dSbus_dV (Calls: 1400, Time: 0.869 s)

Generated 29-Dec-2021 16:49:38 using performance time.
Function in file D:\MatlabLibrary\matpower7.1\lib\dSbus_dV.m
Copy to new window for comparing multiple runs

Parents (calling functions)

Function Name	Function Type	Calls
jacobian_power_flow_half	Function	1400

Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
119	`dSbus_dV2 = diagV * conj(Ybus * diagVnorm) + con…`	1400	0.331	38.1%	█████
118	`dSbus_dV1 = 1j * diagV * conj(diagIbus - Ybus * …`	1400	0.316	36.4%	█████
104	`diagVnorm = sparse(1:n, 1:n, V./abs(V), n, n);`	1400	0.083	9.5%	█
101	`diagV = sparse(1:n, 1:n, V, n, n);`	1400	0.051	5.9%	▌
102	`diagIbus = sparse(1:n, 1:n, Ibus, n, n);`	1400	0.039	4.5%	▌
All other lines			0.049	5.7%	▌
Totals			0.869	100%	

Children (called functions)

No children

Code Analyzer results

No Code Analyzer messages.

Coverage results

Show coverage for parent folder

Total lines in function	120
Non-code lines (comments, blank lines)	95
Code lines (lines that can run)	25
Code lines that did run	17
Code lines that did not run	8
Coverage (did run/can run)	68.00 %

Function listing

```
Time   Calls   Line
                 1  function [dSbus_dV1, dSbus_dV2] = dSbus_dV(Ybus, V, vcart)
                 2  %DSBUS_DV   Computes partial derivatives of power injection w.r.t. voltage.
                 3  %
                 4  %   The derivatives can be take with respect to polar or cartesian coordinates
                 5  %   of voltage, depending on the 3rd argument.
                 6  %
                 7  %   [DSBUS_DVA, DSBUS_DVM] = DSBUS_DV(YBUS, V)
                 8  %   [DSBUS_DVA, DSBUS_DVM] = DSBUS_DV(YBUS, V, 0)
                 9  %
                10  %   Returns two matrices containing partial derivatives of the complex bus
                11  %   power injections w.r.t voltage angle and voltage magnitude, respectively
                12  %   (for all buses).
                13  %
                14  %   [DSBUS_DVR, DSBUS_DVI] = DSBUS_DV(YBUS, V, 1)
                15  %
                16  %   Returns two matrices containing partial derivatives of the complex bus
                17  %   power injections w.r.t the real and imaginary parts of voltage,
                18  %   respectively (for all buses).
                19  %
                20  %   If YBUS is a sparse matrix, the return values will be also. The following
                21  %   explains the expressions used to form the matrices:
```

```
%
%   S = diag(V) * conj(Ibus) = diag(conj(Ibus)) * V
%
%   Polar coordinates:
%     Partials of V & Ibus w.r.t. voltage magnitudes
%       dV/dVm = diag(V./abs(V))
%       dI/dVm = Ybus * dV/dVm = Ybus * diag(V./abs(V))
%
%     Partials of V & Ibus w.r.t. voltage angles
%       dV/dVa = j * diag(V)
%       dI/dVa = Ybus * dV/dVa = Ybus * j * diag(V)
%
%     Partials of S w.r.t. voltage magnitudes
%       dS/dVm = diag(V) * conj(dI/dVm) + diag(conj(Ibus)) * dV/dVm
%              = diag(V) * conj(Ybus * diag(V./abs(V)))
%                                       + conj(diag(Ibus)) * diag(V./abs(V))
%
%     Partials of S w.r.t. voltage angles
%       dS/dVa = diag(V) * conj(dI/dVa) + diag(conj(Ibus)) * dV/dVa
%              = diag(V) * conj(Ybus * j * diag(V))
%                                       + conj(diag(Ibus)) * j * diag(V)
%              = -j * diag(V) * conj(Ybus * diag(V))
%                                       + conj(diag(Ibus)) * j * diag(V)
%              = j * diag(V) * conj(diag(Ibus) - Ybus * diag(V))
%
%   Cartesian coordinates:
%     Partials of V & Ibus w.r.t. real part of complex voltage
%       dV/dVr = diag(ones(n,1))
%       dI/dVr = Ybus * dV/dVr = Ybus
%
%     Partials of V & Ibus w.r.t. imaginary part of complex voltage
%       dV/dVi = j * diag(ones(n,1))
%       dI/dVi = Ybus * dV/dVi = Ybus * j
%
%     Partials of S w.r.t. real part of complex voltage
%       dS/dVr = diag(V) * conj(dI/dVr) + diag(conj(Ibus)) * dV/dVr
%              = diag(V) * conj(Ybus) + conj(diag(Ibus))
%
%     Partials of S w.r.t. imaginary part of complex voltage
%       dS/dVi = diag(V) * conj(dI/dVi) + diag(conj(Ibus)) * dV/dVi
%              = j * (conj(diag(Ibus)) - diag(V) conj(Ybus))
%
%   Examples:
%       [Ybus, Yf, Yt] = makeYbus(baseMVA, bus, branch);
%       [dSbus_dVa, dSbus_dVm] = dSbus_dV(Ybus, V);
%       [dSbus_dVr, dSbus_dVi] = dSbus_dV(Ybus, V, 1);
%
%   For more details on the derivations behind the derivative code used
%   in MATPOWER information, see:
%
%   [TN2]  R. D. Zimmerman, "AC Power Flows, Generalized OPF Costs and
%          their Derivatives using Complex Matrix Notation", MATPOWER
%          Technical Note 2, February 2010. [Online]. Available:
%          https://matpower.org/docs/TN2-OPF-Derivatives.pdf
%          doi: 10.5281/zenodo.3237866
%   [TN4]  B. Sereeter and R. D. Zimmerman, "AC Power Flows and their
%          Derivatives using Complex Matrix Notation and Cartesian
%          Coordinate Voltages," MATPOWER Technical Note 4, April 2018.
%          [Online]. Available: https://matpower.org/docs/TN4-OPF-Derivatives-Cartesian.pdf
%          doi: 10.5281/zenodo.3237909

%   MATPOWER
%   Copyright (c) 1996-2019, Power Systems Engineering Research Center (PSERC)
%   by Ray Zimmerman, PSERC Cornell
%   and Baljinnyam Sereeter, Delft University of Technology
```

```
                      87  %
                      88  %   This file is part of MATPOWER.
                      89  %   Covered by the 3-clause BSD License (see LICENSE file for details).
                      90  %   See https://matpower.org for more info.
                      91
                      92  %% default input args
< 0.001       1400    93  if nargin < 3
< 0.001       1400    94      vcart = 0;      %% default to polar coordinates
< 0.001       1400    95  end
                      96
< 0.001       1400    97  n = length(V);
  0.034       1400    98  Ibus = Ybus * V;
                      99
< 0.001       1400   100  if issparse(Ybus)           %% sparse version (if Ybus is sparse)
  0.051       1400   101      diagV       = sparse(1:n, 1:n, V, n, n);
  0.039       1400   102      diagIbus    = sparse(1:n, 1:n, Ibus, n, n);
< 0.001       1400   103      if ~vcart
  0.083       1400   104          diagVnorm   = sparse(1:n, 1:n, V./abs(V), n, n);
< 0.001       1400   105      end
                     106  else                        %% dense version
                     107      diagV       = diag(V);
                     108      diagIbus    = diag(Ibus);
                     109      if ~vcart
                     110          diagVnorm   = diag(V./abs(V));
                     111      end
< 0.001       1400   112  end
                     113
< 0.001       1400   114  if vcart
                     115      dSbus_dV1 = conj(diagIbus) + diagV * conj(Ybus);         %% dSbus/dVr
                     116      dSbus_dV2 = 1j * (conj(diagIbus) - diagV * conj(Ybus)); %% dSbus/dVi
< 0.001       1400   117  else
  0.316       1400   118      dSbus_dV1 = 1j * diagV * conj(diagIbus - Ybus * diagV);                     %% dSbus/dVa
  0.331       1400   119      dSbus_dV2 = diagV * conj(Ybus * diagVnorm) + conj(diagIbus) * diagVnorm;     %% dSbus/dVm
  0.006       1400   120  end
```

Local functions in this file are not included in this listing.