld	Blade linkage backlash, pu	0.003
hdam	Operating head, pu	1.00
gv0	First abscissa value for wicket gate	0.00
gv1	and blade curves	0.40
gv2	. . . .	0.50
gv3	. . . .	0.55
gv4	. . . .	0.60
gv5	. . . .	0.70
gv6	. . . .	0.80
gv7	. . . .	0.85
gv8	. . . .	0.88
gv9	. . . .	1.00
agv0	Curve of wicket gate flow area	0.00
agv1	as function of gate servo position	0.50
agv2	. . . .	0.62
agv3	. . . .	0.68
agv4	. . . .	0.73
agv5	. . . .	0.83
agv6	. . . .	0.91
agv7	. . . .	0.94
agv8	. . . .	0.955
agv9	. . . .	1.00
bgv0	Curve of Kaplan turbine blade flow area	0.78
bgv1	as function of gate servo position	0.78
bgv2	. . . .	0.78
bgv3	. . . .	0.79
bgv4	. . . .	0.81
bgv5	. . . .	0.88
bgv6	. . . .	0.96

bgv7	. . . .	0.995
bgv8	. . . .	1.00
bgv9	. . . .	1.00
bgvmin	Base value of blade flow area	0.75
deff	Off blade angle power decrease factor	0.00

Notes:

a) Per unit parameters are on the base of turbine MW capability which is entered on the data record of the **h6b** model, or models, that are associated with this model. The "mwcap" entry may be included in the data record of this data model for convenience but it is NOT USED by the dynamic models associated with this data model. Note that the data established by an h6bd model can be used to represent multiple turbine-governer sets of differing real power capabilities.

b) The gates travel over a range of 1.0 per unit from fully closed to fully opened. The gates are at a position greater than zero when the turbine real power output is zero. Gmax and Gmin are operating limits.

c) Tpe, Tsp, Td, Tg, Tbf, Tbs must be greater than zero.

d) Dturb has the dimensions power/speed.

e) Details of the deadband and backlash functions are shown in figure 6.

f) The turbine is described by two curves as follows:

agv The effective flow area of wicket gates versus gate servomotor stroke.

bgv The effective flow area factor of turbine versus gate servomotor command.

The effective flow area of complete turbine is the product of the two factors:

(effective flow area) = agv(g) * (bgvmin + (1-bgvmin)*bgv(g)

where g is gate servo stroke

Each curve is specified by ten points (numbered 0-9) corresponding to ten gate servo positions which are given as gv0 - gv9. The same set of servo stroke positions is used for both curves.

The curve of gate flow area, agv, must pass thru (0,0) and (1,1) for all turbines

For a Kaplan turbine the blade flow area curve, bgv, must pass through (0,0) and (1,1).

For a Francis turbine all the points on the blade flow area curve, bgv, curve must be at be (g,1).

Figure 7 shows typical curves for a Kaplan turbine. Figure 8 shows typical curves for a Francis turbine.

The parameter bgvmin specifies the blade flow area when the blade servo is at zero stroke. The value of bgvmin is in per unit of the blade flow area when the blades are fully open.

g) The gate opening and flow required to support no-load operation is described by the parameter, pnl.

h) The adjustment factor, deff, can be used to account for reduction of Kaplan turbine power when the blade angle is at an off-nominal angle. The adjustment is

(adjusted pwr) = (on-angle pwr)-deff*(ideal blade stroke - actual blade stroke)

Output Channels:

None

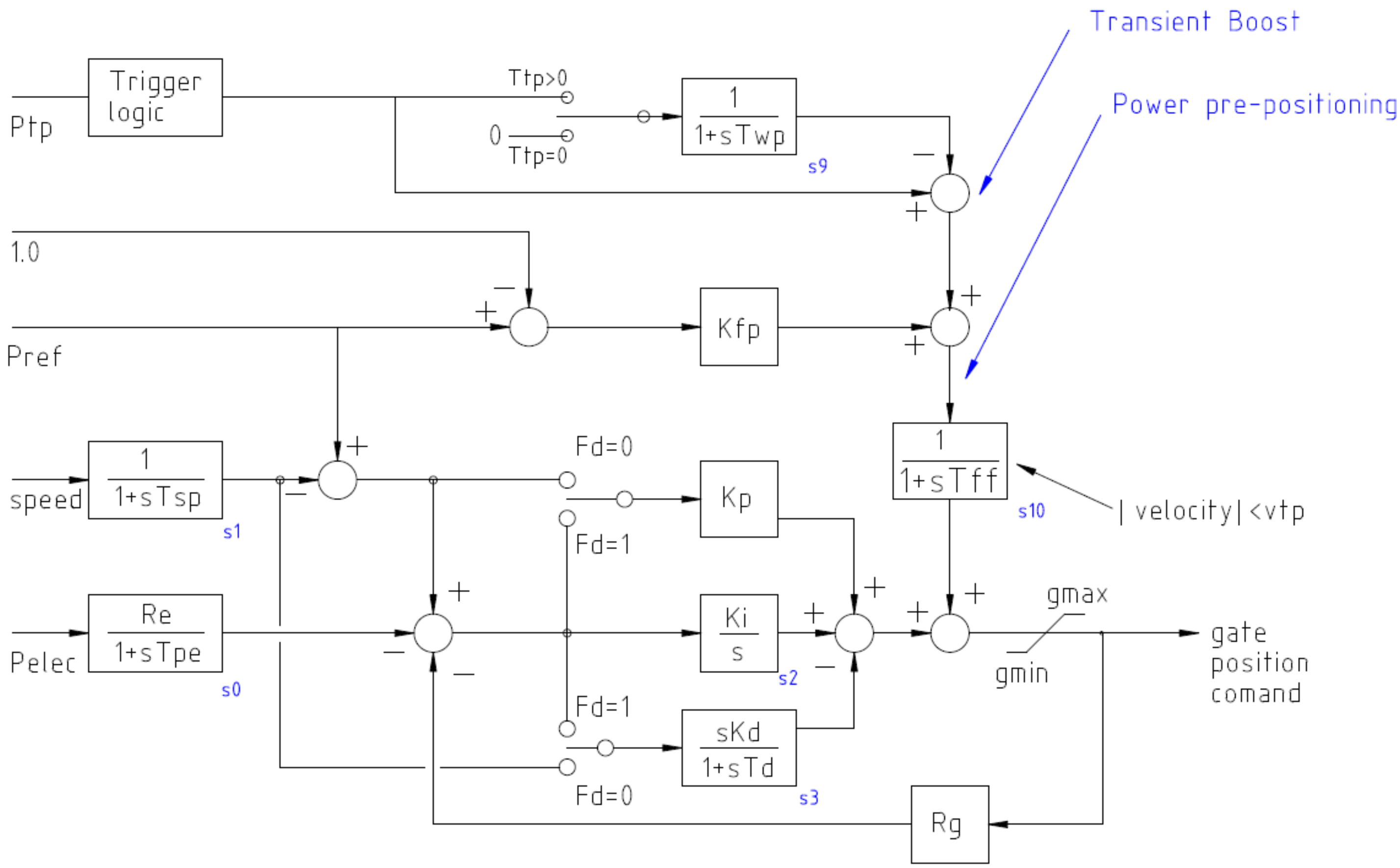

Figure 1 Governor Schematic Diagram

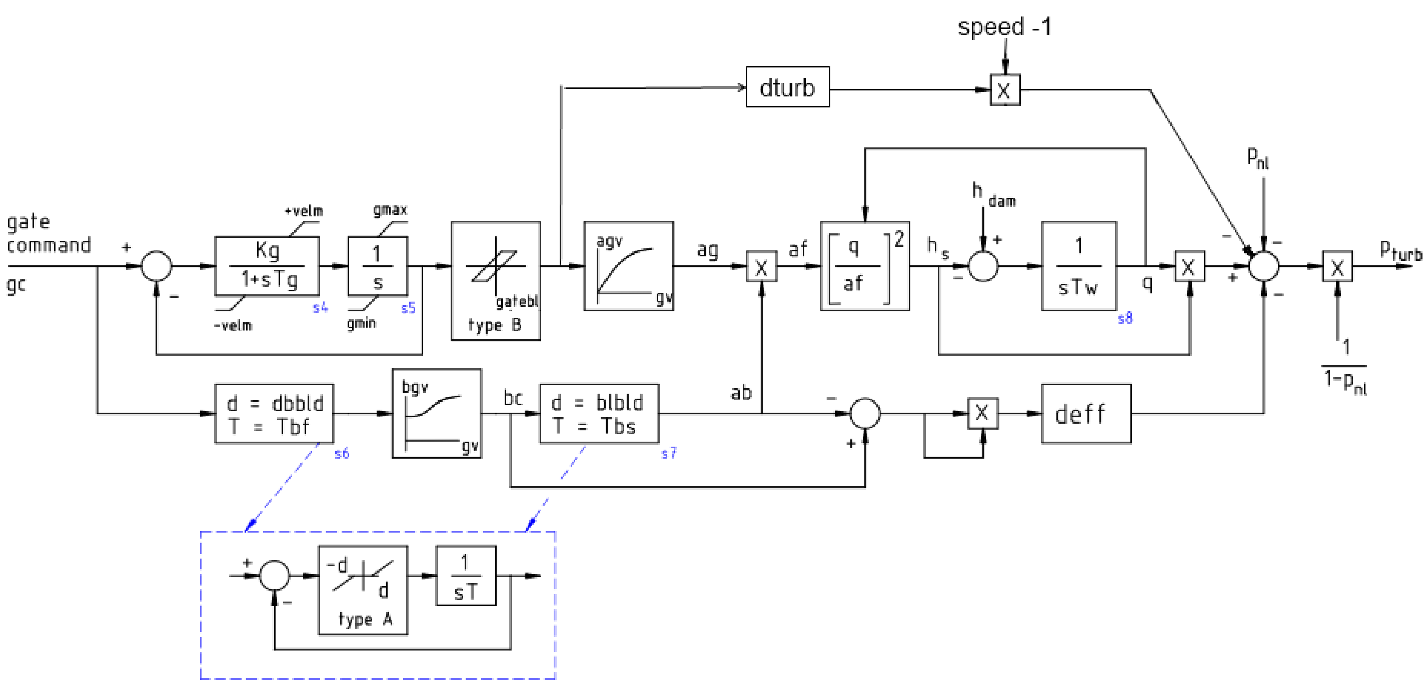

Figure 2 Transfer function for calculation of turbine power

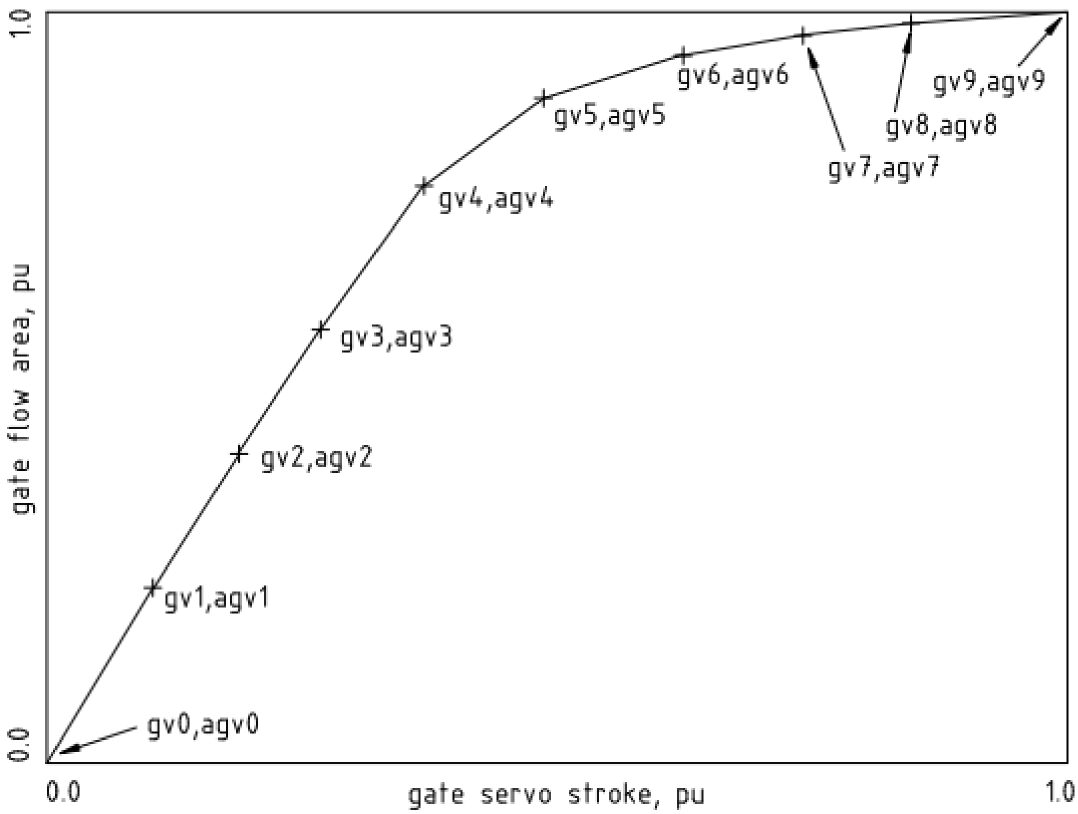

Figure 3 Turbine wicket gate flow curve

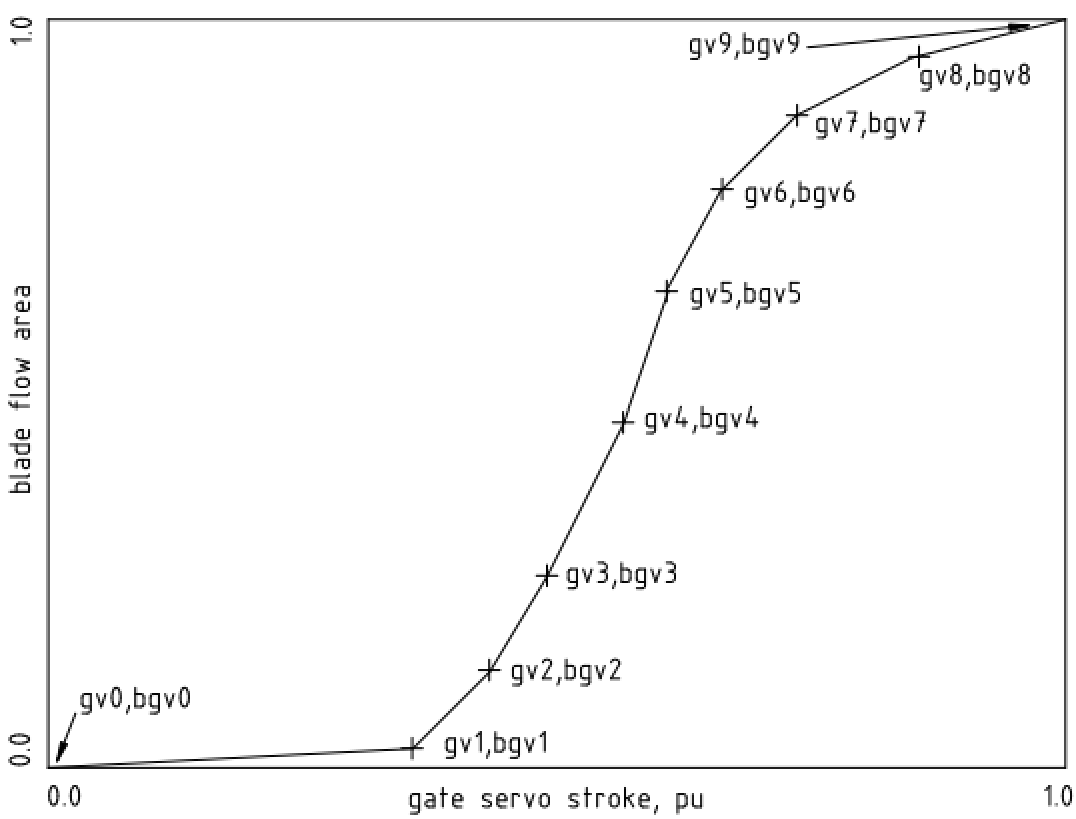

Figure 4 Kaplan turbine blade flow curve

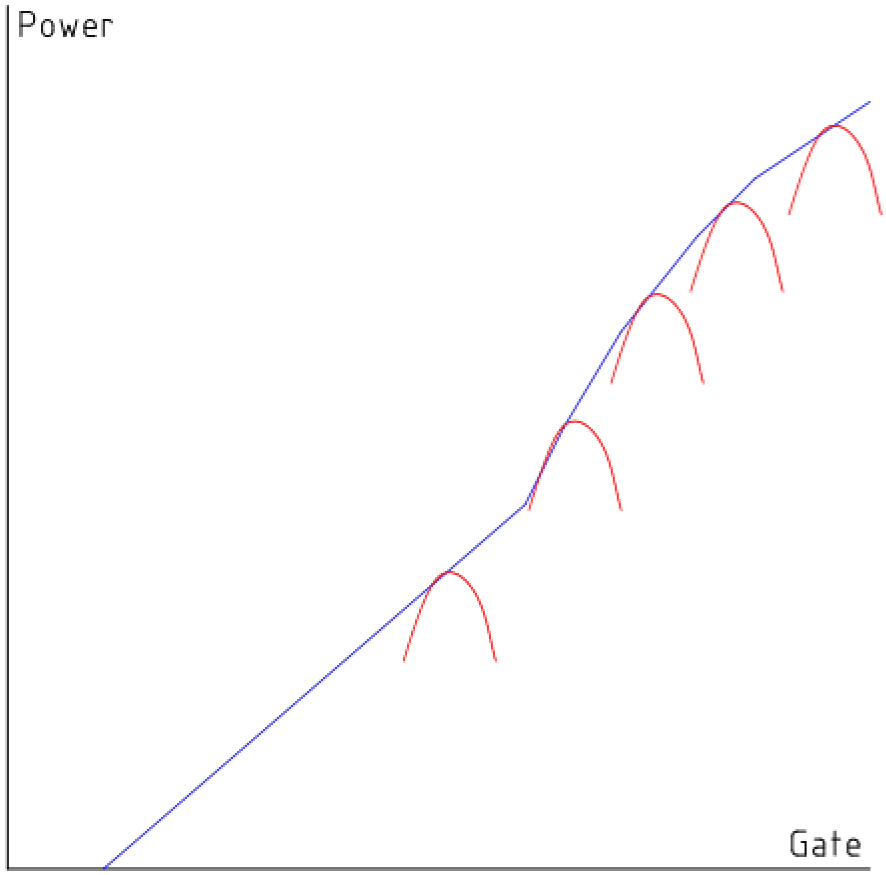

Figure 5 Off-angle power adjustment

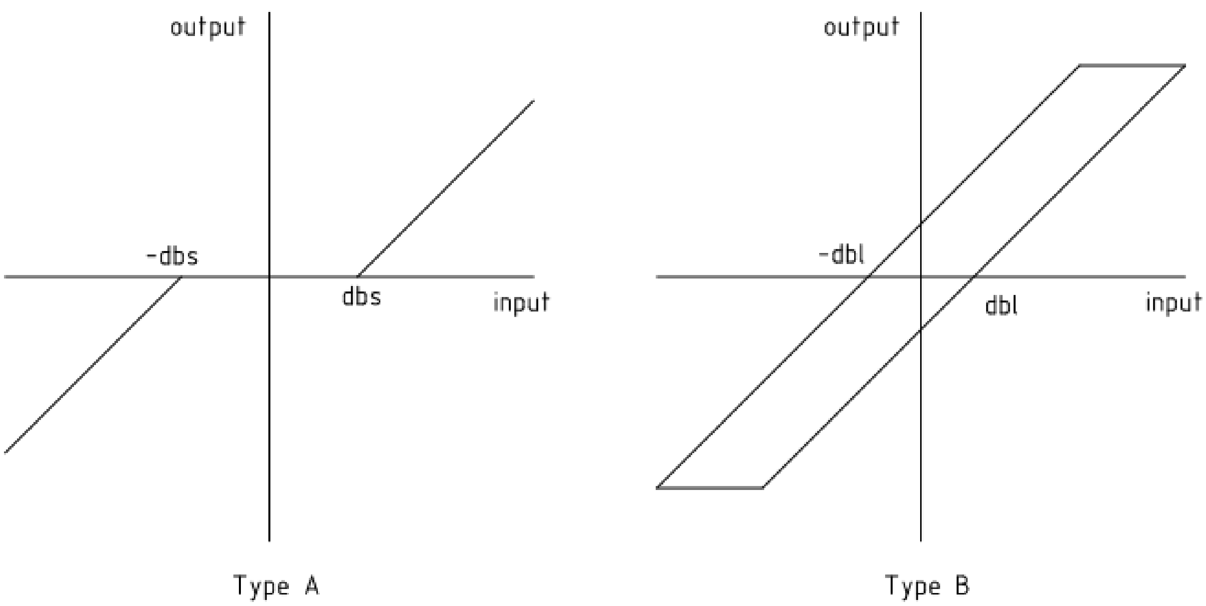

Figure 6 Deadband details

g	ag	ab	Power	Power	Pnl	0.12
Gate	Agate	Ablade	Calc	Test	Bgvmin	0.78
0	0.000	0.000	-0.149	-0.200		
0.400	0.500	0.000	0.334	0.320	Pscale	1.09
0.500	0.620	0.000	0.450	0.450		
0.550	0.680	0.045	0.517	0.500		
0.600	0.730	0.136	0.584	0.580		
0.700	0.830	0.460	0.757	0.750		
0.800	0.910	0.820	0.934	0.920		
0.850	0.940	0.980	1.011	1.010		
0.880	0.960	1.000	1.040	1.040		
1.000	1.000	1	1.090	1.09		

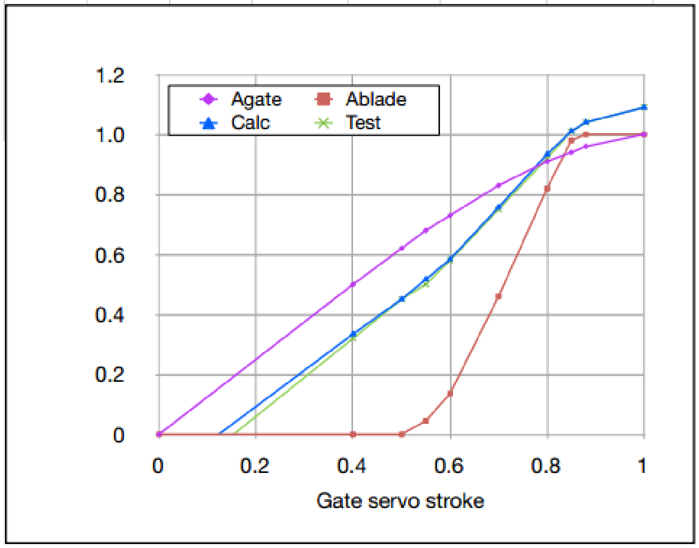

Figure 7 Spreadsheet for preparation of Kaplan turbine data

g	ag	ab	Power	Power	Pnl	0.12
Gate	Agate	Ablade	Calc	Test	Bgvmin	0.78
0	0.000	1.000	-0.149	0.000		
0.400	0.500	1.000	0.471	0.000	Pscale	1.09
0.500	0.620	1.000	0.619	0.000		
0.550	0.680	1.000	0.694	0.000		
0.600	0.730	1.000	0.756	0.000		
0.700	0.830	1.000	0.879	0.000		
0.800	0.910	1.000	0.979	0.000		
0.850	0.940	1.000	1.016	0.000		
0.880	0.960	1.000	1.040	0.000		
1.000	1.000	1	1.090	0.000		

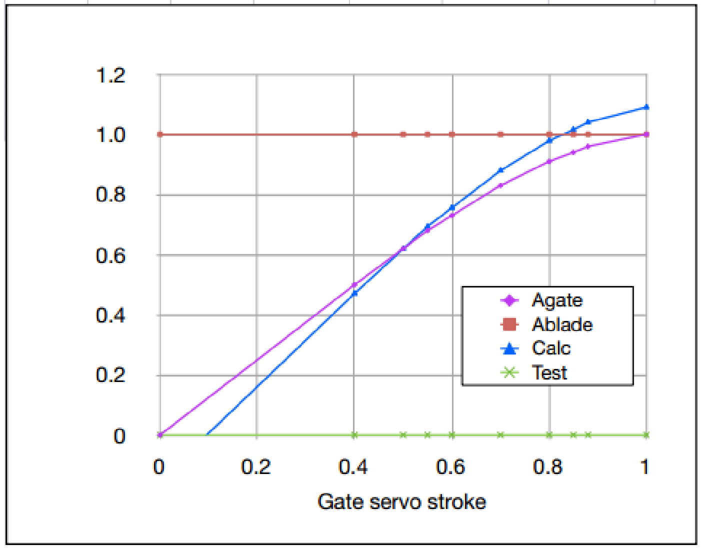

Figure 8 Spreadsheet for preparation of Francis turbine data

h6b.pseudo

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

#include < files needed by your main program >

float   cmag();
float   interp5();

struct h6b
        {
                              // constants - values to be picked up from dynamics data input process
        float tw;                             // water inertia time constant
        float ptp;                            // user-defined gate boost input
        float ftp;                            // frequency deviation for transient boost to pick up
        float ttp;                            // timer for transient boost pickup
        float twp;                            // washout time constant for gate boost
        float vtp;                            // velocity limit on feedforward signal
        float kfp;                            // feedforward gain for speed-load reference
        float tff;                            // feedforward filter time constant
        float fbus;                           // bus number of 'family' model (h6bd)
                              // state variables
        float  s[11];
        float ds[11];
                              // algebraic variables
        float gv;                             // gate servomotor stroke
        float ab;                             // blade angle
        float gout;                           // governor gate command
        float h;
        float ginit;
        float prefinit;
        float tsptp;
        int   tflag;
        float ag;                             // gate area after backlash
        float bc;                             // blade stroke command
        };

struct h6bd
        {
                              // constants - values to be picked up from dynamics data input process
        float re;
        float rg;
        float tpe;
        float tsp;
        float fd;                             // flag controlling input to governor proportional element
        float kp;
        float ki;
        float kd;
        float td;
        float kg;                             // pilot servovavle gain
        float tg;                             // pilot servovalve time constant
        float velm;
        float gmax;
        float gmin;
        float dturb;
        float pnl;                            // hydraulic power needed to support no load operation
        float blgate;                         // Backlash in gate adjustment lankage
        float tbf;                            // Kaplan blade adjustment filter time constant
        float tbs;                            // Kaplan turbine blade servo time constant
        float dbbld;                          // Intentional deadband in blade adjustment path
        float blbld;                          // Backlash in blade adjustment linkage
        float hdam;                           // operating head in per unit of rated head
        float gv0;                            // abscissa values 0-9 of turbine curves
        float gv1;
        float gv2;
        float gv3;
        float gv4;
        float gv5;
        float gv6;
        float gv7;
        float gv8;
        float gv9;
        float agv0;                           // ordinate values 0-9 of gate flow area curve
        float agv1;
        float agv2;
        float agv3;
```

```
        float agv4;
        float agv5;
        float agv6;
        float agv7;
        float agv8;
        float agv9;
        float bgv0;                          // ordinate values 0-9 of blade flow area curves
        float bgv1;
        float bgv2;
        float bgv3;
        float bgv4;
        float bgv5;
        float bgv6;
        float bgv7;
        float bgv8;
        float bgv9;
        float bgvmin;                        // base value of blade flow area
        float deff;                          // off-cam turbine power fall-off factor
        };



int h6b_alge( )
        {
        float   agf, ab,  af, pturb;

        if ( (h6b.s[5] - h6bd.blgate) >= h6b.gv) h6b.gv = h6b.s[5] - h6bd.blgate;   // gate backlash
        if ( (h6b.s[5] + h6bd.blgate) <= h6b.gv) h6b.gv = h6b.s[5] + h6bd.blgate;

        h6b.ag = interp5( h6b.gv, &h6bd.gv0, &h6bd.agv0 );           // gate area as function of gate stroke

        af = h6b.s[7] * h6b.ag;
        h6b.ab = h6b.s[7];
        if ( af > .01 )       h6b.h = ( h6b.s[8] / af ) * ( h6b.s[8] / af );
        else                  h6b.h = h6bd.hdam;

        pturb = ( h6b.s[8] * h6b.h ) - (( <speed> - 1.0) * h6bd.dturb * h6b.gv );
        pturb -= h6bd.deff*(h6b.bc - h6b.ab)*(h6b.bc - h6b.ab);
        pturb -= h6bd.pnl;
        pturb /= (1.-h6bd.pnl);

// pturb is turbine power in per unit on turbine power base

// convert pturb onto the appropriate base for use in the main simulation program

        <pmech> = <converted value>
        return(0);
        }

int h6b_rate(  )
        {
        float   a,     temp, gmax, gmin;
        float   error,  pb,     gp, gc;

        <pgen> = <get generator electrical power from network solution and convert to per unit on turbine power
base>

        if ( h6bd.tpe > 0. ) h6b.ds[0] = <pgen> - h6b.s[0]) / h6bd.tpe;
        else                 h6b.ds[0] = h6b.s[0] = 0.;

        if ( h6bd.tsp < 0. ) h6b.ds[1] = ( <frequency> + 1. - h6b.s[1] )/fabs(h6bd.tsp);
        else
               {
               if (h6bd.tsp > 0. )  h6b.ds[1] = ( <speed> - h6b.s[1] )/h6bd.tsp;
               else
                      {
                      h6b.ds[1] = 0.;  h6b.s[1] = = <speed>;
                      }
               }
        if ( h6bd.re > 0. ) error =   <pref> - h6b.s[0] * h6bd.re -  h6b.s[1];
        if ( h6bd.rg > 0. ) error =   <pref> - h6b.gout * h6bd.rg -  h6b.s[1];

        if ( !h6bd.fd ) gp = h6bd.kp * ( 1. - h6b.s[1] );
        else            gp = h6bd.kp * error;

        if ( h6b.s[2] > (h6bd.gmax - gp ) )  h6b.s[2] = h6b.zs[2] = (h6bd.gmax - gp );
        if ( h6b.s[2] < (h6bd.gmin - gp ) )  h6b.s[2] = h6b.zs[2] = (h6bd.gmin - gp );
        h6b.ds[2] = h6bd.ki * error;
        if ( ((h6b.s[2] + gp) >= h6bd.gmax ) && (h6b.ds[2] > 0.) ) h6b.ds[2] = 0.;
```

```
if ( ((h6b.s[2] + gp) <= h6bd.gmin ) && (h6b.ds[2] < 0.) ) h6b.ds[2] = 0.;

if ( h6bd.fd )  a = error;
else            a = h6b.s[1] - 1.;
if ( h6bd.td > 0. ) h6b.ds[3] = ( a - h6b.s[3] ) / h6bd.td;
else                h6b.ds[3] = h6b.s[3] = 0.;

h6b.gout = gp + h6b.s[2] - h6bd.kd * h6b.ds[3] / h6bd.td;

if ( h6b.gout > h6bd.gmax ) h6b.gout = h6bd.gmax;
if ( h6b.gout < h6bd.gmin ) h6b.gout = h6bd.gmin;

pb = 0.;
switch( h6b.tflag )
        {
case 0: if ( <speed> < (1.-h6b.ftp) ) { h6b.tflag = 1; h6b.tsptp = dypar.time + h6b.ttp; }
        break;
case 1: if ( <speed> > (1.-h6b.ftp) ) { h6b.tflag = 0;      h6b.tsptp = 99999.;    }
        else if ( dypar.time >= h6b.tsptp )  { h6b.tflag = 2; pb = h6b.ptp; }
        break;
case 2: if ( <speed> < (1.-h6b.ftp) )  pb = h6b.ptp;
        else
                {
                if ( dypar.time > (h6b.tsptp + 3.*h6b.twp) )
                        { h6b.tsptp = 99999.; h6b.tflag = 0;  h6b.s[9] = h6b.ds[9] = 0.; }
                else
                        pb = h6b.ptp;
                }
        }

if ( h6b.twp && h6b.ptp )
        h6b.ds[9] = ( pb - h6b.s[9] ) / h6b.twp;
else
        h6b.ds[9] = h6b.s[9] = 0.;

pb -= h6b.s[9];

pb += h6b.kfp*(<pref> - h6b.prefinit);

h6b.ds[10] = ( pb - h6b.s[10] ) / h6b.tff;
if ( h6b.ds[10] >   h6b.vtp ) h6b.ds[10] =   h6b.vtp;
if ( h6b.ds[10] < - h6b.vtp ) h6b.ds[10] = - h6b.vtp;

 h6b.gout += h6b.s[10];

if ( h6b.gout > h6bd.gmax ) h6b.gout = h6bd.gmax;
if ( h6b.gout < h6bd.gmin ) h6b.gout = h6bd.gmin;

switch ( gens[kgen].baseload_flag )
                {
case 1:         gmax = h6b.ginit;
                gmin = h6bd.gmin;
                break;
case 2 :        gmax = gmin = h6b.ginit;
                break;
default:        gmax = h6bd.gmax;
                gmin = h6bd.gmin;
                break;
                }

if ( h6b.s[4] >  h6bd.velm ) h6b.s[4] =  h6bd.velm;
if ( h6b.s[4] < -h6bd.velm ) h6b.s[4] = -h6bd.velm;
h6b.ds[4] = ( h6bd.kg * (h6b.gout - h6b.s[5]) - h6b.s[4] ) / h6bd.tg;
if ( (h6b.s[4] >=   h6bd.velm) && (h6b.ds[4] > 0.) ) h6b.ds[4] = 0.;
if ( (h6b.s[4] <= - h6bd.velm) && (h6b.ds[4] < 0.) ) h6b.ds[4] = 0.;

if( h6b.s[5] > gmax ) h6b.s[5] = h6b.zs[5] = gmax;
if( h6b.s[5] < gmin ) h6b.s[5] = h6b.zs[5] = gmin;
h6b.ds[5] = h6b.s[4];
if( (h6b.s[5] >= gmax) && (h6b.ds[5] > 0.) ) h6b.ds[5] = 0.0;
if( (h6b.s[5] <= gmin) && (h6b.ds[5] < 0.) ) h6b.ds[5] = 0.0;

if( h6b.ds[5] >  h6bd.velm ) h6b.ds[5] =  h6bd.velm;
if( h6b.ds[5] < -h6bd.velm ) h6b.ds[5] = -h6bd.velm;


if ( h6bd.tbf > 0. )
        {
        error = h6b.gout - h6b.s[6];
```

```
                a = 0.;
                if ( error > h6bd.dbbld )            a = error - h6bd.dbbld;
                else  { if ( error < -h6bd.dbbld )   a = error + h6bd.dbbld; }
                h6b.ds[6] = a / h6bd.tbf;
                }
        else { h6b.s[6] = h6b.gout;  h6b.ds[6] = 0.; }

        h6b.bc = h6bd.bgvmin + (1.- h6bd.bgvmin) * interp5( h6b.s[6], &h6bd.gv0, &h6bd.bgv0 );
        if( h6bd.tbs > 0. )
                {
                error = h6b.bc - h6b.s[7];
                a = 0.;
                if ( error > h6bd.blbld )            a = error - h6bd.blbld;
                else  { if ( error < -h6bd.blbld )   a = error + h6bd.blbld; }
                h6b.ds[7] = a / h6bd.tbs;
                }
        else { h6b.s[7] = h6b.bc;  h6b.ds[7] = 0.; }

        h6b.ds[8] = ( h6bd.hdam - h6b.h ) / h6b.tw;
        return(0);
        }


int h6b_init( )
        {
        int     i,      l,      j,      m;
        float   se,     vt,     vmx,    vmi,  temp,   ps, pturb;
        float fa[10], ee, eff, flow, af, oldflow, g;
        float az, bz, cz, dz, aa, bb, cc, g1, g2;
        int kk;

// Get initial generator electrical power from initial condition load flow solution
// and convert to turbine initial power in per unit on turbine power base

        pturb = < get initial turbine power in per unit of turbine base power>


// Turbine initialization
// Build curve of overall flow area vs gate servo stroke
// Note use of C-language indirect addressing convention (& denotes pointer)
        for ( i = 0; i < 10; ++i )
                {
                fa[i] = h6bd.bgvmin + (1.-h6bd.bgvmin) * (*(&h6bd.bgv0+i));
                fa[i] *= *(&h6bd.agv0+i);
                }

        flow = (pturb*(1.-h6bd.pnl) + h6bd.pnl)/h6bd.hdam;
        af = flow / sqrt(h6bd.hdam);
        g = interp5( af, fa, &h6bd.gv0 );

        h6b.s[4] = 0.;
        h6b.s[5] = h6b.s[6] = g;
        h6b.s[7] = h6b.ab = h6b.bc = h6bd.bgvmin + (1.-h6bd.bgvmin) * interp5( g, &h6bd.gv0, &h6bd.bgv0 );
        h6b.s[8] = flow;

        if( h6b.s[5] > 1. )
                {
                <warn - greater than 1.0 gate opening>
                }

        if ( h6b.s[5] > h6bd.gmax)
                {
                <warn - greater than gmax gate opening>
                }

        if ( h6bd.tbs ) h6b.s[7] = h6bd.bgvmin + (1.-h6bd.bgvmin) * interp5( h6b.s[6], &h6bd.gv0, &h6bd.bgv0 );
        else           h6b.s[7] = 1.;

        h6b.ginit = h6b.gv = h6b.gout = h6b.s[2] = h6b.s[5];
        h6b.ag = interp5( h6b.gv  , &h6bd.gv0, &h6bd.agv0 );
        h6b.s[9] = 0.;
        h6b.s[10] = 0.;
        h6b.s[1] = 1.
        h6b.s[3] = 0.;
        h6b.s[0] = <pgen> / <convert generator power to per unit on turbine power base>;
        if ( h6bd.re > 0. ) <pref> = 1. + h6b.s[0] * h6bd.re;
        if ( h6bd.rg > 0. ) <pref> = 1. + h6b.s[5] * h6bd.rg;
        h6b.prefinit = <pref>;

```

```
        if ( h6bd.fd )   h6b.s[2] = h6b.gout;
        else             h6b.s[2] = h6b.gout - h6bd.kp * (genbc[k].pref - 1.);

        h6b.h = h6bd.hdam;

        h6b.tflag = 0;
        h6b.tsptp = 99999.;
        }



float interp5( s, x, y )
float s;
float *x, *y;
        {
        float a;
        int i;
        if ( s >= 1.  ) return(1.);
        if ( s <= 0.  ) return(y[0]);
        for ( i = 0; i<9; ++i )
                if ( s <= x[i+1] ) { a = y[i] + (s-x[i])*(y[i+1]-y[i])/(x[i+1]-x[i]); return(a); }
        return( -1. );
```