jacobian_power_flow_half (Calls: 1400, Time: 2.616 s)

Generated 29-Dec-2021 16:44:14 using performance time.
Function in file D:\Xinliang\Morenet\rapidPF_plus\03_parser\auxfuns\derivatives\jacobian_power_flow_half.m
Copy to new window for comparing multiple runs

Parents (calling functions)

Function Name	Function Type	Calls
...em>@(x)jacobian_power_flow_half(x,pf_eq(x)',state_const,Ybus,entries,copy_buses_local)	Anonymous function	1400

Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
10	`[dS_dVa, dS_dVm] = dSbus_dV(Ybus, V);`	1400	0.805	30.8%	
41	`Hess = Jm'*Jm;`	700	0.804	30.7%	
22	`imag(dS_dVa), imag(dS_dVm), sparse(ntotal, ncore…`	1400	0.302	11.6%	
26	`J = remove_rows(J, buses_to_ignore, ntotal);`	1400	0.288	11.0%	
3	`[Va, Vm, P, Q] = back_to_whole_state(state_var, …`	1400	0.137	5.2%	
All other lines			0.280	10.7%	
Totals			2.616	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
dSbus_dV	Function	1400	0.791	30.3%	
remove_rows	Function	1400	0.280	10.7%	
back_to_whole_state	Function	1400	0.128	4.9%	
Self time (built-ins, overhead, etc.)			1.417	54.2%	
Totals			2.616	100%	

Code Analyzer results

No Code Analyzer messages.

Coverage results

Show coverage for parent folder

Total lines in function	43
Non-code lines (comments, blank lines)	22
Code lines (lines that can run)	21
Code lines that did run	20
Code lines that did not run	1
Coverage (did run/can run)	95.24 %

Function listing

```
Time     Calls   Line
                  1  function [grad, JJp, Hess] = jacobian_power_flow_half(state_var, r, state_0, Ybus, entries, buses_to_ignore)
                  2      % build the whole state
 0.137    1400    3      [Va, Vm, P, Q] = back_to_whole_state(state_var, state_0, entries);
                  4
                  5      % build the derivative
< 0.001   1400    6      if isstruct(Ybus)
                  7          Ybus = makeYbus(Ybus);
< 0.001   1400    8      end
 0.049    1400    9      V = Vm .* exp(1j * Va);
 0.805    1400   10      [dS_dVa, dS_dVm] = dSbus_dV(Ybus, V);
< 0.001   1400   11      assert(numel(Va) == numel(Vm));
< 0.001   1400   12      assert(numel(P) == numel(Q));
< 0.001   1400   13      assert(numel(Va) == numel(buses_to_ignore) + numel(P));
< 0.001   1400   14      ncore = numel(P);
< 0.001   1400   15      ntotal = numel(Va);
                  16
                  17  %     J_P = [ real(dS_dVa), real(dS_dVm), -speye(ntotal, ncore),   sparse(ntotal, ncore) ];
                  18  %     J_Q = [ imag(dS_dVa), imag(dS_dVm),  sparse(ntotal, ncore), -speye(ntotal, ncore) ];
                  19  %     J = [J_P; J_Q];
```

```
0.396   1400   21      J = [real(dS_dVa), real(dS_dVm), -speye(ntotal, ncore),   sparse(ntotal, ncore) ;
        1400   22            imag(dS_dVa), imag(dS_dVm),  sparse(ntotal, ncore), -speye(ntotal, ncore) ];
               23
               24
               25      % remove rows of copy buses
0.288   1400   26      J = remove_rows(J, buses_to_ignore, ntotal);
               27
               28      % only get the columns of variables
               29  %     if iscolumn(y)
               30  %         Jx = J(: , entries.variable.stack)*y;
               31  %     elseif isrow(y)
               32  %         Jx = (y*J(: , entries.variable.stack))';
               33  %     elseif isempty(y)
               34  %         Jm = J(: , entries.variable.stack);
               35  %         Jx = Jm'*Jm;
               36  %     end
0.072   1400   37      Jm   = J(: , entries.variable.stack);
0.033   1400   38      grad = (r*Jm)';
0.010   1400   39      JJp  = @(p)Jm'*(Jm*p);
< 0.001 1400   40      if nargout >2
0.804    700   41          Hess = Jm'*Jm;
< 0.001 1400   42      end
0.013   1400   43  end
```