

Extended Thomas-Fermi

Generated by Doxygen 1.8.15

Contents

| | | |
|----------|--|----------|
| 1 | Module Index | 1 |
| 1.1 | Modules | 1 |
| 2 | Modules Index | 3 |
| 2.1 | Modules List | 3 |
| 3 | File Index | 5 |
| 3.1 | File List | 5 |
| 4 | Module Documentation | 7 |
| 4.1 | Global parameters | 7 |
| 4.1.1 | Detailed Description | 7 |
| 4.1.2 | Variable Documentation | 7 |
| 4.1.2.1 | e2 | 7 |
| 4.1.2.2 | hbar2_2m | 8 |
| 4.1.2.3 | hbarc | 8 |
| 4.1.2.4 | mec2 | 8 |
| 4.1.2.5 | mnc2 | 8 |
| 4.1.2.6 | mpc2 | 8 |
| 4.1.2.7 | pi | 8 |
| 4.1.2.8 | rho0 | 8 |
| 4.2 | Extra constants for calculating $\frac{hbar^2}{2m}$ for BSk forces | 9 |
| 4.2.1 | Detailed Description | 9 |
| 4.2.2 | Variable Documentation | 9 |
| 4.2.2.1 | mamuc2 | 9 |

| | | |
|----------|------------------------|----|
| 4.2.2.2 | rydb | 9 |
| 4.2.2.3 | xmh | 9 |
| 4.2.2.4 | xmn | 9 |
| 4.2.2.5 | xmp | 9 |
| 4.3 | Numerical fractions | 10 |
| 4.3.1 | Detailed Description | 10 |
| 4.3.2 | Variable Documentation | 10 |
| 4.3.2.1 | fr12 | 10 |
| 4.3.2.2 | fr13 | 10 |
| 4.3.2.3 | fr14 | 10 |
| 4.3.2.4 | fr16 | 11 |
| 4.3.2.5 | fr18 | 11 |
| 4.3.2.6 | fr1_12 | 11 |
| 4.3.2.7 | fr1_16 | 11 |
| 4.3.2.8 | fr1_24 | 11 |
| 4.3.2.9 | fr1_32 | 11 |
| 4.3.2.10 | fr1_36 | 11 |
| 4.3.2.11 | fr23 | 11 |
| 4.3.2.12 | fr260_3 | 12 |
| 4.3.2.13 | fr29 | 12 |
| 4.3.2.14 | fr32 | 12 |
| 4.3.2.15 | fr34 | 12 |
| 4.3.2.16 | fr35 | 12 |
| 4.3.2.17 | fr38 | 12 |
| 4.3.2.18 | fr3_10 | 12 |
| 4.3.2.19 | fr43 | 12 |
| 4.3.2.20 | fr53 | 13 |
| 4.3.2.21 | fr54 | 13 |
| 4.3.2.22 | fr83 | 13 |
| 4.4 | Skyrme parameters | 14 |

| | | |
|----------|------------------------|----|
| 4.4.1 | Detailed Description | 14 |
| 4.4.2 | Variable Documentation | 14 |
| 4.4.2.1 | hbar2_2m_q | 14 |
| 4.4.2.2 | j2_terms | 14 |
| 4.4.2.3 | sigma | 14 |
| 4.4.2.4 | t0 | 15 |
| 4.4.2.5 | t1 | 15 |
| 4.4.2.6 | t2 | 15 |
| 4.4.2.7 | t3 | 15 |
| 4.4.2.8 | w0 | 15 |
| 4.4.2.9 | x0 | 15 |
| 4.4.2.10 | x1 | 15 |
| 4.4.2.11 | x2 | 15 |
| 4.4.2.12 | x3 | 15 |
| 4.5 | Skyrme coefficients | 16 |
| 4.5.1 | Detailed Description | 16 |
| 4.5.2 | Variable Documentation | 16 |
| 4.5.2.1 | b1 | 16 |
| 4.5.2.2 | b10 | 16 |
| 4.5.2.3 | b11 | 16 |
| 4.5.2.4 | b12 | 16 |
| 4.5.2.5 | b13 | 17 |
| 4.5.2.6 | b2 | 17 |
| 4.5.2.7 | b3 | 17 |
| 4.5.2.8 | b4 | 17 |
| 4.5.2.9 | b5 | 17 |
| 4.5.2.10 | b6 | 17 |
| 4.5.2.11 | b7 | 17 |
| 4.5.2.12 | b8 | 17 |
| 4.5.2.13 | b9 | 17 |

| | | |
|----------|---------------------------------|----|
| 4.6 | Extra parameters for BSk forces | 18 |
| 4.6.1 | Detailed Description | 18 |
| 4.6.2 | Variable Documentation | 18 |
| 4.6.2.1 | alpha | 18 |
| 4.6.2.2 | beta | 18 |
| 4.6.2.3 | epsilon_lambda | 18 |
| 4.6.2.4 | fn_neg | 18 |
| 4.6.2.5 | fn_pos | 19 |
| 4.6.2.6 | fp_neg | 19 |
| 4.6.2.7 | fp_pos | 19 |
| 4.6.2.8 | gamma | 19 |
| 4.6.2.9 | t4 | 19 |
| 4.6.2.10 | t5 | 19 |
| 4.6.2.11 | x4 | 19 |
| 4.6.2.12 | x5 | 19 |
| 4.7 | Mesh variable | 20 |
| 4.7.1 | Detailed Description | 20 |
| 4.7.2 | Variable Documentation | 20 |
| 4.7.2.1 | r | 20 |
| 4.8 | Global density arrays | 21 |
| 4.8.1 | Detailed Description | 21 |
| 4.8.2 | Variable Documentation | 21 |
| 4.8.2.1 | del2_rho_q | 21 |
| 4.8.2.2 | del2_rho_t | 21 |
| 4.8.2.3 | del_rho_q | 21 |
| 4.8.2.4 | del_rho_t | 21 |
| 4.8.2.5 | rho_q | 21 |
| 4.8.2.6 | rho_t | 21 |
| 4.9 | Global effective mass arrays | 22 |
| 4.9.1 | Detailed Description | 22 |

| | | |
|-----------|--|----|
| 4.9.2 | Variable Documentation | 22 |
| 4.9.2.1 | del2_f_q | 22 |
| 4.9.2.2 | del_f_q | 22 |
| 4.9.2.3 | f_q | 22 |
| 4.10 | Global arrays for other densities and fields | 23 |
| 4.10.1 | Detailed Description | 23 |
| 4.10.2 | Variable Documentation | 23 |
| 4.10.2.1 | del_j_q | 23 |
| 4.10.2.2 | del_j_t | 23 |
| 4.10.2.3 | div_j_2_q | 24 |
| 4.10.2.4 | div_j_q | 24 |
| 4.10.2.5 | div_j_t | 24 |
| 4.10.2.6 | div_w_q | 24 |
| 4.10.2.7 | e_density_c_di | 24 |
| 4.10.2.8 | e_density_c_ex | 24 |
| 4.10.2.9 | e_density_field | 24 |
| 4.10.2.10 | e_density_sky | 24 |
| 4.10.2.11 | j_2_q | 25 |
| 4.10.2.12 | j_q | 25 |
| 4.10.2.13 | j_t | 25 |
| 4.10.2.14 | rho_ch | 25 |
| 4.10.2.15 | tau_2_cont_q | 25 |
| 4.10.2.16 | tau_2_l_q | 25 |
| 4.10.2.17 | tau_2_nl_q | 25 |
| 4.10.2.18 | tau_etf_q | 26 |
| 4.10.2.19 | tau_etf_t | 26 |
| 4.10.2.20 | tau_tf_q | 26 |
| 4.10.2.21 | u_q | 26 |
| 4.10.2.22 | v_c_di | 26 |
| 4.10.2.23 | v_c_ex | 26 |

| | |
|---|----|
| 4.10.2.24 v_c_pe | 26 |
| 4.10.2.25 w_q | 26 |
| 4.11 Global arrays for densities and fields for 4th-order terms | 27 |
| 4.12 Variables for storing properties of Wigner-Seitz cell | 28 |
| 4.12.1 Detailed Description | 28 |
| 4.12.2 Variable Documentation | 28 |
| 4.12.2.1 delta_mu | 29 |
| 4.12.2.2 e_coulomb | 29 |
| 4.12.2.3 e_coulomb_di | 29 |
| 4.12.2.4 e_coulomb_ex | 29 |
| 4.12.2.5 e_field | 29 |
| 4.12.2.6 e_kinetic_q | 29 |
| 4.12.2.7 e_kinetic_t | 30 |
| 4.12.2.8 e_skyrme | 30 |
| 4.12.2.9 e_so | 30 |
| 4.12.2.10 e_total | 30 |
| 4.12.2.11 mu_c | 30 |
| 4.12.2.12 mu_e | 30 |
| 4.12.2.13 mu_q | 31 |
| 4.12.2.14 n_q | 31 |
| 4.12.2.15 n_t | 31 |
| 4.12.2.16 pressure_e | 31 |
| 4.12.2.17 pressure_ex | 31 |
| 4.12.2.18 pressure_nucl | 31 |
| 4.12.2.19 pressure_t | 31 |
| 4.13 Input parameters | 32 |
| 4.13.1 Detailed Description | 33 |
| 4.13.2 Variable Documentation | 33 |
| 4.13.2.1 calc_chem_pots | 33 |
| 4.13.2.2 coulomb_on | 33 |

| | | |
|-----------|-------------------|----|
| 4.13.2.3 | dr | 33 |
| 4.13.2.4 | electrons_on | 34 |
| 4.13.2.5 | emax0_mod | 34 |
| 4.13.2.6 | etf_order | 34 |
| 4.13.2.7 | force | 34 |
| 4.13.2.8 | lmax | 34 |
| 4.13.2.9 | n | 34 |
| 4.13.2.10 | neutron_pairing | 35 |
| 4.13.2.11 | nmaxstate | 35 |
| 4.13.2.12 | pair_alpha | 35 |
| 4.13.2.13 | pair_del_init | 35 |
| 4.13.2.14 | pair_dk | 35 |
| 4.13.2.15 | pair_eta | 35 |
| 4.13.2.16 | pair_k_max | 36 |
| 4.13.2.17 | pair_max_iters | 36 |
| 4.13.2.18 | pair_mix | 36 |
| 4.13.2.19 | pair_num_tol | 36 |
| 4.13.2.20 | pair_qp_cut | 36 |
| 4.13.2.21 | pair_tol | 36 |
| 4.13.2.22 | pair_v0 | 37 |
| 4.13.2.23 | profile_r_max | 37 |
| 4.13.2.24 | proton_bcs | 37 |
| 4.13.2.25 | r_max | 37 |
| 4.13.2.26 | run_mode | 37 |
| 4.13.2.27 | smooth_qp_cut | 37 |
| 4.13.2.28 | specify_n | 38 |
| 4.13.2.29 | strut_r_max | 38 |
| 4.13.2.30 | strutinsky_on | 38 |
| 4.13.2.31 | verbose | 38 |
| 4.14 | External profiles | 39 |

| | |
|---|----|
| 4.14.1 Detailed Description | 39 |
| 4.14.2 Variable Documentation | 39 |
| 4.14.2.1 n_profile | 39 |
| 4.14.2.2 num_n | 39 |
| 4.14.2.3 num_p | 39 |
| 4.14.2.4 p_profile | 39 |
| 4.14.2.5 r_ws | 39 |
| 4.15 Format statements | 40 |
| 4.15.1 Detailed Description | 40 |
| 4.15.2 Variable Documentation | 40 |
| 4.15.2.1 info_format | 40 |
| 4.15.2.2 n_floats | 40 |
| 4.16 Global strings for printing output | 41 |
| 4.16.1 Detailed Description | 41 |
| 4.16.2 Variable Documentation | 41 |
| 4.16.2.1 force_string | 41 |
| 4.16.2.2 iso_string | 41 |
| 4.16.2.3 line_break | 41 |
| 4.16.2.4 space4 | 41 |
| 4.16.2.5 table_string | 41 |
| 4.17 Variables for use in strutinsky module | 42 |
| 4.17.1 Detailed Description | 42 |
| 4.17.2 Variable Documentation | 42 |
| 4.17.2.1 d2hmen | 42 |
| 4.17.2.2 dhmen | 42 |
| 4.17.2.3 e_sc_q | 43 |
| 4.17.2.4 e_sc_t | 43 |
| 4.17.2.5 ecut | 43 |
| 4.17.2.6 ep | 43 |
| 4.17.2.7 hb2m | 43 |

| | | |
|-----------|--------------------------------------|----|
| 4.17.2.8 | jip | 43 |
| 4.17.2.9 | lp | 44 |
| 4.17.2.10 | nnst | 44 |
| 4.17.2.11 | ps | 44 |
| 4.17.2.12 | strut_n | 44 |
| 4.17.2.13 | vpot | 44 |
| 4.17.2.14 | vso | 44 |
| 4.18 | Variables for use in pairing module | 45 |
| 4.18.1 | Detailed Description | 45 |
| 4.18.2 | Variable Documentation | 45 |
| 4.18.2.1 | delta_n | 45 |
| 4.18.2.2 | delta_p | 46 |
| 4.18.2.3 | e_pair_bcs | 46 |
| 4.18.2.4 | e_pair_q | 46 |
| 4.18.2.5 | e_pair_t | 46 |
| 4.18.2.6 | e_qp_p | 46 |
| 4.18.2.7 | num_p_bcs | 46 |
| 4.18.2.8 | occ_u | 47 |
| 4.18.2.9 | occ_v | 47 |
| 4.18.2.10 | pair_gap_n | 47 |
| 4.18.2.11 | pair_gap_p | 47 |
| 4.18.2.12 | rho_anom_p_bcs | 47 |
| 4.18.2.13 | rho_p_bcs | 47 |
| 4.18.2.14 | strength_bcs | 47 |
| 4.19 | Variables for electron contributions | 48 |
| 4.19.1 | Detailed Description | 48 |
| 4.19.2 | Variable Documentation | 48 |
| 4.19.2.1 | e_coulomb_e | 48 |
| 4.19.2.2 | e_coulomb_pe | 48 |
| 4.19.2.3 | e_kinetic_e | 48 |
| 4.19.2.4 | rho_e | 48 |

| | | |
|----------|-----------------------------------|-----------|
| 5 | Module Documentation | 49 |
| 5.1 | pairing Module Reference | 49 |
| 5.1.1 | Detailed Description | 49 |
| 5.1.2 | Function/Subroutine Documentation | 49 |
| 5.1.2.1 | bcs_protons() | 50 |
| 5.1.2.2 | calc_bcs_num_p() | 50 |
| 5.1.2.3 | calc_e_pair() | 51 |
| 5.1.2.4 | calc_gap() | 52 |
| 5.1.2.5 | delta_parametric() | 52 |
| 5.1.2.6 | interazchamelanalyt() | 53 |
| 5.1.2.7 | llambda() | 54 |
| 5.2 | parameters Module Reference | 55 |
| 5.2.1 | Detailed Description | 61 |
| 5.2.2 | Function/Subroutine Documentation | 61 |
| 5.2.2.1 | force_initialise() | 61 |
| 5.2.3 | Variable Documentation | 61 |
| 5.2.3.1 | a_q | 62 |
| 5.2.3.2 | d1_a_q | 62 |
| 5.2.3.3 | d2_a_q | 62 |
| 5.2.3.4 | d2_f_q | 62 |
| 5.2.3.5 | d2_rho_q | 62 |
| 5.2.3.6 | d3_a_q | 62 |
| 5.2.3.7 | d3_f_q | 62 |
| 5.2.3.8 | d3_rho_q | 62 |
| 5.2.3.9 | d4_a_q | 63 |
| 5.2.3.10 | d4_f_q | 63 |
| 5.2.3.11 | d4_rho_q | 63 |
| 5.2.3.12 | div_j_4_q | 63 |
| 5.2.3.13 | dp | 63 |
| 5.2.3.14 | j_4_q | 63 |

| | | |
|----------|-------------------------------------|----|
| 5.2.3.15 | <code>tau_4_cont_q</code> | 63 |
| 5.2.3.16 | <code>tau_4_no_spin_q</code> | 64 |
| 5.2.3.17 | <code>tau_4_so_q</code> | 64 |
| 5.3 | routines Module Reference | 64 |
| 5.3.1 | Detailed Description | 66 |
| 5.3.2 | Function/Subroutine Documentation | 66 |
| 5.3.2.1 | <code>allocate_arrays()</code> | 66 |
| 5.3.2.2 | <code>calc_div_j_2_q()</code> | 67 |
| 5.3.2.3 | <code>calc_div_j_4_q()</code> | 67 |
| 5.3.2.4 | <code>calc_e_f_q()</code> | 68 |
| 5.3.2.5 | <code>calc_f_q()</code> | 69 |
| 5.3.2.6 | <code>calc_j_2_q()</code> | 70 |
| 5.3.2.7 | <code>calc_j_4_q()</code> | 70 |
| 5.3.2.8 | <code>calc_particle_number()</code> | 71 |
| 5.3.2.9 | <code>calc_rho_q()</code> | 72 |
| 5.3.2.10 | <code>calc_tau_2_l()</code> | 73 |
| 5.3.2.11 | <code>calc_tau_2_nl()</code> | 73 |
| 5.3.2.12 | <code>calc_tau_4_no_spin()</code> | 74 |
| 5.3.2.13 | <code>calc_tau_4_so()</code> | 75 |
| 5.3.2.14 | <code>calc_tau_tf()</code> | 76 |
| 5.3.2.15 | <code>calc_u_q()</code> | 77 |
| 5.3.2.16 | <code>calc_w_q()</code> | 78 |
| 5.3.2.17 | <code>charge()</code> | 79 |
| 5.3.2.18 | <code>d1()</code> | 79 |
| 5.3.2.19 | <code>d2()</code> | 80 |
| 5.3.2.20 | <code>d3()</code> | 81 |
| 5.3.2.21 | <code>d4()</code> | 81 |
| 5.3.2.22 | <code>deallocate_arrays()</code> | 82 |
| 5.3.2.23 | <code>deallocate_mesh()</code> | 82 |
| 5.3.2.24 | <code>eval_coulomb()</code> | 83 |

| | | |
|----------|---|-----|
| 5.3.2.25 | <code>eval_electrons()</code> | 83 |
| 5.3.2.26 | <code>eval_fields()</code> | 84 |
| 5.3.2.27 | <code>eval_skyrme_energy_density()</code> | 86 |
| 5.3.2.28 | <code>eval_tau_etf()</code> | 86 |
| 5.3.2.29 | <code>eval_u_q()</code> | 87 |
| 5.3.2.30 | <code>eval_ws_quantities()</code> | 88 |
| 5.3.2.31 | <code>extrapolate_back_3()</code> | 90 |
| 5.3.2.32 | <code>initialise_uniform_mesh()</code> | 90 |
| 5.3.2.33 | <code>lap_sphe_symm()</code> | 91 |
| 5.3.2.34 | <code>pressure()</code> | 91 |
| 5.3.2.35 | <code>str2int()</code> | 92 |
| 5.3.2.36 | <code>write_densities()</code> | 93 |
| 5.3.2.37 | <code>write_fields()</code> | 93 |
| 5.3.2.38 | <code>write_sp_states_p()</code> | 94 |
| 5.3.2.39 | <code>ws_integral()</code> | 94 |
| 5.3.3 | Variable Documentation | 95 |
| 5.3.3.1 | <code>file_unit_0</code> | 95 |
| 5.3.3.2 | <code>file_unit_1</code> | 95 |
| 5.4 | rspace Module Reference | 95 |
| 5.4.1 | Detailed Description | 96 |
| 5.4.2 | Function/Subroutine Documentation | 96 |
| 5.4.2.1 | <code>boundary()</code> | 96 |
| 5.4.2.2 | <code>numerov()</code> | 97 |
| 5.5 | strutinsky Module Reference | 97 |
| 5.5.1 | Detailed Description | 97 |
| 5.5.2 | Function/Subroutine Documentation | 98 |
| 5.5.2.1 | <code>calc_e_sc()</code> | 98 |
| 5.5.2.2 | <code>calc_e_sc_pair()</code> | 99 |
| 5.5.2.3 | <code>sp_states_deallocate()</code> | 99 |
| 5.5.2.4 | <code>sp_states_setup()</code> | 100 |

| | | |
|----------|------------------------------------|------------|
| 5.5.2.5 | state_sort() | 100 |
| 5.6 | test Module Reference | 101 |
| 5.6.1 | Detailed Description | 102 |
| 5.6.2 | Function/Subroutine Documentation | 102 |
| 5.6.2.1 | bartel_bencheikh_benchmark() | 102 |
| 5.6.2.2 | orders_kinetic_densities() | 103 |
| 5.6.2.3 | read_test_params() | 103 |
| 5.6.2.4 | test_calc_coulomb_energy() | 104 |
| 5.6.2.5 | test_calc_particle_number() | 104 |
| 5.6.2.6 | test_calc_skyrme_energy() | 105 |
| 5.6.2.7 | test_central_potentials() | 105 |
| 5.6.2.8 | test_densities() | 106 |
| 5.6.2.9 | test_density_derivs() | 106 |
| 5.6.2.10 | test_eff_mass() | 107 |
| 5.6.2.11 | test_kinetic_densities() | 107 |
| 5.6.2.12 | test_si_correction() | 108 |
| 5.6.2.13 | test_sp_energies() | 108 |
| 5.6.2.14 | test_spin_current_densities() | 109 |
| 5.6.2.15 | write_ws_cell_array_quantity() | 110 |
| 5.6.3 | Variable Documentation | 110 |
| 5.6.3.1 | file_unit | 110 |
| 5.6.3.2 | file_unit_2 | 110 |
| 5.6.3.3 | sample_profile | 110 |
| 6 | File Documentation | 111 |
| 6.1 | srcs/boundary.f90 File Reference | 111 |
| 6.2 | srcs/etf.f90 File Reference | 111 |
| 6.2.1 | Function/Subroutine Documentation | 111 |
| 6.2.1.1 | etf() | 112 |
| 6.2.1.2 | total_energy_with_corrections() | 113 |
| 6.3 | srcs/pairing.f90 File Reference | 115 |
| 6.4 | srcs/parameters.f90 File Reference | 115 |
| 6.5 | srcs/routines.f90 File Reference | 122 |
| 6.6 | srcs/strutinsky.f90 File Reference | 124 |
| 6.7 | srcs/test.f90 File Reference | 124 |
| | Index | 127 |

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

| | |
|--|----|
| Global parameters | 7 |
| Extra constants for calculating $\frac{\hbar^2}{2m}f$ for BSk forces | 9 |
| Numerical fractions | 10 |
| Skyrme parameters | 14 |
| Skyrme coefficients | 16 |
| Extra parameters for BSk forces | 18 |
| Mesh variable | 20 |
| Global density arrays | 21 |
| Global effective mass arrays | 22 |
| Global arrays for other densities and fields | 23 |
| Global arrays for densities and fields for 4th-order terms | 27 |
| Variables for storing properties of Wigner-Seitz cell | 28 |
| Input parameters | 32 |
| External profiles | 39 |
| Format statements | 40 |
| Global strings for printing output | 41 |
| Variables for use in strutinsky module | 42 |
| Variables for use in pairing module | 45 |
| Variables for electron contributions | 48 |

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

| | | |
|----------------------------|---|-----|
| pairing | Module to hold routines for carrying out Strutinsky correction | 49 |
| parameters | Module to hold global parameters and variables | 55 |
| routines | Module to hold main routines: densities, energies | 64 |
| rspace | Module to hold routines for calculating single particle energies. Written by A. Pastore, some variables and use statements modified by M. Shelley for integration into etf code | 95 |
| strutinsky | Module to hold routines for carrying out Strutinsky correction | 97 |
| test | Module to hold test routines | 101 |

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| | |
|--|-----|
| srcs/ boundary.f90 | 111 |
| srcs/ etf.f90 | 111 |
| srcs/ pairing.f90 | 115 |
| srcs/ parameters.f90 | 115 |
| srcs/ routines.f90 | 122 |
| srcs/ strutinsky.f90 | 124 |
| srcs/ test.f90 | 124 |

Chapter 4

Module Documentation

4.1 Global parameters

Variables

- `real(kind=dp), parameter parameters::pi = 3.1415926535897932_dp`
 π
- `real(kind=dp), parameter parameters::hbar2_2m = 20.73553_dp`
 $\frac{\hbar^2}{2m}$ for SLy forces
- `real(kind=dp), parameter parameters::rho0 = 0.16_dp`
 ρ_0 , nuclear saturation density [fm^{-3}]
- `real(kind=dp), parameter parameters::e2 = 1.439978408596513_dp`
 e^2 , electric charge squared [$MeV \cdot fm$]
- `real(kind=dp), parameter parameters::mec2 = 0.51099895_dp`
Electron rest mass [MeV/c^2].
- `real(kind=dp), parameter parameters::mnc2 = 939.56542052_dp`
- `real(kind=dp), parameter parameters::mpc2 = 938.27208816_dp`
- `real(kind=dp), parameter parameters::hbarc = 197.3269804_dp`
 $\hbar * c$

4.1.1 Detailed Description

4.1.2 Variable Documentation

4.1.2.1 e2

`real(kind=dp), parameter parameters::e2 = 1.439978408596513_dp`

e^2 , electric charge squared [$MeV \cdot fm$]

4.1.2.2 hbar2_2m

`real(kind=dp), parameter parameters::hbar2_2m = 20.73553_dp`

$\frac{\hbar^2}{2m}$ for SLy forces

4.1.2.3 hbarc

`real(kind=dp), parameter parameters::hbarc = 197.3269804_dp`

$\hbar * c$

4.1.2.4 mec2

`real(kind=dp), parameter parameters::mec2 = 0.51099895_dp`

Electron rest mass $[MeV/c^2]$.

4.1.2.5 mnc2

`real(kind=dp), parameter parameters::mnc2 = 939.56542052_dp`

4.1.2.6 mpc2

`real(kind=dp), parameter parameters::mpc2 = 938.27208816_dp`

4.1.2.7 pi

`real(kind=dp), parameter parameters::pi = 3.1415926535897932_dp`

π

4.1.2.8 rho0

`real(kind=dp), parameter parameters::rho0 = 0.16_dp`

ρ_0 , nuclear saturation density $[fm^{-3}]$

4.2 Extra constants for calculating $\frac{\hbar^2}{2m}$ for BSk forces

Variables

- `real(kind=dp), parameter parameters::mamuc2 = 931.49386_dp`
Atomic mass unit u .
- `real(kind=dp), parameter parameters::xmh = 7.28896940_dp`
- `real(kind=dp), parameter parameters::rydb = 13.6056981e-6_dp`
Rydberg constant.
- `real(kind=dp), parameter parameters::xmp = xmh - mec2 + rydb`
- `real(kind=dp), parameter parameters::xmn = 8.07132281_dp`

4.2.1 Detailed Description

4.2.2 Variable Documentation

4.2.2.1 mamuc2

```
real(kind=dp), parameter parameters::mamuc2 = 931.49386_dp
```

Atomic mass unit u .

4.2.2.2 rydb

```
real(kind=dp), parameter parameters::rydb = 13.6056981e-6_dp
```

Rydberg constant.

4.2.2.3 xmh

```
real(kind=dp), parameter parameters::xmh = 7.28896940_dp
```

4.2.2.4 xmn

```
real(kind=dp), parameter parameters::xmn = 8.07132281_dp
```

4.2.2.5 xmp

```
real(kind=dp), parameter parameters::xmp = xmh - mec2 + rydb
```

4.3 Numerical fractions

Variables

- `real(kind=dp), parameter parameters::fr12 = 1._dp/2`
- `real(kind=dp), parameter parameters::fr14 = 1._dp/4`
- `real(kind=dp), parameter parameters::fr18 = 1._dp/8`
- `real(kind=dp), parameter parameters::fr1_16 = 1._dp/16`
- `real(kind=dp), parameter parameters::fr1_32 = 1._dp/32`
- `real(kind=dp), parameter parameters::fr13 = 1._dp/3`
- `real(kind=dp), parameter parameters::fr16 = 1._dp/6`
- `real(kind=dp), parameter parameters::fr1_12 = 1._dp/12`
- `real(kind=dp), parameter parameters::fr1_24 = 1._dp/24`
- `real(kind=dp), parameter parameters::fr1_36 = 1._dp/36`
- `real(kind=dp), parameter parameters::fr23 = 2._dp/3`
- `real(kind=dp), parameter parameters::fr29 = 2._dp/9`
- `real(kind=dp), parameter parameters::fr32 = 3._dp/2`
- `real(kind=dp), parameter parameters::fr34 = 3._dp/4`
- `real(kind=dp), parameter parameters::fr35 = 3._dp/5`
- `real(kind=dp), parameter parameters::fr38 = 3._dp/8`
- `real(kind=dp), parameter parameters::fr3_10 = 3._dp/10`
- `real(kind=dp), parameter parameters::fr43 = 4._dp/3`
- `real(kind=dp), parameter parameters::fr53 = 5._dp/3`
- `real(kind=dp), parameter parameters::fr54 = 5._dp/4`
- `real(kind=dp), parameter parameters::fr83 = 8._dp/3`
- `real(kind=dp), parameter parameters::fr260_3 = 260._dp/3`

4.3.1 Detailed Description

4.3.2 Variable Documentation

4.3.2.1 fr12

`real(kind=dp), parameter parameters::fr12 = 1._dp/2`

4.3.2.2 fr13

`real(kind=dp), parameter parameters::fr13 = 1._dp/3`

4.3.2.3 fr14

`real(kind=dp), parameter parameters::fr14 = 1._dp/4`

4.3.2.4 fr16

```
real(kind=dp), parameter parameters::fr16 = 1._dp/6
```

4.3.2.5 fr18

```
real(kind=dp), parameter parameters::fr18 = 1._dp/8
```

4.3.2.6 fr1_12

```
real(kind=dp), parameter parameters::fr1_12 = 1._dp/12
```

4.3.2.7 fr1_16

```
real(kind=dp), parameter parameters::fr1_16 = 1._dp/16
```

4.3.2.8 fr1_24

```
real(kind=dp), parameter parameters::fr1_24 = 1._dp/24
```

4.3.2.9 fr1_32

```
real(kind=dp), parameter parameters::fr1_32 = 1._dp/32
```

4.3.2.10 fr1_36

```
real(kind=dp), parameter parameters::fr1_36 = 1._dp/36
```

4.3.2.11 fr23

```
real(kind=dp), parameter parameters::fr23 = 2._dp/3
```

4.3.2.12 fr260_3

`real(kind=dp), parameter parameters::fr260_3 = 260._dp/3`

4.3.2.13 fr29

`real(kind=dp), parameter parameters::fr29 = 2._dp/9`

4.3.2.14 fr32

`real(kind=dp), parameter parameters::fr32 = 3._dp/2`

4.3.2.15 fr34

`real(kind=dp), parameter parameters::fr34 = 3._dp/4`

4.3.2.16 fr35

`real(kind=dp), parameter parameters::fr35 = 3._dp/5`

4.3.2.17 fr38

`real(kind=dp), parameter parameters::fr38 = 3._dp/8`

4.3.2.18 fr3_10

`real(kind=dp), parameter parameters::fr3_10 = 3._dp/10`

4.3.2.19 fr43

`real(kind=dp), parameter parameters::fr43 = 4._dp/3`

4.3.2.20 fr53

```
real(kind=dp), parameter parameters::fr53 = 5._dp/3
```

4.3.2.21 fr54

```
real(kind=dp), parameter parameters::fr54 = 5._dp/4
```

4.3.2.22 fr83

```
real(kind=dp), parameter parameters::fr83 = 8._dp/3
```

4.4 Skyrme parameters

Variables

- `real(kind=dp) parameters::w0`
Spin-orbit strength W_0 .
- `real(kind=dp) parameters::t0`
- `real(kind=dp) parameters::x0`
- `real(kind=dp) parameters::t1`
- `real(kind=dp) parameters::x1`
- `real(kind=dp) parameters::t2`
- `real(kind=dp) parameters::x2`
- `real(kind=dp) parameters::t3`
- `real(kind=dp) parameters::x3`
- `real(kind=dp) parameters::sigma`
- `real(kind=dp), dimension(0:1) parameters::hbar2_2m_q`
 $\frac{\hbar^2}{2m}$ with isospin dependence, as required by BSk forces
- `logical parameters::j2_terms`
Whether functional uses \mathbf{J}^2 terms.

4.4.1 Detailed Description

4.4.2 Variable Documentation

4.4.2.1 hbar2_2m_q

`real(kind=dp), dimension(0:1) parameters::hbar2_2m_q`

$\frac{\hbar^2}{2m}$ with isospin dependence, as required by BSk forces

4.4.2.2 j2_terms

`logical parameters::j2_terms`

Whether functional uses \mathbf{J}^2 terms.

4.4.2.3 sigma

`real(kind=dp) parameters::sigma`

4.4.2.4 t0

```
real(kind=dp) parameters::t0
```

4.4.2.5 t1

```
real(kind=dp) parameters::t1
```

4.4.2.6 t2

```
real(kind=dp) parameters::t2
```

4.4.2.7 t3

```
real(kind=dp) parameters::t3
```

4.4.2.8 w0

```
real(kind=dp) parameters::w0
```

Spin-orbit strength W_0 .

4.4.2.9 x0

```
real(kind=dp) parameters::x0
```

4.4.2.10 x1

```
real(kind=dp) parameters::x1
```

4.4.2.11 x2

```
real(kind=dp) parameters::x2
```

4.4.2.12 x3

```
real(kind=dp) parameters::x3
```

4.5 Skyrme coefficients

Variables

- `real(kind=dp) parameters::b1`
- `real(kind=dp) parameters::b2`
- `real(kind=dp) parameters::b3`
- `real(kind=dp) parameters::b4`
- `real(kind=dp) parameters::b5`
- `real(kind=dp) parameters::b6`
- `real(kind=dp) parameters::b7`
- `real(kind=dp) parameters::b8`
- `real(kind=dp) parameters::b9`
- `real(kind=dp) parameters::b10`
- `real(kind=dp) parameters::b11`
- `real(kind=dp) parameters::b12`
- `real(kind=dp) parameters::b13`

4.5.1 Detailed Description

4.5.2 Variable Documentation

4.5.2.1 b1

`real(kind=dp) parameters::b1`

4.5.2.2 b10

`real(kind=dp) parameters::b10`

4.5.2.3 b11

`real(kind=dp) parameters::b11`

4.5.2.4 b12

`real(kind=dp) parameters::b12`

4.5.2.5 b13

```
real(kind=dp) parameters::b13
```

4.5.2.6 b2

```
real(kind=dp) parameters::b2
```

4.5.2.7 b3

```
real(kind=dp) parameters::b3
```

4.5.2.8 b4

```
real(kind=dp) parameters::b4
```

4.5.2.9 b5

```
real(kind=dp) parameters::b5
```

4.5.2.10 b6

```
real(kind=dp) parameters::b6
```

4.5.2.11 b7

```
real(kind=dp) parameters::b7
```

4.5.2.12 b8

```
real(kind=dp) parameters::b8
```

4.5.2.13 b9

```
real(kind=dp) parameters::b9
```

4.6 Extra parameters for BSk forces

Variables

- `real(kind=dp) parameters::alpha`
- `real(kind=dp) parameters::beta`
- `real(kind=dp) parameters::gamma`
- `real(kind=dp) parameters::t4`
- `real(kind=dp) parameters::x4`
- `real(kind=dp) parameters::t5`
- `real(kind=dp) parameters::x5`
- `real(kind=dp) parameters::fn_pos`
- `real(kind=dp) parameters::fn_neg`
- `real(kind=dp) parameters::fp_pos`
- `real(kind=dp) parameters::fp_neg`
- `real(kind=dp) parameters::epsilon_lambda`

4.6.1 Detailed Description

4.6.2 Variable Documentation

4.6.2.1 alpha

`real(kind=dp) parameters::alpha`

4.6.2.2 beta

`real(kind=dp) parameters::beta`

4.6.2.3 epsilon_lambda

`real(kind=dp) parameters::epsilon_lambda`

4.6.2.4 fn_neg

`real(kind=dp) parameters::fn_neg`

4.6.2.5 fn_pos

```
real(kind=dp) parameters::fn_pos
```

4.6.2.6 fp_neg

```
real(kind=dp) parameters::fp_neg
```

4.6.2.7 fp_pos

```
real(kind=dp) parameters::fp_pos
```

4.6.2.8 gamma

```
real(kind=dp) parameters::gamma
```

4.6.2.9 t4

```
real(kind=dp) parameters::t4
```

4.6.2.10 t5

```
real(kind=dp) parameters::t5
```

4.6.2.11 x4

```
real(kind=dp) parameters::x4
```

4.6.2.12 x5

```
real(kind=dp) parameters::x5
```

4.7 Mesh variable

Variables

- `real(kind=dp), dimension(:), allocatable parameters::r`
Mesh for r.

4.7.1 Detailed Description

4.7.2 Variable Documentation

4.7.2.1 r

```
real(kind=dp), dimension(:), allocatable parameters::r
```

Mesh for r.

4.8 Global density arrays

Variables

- `real(kind=dp), dimension(:, :), allocatable parameters::rho_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::del_rho_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::del2_rho_q`
- `real(kind=dp), dimension(:), allocatable parameters::rho_t`
- `real(kind=dp), dimension(:), allocatable parameters::del_rho_t`
- `real(kind=dp), dimension(:), allocatable parameters::del2_rho_t`

4.8.1 Detailed Description

4.8.2 Variable Documentation

4.8.2.1 del2_rho_q

```
real(kind=dp), dimension(:, :), allocatable parameters::del2_rho_q
```

4.8.2.2 del2_rho_t

```
real(kind=dp), dimension(:), allocatable parameters::del2_rho_t
```

4.8.2.3 del_rho_q

```
real(kind=dp), dimension(:, :), allocatable parameters::del_rho_q
```

4.8.2.4 del_rho_t

```
real(kind=dp), dimension(:), allocatable parameters::del_rho_t
```

4.8.2.5 rho_q

```
real(kind=dp), dimension(:, :), allocatable parameters::rho_q
```

4.8.2.6 rho_t

```
real(kind=dp), dimension(:), allocatable parameters::rho_t
```

4.9 Global effective mass arrays

Variables

- `real(kind=dp), dimension(:, :), allocatable parameters::f_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::del_f_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::del2_f_q`

4.9.1 Detailed Description

4.9.2 Variable Documentation

4.9.2.1 del2_f_q

`real(kind=dp), dimension(:, :), allocatable parameters::del2_f_q`

4.9.2.2 del_f_q

`real(kind=dp), dimension(:, :), allocatable parameters::del_f_q`

4.9.2.3 f_q

`real(kind=dp), dimension(:, :), allocatable parameters::f_q`

4.10 Global arrays for other densities and fields

Variables

- `real(kind=dp), dimension(:, :), allocatable parameters::w_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::div_w_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::u_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::j_2_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::j_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::del_j_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::div_j_2_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::div_j_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::tau_tf_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::tau_2_l_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::tau_2_nl_q`
- `real(kind=dp), dimension(:, :), allocatable parameters::tau_etf_q`
- `real(kind=dp), dimension(:), allocatable parameters::j_t`
- `real(kind=dp), dimension(:), allocatable parameters::del_j_t`
- `real(kind=dp), dimension(:), allocatable parameters::div_j_t`
- `real(kind=dp), dimension(:), allocatable parameters::tau_etf_t`
- `real(kind=dp), dimension(:), allocatable parameters::e_density_field`
- `real(kind=dp), dimension(:), allocatable parameters::e_density_sky`
- `real(kind=dp), dimension(:), allocatable parameters::v_c_di`
- `real(kind=dp), dimension(:), allocatable parameters::v_c_ex`
- `real(kind=dp), dimension(:), allocatable parameters::e_density_c_di`
- `real(kind=dp), dimension(:), allocatable parameters::e_density_c_ex`
- `real(kind=dp), dimension(:), allocatable parameters::v_c_pe`
- `real(kind=dp), dimension(:, :, :), allocatable parameters::tau_2_cont_q`
- `real(kind=dp), dimension(:), allocatable parameters::rho_ch`

Charge density.

4.10.1 Detailed Description

4.10.2 Variable Documentation

4.10.2.1 `del_j_q`

```
real(kind=dp), dimension(:, :), allocatable parameters::del_j_q
```

4.10.2.2 `del_j_t`

```
real(kind=dp), dimension(:), allocatable parameters::del_j_t
```

4.10.2.3 div_j_2_q

`real(kind=dp), dimension(:, :), allocatable parameters::div_j_2_q`

4.10.2.4 div_j_q

`real(kind=dp), dimension(:, :), allocatable parameters::div_j_q`

4.10.2.5 div_j_t

`real(kind=dp), dimension(:), allocatable parameters::div_j_t`

4.10.2.6 div_w_q

`real(kind=dp), dimension(:, :), allocatable parameters::div_w_q`

4.10.2.7 e_density_c_di

`real(kind=dp), dimension(:), allocatable parameters::e_density_c_di`

4.10.2.8 e_density_c_ex

`real(kind=dp), dimension(:), allocatable parameters::e_density_c_ex`

4.10.2.9 e_density_field

`real(kind=dp), dimension(:), allocatable parameters::e_density_field`

4.10.2.10 e_density_sky

`real(kind=dp), dimension(:), allocatable parameters::e_density_sky`

4.10.2.11 j_2_q

```
real(kind=dp), dimension(:,:), allocatable parameters::j_2_q
```

4.10.2.12 j_q

```
real(kind=dp), dimension(:,:), allocatable parameters::j_q
```

4.10.2.13 j_t

```
real(kind=dp), dimension(:), allocatable parameters::j_t
```

4.10.2.14 rho_ch

```
real(kind=dp), dimension(:), allocatable parameters::rho_ch
```

Charge density.

4.10.2.15 tau_2_cont_q

```
real(kind=dp), dimension(:,:,:), allocatable parameters::tau_2_cont_q
```

4.10.2.16 tau_2_l_q

```
real(kind=dp), dimension(:,:), allocatable parameters::tau_2_l_q
```

4.10.2.17 tau_2_nl_q

```
real(kind=dp), dimension(:,:), allocatable parameters::tau_2_nl_q
```

4.10.2.18 tau_etf_q

```
real(kind=dp), dimension(:,:), allocatable parameters::tau_etf_q
```

4.10.2.19 tau_etf_t

```
real(kind=dp), dimension(:), allocatable parameters::tau_etf_t
```

4.10.2.20 tau_tf_q

```
real(kind=dp), dimension(:,:), allocatable parameters::tau_tf_q
```

4.10.2.21 u_q

```
real(kind=dp), dimension(:,:), allocatable parameters::u_q
```

4.10.2.22 v_c_di

```
real(kind=dp), dimension(:), allocatable parameters::v_c_di
```

4.10.2.23 v_c_ex

```
real(kind=dp), dimension(:), allocatable parameters::v_c_ex
```

4.10.2.24 v_c_pe

```
real(kind=dp), dimension(:), allocatable parameters::v_c_pe
```

4.10.2.25 w_q

```
real(kind=dp), dimension(:,:), allocatable parameters::w_q
```

4.11 Global arrays for densities and fields for 4th-order terms

4.12 Variables for storing properties of Wigner-Seitz cell

Variables

- `real(kind=dp), dimension(0:1) parameters::n_q`
Number of neutrons and protons.
- `real(kind=dp) parameters::n_t`
Total number of particles.
- `real(kind=dp) parameters::e_field`
Field energy.
- `real(kind=dp) parameters::e_skyrme`
Skyrme energy.
- `real(kind=dp), dimension(0:1) parameters::e_kinetic_q`
Kinetic energy for neutron and protons.
- `real(kind=dp) parameters::e_kinetic_t`
Total kinetic energy.
- `real(kind=dp) parameters::e_so`
Spin-orbit energy.
- `real(kind=dp) parameters::e_coulomb_di`
Direct Coulomb energy.
- `real(kind=dp) parameters::e_coulomb_ex`
Exchange Coulomb energy.
- `real(kind=dp) parameters::e_coulomb`
Total Coulomb energy.
- `real(kind=dp) parameters::e_total`
Total energy.
- `real(kind=dp), dimension(0:1) parameters::mu_q`
Neutron and proton chemical potential.
- `real(kind=dp) parameters::mu_e`
Electron chemical potential.
- `real(kind=dp) parameters::mu_c`
Coulomb interaction contribution to electron chemical potential.
- `real(kind=dp) parameters::delta_mu`
Overall chemical potential (beta-equilibrium condition)
- `real(kind=dp) parameters::pressure_nucl`
Nuclear pressure.
- `real(kind=dp) parameters::pressure_e`
Electron pressure.
- `real(kind=dp) parameters::pressure_ex`
Coulomb exchange pressure.
- `real(kind=dp) parameters::pressure_t`
Total pressure.

4.12.1 Detailed Description

4.12.2 Variable Documentation

4.12.2.1 delta_mu

```
real(kind=dp) parameters::delta_mu
```

Overall chemical potential (beta-equilibrium condition)

4.12.2.2 e_coulomb

```
real(kind=dp) parameters::e_coulomb
```

Total Coulomb energy.

4.12.2.3 e_coulomb_di

```
real(kind=dp) parameters::e_coulomb_di
```

Direct Coulomb energy.

4.12.2.4 e_coulomb_ex

```
real(kind=dp) parameters::e_coulomb_ex
```

Exchange Coulomb energy.

4.12.2.5 e_field

```
real(kind=dp) parameters::e_field
```

Field energy.

4.12.2.6 e_kinetic_q

```
real(kind=dp), dimension(0:1) parameters::e_kinetic_q
```

Kinetic energy for neutron and protons.

4.12.2.7 e_kinetic_t

```
real(kind=dp) parameters::e_kinetic_t
```

Total kinetic energy.

4.12.2.8 e_skyrme

```
real(kind=dp) parameters::e_skyrme
```

Skyrme energy.

4.12.2.9 e_so

```
real(kind=dp) parameters::e_so
```

Spin-orbit energy.

4.12.2.10 e_total

```
real(kind=dp) parameters::e_total
```

Total energy.

4.12.2.11 mu_c

```
real(kind=dp) parameters::mu_c
```

Coulomb interaction contribution to electron chemical potential.

4.12.2.12 mu_e

```
real(kind=dp) parameters::mu_e
```

Electron chemical potential.

4.12.2.13 mu_q

```
real(kind=dp), dimension(0:1) parameters::mu_q
```

Neutron and proton chemical potential.

4.12.2.14 n_q

```
real(kind=dp), dimension(0:1) parameters::n_q
```

Number of neutrons and protons.

4.12.2.15 n_t

```
real(kind=dp) parameters::n_t
```

Total number of particles.

4.12.2.16 pressure_e

```
real(kind=dp) parameters::pressure_e
```

Electron pressure.

4.12.2.17 pressure_ex

```
real(kind=dp) parameters::pressure_ex
```

Coulomb exchange pressure.

4.12.2.18 pressure_nucl

```
real(kind=dp) parameters::pressure_nucl
```

Nuclear pressure.

4.12.2.19 pressure_t

```
real(kind=dp) parameters::pressure_t
```

Total pressure.

4.13 Input parameters

Variables

- integer `parameters::run_mode`
Mode to run code in (normal, test)
- logical `parameters::verbose`
Whether to print all extra messages about run.
- real(kind=dp) `parameters::dr`
Mesh spacing of r (fm).
- logical `parameters::profile_r_max`
Whether to take "r_max" from "r_ws".
- real(kind=dp) `parameters::r_max`
Max value of r (fm).
- logical `parameters::specify_n`
Whether to specify "n" instead of "dr".
- integer `parameters::n`
Number of mesh points.
- integer `parameters::force`
Force to use.
- integer `parameters::etf_order`
Order at which to calculate kinetic energy densities.
- logical `parameters::coulomb_on`
Use Coulomb interaction or not.
- logical `parameters::electrons_on`
Add electrons.
- integer `parameters::nmaxstate`
Number of states that can be stored.
- integer `parameters::lmax`
Maximum angular momentum of states to find.
- logical `parameters::calc_chem_pots`
Calculate chemical potentials.
- logical, dimension(0:1) `parameters::strutinsky_on`
Use Strutinsky Integral (SI) correction for neutrons and protons.
- real(kind=dp) `parameters::strut_r_max`
Max value of r to use for box for single particle states (fm).
- logical `parameters::emax0_mod`
Whether to modify "emax0" to always = 0.
- integer `parameters::neutron_pairing`
Type of pairing calculation to perform for neutrons.
- logical `parameters::proton_bcs`
Whether to do BCS for protons.
- real(kind=dp) `parameters::pair_qp_cut`
Cut-off for quasiparticle energy for solving gap equation.
- logical `parameters::smooth_qp_cut`
Whether to use smooth cut-off on quasiparticle energy.
- real(kind=dp) `parameters::pair_k_max`
Maximum momentum for integral in gap equation.
- real(kind=dp) `parameters::pair_v0`
Interaction strength parameter.

- `real(kind=dp) parameters::pair_eta`
Interaction parameter eta.
- `real(kind=dp) parameters::pair_alpha`
Interaction parameter alpha.
- `real(kind=dp) parameters::pair_tol`
Gap equation self-consistency tolerance.
- `real(kind=dp) parameters::pair_mix`
Mix of previous iteration in gap equation self-consistency loop.
- `real(kind=dp) parameters::pair_dk`
Step size in integral in gap equation.
- `real(kind=dp) parameters::pair_del_init`
Initial guess for pairing gap.
- `integer parameters::pair_max_iters`
Maximum iterations for gap equation self-consistency loop.
- `real(kind=dp) parameters::pair_num_tol`
BCS number equation self-consistency tolerance.

4.13.1 Detailed Description

4.13.2 Variable Documentation

4.13.2.1 `calc_chem_pots`

`logical parameters::calc_chem_pots`

Calculate chemical potentials.

4.13.2.2 `coulomb_on`

`logical parameters::coulomb_on`

Use Coulomb interaction or not.

4.13.2.3 `dr`

`real(kind=dp) parameters::dr`

Mesh spacing of r (fm).

4.13.2.4 electrons_on

```
logical parameters::electrons_on
```

Add electrons.

4.13.2.5 emax0_mod

```
logical parameters::emax0_mod
```

Whether to modify "emax0" to always = 0.

4.13.2.6 etf_order

```
integer parameters::etf_order
```

Order at which to calculate kinetic energy densities.

4.13.2.7 force

```
integer parameters::force
```

Force to use.

4.13.2.8 lmax

```
integer parameters::lmax
```

Maximum angular momentum of states to find.

4.13.2.9 n

```
integer parameters::n
```

Number of mesh points.

4.13.2.10 neutron_pairing

```
integer parameters::neutron_pairing
```

Type of pairing calculation to perform for neutrons.

4.13.2.11 nmaxstate

```
integer parameters::nmaxstate
```

Number of states that can be stored.

4.13.2.12 pair_alpha

```
real(kind=dp) parameters::pair_alpha
```

Interaction parameter alpha.

4.13.2.13 pair_del_init

```
real(kind=dp) parameters::pair_del_init
```

Initial guess for pairing gap.

4.13.2.14 pair_dk

```
real(kind=dp) parameters::pair_dk
```

Step size in integral in gap equation.

4.13.2.15 pair_eta

```
real(kind=dp) parameters::pair_eta
```

Interaction parameter eta.

4.13.2.16 pair_k_max

```
real(kind=dp) parameters::pair_k_max
```

Maximum momentum for integral in gap equation.

4.13.2.17 pair_max_iters

```
integer parameters::pair_max_iters
```

Maximum iterations for gap equation self-consistency loop.

4.13.2.18 pair_mix

```
real(kind=dp) parameters::pair_mix
```

Mix of previous iteration in gap equation self-consistency loop.

4.13.2.19 pair_num_tol

```
real(kind=dp) parameters::pair_num_tol
```

BCS number equation self-consistency tolerance.

4.13.2.20 pair_qp_cut

```
real(kind=dp) parameters::pair_qp_cut
```

Cut-off for quasiparticle energy for solving gap equation.

4.13.2.21 pair_tol

```
real(kind=dp) parameters::pair_tol
```

Gap equation self-consistency tolerance.

4.13.2.22 pair_v0

```
real(kind=dp) parameters::pair_v0
```

Interaction strength parameter.

4.13.2.23 profile_r_max

```
logical parameters::profile_r_max
```

Whether to take "r_max" from "r_ws".

4.13.2.24 proton_bcs

```
logical parameters::proton_bcs
```

Whether to do BCS for protons.

4.13.2.25 r_max

```
real(kind=dp) parameters::r_max
```

Max value of r (fm).

4.13.2.26 run_mode

```
integer parameters::run_mode
```

Mode to run code in (normal, test)

4.13.2.27 smooth_qp_cut

```
logical parameters::smooth_qp_cut
```

Whether to use smooth cut-off on quasiparticle energy.

4.13.2.28 specify_n

```
logical parameters::specify_n
```

Whether to specify "n" instead of "dr".

4.13.2.29 strut_r_max

```
real(kind=dp) parameters::strut_r_max
```

Max value of r to use for box for single particle states (fm).

4.13.2.30 strutinsky_on

```
logical, dimension(0:1) parameters::strutinsky_on
```

Use Strutinsky Integral (SI) correction for neutrons and protons.

4.13.2.31 verbose

```
logical parameters::verbose
```

Whether to print all extra messages about run.

4.14 External profiles

Variables

- `real(kind=dp), dimension(5) parameters::n_profile`
- `real(kind=dp), dimension(5) parameters::p_profile`
- `real(kind=dp) parameters::num_n`
- `integer parameters::num_p`
- `real(kind=dp) parameters::r_ws`

4.14.1 Detailed Description

4.14.2 Variable Documentation

4.14.2.1 n_profile

```
real(kind=dp), dimension(5) parameters::n_profile
```

4.14.2.2 num_n

```
real(kind=dp) parameters::num_n
```

4.14.2.3 num_p

```
integer parameters::num_p
```

4.14.2.4 p_profile

```
real(kind=dp), dimension(5) parameters::p_profile
```

4.14.2.5 r_ws

```
real(kind=dp) parameters::r_ws
```

4.15 Format statements

Variables

- `character(18), dimension(10) parameters::n_floats = (/ 1(es24.16e3,1x)), ' 2(es24.16e3,1x)), ' 3(es24.16e3,1x)), ' 4(es24.16e3,1x)), ' 5(es24.16e3,1x)), ' 6(es24.16e3,1x)), ' 7(es24.16e3,1x)), ' 8(es24.16e3,1x)), ' 9(es24.16e3,1x)), ' 10(es24.16e3,1x)) /)`
- `character(27) parameters::info_format = 'a27,1x,a1,1x,es24.16e3'`

4.15.1 Detailed Description

4.15.2 Variable Documentation

4.15.2.1 info_format

```
character(27) parameters::info_format = 'a27,1x,a1,1x,es24.16e3'
```

4.15.2.2 n_floats

```
character(18), dimension(10) parameters::n_floats = (/ 1(es24.16e3,1x)), ' 2(es24.16e3,1x)), ' 3(es24.16e3,1x)), ' 4(es24.16e3,1x)), ' 5(es24.16e3,1x)), ' 6(es24.16e3,1x)), ' 7(es24.16e3,1x)), ' 8(es24.16e3,1x)), ' 9(es24.16e3,1x)), ' 10(es24.16e3,1x)) /)
```


4.16 Global strings for printing output

Variables

- character(8) `parameters::force_string`
- character(80), parameter `parameters::line_break` = '#####'
Separating output.
- character(62), parameter `parameters::table_string` = '-----+-----'
- character(1), dimension(0:1), parameter `parameters::iso_string` = (/ 'n', 'p' /)
Isospin labels.
- character(4), parameter `parameters::space4` = ' '
Whitespace of length 4.

4.16.1 Detailed Description

4.16.2 Variable Documentation

4.16.2.1 `force_string`

character(8) `parameters::force_string`

4.16.2.2 `iso_string`

character(1), dimension(0:1), parameter `parameters::iso_string` = (/ 'n', 'p' /)

Isospin labels.

4.16.2.3 `line_break`

character(80), parameter `parameters::line_break` = '#####'

Separating output.

4.16.2.4 `space4`

character(4), parameter `parameters::space4` = ' '

Whitespace of length 4.

4.16.2.5 `table_string`

character(62), parameter `parameters::table_string` = '-----+-----'

4.17 Variables for use in strutinsky module

Variables

- integer `parameters::strut_n`
- real(kind=`dp`), dimension(:, :), allocatable `parameters::dhmen`
- real(kind=`dp`), dimension(:, :), allocatable `parameters::d2hmen`
- real(kind=`dp`), dimension(:, :), allocatable `parameters::hb2m`
- real(kind=`dp`), dimension(:, :), allocatable `parameters::vpot`
- real(kind=`dp`), dimension(:, :), allocatable `parameters::vso`
- real(kind=`dp`), dimension(0:1) `parameters::ecut`
Cutoff energies for neutrons and protons.
- real(kind=`dp`), dimension(:, :), allocatable `parameters::ps`
Storage for single particle wavefunctions.
- integer `parameters::nnst`
Number of states found within cutoff energy.
- integer, dimension(:), allocatable `parameters::jjp`
Storage for J of states found.
- integer, dimension(:), allocatable `parameters::lp`
Storage for L of states found.
- real(kind=`dp`), dimension(:), allocatable `parameters::ep`
Storage for energy of states found.
- real(kind=`dp`), dimension(0:1) `parameters::e_sc_q`
Storage for shell correction energies.
- real(kind=`dp`) `parameters::e_sc_t`
Storage for total shell correction energy.

4.17.1 Detailed Description

4.17.2 Variable Documentation

4.17.2.1 d2hmen

```
real(kind=dp), dimension(:, :), allocatable parameters::d2hmen
```

4.17.2.2 dhmen

```
real(kind=dp), dimension(:, :), allocatable parameters::dhmen
```

4.17.2.3 e_sc_q

```
real(kind=dp), dimension(0:1) parameters::e_sc_q
```

Storage for shell correction energies.

4.17.2.4 e_sc_t

```
real(kind=dp) parameters::e_sc_t
```

Storage for total shell correction energy.

4.17.2.5 ecut

```
real(kind=dp), dimension(0:1) parameters::ecut
```

Cutoff energies for neutrons and protons.

4.17.2.6 ep

```
real(kind=dp), dimension(:), allocatable parameters::ep
```

Storage for energy of states found.

4.17.2.7 hb2m

```
real(kind=dp), dimension(:, :), allocatable parameters::hb2m
```

4.17.2.8 jjp

```
integer, dimension(:), allocatable parameters::jjp
```

Storage for J of states found.

4.17.2.9 lp

`integer, dimension(:), allocatable parameters::lp`

Storage for L of states found.

4.17.2.10 nnst

`integer parameters::nnst`

Number of states found within cutoff energy.

4.17.2.11 ps

`real(kind=dp), dimension(:,:), allocatable parameters::ps`

Storage for single particle wavefunctions.

4.17.2.12 strut_n

`integer parameters::strut_n`

4.17.2.13 vpot

`real(kind=dp), dimension(:,:), allocatable parameters::vpot`

4.17.2.14 vso

`real(kind=dp), dimension(:,:), allocatable parameters::vso`

4.18 Variables for use in pairing module

Variables

- `real(kind=dp), dimension(:), allocatable parameters::pair_gap_n`
Neutron pairing gap.
- `real(kind=dp), dimension(:), allocatable parameters::delta_n`
Neutron pairing field.
- `real(kind=dp), dimension(:), allocatable parameters::pair_gap_p`
Proton pairing gap.
- `real(kind=dp), dimension(:), allocatable parameters::delta_p`
Proton pairing field.
- `real(kind=dp), dimension(:), allocatable parameters::rho_p_bcs`
Proton density (from wavefunctions)
- `real(kind=dp), dimension(:), allocatable parameters::rho_anom_p_bcs`
Anomalous density.
- `real(kind=dp), dimension(:), allocatable parameters::strength_bcs`
Interaction strength for protons.
- `real(kind=dp), dimension(:), allocatable parameters::occ_v`
Particle occupation probabilities.
- `real(kind=dp), dimension(:), allocatable parameters::occ_u`
Hole occupation probabilities.
- `real(kind=dp), dimension(:), allocatable parameters::e_qp_p`
Quasiparticle energies.
- `real(kind=dp) parameters::num_p_bcs`
Number of protons calculated from occupations.
- `real(kind=dp) parameters::e_pair_bcs`
BCS pairing energy.
- `real(kind=dp), dimension(0:1) parameters::e_pair_q`
Storage for pairing condensation energies.
- `real(kind=dp) parameters::e_pair_t`
Storage for total pairing condensation energy.

4.18.1 Detailed Description

4.18.2 Variable Documentation

4.18.2.1 `delta_n`

```
real(kind=dp), dimension(:), allocatable parameters::delta_n
```

Neutron pairing field.

4.18.2.2 delta_p

```
real(kind=dp), dimension(:), allocatable parameters::delta_p
```

Proton pairing field.

4.18.2.3 e_pair_bcs

```
real(kind=dp) parameters::e_pair_bcs
```

BCS pairing energy.

4.18.2.4 e_pair_q

```
real(kind=dp), dimension(0:1) parameters::e_pair_q
```

Storage for pairing condensation energies.

4.18.2.5 e_pair_t

```
real(kind=dp) parameters::e_pair_t
```

Storage for total pairing condensation energy.

4.18.2.6 e_qp_p

```
real(kind=dp), dimension(:), allocatable parameters::e_qp_p
```

Quasiparticle energies.

4.18.2.7 num_p_bcs

```
real(kind=dp) parameters::num_p_bcs
```

Number of protons calculated from occupations.

4.18.2.8 occ_u

```
real(kind=dp), dimension(:), allocatable parameters::occ_u
```

Hole occupation probabilities.

4.18.2.9 occ_v

```
real(kind=dp), dimension(:), allocatable parameters::occ_v
```

Particle occupation probabilities.

4.18.2.10 pair_gap_n

```
real(kind=dp), dimension(:), allocatable parameters::pair_gap_n
```

Neutron pairing gap.

4.18.2.11 pair_gap_p

```
real(kind=dp), dimension(:), allocatable parameters::pair_gap_p
```

Proton pairing gap.

4.18.2.12 rho_anom_p_bcs

```
real(kind=dp), dimension(:), allocatable parameters::rho_anom_p_bcs
```

Anomalous density.

4.18.2.13 rho_p_bcs

```
real(kind=dp), dimension(:), allocatable parameters::rho_p_bcs
```

Proton density (from wavefunctions)

4.18.2.14 strength_bcs

```
real(kind=dp), dimension(:), allocatable parameters::strength_bcs
```

Interaction strength for protons.

4.19 Variables for electron contributions

Variables

- `real(kind=dp) parameters::rho_e`
Electron density.
- `real(kind=dp) parameters::e_kinetic_e`
Total electron kinetic energy.
- `real(kind=dp) parameters::e_coulomb_e`
Total electron-electron potential energy from Coulomb interaction.
- `real(kind=dp) parameters::e_coulomb_pe`
Total proton-electron potential energy from Coulomb interaction.

4.19.1 Detailed Description

4.19.2 Variable Documentation

4.19.2.1 e_coulomb_e

```
real(kind=dp) parameters::e_coulomb_e
```

Total electron-electron potential energy from Coulomb interaction.

4.19.2.2 e_coulomb_pe

```
real(kind=dp) parameters::e_coulomb_pe
```

Total proton-electron potential energy from Coulomb interaction.

4.19.2.3 e_kinetic_e

```
real(kind=dp) parameters::e_kinetic_e
```

Total electron kinetic energy.

4.19.2.4 rho_e

```
real(kind=dp) parameters::rho_e
```

Electron density.

Chapter 5

Module Documentation

5.1 pairing Module Reference

Module to hold routines for carrying out Strutinsky correction.

Functions/Subroutines

- subroutine [calc_e_pair](#) ()
Subroutine to calculate the pairing energy for the WS cell, using the method specified in "input.in".
- real(kind=dp) function [calc_gap](#) (r, rhon, rhop, fq, mu)
Function to calculate the pairing gap in infinite neutron matter, for specified density and effective mass.
- subroutine [bcs_protons](#) ()
Subroutine to perform BCS for protons.
- real(kind=dp) function [calc_bcs_num_p](#) (mu_p)
Function to solve gap equation at given chemical potential "mu_p", returning the (BCS) number of protons.
- real(kind=dp) function [interazchamelanalyt](#) (rhon, rhop, hbm, itz)
Function to calculate the interaction strength for the effective contact pairing force. Modified by M. Shelley.
- real(kind=dp) function [llambda](#) (x)
Function to calculate the pairing cutoff for the effective contact pairing force.
- real(kind=dp) function [delta_parametric](#) (kf, YY, itz, xk0)
Function to calculate the pairing gap using the analytical BSk expression. Modified by M. Shelley.

5.1.1 Detailed Description

Module to hold routines for carrying out Strutinsky correction.

Author

M. Shelley

5.1.2 Function/Subroutine Documentation

5.1.2.1 bcs_protons()

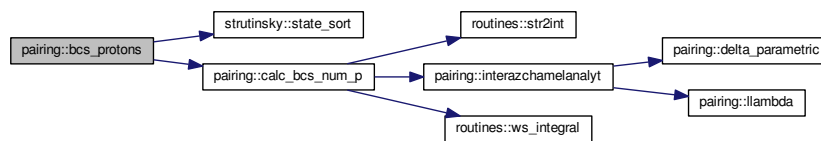
```
subroutine pairing::bcs_protons ( )
```

Subroutine to perform BCS for protons.

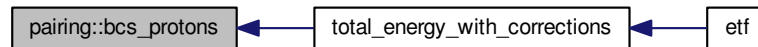
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.2 calc_bcs_num_p()

```
real(kind=dp) function pairing::calc_bcs_num_p (
    real(kind=dp), intent(in) mu_p )
```

Function to solve gap equation at given chemical potential "mu_p", returning the (BCS) number of protons.

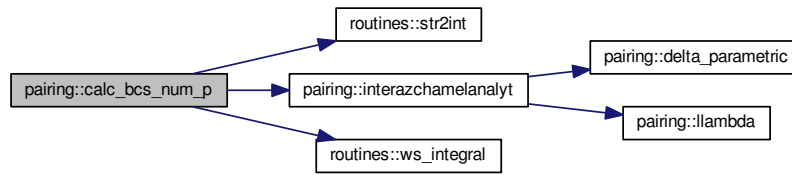
Author

M. Shelley, A. Pastore

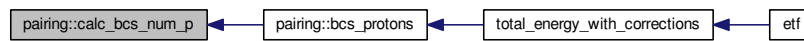
Parameters

| | | |
|----|---------|---------------------------|
| in | μ_p | Proton chemical potential |
|----|---------|---------------------------|

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.3 calc_e_pair()

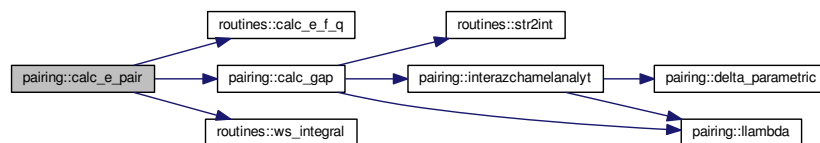
```
subroutine pairing::calc_e_pair ( )
```

Subroutine to calculate the pairing energy for the WS cell, using the method specified in "input.in".

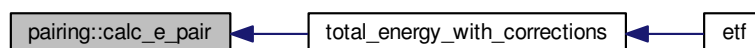
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.4 calc_gap()

```
real(kind=dp) function pairing::calc_gap (
    real(kind=dp), intent(in) r,
    real(kind=dp), intent(in) rhon,
    real(kind=dp), intent(in) rhop,
    real(kind=dp), intent(in) fq,
    real(kind=dp), intent(in) mu )
```

Function to calculate the pairing gap in infinite neutron matter, for specified density and effective mass.

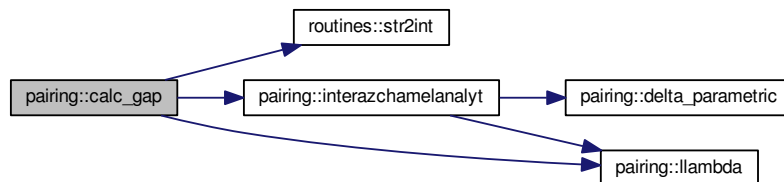
Author

M. Shelley

Parameters

| | | |
|----|-------------|------------------------------|
| in | <i>r</i> | Radius |
| in | <i>rhon</i> | Neutron density |
| in | <i>rhop</i> | Proton density |
| in | <i>fq</i> | Effective mass |
| in | <i>mu</i> | Effective chemical potential |

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.5 delta_parametric()

```
real(kind=dp) function pairing::delta_parametric (
    real(kind=dp), intent(in) kf,
```

```

real(kind=dp), intent(in) YY,
integer, intent(in) itz,
real(kind=dp), intent(in) xk0 )

```

Function to calculate the pairing gap using the analytical BSk expression. Modified by M. Shelley.

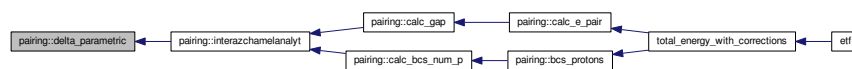
Author

A. Pastore, M. Shelley

Parameters

| | | |
|----|------------|----------------------------|
| in | <i>kf</i> | Fermi momentum |
| in | <i>YY</i> | Neutron-proton composition |
| in | <i>itz</i> | Isospin |
| in | <i>xk0</i> | "Average" fermi momentum |

Here is the caller graph for this function:



5.1.2.6 interazchamelanalyt()

```

real(kind=dp) function pairing::interazchamelanalyt (
    real(kind=dp), intent(in) rhon,
    real(kind=dp), intent(in) rhop,
    real(kind=dp), intent(in) hbm,
    integer, intent(in) itz )

```

Function to calculate the interaction strength for the effective contact pairing force. Modified by M. Shelley.

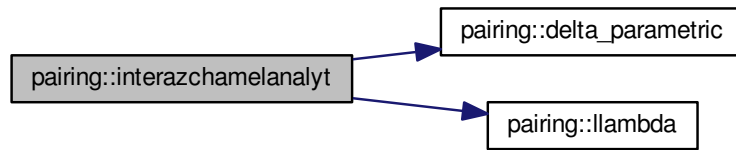
Author

A. Pastore, M. Shelley

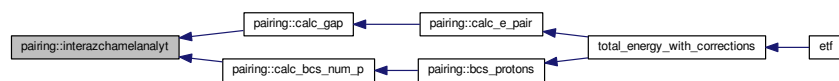
Parameters

| | | |
|----|-------------|------------------------|
| in | <i>rhon</i> | Neutron density |
| in | <i>rhop</i> | Proton density |
| in | <i>hbm</i> | $\frac{\hbar^2}{2m^*}$ |
| in | <i>itz</i> | Isospin |

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.7 llambda()

```

real(kind=dp) function pairing::llambda (
    real(kind=dp), intent(in) x )
  
```

Function to calculate the pairing cutoff for the effective contact pairing force.

Author

A. Pastore

Parameters

| | | |
|----|---|--------------------------------|
| in | x | (DDCI cutoff) / (Fermi energy) |
|----|---|--------------------------------|

Here is the caller graph for this function:



5.2 parameters Module Reference

Module to hold global parameters and variables.

Functions/Subroutines

- subroutine `force_initialise ()`
Subroutine to initialise Skyrme force parameters with values for a given force specified in 'input.dat'.

Variables

- integer, parameter `dp` = selected_real_kind(15, 300)
Precision.
- real(kind=`dp`), parameter `pi` = 3.1415926535897932_dp
 π
- real(kind=`dp`), parameter `hbar2_2m` = 20.73553_dp
 $\frac{\hbar^2}{2m}$ for SLy forces
- real(kind=`dp`), parameter `rho0` = 0.16_dp
 ρ_0 , nuclear saturation density [fm^{-3}]
- real(kind=`dp`), parameter `e2` = 1.439978408596513_dp
 e^2 , electric charge squared [$MeV \cdot fm$]
- real(kind=`dp`), parameter `mec2` = 0.51099895_dp
Electron rest mass [MeV/c^2].
- real(kind=`dp`), parameter `mnc2` = 939.56542052_dp
- real(kind=`dp`), parameter `mpc2` = 938.27208816_dp
- real(kind=`dp`), parameter `hbarc` = 197.3269804_dp
 $\hbar * c$
- real(kind=`dp`), parameter `mamuc2` = 931.49386_dp
Atomic mass unit u.
- real(kind=`dp`), parameter `xmh` = 7.28896940_dp
- real(kind=`dp`), parameter `rydb` = 13.6056981e-6_dp
Rydberg constant.
- real(kind=`dp`), parameter `xmp` = `xmh` - `mec2` + `rydb`
- real(kind=`dp`), parameter `xmn` = 8.07132281_dp
- real(kind=`dp`), parameter `fr12` = 1._dp/2
- real(kind=`dp`), parameter `fr14` = 1._dp/4
- real(kind=`dp`), parameter `fr18` = 1._dp/8
- real(kind=`dp`), parameter `fr1_16` = 1._dp/16
- real(kind=`dp`), parameter `fr1_32` = 1._dp/32
- real(kind=`dp`), parameter `fr13` = 1._dp/3
- real(kind=`dp`), parameter `fr16` = 1._dp/6
- real(kind=`dp`), parameter `fr1_12` = 1._dp/12
- real(kind=`dp`), parameter `fr1_24` = 1._dp/24
- real(kind=`dp`), parameter `fr1_36` = 1._dp/36
- real(kind=`dp`), parameter `fr23` = 2._dp/3
- real(kind=`dp`), parameter `fr29` = 2._dp/9
- real(kind=`dp`), parameter `fr32` = 3._dp/2
- real(kind=`dp`), parameter `fr34` = 3._dp/4
- real(kind=`dp`), parameter `fr35` = 3._dp/5
- real(kind=`dp`), parameter `fr38` = 3._dp/8

- real(kind=dp), parameter fr3_10 = 3._dp/10
- real(kind=dp), parameter fr43 = 4._dp/3
- real(kind=dp), parameter fr53 = 5._dp/3
- real(kind=dp), parameter fr54 = 5._dp/4
- real(kind=dp), parameter fr83 = 8._dp/3
- real(kind=dp), parameter fr260_3 = 260._dp/3
- real(kind=dp) w0
- Spin-orbit strength W_0 .*
- real(kind=dp) t0
- real(kind=dp) x0
- real(kind=dp) t1
- real(kind=dp) x1
- real(kind=dp) t2
- real(kind=dp) x2
- real(kind=dp) t3
- real(kind=dp) x3
- real(kind=dp) sigma
- real(kind=dp), dimension(0:1) hbar2_2m_q
- $\frac{\hbar^2}{2m}$ with isospin dependence, as required by BSk forces
- logical j2_terms
- Whether functional uses J^2 terms.*
- real(kind=dp) b1
- real(kind=dp) b2
- real(kind=dp) b3
- real(kind=dp) b4
- real(kind=dp) b5
- real(kind=dp) b6
- real(kind=dp) b7
- real(kind=dp) b8
- real(kind=dp) b9
- real(kind=dp) b10
- real(kind=dp) b11
- real(kind=dp) b12
- real(kind=dp) b13
- real(kind=dp) alpha
- real(kind=dp) beta
- real(kind=dp) gamma
- real(kind=dp) t4
- real(kind=dp) x4
- real(kind=dp) t5
- real(kind=dp) x5
- real(kind=dp) fn_pos
- real(kind=dp) fn_neg
- real(kind=dp) fp_pos
- real(kind=dp) fp_neg
- real(kind=dp) epsilon_lambda
- real(kind=dp), dimension(:), allocatable r
- Mesh for r.*
- real(kind=dp), dimension(:,:), allocatable rho_q
- real(kind=dp), dimension(:,:), allocatable del_rho_q
- real(kind=dp), dimension(:,:), allocatable del2_rho_q
- real(kind=dp), dimension(:), allocatable rho_t
- real(kind=dp), dimension(:), allocatable del_rho_t
- real(kind=dp), dimension(:), allocatable del2_rho_t

- `real(kind=dp), dimension(:, :), allocatable f_q`
- `real(kind=dp), dimension(:, :), allocatable del_f_q`
- `real(kind=dp), dimension(:, :), allocatable del2_f_q`
- `real(kind=dp), dimension(:, :), allocatable w_q`
- `real(kind=dp), dimension(:, :), allocatable div_w_q`
- `real(kind=dp), dimension(:, :), allocatable u_q`
- `real(kind=dp), dimension(:, :), allocatable j_2_q`
- `real(kind=dp), dimension(:, :), allocatable j_q`
- `real(kind=dp), dimension(:, :), allocatable del_j_q`
- `real(kind=dp), dimension(:, :), allocatable div_j_2_q`
- `real(kind=dp), dimension(:, :), allocatable div_j_q`
- `real(kind=dp), dimension(:, :), allocatable tau_tf_q`
- `real(kind=dp), dimension(:, :), allocatable tau_2_l_q`
- `real(kind=dp), dimension(:, :), allocatable tau_2_nl_q`
- `real(kind=dp), dimension(:, :), allocatable tau_etf_q`
- `real(kind=dp), dimension(:), allocatable j_t`
- `real(kind=dp), dimension(:), allocatable del_j_t`
- `real(kind=dp), dimension(:), allocatable div_j_t`
- `real(kind=dp), dimension(:), allocatable tau_etf_t`
- `real(kind=dp), dimension(:), allocatable e_density_field`
- `real(kind=dp), dimension(:), allocatable e_density_sky`
- `real(kind=dp), dimension(:), allocatable v_c_di`
- `real(kind=dp), dimension(:), allocatable v_c_ex`
- `real(kind=dp), dimension(:), allocatable e_density_c_di`
- `real(kind=dp), dimension(:), allocatable e_density_c_ex`
- `real(kind=dp), dimension(:), allocatable v_c_pe`
- `real(kind=dp), dimension(:, :, :), allocatable tau_2_cont_q`
- `real(kind=dp), dimension(:), allocatable rho_ch`

Charge density.

- `real(kind=dp), dimension(:, :), allocatable d2_rho_q`
- `real(kind=dp), dimension(:, :), allocatable d3_rho_q`
- `real(kind=dp), dimension(:, :), allocatable d4_rho_q`
- `real(kind=dp), dimension(:, :), allocatable d2_f_q`
- `real(kind=dp), dimension(:, :), allocatable d3_f_q`
- `real(kind=dp), dimension(:, :), allocatable d4_f_q`
- `real(kind=dp), dimension(:, :), allocatable a_q`
- `real(kind=dp), dimension(:, :), allocatable d1_a_q`
- `real(kind=dp), dimension(:, :), allocatable d2_a_q`
- `real(kind=dp), dimension(:, :), allocatable d3_a_q`
- `real(kind=dp), dimension(:, :), allocatable d4_a_q`
- `real(kind=dp), dimension(:, :), allocatable tau_4_no_spin_q`
- `real(kind=dp), dimension(:, :), allocatable tau_4_so_q`
- `real(kind=dp), dimension(:, :), allocatable j_4_q`
- `real(kind=dp), dimension(:, :), allocatable div_j_4_q`
- `real(kind=dp), dimension(:, :, :), allocatable tau_4_cont_q`
- `real(kind=dp), dimension(0:1) n_q`

Number of neutrons and protons.

- `real(kind=dp) n_t`

Total number of particles.

- `real(kind=dp) e_field`

Field energy.

- `real(kind=dp) e_skyrme`

Skyrme energy.

- `real(kind=dp), dimension(0:1) e_kinetic_q`

- *Kinetic energy for neutron and protons.*
real(kind=dp) [e_kinetic_t](#)
- *Total kinetic energy.*
real(kind=dp) [e_so](#)
- *Spin-orbit energy.*
real(kind=dp) [e_coulomb_di](#)
- *Direct Coulomb energy.*
real(kind=dp) [e_coulomb_ex](#)
- *Exchange Coulomb energy.*
real(kind=dp) [e_coulomb](#)
- *Total Coulomb energy.*
real(kind=dp) [e_total](#)
- *Total energy.*
real(kind=dp), dimension(0:1) [mu_q](#)
- *Neutron and proton chemical potential.*
real(kind=dp) [mu_e](#)
- *Electron chemical potential.*
real(kind=dp) [mu_c](#)
- *Coulomb interaction contribution to electron chemical potential.*
real(kind=dp) [delta_mu](#)
- *Overall chemical potential (beta-equilibrium condition)*
real(kind=dp) [pressure_nucl](#)
- *Nuclear pressure.*
real(kind=dp) [pressure_e](#)
- *Electron pressure.*
real(kind=dp) [pressure_ex](#)
- *Coulomb exchange pressure.*
real(kind=dp) [pressure_t](#)
- *Total pressure.*
integer [run_mode](#)
- *Mode to run code in (normal, test)*
logical [verbose](#)
- *Whether to print all extra messages about run.*
real(kind=dp) [dr](#)
- *Mesh spacing of r (fm).*
logical [profile_r_max](#)
- *Whether to take "r_max" from "r_ws".*
real(kind=dp) [r_max](#)
- *Max value of r (fm).*
logical [specify_n](#)
- *Whether to specify "n" instead of "dr".*
integer [n](#)
- *Number of mesh points.*
integer [force](#)
- *Force to use.*
integer [etf_order](#)
- *Order at which to calculate kinetic energy densities.*
logical [coulomb_on](#)
- *Use Coulomb interaction or not.*
logical [electrons_on](#)
- *Add electrons.*

- integer `nmaxstate`
Number of states that can be stored.
- integer `lmax`
Maximum angular momentum of states to find.
- logical `calc_chem_pots`
Calculate chemical potentials.
- logical, dimension(0:1) `strutinsky_on`
Use Strutinsky Integral (SI) correction for neutrons and protons.
- real(kind=dp) `strut_r_max`
Max value of r to use for box for single particle states ($f.m$).
- logical `emax0_mod`
Whether to modify "emax0" to always = 0.
- integer `neutron_pairing`
Type of pairing calculation to perform for neutrons.
- logical `proton_bcs`
Whether to do BCS for protons.
- real(kind=dp) `pair_qp_cut`
Cut-off for quasiparticle energy for solving gap equation.
- logical `smooth_qp_cut`
Whether to use smooth cut-off on quasiparticle energy.
- real(kind=dp) `pair_k_max`
Maximum momentum for integral in gap equation.
- real(kind=dp) `pair_v0`
Interaction strength parameter.
- real(kind=dp) `pair_eta`
Interaction parameter eta.
- real(kind=dp) `pair_alpha`
Interaction parameter alpha.
- real(kind=dp) `pair_tol`
Gap equation self-consistency tolerance.
- real(kind=dp) `pair_mix`
Mix of previous iteration in gap equation self-consistency loop.
- real(kind=dp) `pair_dk`
Step size in integral in gap equation.
- real(kind=dp) `pair_del_init`
Initial guess for pairing gap.
- integer `pair_max_iters`
Maximum iterations for gap equation self-consistency loop.
- real(kind=dp) `pair_num_tol`
BCS number equation self-consistency tolerance.
- real(kind=dp), dimension(5) `n_profile`
- real(kind=dp), dimension(5) `p_profile`
- real(kind=dp) `num_n`
- integer `num_p`
- real(kind=dp) `r_ws`
- character(18), dimension(10) `n_floats` = `('/(1(es24.16e3,1x))','(2(es24.16e3,1x))','(3(es24.16e3,1x))',`
`'(4(es24.16e3,1x))','(5(es24.16e3,1x))','(6(es24.16e3,1x))','(7(es24.16e3,1x))', '(8(es24.16e3,1x))','(`
`9(es24.16e3,1x))','(10(es24.16e3,1x))')/)`
- character(27) `info_format` = `'(a27,1x,a1,1x,es24.16e3)'`
- character(8) `force_string`
- character(80), parameter `line_break` = `'#####'`

Separating output.

- character(62), parameter `table_string` = '-----+-----'
- character(1), dimension(0:1), parameter `iso_string` = ('n','p')

Isospin labels.

- character(4), parameter `space4` = ''

Whitespace of length 4.

- integer `strut_n`
- real(kind=`dp`), dimension(:,:), allocatable `dhmen`
- real(kind=`dp`), dimension(:,:), allocatable `d2hmen`
- real(kind=`dp`), dimension(:,:), allocatable `hb2m`
- real(kind=`dp`), dimension(:,:), allocatable `vpot`
- real(kind=`dp`), dimension(:,:), allocatable `vso`
- real(kind=`dp`), dimension(0:1) `ecut`

Cutoff energies for neutrons and protons.

- real(kind=`dp`), dimension(:,:), allocatable `ps`

Storage for single particle wavefunctions.

- integer `nnst`

Number of states found within cutoff energy.

- integer, dimension(:), allocatable `jip`

Storage for J of states found.

- integer, dimension(:), allocatable `lp`

Storage for L of states found.

- real(kind=`dp`), dimension(:), allocatable `ep`

Storage for energy of states found.

- real(kind=`dp`), dimension(0:1) `e_sc_q`

Storage for shell correction energies.

- real(kind=`dp`) `e_sc_t`

Storage for total shell correction energy.

- real(kind=`dp`), dimension(:), allocatable `pair_gap_n`

Neutron pairing gap.

- real(kind=`dp`), dimension(:), allocatable `delta_n`

Neutron pairing field.

- real(kind=`dp`), dimension(:), allocatable `pair_gap_p`

Proton pairing gap.

- real(kind=`dp`), dimension(:), allocatable `delta_p`

Proton pairing field.

- real(kind=`dp`), dimension(:), allocatable `rho_p_bcs`

Proton density (from wavefunctions)

- real(kind=`dp`), dimension(:), allocatable `rho_anom_p_bcs`

Anomalous density.

- real(kind=`dp`), dimension(:), allocatable `strength_bcs`

Interaction strength for protons.

- real(kind=`dp`), dimension(:), allocatable `occ_v`

Particle occupation probabilities.

- real(kind=`dp`), dimension(:), allocatable `occ_u`

Hole occupation probabilities.

- real(kind=`dp`), dimension(:), allocatable `e_qp_p`

Quasiparticle energies.

- real(kind=`dp`) `num_p_bcs`

Number of protons calculated from occupations.

- real(kind=`dp`) `e_pair_bcs`

- *BCS pairing energy.*
real(kind=dp), dimension(0:1) `e_pair_q`
- *Storage for pairing condensation energies.*
real(kind=dp) `e_pair_t`
- *Storage for total pairing condensation energy.*
real(kind=dp) `rho_e`
- *Electron density.*
real(kind=dp) `e_kinetic_e`
- *Total electron kinetic energy.*
real(kind=dp) `e_coulomb_e`
- *Total electron-electron potential energy from Coulomb interaction.*
real(kind=dp) `e_coulomb_pe`
- *Total proton-electron potential energy from Coulomb interaction.*

5.2.1 Detailed Description

Module to hold global parameters and variables.

Author

M. Shelley

5.2.2 Function/Subroutine Documentation

5.2.2.1 `force_initialise()`

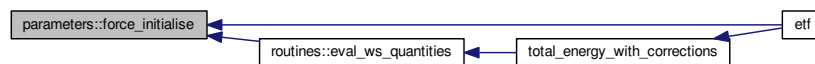
```
subroutine parameters::force_initialise ( )
```

Subroutine to initialise Skyrme force parameters with values for a given force specified in 'input.dat'.

Author

M. Shelley

Here is the caller graph for this function:



5.2.3 Variable Documentation

5.2.3.1 a_q

```
real(kind=dp), dimension(:, :), allocatable parameters::a_q
```

5.2.3.2 d1_a_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d1_a_q
```

5.2.3.3 d2_a_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d2_a_q
```

5.2.3.4 d2_f_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d2_f_q
```

5.2.3.5 d2_rho_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d2_rho_q
```

5.2.3.6 d3_a_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d3_a_q
```

5.2.3.7 d3_f_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d3_f_q
```

5.2.3.8 d3_rho_q

```
real(kind=dp), dimension(:, :), allocatable parameters::d3_rho_q
```

5.2.3.9 d4_a_q

```
real(kind=dp), dimension(:,:), allocatable parameters::d4_a_q
```

5.2.3.10 d4_f_q

```
real(kind=dp), dimension(:,:), allocatable parameters::d4_f_q
```

5.2.3.11 d4_rho_q

```
real(kind=dp), dimension(:,:), allocatable parameters::d4_rho_q
```

5.2.3.12 div_j_4_q

```
real(kind=dp), dimension(:,:), allocatable parameters::div_j_4_q
```

5.2.3.13 dp

```
integer, parameter parameters::dp = selected_real_kind(15, 300)
```

Precision.

5.2.3.14 j_4_q

```
real(kind=dp), dimension(:,:), allocatable parameters::j_4_q
```

5.2.3.15 tau_4_cont_q

```
real(kind=dp), dimension(:,:,:), allocatable parameters::tau_4_cont_q
```

5.2.3.16 tau_4_no_spin_q

```
real(kind=dp), dimension(:, :), allocatable parameters::tau_4_no_spin_q
```

5.2.3.17 tau_4_so_q

```
real(kind=dp), dimension(:, :), allocatable parameters::tau_4_so_q
```

5.3 routines Module Reference

Module to hold main routines: densities, energies.

Functions/Subroutines

- integer function [str2int](#) (string)
- subroutine [allocate_arrays](#) ()

Subroutine to allocate arrays for all densities and effective masses, and for their derivatives.
- subroutine [initialise_uniform_mesh](#) ()

Subroutine to calculate number of mesh points for a uniform grid, allocate array for r, and then populate r with the mesh points.
- subroutine [deallocate_mesh](#) ()

Subroutine to deallocate array for r.
- subroutine [deallocate_arrays](#) ()

Subroutine to deallocate arrays for all densities and effective masses, and for their derivatives.
- real(kind=dp) function [calc_rho_q](#) (rhoqgas, rhoqliq, rq, aq, gammaq, r)

Function to calculate the neutron or proton density at a given radius r, using supplied input parameters.
- subroutine [calc_f_q](#) (rhot, rhoq, q, fq)

Subroutine to calculate the effective mass ratio $f_q = \frac{m}{m_q^}$ for neutrons or protons, for standard Skyrme or for BSk forces.*
- subroutine [calc_w_q](#) (delrho, delrhoq, Wq)

Function to calculate the spin-orbit field W_q for neutrons or protons.
- subroutine [calc_tau_tf](#) (rhoq, tauTF)

Function to calculate the zeroth-order contribution to the kinetic energy density for neutrons or protons.
- subroutine [calc_tau_2_l](#) (rhoq, delrhoq, del2rhoq, tau2Lq, t2contq)

Function to calculate the local second-order contribution to the kinetic energy density for neutrons or protons.
- subroutine [calc_tau_2_nl](#) (rhoq, delrhoq, fq, delfq, del2fq, Wq, q, tau2NLq, t2contq)

Subroutine to calculate the non-local second-order contribution to the kinetic energy density for neutrons or protons.
- subroutine [calc_tau_4_no_spin](#) (rhoq, d1rhoq, d2rhoq, d3rhoq, d4rhoq, fq, d1fq, d2fq, d3fq, d4fq, tau4nospinq, t4contq)

Subroutine to calculate the fourth-order contributions to the kinetic energy density (without spin-orbit contributions) τ_{q_i} for neutrons or protons.
- subroutine [calc_tau_4_so](#) (rhoq, d1rhoq, fq, d1fq, d2fq, d1Aq, d2Aq, d3Aq, q, tau_4_so_q, t4contq)

Subroutine to calculate the fourth-order contributions to the kinetic energy density (spin-orbit contributions) τ_{q_i} for neutrons or protons.
- subroutine [calc_j_2_q](#) (rhoq, Wq, fq, q, J2q)

Subroutine to calculate the second-order contributions to the spin current density J_q for neutrons or protons.

- subroutine `calc_j_4_q` (rhoq, d1rhoq, fq, d1fq, d2fq, d1Aq, d2Aq, d3Aq, q, J4q)
Subroutine to calculate the fourth-order contributions to the spin current density \mathbf{J}_q for neutrons or protons.
- subroutine `calc_div_j_2_q` (rhoq, d1rhoq, fq, d1fq, d1Aq, d2Aq, q, divJ2q)
Subroutine to calculate the second-order contributions to the divergence of the current density \mathbf{J}_q for neutrons or protons.
- subroutine `calc_div_j_4_q` (rhoq, d1rhoq, d2rhoq, fq, d1fq, d2fq, d3fq, d1Aq, d2Aq, d3Aq, d4Aq, q, divJ4q)
Subroutine to calculate the fourth-order contributions to the divergence of the current density \mathbf{J}_q for neutrons or protons.
- real(kind=dp) function `calc_u_q` (rhot, rhon, rhop, delrhot, delrhon, delrhon, del2rhot, del2rhon, del2rhon, taut, taun, taup, divJt, divJq, q)
Function to calculate the central potential U_q for neutrons or protons, for standard Skyrme or for BSk forces with extra terms.
- subroutine `eval_fields` ()
Subroutine to evaluate matter density derivatives, and all fields: effective masses (and derivatives), spin-orbit fields, spin current densities (and derivatives)
- subroutine `eval_tau_etc` ()
Subroutine to calculate the kinetic energy density τ_{ETF} with the (extended) Thomas-Fermi approximation at order specified by "etc_order".
- subroutine `charge` (rho, Neutr, Nprot, Ngrid1, del1, rhoch)
Subroutine to calculate the charge density from the matter densities. Modified by M. Shelley.
- subroutine `eval_coulomb` (prot_dens, calc_exchange)
Subroutine to evaluate the Coulomb potentials and energy densities, using either the proton or charge density.
- subroutine `eval_electrons` ()
Subroutine to evaluate the proton-electron potential which contributes to U_q , all energy contributions, and the pressure, coming from a homogeneous electron gas. Also calculates the electron chemical potential.
- subroutine `eval_u_q` ()
Subroutine to evaluate the central fields U_q .
- subroutine `eval_skyrme_energy_density` ()
Subroutine to evaluate the Skyrme energy density, after first separately evaluating the field energy density.
- subroutine `pressure` ()
Subroutine to calculate the nuclear contribution to the pressure, and then the total pressure.
- subroutine `calc_particle_number` ()
Subroutine to calculate the number of particles in the Wigner-Seitz cell.
- subroutine `eval_ws_quantities` ()
Subroutine to evaluate the various densities, fields, derivatives, energies, and particle numbers in the WS cell with densities ρ_{hoq} .
- real(kind=dp) function `calc_e_f_q` (rhoq, fq, q)
Subroutine to evaluate the Fermi energy at a given density and effective mass, for neutrons or protons.
- subroutine `d1` (n, h, f, df)
This subroutine computes the first derivative of function evaluated on the meshpoints 1,...,npt. The input is the function f with extrapolated values in -1, 0. Modified by M. Shelley.
- subroutine `d2` (n, h, f, d2f)
This subroutine computes the second derivative of function evaluated on the meshpoints 1,...,npt. The input is the function f with extrapolated values in -1, 0. Modified by M. Shelley.
- subroutine `lap_sphe_symm` (f, df_dr, d2f_dr2)
Subroutine to carry out Laplacian in spherical symmetry.
- subroutine `d3` (f, d3f_dr3)
Subroutine to calculate third derivative of function f evaluated on evenly-spaced meshpoints from 1 to n. Works with extrapolated values in -2,-1,0. Uses 7-point stencil.
- subroutine `d4` (f, d4f_dr4)
Subroutine to calculate fourth derivative of function f evaluated on evenly-spaced meshpoints from 1 to n. Works with extrapolated values in -2,-1,0. Uses 7-point stencil.
- subroutine `extrapolate_back_3` (grid)

Subroutine to extrapolate back 3 points on the r mesh, to facilitate the calculation of first and second derivatives using a 5-point stencil. Values in grid start at 4, extrapolated values are put in elements 3,2,1. Coefficients come from solving 4th-order polynomial, assuming that $f'(0) = 0$, and $f(-h) = f(h)$.

- real(kind=dp) function [ws_integral](#) (quantity, n_max)

Function to integrate a density or field over the whole W-S cell, using Simpson's rule (+ Simpson's 3/8 rule if odd number of points)

- subroutine [write_densities](#) ()

Subroutine to write neutron and proton densities to files.

- subroutine [write_fields](#) ()

Subroutine to write neutron and proton fields to files.

- subroutine [write_sp_states_p](#) ()

Subroutine to write proton single particle energies to file, and extra details if BCS has been performed.

Variables

- integer [file_unit_0](#)
- integer [file_unit_1](#)

Unit numbers for opening files.

5.3.1 Detailed Description

Module to hold main routines: densities, energies.

Author

M. Shelley

5.3.2 Function/Subroutine Documentation

5.3.2.1 [allocate_arrays](#)()

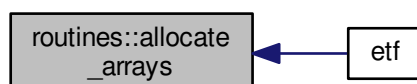
```
subroutine routines::allocate_arrays ( )
```

Subroutine to allocate arrays for all densities and effective masses, and for their derivatives.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.2 calc_div_j_2_q()

```

subroutine routines::calc_div_j_2_q (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) d1rhoq,
    real(kind=dp), dimension(1:n), intent(in) fq,
    real(kind=dp), dimension(1:n), intent(in) d1fq,
    real(kind=dp), dimension(1:n), intent(in) d1Aq,
    real(kind=dp), dimension(1:n), intent(in) d2Aq,
    integer, intent(in) q,
    real(kind=dp), dimension(1:n), intent(inout) divJ2q )

```

Subroutine to calculate the second-order contributions to the divergence of the current density \mathbf{J}_q for neutrons or protons.

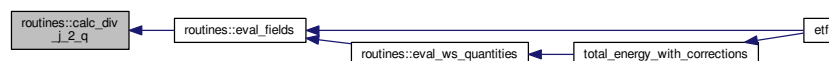
Author

M. Shelley

Parameters

| | | |
|---------|---------------|--|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>d1rhoq</i> | First derivative of neutron or proton density |
| in | <i>fq</i> | Effective mass for protons or neutrons |
| in | <i>d1fq</i> | First derivative of effective mass for neutrons or protons |
| in | <i>d1Aq</i> | First derivative of "composite" neutron or proton density A_q |
| in | <i>d2Aq</i> | Second derivative of "composite" neutron or proton density A_q |
| in | <i>q</i> | Isospin |
| in, out | <i>divJ2q</i> | Array for second-order contributions |

Here is the caller graph for this function:



5.3.2.3 calc_div_j_4_q()

```

subroutine routines::calc_div_j_4_q (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) d1rhoq,
    real(kind=dp), dimension(1:n), intent(in) d2rhoq,
    real(kind=dp), dimension(1:n), intent(in) fq,
    real(kind=dp), dimension(1:n), intent(in) d1fq,
    real(kind=dp), dimension(1:n), intent(in) d2fq,
    real(kind=dp), dimension(1:n), intent(in) d3fq,

```

```

real(kind=dp), dimension(1:n), intent(in) d1Aq,
real(kind=dp), dimension(1:n), intent(in) d2Aq,
real(kind=dp), dimension(1:n), intent(in) d3Aq,
real(kind=dp), dimension(1:n), intent(in) d4Aq,
integer, intent(in) q,
real(kind=dp), dimension(1:n), intent(inout) divJ4q )

```

Subroutine to calculate the fourth-order contributions to the divergence of the current density \mathbf{J}_q for neutrons or protons.

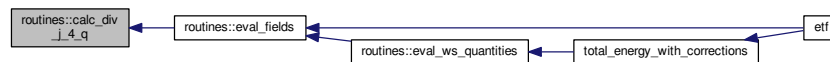
Author

M. Shelley

Parameters

| | | |
|---------|---------------|--|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>d1rhoq</i> | First derivative of neutron or proton density |
| in | <i>d2rhoq</i> | Second derivative of neutron or proton density |
| in | <i>fq</i> | Effective mass for protons or neutrons |
| in | <i>d1fq</i> | First derivative of effective mass for neutrons or protons |
| in | <i>d2fq</i> | Second derivative of effective mass for neutrons or protons |
| in | <i>d3fq</i> | Third derivative of effective mass for neutrons or protons |
| in | <i>d1Aq</i> | First derivative of "composite" neutron or proton density A_q |
| in | <i>d2Aq</i> | Second derivative of "composite" neutron or proton density A_q |
| in | <i>d3Aq</i> | Third derivative of "composite" neutron or proton density A_q |
| in | <i>d4Aq</i> | Fourth derivative of "composite" neutron or proton density A_q |
| in | <i>q</i> | Isospin |
| in, out | <i>divJ4q</i> | Array for fourth-order contributions |

Here is the caller graph for this function:



5.3.2.4 calc_e_f_q()

```

real(kind=dp) function routines::calc_e_f_q (
    real(kind=dp), intent(in) rhoq,
    real(kind=dp), intent(in) fq,
    integer, intent(in) q )

```

Subroutine to evaluate the Fermi energy at a given density and effective mass, for neutrons or protons.

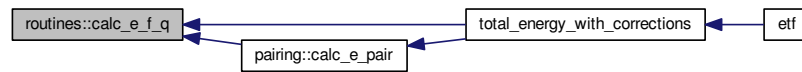
Author

M. Shelley

Parameters

| | | |
|----|-------------|----------------------------------|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>fn</i> | Neutron or proton effective mass |

Here is the caller graph for this function:

5.3.2.5 `calc_f_q()`

```

subroutine routines::calc_f_q (
    real(kind=dp), dimension(:), intent(in) rhot,
    real(kind=dp), dimension(:), intent(in) rhoq,
    integer, intent(in) q,
    real(kind=dp), dimension(:), intent(inout) fq )
  
```

Subroutine to calculate the effective mass ratio $f_q = \frac{m}{m_q^*}$ for neutrons or protons, for standard Skyrme or for BSk forces.

Author

M. Shelley

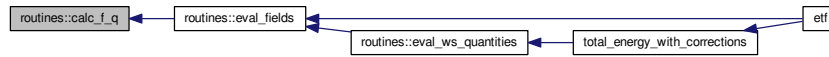
Parameters

| | | |
|---------|-------------|----------------------------------|
| in | <i>rhot</i> | Total density |
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>q</i> | Isospin |
| in, out | <i>fq</i> | Neutron or proton effective mass |

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.6 calc_j_2_q()

```

subroutine routines::calc_j_2_q (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) Wq,
    real(kind=dp), dimension(1:n), intent(in) fq,
    integer, intent(in) q,
    real(kind=dp), dimension(1:n), intent(inout) J2q )
  
```

Subroutine to calculate the second-order contributions to the spin current density \mathbf{J}_q for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|----|-------------|---|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>Wq</i> | Spin-orbit field |
| in | <i>fq</i> | Effective mass for neutrons or protons |
| in | <i>q</i> | Isospin [in,out] J2q Array for second-order contributions |

Here is the caller graph for this function:



5.3.2.7 calc_j_4_q()

```

subroutine routines::calc_j_4_q (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) dlrhoq,
    real(kind=dp), dimension(1:n), intent(in) fq,
  
```

```

real(kind=dp), dimension(1:n), intent(in) d1fq,
real(kind=dp), dimension(1:n), intent(in) d2fq,
real(kind=dp), dimension(1:n), intent(in) d1Aq,
real(kind=dp), dimension(1:n), intent(in) d2Aq,
real(kind=dp), dimension(1:n), intent(in) d3Aq,
integer, intent(in) q,
real(kind=dp), dimension(1:n), intent(inout) J4q )

```

Subroutine to calculate the fourth-order contributions to the spin current density \mathbf{J}_q for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|---------------|--|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>d1rhoq</i> | First derivative of neutron or proton density |
| in | <i>fq</i> | Effective mass for protons or neutrons |
| in | <i>d1fq</i> | First derivative of effective mass for neutrons or protons |
| in | <i>d2fq</i> | Second derivative of effective mass for neutrons or protons |
| in | <i>d1Aq</i> | First derivative of "composite" neutron or proton density A_q |
| in | <i>d2Aq</i> | Second derivative of "composite" neutron or proton density A_q |
| in | <i>d3Aq</i> | Third derivative of "composite" neutron or proton density A_q |
| in | <i>q</i> | Isospin |
| in, out | <i>J4q</i> | Array for fourth-order contributions |

Here is the caller graph for this function:



5.3.2.8 calc_particle_number()

```

subroutine routines::calc_particle_number ( )

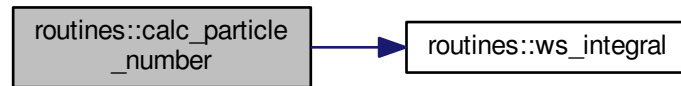
```

Subroutine to calculate the number of particles in the Wigner-Seitz cell.

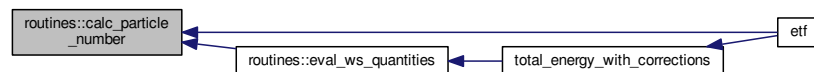
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:

**5.3.2.9 calc_rho_q()**

```

real(kind=dp) function routines::calc_rho_q (
    real(kind=dp), intent(in) rhoqgas,
    real(kind=dp), intent(in) rhoqliq,
    real(kind=dp), intent(in) rq,
    real(kind=dp), intent(in) aq,
    real(kind=dp), intent(in) gammaq,
    real(kind=dp), intent(in) r )
  
```

Function to calculate the neutron or proton density at a given radius r , using supplied input parameters.

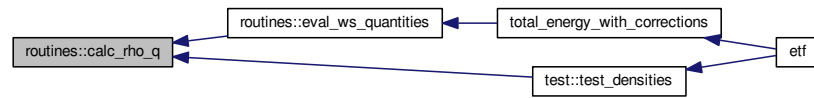
Author

M. Shelley

Parameters

| | | |
|----|----------------|--|
| in | <i>rhoqgas</i> | Asymptotic density in gas far from surface |
| in | <i>rhoqliq</i> | Asymptotic density in cluster far from surface |
| in | <i>rq</i> | Cluster radius |
| in | <i>aq</i> | Surface diffuseness |
| in | <i>gammaq</i> | Parameter allowing for "asymmetric surface" |
| in | <i>r</i> | Radius |

Here is the caller graph for this function:



5.3.2.10 calc_tau_2_l()

```

subroutine routines::calc_tau_2_l (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) delrhoq,
    real(kind=dp), dimension(1:n), intent(in) del2rhoq,
    real(kind=dp), dimension(1:n), intent(inout) tau2Lq,
    real(kind=dp), dimension(1:n), intent(inout) t2contq )
  
```

Function to calculate the local second-order contribution to the kinetic energy density for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|-----------------|---|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>delrhoq</i> | Gradient of proton or neutron density |
| in | <i>del2rhoq</i> | Laplacian of proton or neutron density |
| in, out | <i>tau2Lq</i> | Array for local second-order contributions |
| in, out | <i>t2contq</i> | Array for individual second-order contributions |

Here is the caller graph for this function:



5.3.2.11 calc_tau_2_nl()

```

subroutine routines::calc_tau_2_nl (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
  
```

```

real(kind=dp), dimension(1:n), intent(in) delrhoq,
real(kind=dp), dimension(1:n), intent(in) fq,
real(kind=dp), dimension(1:n), intent(in) delfq,
real(kind=dp), dimension(1:n), intent(in) del2fq,
real(kind=dp), dimension(1:n), intent(in) Wq,
integer, intent(in) q,
real(kind=dp), dimension(1:n), intent(inout) tau2NLq,
real(kind=dp), dimension(1:n,2:4), intent(inout) t2contq )

```

Subroutine to calculate the non-local second-order contribution to the kinetic energy density for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|----------------|---|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>delrhoq</i> | Gradient of neutron or proton density |
| in | <i>fq</i> | Effective mass for neutrons or protons |
| in | <i>delfq</i> | Gradient of effective mass for neutrons or protons |
| in | <i>del2fq</i> | Laplacian of effective mass for neutrons or protons |
| in | <i>Wq</i> | Spin-orbit field |
| in | <i>q</i> | Isospin |
| in, out | <i>tau2NLq</i> | Array for non-local second-order contributions |
| in, out | <i>t2contq</i> | Array for individual second-order contributions |

Here is the caller graph for this function:



5.3.2.12 calc_tau_4_no_spin()

```

subroutine routines::calc_tau_4_no_spin (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) d1rhoq,
    real(kind=dp), dimension(1:n), intent(in) d2rhoq,
    real(kind=dp), dimension(1:n), intent(in) d3rhoq,
    real(kind=dp), dimension(1:n), intent(in) d4rhoq,
    real(kind=dp), dimension(1:n), intent(in) fq,
    real(kind=dp), dimension(1:n), intent(in) d1fq,
    real(kind=dp), dimension(1:n), intent(in) d2fq,
    real(kind=dp), dimension(1:n), intent(in) d3fq,
    real(kind=dp), dimension(1:n), intent(in) d4fq,

```

```

real(kind=dp), dimension(1:n), intent(inout) tau4nospiq,
real(kind=dp), dimension(1:n,1:3), intent(inout) t4contq )

```

Subroutine to calculate the fourth-order contributions to the kinetic energy density (without spin-orbit contributions) τ_q for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|-------------------|---|
| in | <i>rhoq</i> | Neutron or proton density |
| in | <i>d1rhoq</i> | First derivative of neutron or proton density |
| in | <i>d2rhoq</i> | Second derivative of neutron or proton density |
| in | <i>d3rhoq</i> | Third derivative of neutron or proton density |
| in | <i>d4rhoq</i> | Fourth derivative of neutron or proton density |
| in | <i>fq</i> | Effective mass for neutrons or protons |
| in | <i>d1fq</i> | First derivative of effective mass for neutrons or protons |
| in | <i>d2fq</i> | Second derivative of effective mass for neutrons or protons |
| in | <i>d3fq</i> | Third derivative of effective mass for neutrons or protons |
| in | <i>d4fq</i> | Fourth derivative of effective mass for neutrons or protons |
| in, out | <i>tau4nospiq</i> | Array for fourth-order contributions (no spin-orbit) |
| in, out | <i>t4contq</i> | Array for individual fourth-order contributions |

Here is the caller graph for this function:



5.3.2.13 calc_tau_4_so()

```

subroutine routines::calc_tau_4_so (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(in) d1rhoq,
    real(kind=dp), dimension(1:n), intent(in) fq,
    real(kind=dp), dimension(1:n), intent(in) d1fq,
    real(kind=dp), dimension(1:n), intent(in) d2fq,
    real(kind=dp), dimension(1:n), intent(in) d1Aq,
    real(kind=dp), dimension(1:n), intent(in) d2Aq,
    real(kind=dp), dimension(1:n), intent(in) d3Aq,
    integer, intent(in) q,
    real(kind=dp), dimension(1:n), intent(inout) tau_4_so_q,
    real(kind=dp), dimension(1:n), intent(inout) t4contq )

```

Subroutine to calculate the fourth-order contributions to the kinetic energy density (spin-orbit contributions) τ_q for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|-------------------|--|
| in | ρq | Neutron or proton density |
| in | $d1\rho q$ | First derivative of neutron or proton density |
| in | $f q$ | Effective mass for protons or neutrons |
| in | $d1f q$ | First derivative of effective mass for neutrons or protons |
| in | $d2f q$ | Second derivative of effective mass for neutrons or protons |
| in | $d1A q$ | First derivative of "composite" neutron or proton density A_q |
| in | $d2A q$ | Second derivative of "composite" neutron or proton density A_q |
| in | $d3A q$ | Third derivative of "composite" neutron or proton density A_q |
| in | q | Isospin |
| in, out | $\tau_{4_so_q}$ | Array for fourth-order contributions (spin-orbit) |
| in, out | $t4contq$ | Array for individual fourth-order contributions |

Here is the caller graph for this function:

**5.3.2.14 calc_tau_tf()**

```

subroutine routines::calc_tau_tf (
    real(kind=dp), dimension(1:n), intent(in) rhoq,
    real(kind=dp), dimension(1:n), intent(inout) tauTF )

```

Function to calculate the zeroth-order contribution to the kinetic energy density for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|-----------|--------------------------------------|
| in | ρq | Neutron or proton density |
| in, out | τTF | Array for zeroth-order contributions |

Here is the caller graph for this function:



5.3.2.15 calc_u_q()

```

real(kind=dp) function routines::calc_u_q (
    real(kind=dp), intent(in) rhot,
    real(kind=dp), intent(in) rhon,
    real(kind=dp), intent(in) rhop,
    real(kind=dp), intent(in) delrhot,
    real(kind=dp), intent(in) delrhon,
    real(kind=dp), intent(in) delrhop,
    real(kind=dp), intent(in) del2rhot,
    real(kind=dp), intent(in) del2rhon,
    real(kind=dp), intent(in) del2rhop,
    real(kind=dp), intent(in) taut,
    real(kind=dp), intent(in) taun,
    real(kind=dp), intent(in) taup,
    real(kind=dp), intent(in) divJt,
    real(kind=dp), intent(in) divJq,
    integer, intent(in) q )
  
```

Function to calculate the central potential U_q for neutrons or protons, for standard Skyrme or for BSk forces with extra terms.

Author

M. Shelley

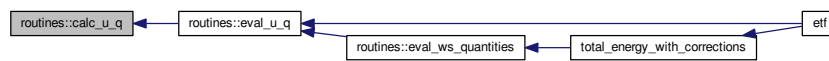
Parameters

| | | |
|----|-----------------|--|
| in | <i>rhot</i> | Total density |
| in | <i>rhon</i> | Neutron density |
| in | <i>rhop</i> | Proton density |
| in | <i>delrhot</i> | First derivative of total density |
| in | <i>delrhon</i> | First derivative of neutron density |
| in | <i>delrhop</i> | First derivative of proton density |
| in | <i>del2rhot</i> | Laplacian of total density |
| in | <i>del2rhon</i> | Laplacian of neutron density |
| in | <i>del2rhop</i> | Laplacian of proton density |
| in | <i>taut</i> | Total kinetic density |
| in | <i>taun</i> | Neutron kinetic density |
| in | <i>taup</i> | Proton kinetic density |
| in | <i>divJt</i> | Divergence of total spin current density |
| in | <i>divJq</i> | Divergence of proton or neutron spin current density |
| in | <i>q</i> | Isospin |

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.16 calc_w_q()

```

subroutine routines::calc_w_q (
    real(kind=dp), dimension(1:n), intent(in) delrho,
    real(kind=dp), dimension(1:n), intent(in) delrhoq,
    real(kind=dp), dimension(1:n), intent(inout) Wq )
  
```

Function to calculate the spin-orbit field \mathbf{W}_q for neutrons or protons.

Author

M. Shelley

Parameters

| | | |
|---------|----------------|---------------------------------------|
| in | <i>delrho</i> | Gradient of total density |
| in | <i>delrhoq</i> | Gradient of neutron or proton density |
| in, out | <i>Wq</i> | Array for spin-orbit field |

Here is the caller graph for this function:



5.3.2.17 charge()

```

subroutine routines::charge (
    real(kind=dp), dimension(ngrid1,0:1), intent(in) rho,
    real(kind=dp), intent(in) Neutr,
    real(kind=dp), intent(in) Nprot,
    integer, intent(in) Ngrid1,
    real(kind=dp), intent(in) del1,
    real(kind=dp), dimension(ngrid1), intent(inout) rhoch )

```

Subroutine to calculate the charge density from the matter densities. Modified by M. Shelley.

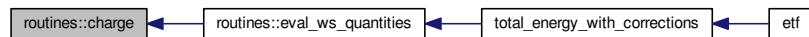
Author

A. Pastore, M. Shelley

Parameters

| | | |
|---------|---------------|------------------------------|
| in | <i>rho</i> | Neutron and proton densities |
| in | <i>Neutr</i> | Number of neutrons |
| in | <i>Nprot</i> | Number of protons |
| in | <i>Ngrid1</i> | Number of mesh points |
| in | <i>del1</i> | Step size |
| in, out | <i>rhoch</i> | Array for charge density |

Here is the caller graph for this function:



5.3.2.18 d1()

```

subroutine routines::d1 (
    integer, intent(in) n,
    real(kind=dp), intent(in) h,
    real(kind=dp), dimension(-1:n), intent(in) f,
    real(kind=dp), dimension(1:n), intent(inout) df )

```

This subroutine computes the first derivative of function evaluated on the meshpoints 1,...,npt. The input is the function f with extrapolated values in -1, 0. Modified by M. Shelley.

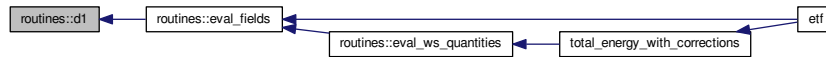
Author

A. Pastore, M. Shelley

Parameters

| | | |
|---------|------|---|
| in | n | Number of meshpoints |
| in | f | Input function (1:n) with extrapolated values in -1,0 |
| in | h | Step size |
| in, out | df | Array for first derivatives |

Here is the caller graph for this function:

**5.3.2.19 d2()**

```

subroutine routines::d2 (
    integer, intent(in) n,
    real(kind=dp), intent(in) h,
    real(kind=dp), dimension(-1:n), intent(in) f,
    real(kind=dp), dimension(1:n), intent(inout) d2f )
  
```

This subroutine computes the second derivative of function evaluated on the meshpoints 1,...,npt. The input is the function f with extrapolated values in -1, 0. Modified by M. Shelley.

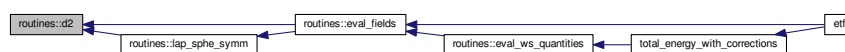
Author

A. Pastore, M. Shelley

Parameters

| | | |
|---------|-------|---|
| in | n | Number of meshpoints |
| in | f | Input function (1:n) with extrapolated values in -1,0 |
| in | h | Step size |
| in, out | $d2f$ | Array for second derivatives |

Here is the caller graph for this function:



5.3.2.20 d3()

```

subroutine routines::d3 (
    real(kind=dp), dimension(-2:n), intent(in) f,
    real(kind=dp), dimension(1:n), intent(inout) d3f_dr3 )

```

Subroutine to calculate third derivative of function f evaluated on evenly-spaced meshpoints from 1 to n . Works with extrapolated values in -2,-1,0. Uses 7-point stencil.

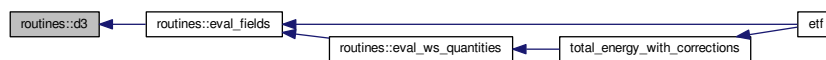
Author

M. Shelley

Parameters

| | | |
|---------|------------|--|
| in | f | Input function (1:n) with extrapolated values in -2,-1,0 |
| in, out | $d3f_dr3$ | Array for third derivative |

Here is the caller graph for this function:



5.3.2.21 d4()

```

subroutine routines::d4 (
    real(kind=dp), dimension(-2:n), intent(in) f,
    real(kind=dp), dimension(1:n), intent(inout) d4f_dr4 )

```

Subroutine to calculate fourth derivative of function f evaluated on evenly-spaced meshpoints from 1 to n . Works with extrapolated values in -2,-1,0. Uses 7-point stencil.

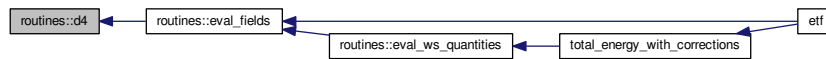
Author

M. Shelley

Parameters

| | | |
|---------|------------|--|
| in | f | Input function (1:n) with extrapolated values in -2,-1,0 |
| in, out | $d4f_dr4$ | Array for fourth derivative |

Here is the caller graph for this function:



5.3.2.22 `deallocate_arrays()`

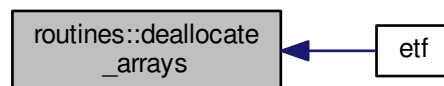
```
subroutine routines::deallocate_arrays ( )
```

Subroutine to deallocate arrays for all densities and effective masses, and for their derivatives.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.23 `deallocate_mesh()`

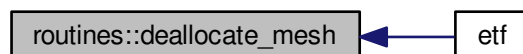
```
subroutine routines::deallocate_mesh ( )
```

Subroutine to deallocate array for `r`.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.24 eval_coulomb()

```

subroutine routines::eval_coulomb (
    real(kind=dp), dimension(1:n), intent(in) prot_dens,
    logical, intent(in) calc_exchange )

```

Subroutine to evaluate the Coulomb potentials and energy densities, using either the proton or charge density.

Author

M. Shelley

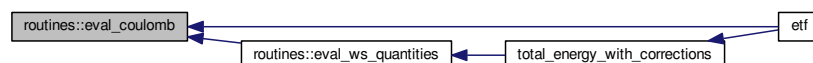
Parameters

| | | |
|----|----------------------|---|
| in | <i>prot_dens</i> | Density to use (proton or charge) |
| in | <i>calc_exchange</i> | Whether to calculate exchange potential |

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.25 eval_electrons()

```

subroutine routines::eval_electrons ( )

```

Subroutine to evaluate the proton-electron potential which contributes to U_q , all energy contributions, and the pressure, coming from a homogeneous electron gas. Also calculates the electron chemical potential.

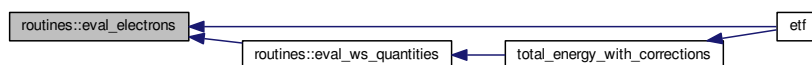
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.26 eval_fields()

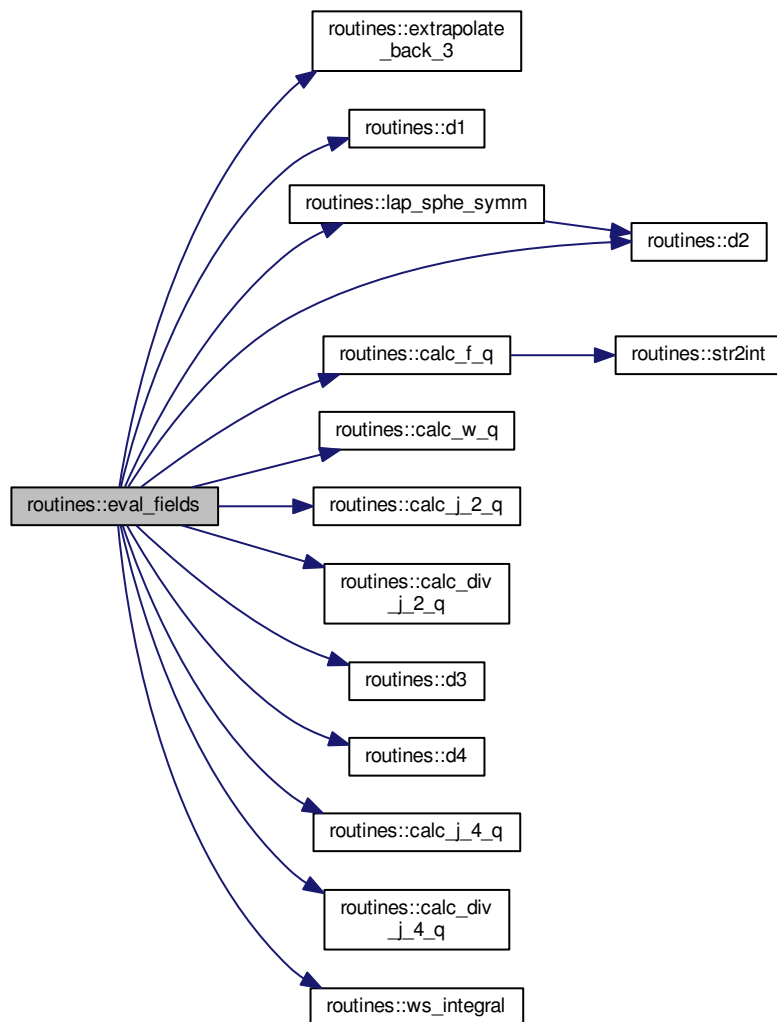
```
subroutine routines::eval_fields ( )
```

Subroutine to evaluate matter density derivatives, and all fields: effective masses (and derivatives), spin-orbit fields, spin current densities (and derivatives)

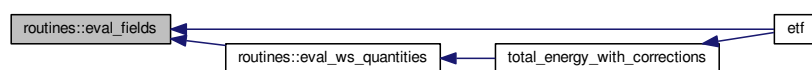
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.27 eval_skyrme_energy_density()

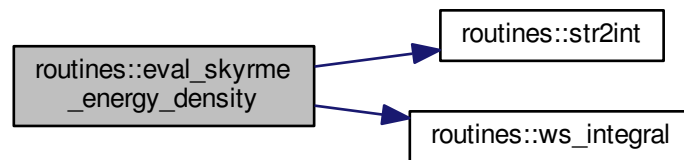
```
subroutine routines::eval_skyrme_energy_density ( )
```

Subroutine to evaluate the Skyrme energy density, after first separately evaluating the field energy density.

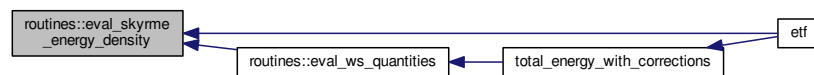
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.28 eval_tau_etf()

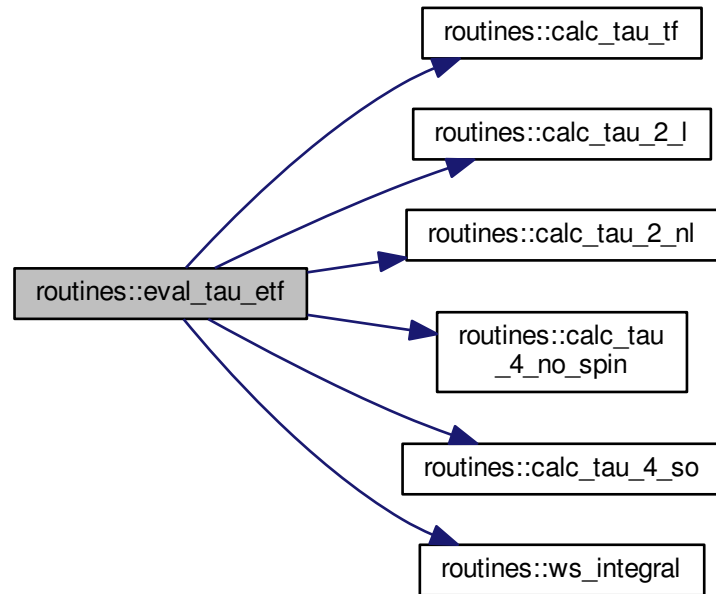
```
subroutine routines::eval_tau_etf ( )
```

Subroutine to calculate the kinetic energy density τ_{ETF} with the (extended) Thomas-Fermi approximation at order specified by "etf_order".

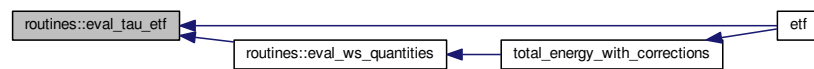
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.29 eval_u_q()

```
subroutine routines::eval_u_q ( )
```

Subroutine to evaluate the central fields U_q .

Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.30 eval_ws_quantities()

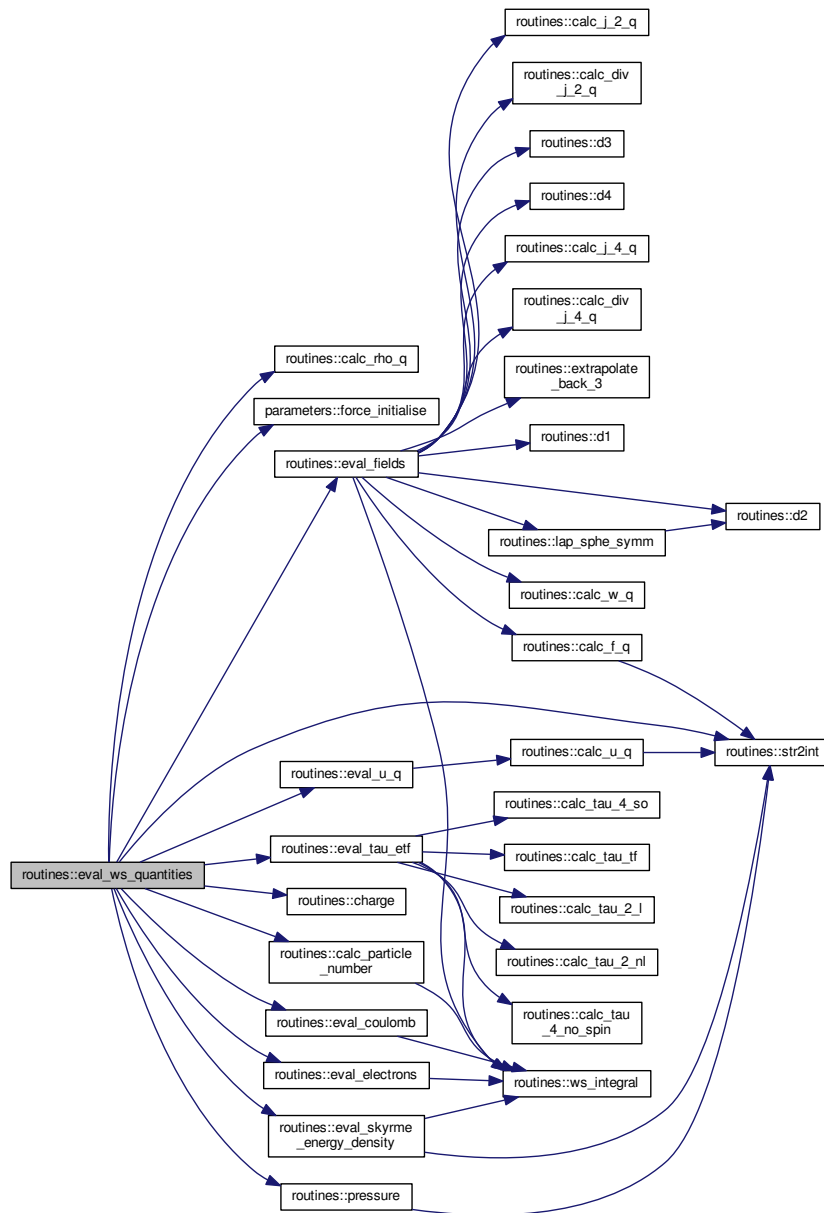
```
subroutine routines::eval_ws_quantities ( )
```

Subroutine to evaluate the various densities, fields, derivatives, energies, and particle numbers in the WS cell with densities ρ_q .

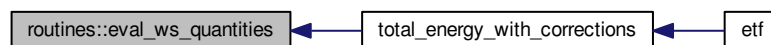
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.31 extrapolate_back_3()

```
subroutine routines::extrapolate_back_3 (
    real(kind=dp), dimension(:), intent(inout) grid )
```

Subroutine to extrapolate back 3 points on the r mesh, to facilitate the calculation of first and second derivatives using a 5-point stencil. Values in grid start at 4, extrapolated values are put in elements 3,2,1. Coefficients come from solving 4th-order polynomial, assuming that $f'(0) = 0$, and $f(-h) = f(h)$.

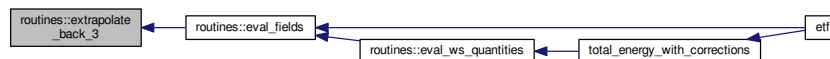
Author

M. Shelley

Parameters

| | | |
|---------|------|--|
| in, out | grid | Array of quantity evaluated on r mesh, with 3 elements before $r = dr$ |
|---------|------|--|

Here is the caller graph for this function:



5.3.2.32 initialise_uniform_mesh()

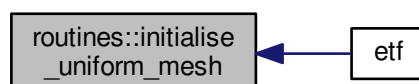
```
subroutine routines::initialise_uniform_mesh ( )
```

Subroutine to calculate number of mesh points for a uniform grid, allocate array for r, and then populate r with the mesh points.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.33 lap_sphe_symm()

```

subroutine routines::lap_sphe_symm (
    real(kind=dp), dimension(-2:n), intent(in) f,
    real(kind=dp), dimension(1:n), intent(in) df_dr,
    real(kind=dp), dimension(1:n), intent(inout) d2f_dr2 )

```

Subroutine to carry out Laplacian in spherical symmetry.

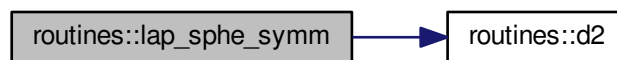
Author

M. Shelley

Parameters

| | | |
|---------|----------------|--|
| in | <i>f</i> | Input function (1:n) with extrapolated values in -2,-1,0 |
| in | <i>df_dr</i> | First derivative array |
| in, out | <i>d2f_dr2</i> | Array for Laplacian |

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.34 pressure()

```

subroutine routines::pressure ( )

```

Subroutine to calculate the nuclear contribution to the pressure, and then the total pressure.

Author

M. Shelley

Here is the call graph for this function:



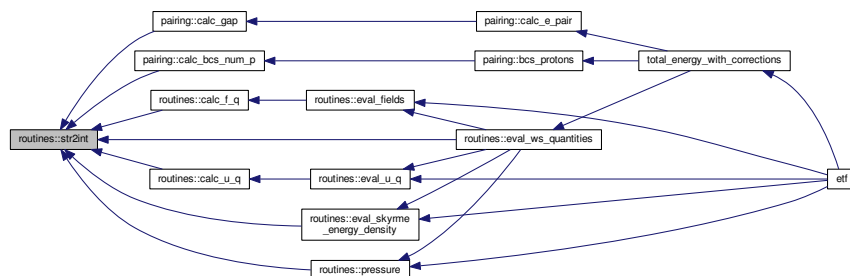
Here is the caller graph for this function:



5.3.2.35 str2int()

```
integer function routines::str2int (
    character(len=*), intent(in) string )
```

Here is the caller graph for this function:



5.3.2.36 write_densities()

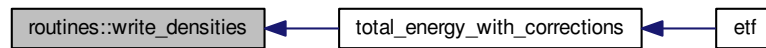
```
subroutine routines::write_densities ( )
```

Subroutine to write neutron and proton densities to files.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.37 write_fields()

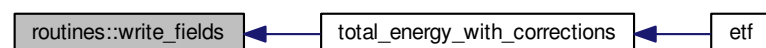
```
subroutine routines::write_fields ( )
```

Subroutine to write neutron and proton fields to files.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.38 write_sp_states_p()

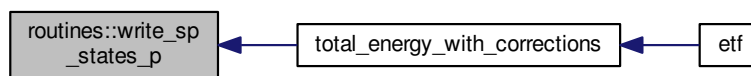
```
subroutine routines::write_sp_states_p ( )
```

Subroutine to write proton single particle energies to file, and extra details if BCS has been performed.

Author

M. Shelley

Here is the caller graph for this function:



5.3.2.39 ws_integral()

```
real(kind=dp) function routines::ws_integral (
    real(kind=dp), dimension(:), intent(in) quantity,
    integer, intent(in) n_max )
```

Function to integrate a density or field over the whole W-S cell, using Simpson's rule (+ Simpson's 3/8 rule if odd number of points)

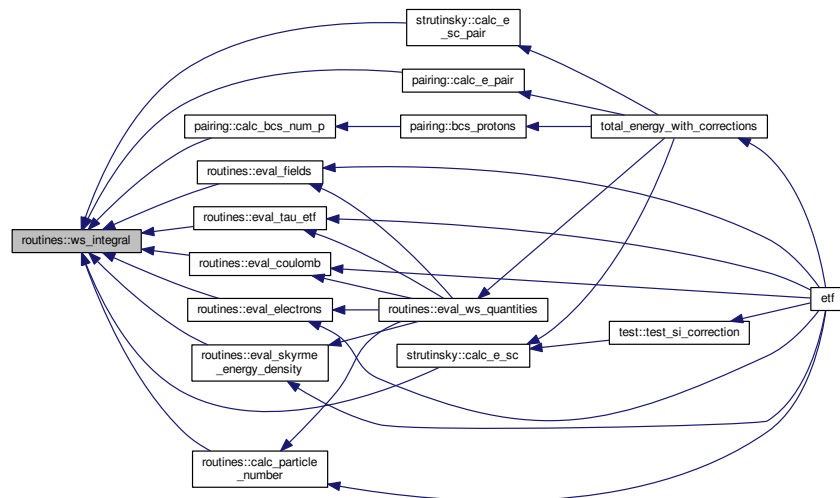
Author

M. Shelley

Parameters

| | | |
|----|-----------------|--|
| in | <i>quantity</i> | Array of quantity, evaluated on r mesh, to be integrated |
| in | <i>n_max</i> | Maximum mesh point in array for integration |

Here is the caller graph for this function:



5.3.3 Variable Documentation

5.3.3.1 file_unit_0

```
integer routines::file_unit_0
```

5.3.3.2 file_unit_1

```
integer routines::file_unit_1
```

Unit numbers for opening files.

5.4 rspace Module Reference

Module to hold routines for calculating single particle energies. Written by A. Pastore, some variables and use statements modified by M. Shelley for integration into etf code.

Functions/Subroutines

- subroutine **boundary** (h, Lmax, Ecut, Nmaxt, PS, NNST, JJP, LP, EP, isospin)
- subroutine **numerov** (h, Nrmax, Emax0, N, L, J, EFINAL, U, V, VSO, HME, GI, GIprimo, no)

5.4.1 Detailed Description

Module to hold routines for calculating single particle energies. Written by A. Pastore, some variables and use statements modified by M. Shelley for integration into etf code.

Author

A. Pastore, M. Shelley

5.4.2 Function/Subroutine Documentation

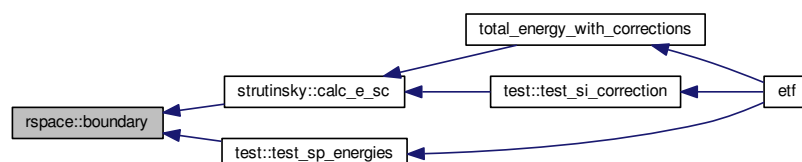
5.4.2.1 boundary()

```
subroutine rspace::boundary (
    double precision h,
    integer Lmax,
    double precision Ecut,
    integer Nmaxt,
    double precision, dimension(nmaxstate,nmaxt) PS,
    integer NNST,
    integer, dimension(nmaxstate) JJP,
    integer, dimension(nmaxstate) LP,
    double precision, dimension(nmaxstate) EP,
    integer isospin )
```

Here is the call graph for this function:



Here is the caller graph for this function:



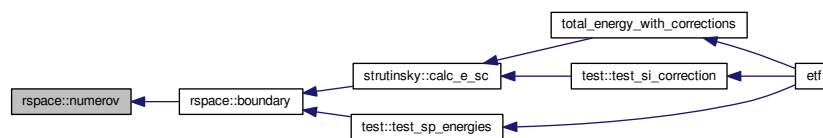
5.4.2.2 numerov()

```

subroutine rspace::numerov (
    double precision h,
    integer Nrmax,
    double precision Emax0,
    integer N,
    integer L,
    integer J,
    double precision EFINAL,
    double precision, dimension(nrmax) U,
    double precision, dimension(nrmax) V,
    double precision, dimension(nrmax) VSO,
    double precision, dimension(nrmax) HME,
    double precision, dimension(nrmax) GI,
    double precision, dimension(nrmax) GIprimo,
    integer no )

```

Here is the caller graph for this function:



5.5 strutinsky Module Reference

Module to hold routines for carrying out Strutinsky correction.

Functions/Subroutines

- subroutine [sp_states_setup](#) ()
Subroutine to allocate + initialise variables needed in 'boundary' routine, and allocate arrays needed for BCS.
- subroutine [sp_states_deallocate](#) ()
Subroutine to deallocate arrays needed in 'boundary' routine, and deallocate arrays needed for BCS.
- subroutine [state_sort](#) (sp_J, sp_L, sp_E, sp_wfns)
Subroutine to sort single particle states by their energies. Use bubble sort.
- subroutine [calc_e_sc](#) (q)
Subroutine to calculate the Strutinsky shell correction energy for isospin q. First, sum over occupied states of s.p. energies. Second, integrate over W-S cell of smoothed ETF densities and fields.
- subroutine [calc_e_sc_pair](#) ()
Subroutine to calculate the Strutinsky shell correction energy for protons, using BCS occupation probabilities.

5.5.1 Detailed Description

Module to hold routines for carrying out Strutinsky correction.

Author

M. Shelley

5.5.2 Function/Subroutine Documentation

5.5.2.1 `calc_e_sc()`

```
subroutine strutinsky::calc_e_sc (
    integer, intent(in) q )
```

Subroutine to calculate the Strutinsky shell correction energy for isospin q . First, sum over occupied states of s.p. energies. Second, integrate over W-S cell of smoothed ETF densities and fields.

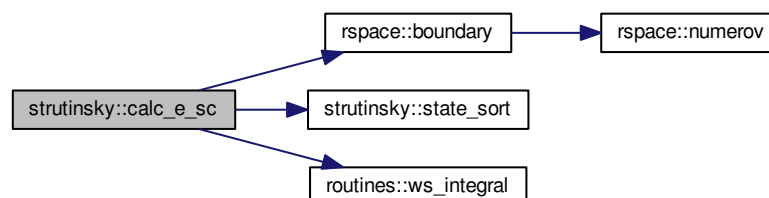
Author

M. Shelley

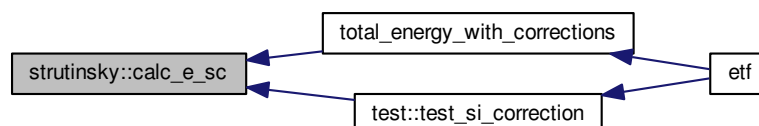
Parameters

| | | |
|----|-----|---------|
| in | q | Isospin |
| in | q | Isospin |

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.2 calc_e_sc_pair()

```
subroutine strutinsky::calc_e_sc_pair ( )
```

Subroutine to calculate the Strutinsky shell correction energy for protons, using BCS occupation probabilities.

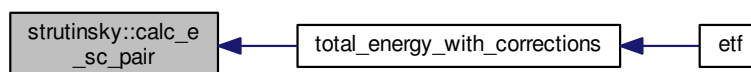
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.3 sp_states_deallocate()

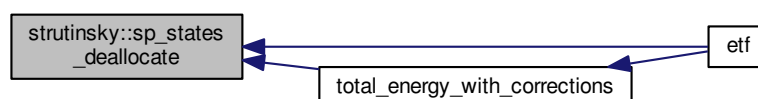
```
subroutine strutinsky::sp_states_deallocate ( )
```

Subroutine to deallocate arrays needed in 'boundary' routine, and deallocate arrays needed for BCS.

Author

M. Shelley

Here is the caller graph for this function:



5.5.2.4 sp_states_setup()

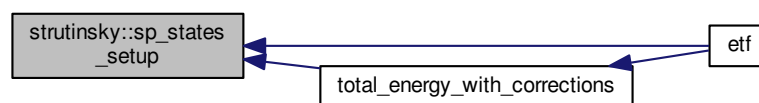
```
subroutine strutinsky::sp_states_setup ( )
```

Subroutine to allocate + initialise variables needed in 'boundary' routine, and allocate arrays needed for BCS.

Author

M. Shelley

Here is the caller graph for this function:



5.5.2.5 state_sort()

```
subroutine strutinsky::state_sort (
    integer, dimension(:), intent(inout) sp_J,
    integer, dimension(:), intent(inout) sp_L,
    real(kind=dp), dimension(:), intent(inout) sp_E,
    real(kind=dp), dimension(:,:), intent(inout) sp_wfns )
```

Subroutine to sort single particle states by their energies. Use bubble sort.

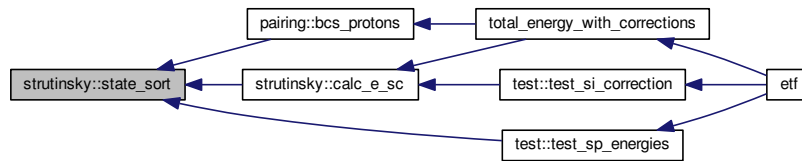
Author

M. Shelley

Parameters

| | | |
|---------|----------------|---|
| in, out | <i>sp_J</i> | Array for (2)J's of single particle states |
| in, out | <i>sp_L</i> | Array for L's of single particle states |
| in, out | <i>sp_E</i> | Array for energies of single particle states |
| in, out | <i>sp_wfns</i> | Array for wavefunctions of single particle states |

Here is the caller graph for this function:



5.6 test Module Reference

Module to hold test routines.

Functions/Subroutines

- subroutine [read_test_params](#) ()
Subroutine to read from "test_input.in" parameters used for test routines.
- subroutine [test_densities](#) ()
Subroutine to calculate the density profiles using sample parameters.
- subroutine [test_density_derivs](#) ()
Subroutine to test some derivatives of the total and individual densities.
- subroutine [test_eff_mass](#) ()
Subroutine to test effective masses $f m^{-3}$.
- subroutine [test_kinetic_densities](#) ()
Subroutine to test kinetic energy densities.
- subroutine [test_spin_current_densities](#) ()
Subroutine to write spin current densities and spin-orbit fields to file.
- subroutine [orders_kinetic_densities](#) ()
Subroutine to write order-by-order comparison of kinetic densities to file.
- subroutine [test_central_potentials](#) ()
Subroutine to write spin central potentials to file.
- subroutine [write_ws_cell_array_quantity](#) (quantity)
Utility subroutine that can be called anywhere at any time, for writing a quantity (evaluated over WS cell) to file.
- subroutine [test_calc_particle_number](#) ()
Subroutine to test calculation of neutron and proton particle numbers.
- subroutine [test_calc_coulomb_energy](#) ()
Subroutine to test calculation of Coulomb energy.
- subroutine [test_calc_skyrme_energy](#) ()
Subroutine to test calculation of Skyrme energy.
- subroutine [test_sp_energies](#) ()
Subroutine to test calculation of single particle energies.
- subroutine [test_si_correction](#) ()
Subroutine to test the Strutinsky integral correction.
- subroutine [bartel_bencheikh_benchmark](#) ()
Subroutine to check different contributions to τ_{ETF} at 2nd order, for comparison with results in Bartel and Bencheikh (2002)

Variables

- integer `file_unit`
- integer `file_unit_2`
Unit numbers for opening files.
- integer `sample_profile`
Test density profile to use.

5.6.1 Detailed Description

Module to hold test routines.

Author

M. Shelley

5.6.2 Function/Subroutine Documentation

5.6.2.1 `bartel_bencheikh_benchmark()`

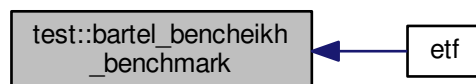
```
subroutine test::bartel_bencheikh_benchmark ( )
```

Subroutine to check different contributions to τ_{ETF} at 2nd order, for comparison with results in Bartel and Bencheikh (2002)

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.2 orders_kinetic_densities()

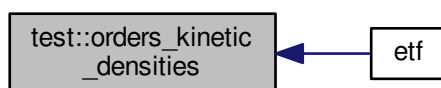
```
subroutine test::orders_kinetic_densities ( )
```

Subroutine to write order-by-order comparison of kinetic densities to file.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.3 read_test_params()

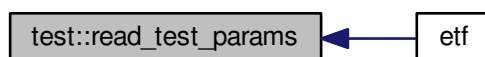
```
subroutine test::read_test_params ( )
```

Subroutine to read from "test_input.in" parameters used for test routines.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.4 test_calc_coulomb_energy()

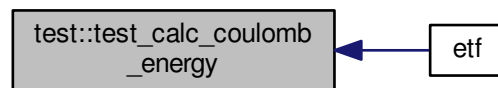
```
subroutine test::test_calc_coulomb_energy ( )
```

Subroutine to test calculation of Coulomb energy.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.5 test_calc_particle_number()

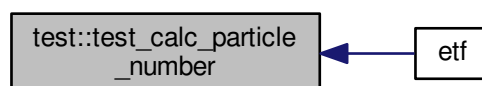
```
subroutine test::test_calc_particle_number ( )
```

Subroutine to test calculation of neutron and proton particle numbers.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.6 test_calc_skyrme_energy()

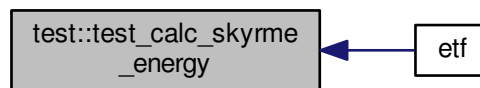
```
subroutine test::test_calc_skyrme_energy ( )
```

Subroutine to test calculation of Skyrme energy.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.7 test_central_potentials()

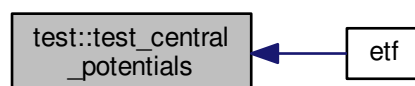
```
subroutine test::test_central_potentials ( )
```

Subroutine to write spin central potentials to file.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.8 test_densities()

```
subroutine test::test_densities ( )
```

Subroutine to calculate the density profiles using sample parameters.

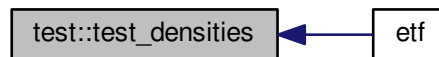
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.9 test_density_derivs()

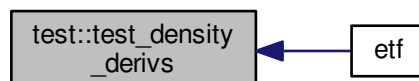
```
subroutine test::test_density_derivs ( )
```

Subroutine to test some derivatives of the total and individual densities.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.10 test_eff_mass()

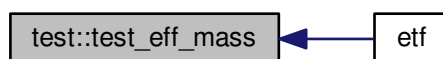
```
subroutine test::test_eff_mass ( )
```

Subroutine to test effective masses fm^{-3} .

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.11 test_kinetic_densities()

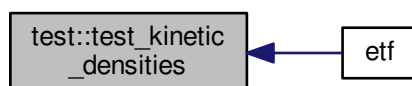
```
subroutine test::test_kinetic_densities ( )
```

Subroutine to test kinetic energy densities.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.12 test_si_correction()

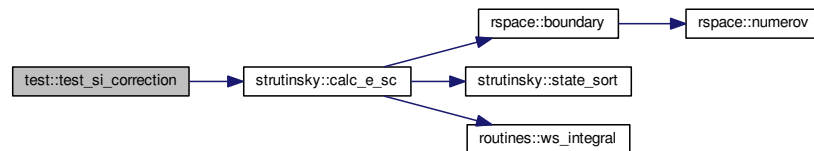
```
subroutine test::test_si_correction ( )
```

Subroutine to test the Strutinsky integral correction.

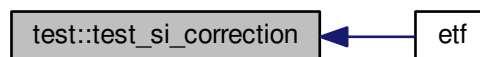
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.13 test_sp_energies()

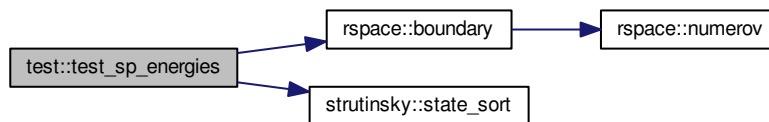
```
subroutine test::test_sp_energies ( )
```

Subroutine to test calculation of single particle energies.

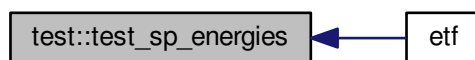
Author

M. Shelley

Here is the call graph for this function:



Here is the caller graph for this function:

**5.6.2.14 test_spin_current_densities()**

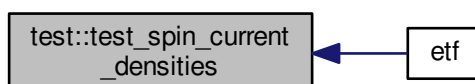
```
subroutine test::test_spin_current_densities ( )
```

Subroutine to write spin current densities and spin-orbit fields to file.

Author

M. Shelley

Here is the caller graph for this function:



5.6.2.15 write_ws_cell_array_quantity()

```
subroutine test::write_ws_cell_array_quantity (
    real(kind=dp), dimension(1:n), intent(in) quantity )
```

Utility subroutine that can be called anywhere at any time, for writing a quantity (evaluated over WS cell) to file.

Author

M. Shelley

Parameters

| | | |
|----|-----------------|-----------------------------------|
| in | <i>quantity</i> | Array to be written to file (1:n) |
|----|-----------------|-----------------------------------|

5.6.3 Variable Documentation

5.6.3.1 file_unit

```
integer test::file_unit
```

5.6.3.2 file_unit_2

```
integer test::file_unit_2
```

Unit numbers for opening files.

5.6.3.3 sample_profile

```
integer test::sample_profile
```

Test density profile to use.

Chapter 6

File Documentation

6.1 srcs/boundary.f90 File Reference

Modules

- module [rspace](#)

Module to hold routines for calculating single particle energies. Written by A. Pastore, some variables and use statements modified by M. Shelley for integration into etf code.

Functions/Subroutines

- subroutine [rspace::boundary](#) (h, Lmax, Ecut, Nmaxt, PS, NNST, JJP, LP, EP, isospin)
- subroutine [rspace::numerov](#) (h, Nrmax, Emax0, N, L, J, EFINAL, U, V, VSO, HME, GI, GIprimo, no)

6.2 srcs/etf.f90 File Reference

Functions/Subroutines

- program [etf](#)

Program to calculate the equation of state in the inner crust with the extended Thomas-Fermi (ETF) approach.

- subroutine [total_energy_with_corrections](#) (write_files_print_info)

Subroutine to evaluate all WS quantities using supplied profile parameters, and calculate total energy, with pairing and shell corrections included. Write densities and fields to files, and all info to screen, if specified.

6.2.1 Function/Subroutine Documentation

6.2.1.2 total_energy_with_corrections()

```
subroutine etf::total_energy_with_corrections (
    logical, intent(in) write_files_print_info )
```

Subroutine to evaluate all WS quantities using supplied profile parameters, and calculate total energy, with pairing and shell corrections included. Write densities and fields to files, and all info to screen, if specified.

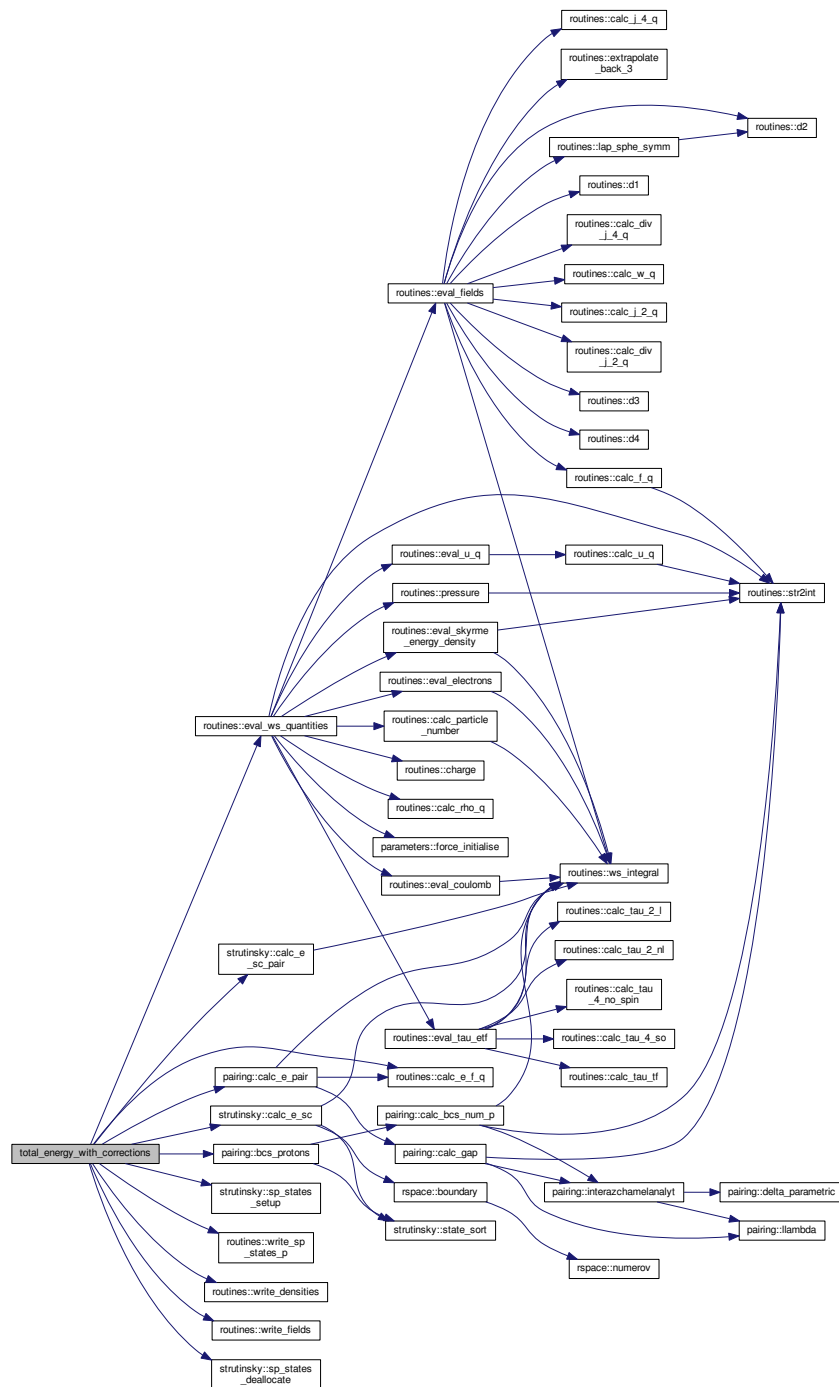
Author

M. Shelley

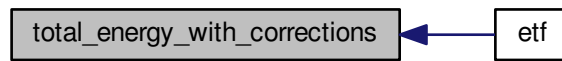
Parameters

| | | |
|----|-------------------------------|--------------------------------------|
| in | <i>write_files_print_info</i> | Whether to write to files and screen |
|----|-------------------------------|--------------------------------------|

Here is the call graph for this function:



Here is the caller graph for this function:



6.3 srcs/pairing.f90 File Reference

Modules

- module [pairing](#)
Module to hold routines for carrying out Strutinsky correction.

Functions/Subroutines

- subroutine [pairing::calc_e_pair](#) ()
Subroutine to calculate the pairing energy for the WS cell, using the method specified in "input.in".
- real(kind=dp) function [pairing::calc_gap](#) (r, rhon, rhop, fq, mu)
Function to calculate the pairing gap in infinite neutron matter, for specified density and effective mass.
- subroutine [pairing::bcs_protons](#) ()
Subroutine to perform BCS for protons.
- real(kind=dp) function [pairing::calc_bcs_num_p](#) (mu_p)
Function to solve gap equation at given chemical potential "mu_p", returning the (BCS) number of protons.
- real(kind=dp) function [pairing::interazchamelanalyt](#) (rhon, rhop, hbm, itz)
Function to calculate the interaction strength for the effective contact pairing force. Modified by M. Shelley.
- real(kind=dp) function [pairing::llambda](#) (x)
Function to calculate the pairing cutoff for the effective contact pairing force.
- real(kind=dp) function [pairing::delta_parametric](#) (kf, YY, itz, xk0)
Function to calculate the pairing gap using the analytical BSk expression. Modified by M. Shelley.

6.4 srcs/parameters.f90 File Reference

Modules

- module [parameters](#)
Module to hold global parameters and variables.

Functions/Subroutines

- subroutine [parameters::force_initialise](#) ()
Subroutine to initialise Skyrme force parameters with values for a given force specified in 'input.dat'.

Variables

- integer, parameter `parameters::dp` = selected_real_kind(15, 300)
Precision.
- real(kind=dp), parameter `parameters::pi` = 3.1415926535897932_dp
 π
- real(kind=dp), parameter `parameters::hbar2_2m` = 20.73553_dp
 $\frac{\hbar^2}{2m}$ for *SLy forces*
- real(kind=dp), parameter `parameters::rho0` = 0.16_dp
 ρ_0 , nuclear saturation density [fm^{-3}]
- real(kind=dp), parameter `parameters::e2` = 1.439978408596513_dp
 e^2 , electric charge squared [$MeV \cdot fm$]
- real(kind=dp), parameter `parameters::mec2` = 0.51099895_dp
Electron rest mass [MeV/c^2].
- real(kind=dp), parameter `parameters::mnc2` = 939.56542052_dp
- real(kind=dp), parameter `parameters::mpc2` = 938.27208816_dp
- real(kind=dp), parameter `parameters::hbarc` = 197.3269804_dp
 $\hbar * c$
- real(kind=dp), parameter `parameters::mamuc2` = 931.49386_dp
Atomic mass unit u .
- real(kind=dp), parameter `parameters::xmh` = 7.28896940_dp
- real(kind=dp), parameter `parameters::rydb` = 13.6056981e-6_dp
Rydberg constant.
- real(kind=dp), parameter `parameters::xmp` = xmh - mec2 + rydb
- real(kind=dp), parameter `parameters::xmn` = 8.07132281_dp
- real(kind=dp), parameter `parameters::fr12` = 1._dp/2
- real(kind=dp), parameter `parameters::fr14` = 1._dp/4
- real(kind=dp), parameter `parameters::fr18` = 1._dp/8
- real(kind=dp), parameter `parameters::fr1_16` = 1._dp/16
- real(kind=dp), parameter `parameters::fr1_32` = 1._dp/32
- real(kind=dp), parameter `parameters::fr13` = 1._dp/3
- real(kind=dp), parameter `parameters::fr16` = 1._dp/6
- real(kind=dp), parameter `parameters::fr1_12` = 1._dp/12
- real(kind=dp), parameter `parameters::fr1_24` = 1._dp/24
- real(kind=dp), parameter `parameters::fr1_36` = 1._dp/36
- real(kind=dp), parameter `parameters::fr23` = 2._dp/3
- real(kind=dp), parameter `parameters::fr29` = 2._dp/9
- real(kind=dp), parameter `parameters::fr32` = 3._dp/2
- real(kind=dp), parameter `parameters::fr34` = 3._dp/4
- real(kind=dp), parameter `parameters::fr35` = 3._dp/5
- real(kind=dp), parameter `parameters::fr38` = 3._dp/8
- real(kind=dp), parameter `parameters::fr3_10` = 3._dp/10
- real(kind=dp), parameter `parameters::fr43` = 4._dp/3
- real(kind=dp), parameter `parameters::fr53` = 5._dp/3
- real(kind=dp), parameter `parameters::fr54` = 5._dp/4
- real(kind=dp), parameter `parameters::fr83` = 8._dp/3
- real(kind=dp), parameter `parameters::fr260_3` = 260._dp/3
- real(kind=dp) `parameters::w0`
Spin-orbit strength W_0 .
- real(kind=dp) `parameters::t0`
- real(kind=dp) `parameters::x0`
- real(kind=dp) `parameters::t1`
- real(kind=dp) `parameters::x1`

- real(kind=dp) [parameters::t2](#)
- real(kind=dp) [parameters::x2](#)
- real(kind=dp) [parameters::t3](#)
- real(kind=dp) [parameters::x3](#)
- real(kind=dp) [parameters::sigma](#)
- real(kind=dp), dimension(0:1) [parameters::hbar2_2m_q](#)

$\frac{\hbar^2}{2m}$ with isospin dependence, as required by BSk forces

- logical [parameters::j2_terms](#)

Whether functional uses \mathbf{J}^2 terms.

- real(kind=dp) [parameters::b1](#)
- real(kind=dp) [parameters::b2](#)
- real(kind=dp) [parameters::b3](#)
- real(kind=dp) [parameters::b4](#)
- real(kind=dp) [parameters::b5](#)
- real(kind=dp) [parameters::b6](#)
- real(kind=dp) [parameters::b7](#)
- real(kind=dp) [parameters::b8](#)
- real(kind=dp) [parameters::b9](#)
- real(kind=dp) [parameters::b10](#)
- real(kind=dp) [parameters::b11](#)
- real(kind=dp) [parameters::b12](#)
- real(kind=dp) [parameters::b13](#)
- real(kind=dp) [parameters::alpha](#)
- real(kind=dp) [parameters::beta](#)
- real(kind=dp) [parameters::gamma](#)
- real(kind=dp) [parameters::t4](#)
- real(kind=dp) [parameters::x4](#)
- real(kind=dp) [parameters::t5](#)
- real(kind=dp) [parameters::x5](#)
- real(kind=dp) [parameters::fn_pos](#)
- real(kind=dp) [parameters::fn_neg](#)
- real(kind=dp) [parameters::fp_pos](#)
- real(kind=dp) [parameters::fp_neg](#)
- real(kind=dp) [parameters::epsilon_lambda](#)
- real(kind=dp), dimension(:), allocatable [parameters::r](#)

Mesh for r .

- real(kind=dp), dimension(:,:), allocatable [parameters::rho_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::del_rho_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::del2_rho_q](#)
- real(kind=dp), dimension(:), allocatable [parameters::rho_t](#)
- real(kind=dp), dimension(:), allocatable [parameters::del_rho_t](#)
- real(kind=dp), dimension(:), allocatable [parameters::del2_rho_t](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::f_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::del_f_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::del2_f_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::w_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::div_w_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::u_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::j_2_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::j_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::del_j_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::div_j_2_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::div_j_q](#)
- real(kind=dp), dimension(:,:), allocatable [parameters::tau_tf_q](#)

- `real(kind=dp), dimension(:, :), allocatable` [parameters::tau_2_l_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::tau_2_nl_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::tau_etf_q](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::j_t](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::del_j_t](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::div_j_t](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::tau_etf_t](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::e_density_field](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::e_density_sky](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::v_c_di](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::v_c_ex](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::e_density_c_di](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::e_density_c_ex](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::v_c_pe](#)
- `real(kind=dp), dimension(:, :, :), allocatable` [parameters::tau_2_cont_q](#)
- `real(kind=dp), dimension(:), allocatable` [parameters::rho_ch](#)

Charge density.

- `real(kind=dp), dimension(:, :), allocatable` [parameters::d2_rho_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d3_rho_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d4_rho_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d2_f_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d3_f_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d4_f_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::a_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d1_a_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d2_a_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d3_a_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::d4_a_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::tau_4_no_spin_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::tau_4_so_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::j_4_q](#)
- `real(kind=dp), dimension(:, :), allocatable` [parameters::div_j_4_q](#)
- `real(kind=dp), dimension(:, :, :), allocatable` [parameters::tau_4_cont_q](#)
- `real(kind=dp), dimension(0:1)` [parameters::n_q](#)

Number of neutrons and protons.

- `real(kind=dp)` [parameters::n_t](#)

Total number of particles.

- `real(kind=dp)` [parameters::e_field](#)

Field energy.

- `real(kind=dp)` [parameters::e_skyrme](#)

Skyrme energy.

- `real(kind=dp), dimension(0:1)` [parameters::e_kinetic_q](#)

Kinetic energy for neutron and protons.

- `real(kind=dp)` [parameters::e_kinetic_t](#)

Total kinetic energy.

- `real(kind=dp)` [parameters::e_so](#)

Spin-orbit energy.

- `real(kind=dp)` [parameters::e_coulomb_di](#)

Direct Coulomb energy.

- `real(kind=dp)` [parameters::e_coulomb_ex](#)

Exchange Coulomb energy.

- `real(kind=dp)` [parameters::e_coulomb](#)

Total Coulomb energy.

- real(kind=dp) `parameters::e_total`
Total energy.
- real(kind=dp), dimension(0:1) `parameters::mu_q`
Neutron and proton chemical potential.
- real(kind=dp) `parameters::mu_e`
Electron chemical potential.
- real(kind=dp) `parameters::mu_c`
Coulomb interaction contribution to electron chemical potential.
- real(kind=dp) `parameters::delta_mu`
Overall chemical potential (beta-equilibrium condition)
- real(kind=dp) `parameters::pressure_nucl`
Nuclear pressure.
- real(kind=dp) `parameters::pressure_e`
Electron pressure.
- real(kind=dp) `parameters::pressure_ex`
Coulomb exchange pressure.
- real(kind=dp) `parameters::pressure_t`
Total pressure.
- integer `parameters::run_mode`
Mode to run code in (normal, test)
- logical `parameters::verbose`
Whether to print all extra messages about run.
- real(kind=dp) `parameters::dr`
Mesh spacing of r (fm).
- logical `parameters::profile_r_max`
Whether to take "r_max" from "r_ws".
- real(kind=dp) `parameters::r_max`
Max value of r (fm).
- logical `parameters::specify_n`
Whether to specify "n" instead of "dr".
- integer `parameters::n`
Number of mesh points.
- integer `parameters::force`
Force to use.
- integer `parameters::etf_order`
Order at which to calculate kinetic energy densities.
- logical `parameters::coulomb_on`
Use Coulomb interaction or not.
- logical `parameters::electrons_on`
Add electrons.
- integer `parameters::nmaxstate`
Number of states that can be stored.
- integer `parameters::lmax`
Maximum angular momentum of states to find.
- logical `parameters::calc_chem_pots`
Calculate chemical potentials.
- logical, dimension(0:1) `parameters::strutinsky_on`
Use Strutinsky Integral (SI) correction for neutrons and protons.
- real(kind=dp) `parameters::strut_r_max`
Max value of r to use for box for single particle states (fm).
- logical `parameters::emax0_mod`

- Whether to modify "emax0" to always = 0.*

 - integer `parameters::neutron_pairing`

Type of pairing calculation to perform for neutrons.
 - logical `parameters::proton_bcs`

Whether to do BCS for protons.
 - real(kind=dp) `parameters::pair_qp_cut`

Cut-off for quasiparticle energy for solving gap equation.
 - logical `parameters::smooth_qp_cut`

Whether to use smooth cut-off on quasiparticle energy.
 - real(kind=dp) `parameters::pair_k_max`

Maximum momentum for integral in gap equation.
 - real(kind=dp) `parameters::pair_v0`

Interaction strength parameter.
 - real(kind=dp) `parameters::pair_eta`

Interaction parameter eta.
 - real(kind=dp) `parameters::pair_alpha`

Interaction parameter alpha.
 - real(kind=dp) `parameters::pair_tol`

Gap equation self-consistency tolerance.
 - real(kind=dp) `parameters::pair_mix`

Mix of previous iteration in gap equation self-consistency loop.
 - real(kind=dp) `parameters::pair_dk`

Step size in integral in gap equation.
 - real(kind=dp) `parameters::pair_del_init`

Initial guess for pairing gap.
 - integer `parameters::pair_max_iters`

Maximum iterations for gap equation self-consistency loop.
 - real(kind=dp) `parameters::pair_num_tol`

BCS number equation self-consistency tolerance.
 - real(kind=dp), dimension(5) `parameters::n_profile`
 - real(kind=dp), dimension(5) `parameters::p_profile`
 - real(kind=dp) `parameters::num_n`
 - integer `parameters::num_p`
 - real(kind=dp) `parameters::r_ws`
 - character(18), dimension(10) `parameters::n_floats` = `('/(1(es24.16e3,1x))','(2(es24.16e3,1x))','(3(es24.16e3,1x))','(4(es24.16e3,1x))','(5(es24.16e3,1x))','(6(es24.16e3,1x))','(7(es24.16e3,1x))','(8(es24.16e3,1x))','(9(es24.16e3,1x))','(10(es24.16e3,1x))')'`
 - character(27) `parameters::info_format` = `'(a27,1x,a1,1x,es24.16e3)'`
 - character(8) `parameters::force_string`
 - character(80), parameter `parameters::line_break` = `#####`
- Separating output.*
- character(62), parameter `parameters::table_string` = `'-----+-----'`
 - character(1), dimension(0:1), parameter `parameters::iso_string` = `('n','p')`
- Isospin labels.*
- character(4), parameter `parameters::space4` = `' '`
- Whitespace of length 4.*
- integer `parameters::strut_n`
 - real(kind=dp), dimension(:,,:), allocatable `parameters::dhmen`
 - real(kind=dp), dimension(:,,:), allocatable `parameters::d2hmen`
 - real(kind=dp), dimension(:,,:), allocatable `parameters::hb2m`
 - real(kind=dp), dimension(:,,:), allocatable `parameters::vpot`
 - real(kind=dp), dimension(:,,:), allocatable `parameters::vso`

- real(kind=dp), dimension(0:1) [parameters::ecut](#)
Cutoff energies for neutrons and protons.
- real(kind=dp), dimension(:,:), allocatable [parameters::ps](#)
Storage for single particle wavefunctions.
- integer [parameters::nnst](#)
Number of states found within cutoff energy.
- integer, dimension(:), allocatable [parameters::jjp](#)
Storage for J of states found.
- integer, dimension(:), allocatable [parameters::lp](#)
Storage for L of states found.
- real(kind=dp), dimension(:), allocatable [parameters::ep](#)
Storage for energy of states found.
- real(kind=dp), dimension(0:1) [parameters::e_sc_q](#)
Storage for shell correction energies.
- real(kind=dp) [parameters::e_sc_t](#)
Storage for total shell correction energy.
- real(kind=dp), dimension(:), allocatable [parameters::pair_gap_n](#)
Neutron pairing gap.
- real(kind=dp), dimension(:), allocatable [parameters::delta_n](#)
Neutron pairing field.
- real(kind=dp), dimension(:), allocatable [parameters::pair_gap_p](#)
Proton pairing gap.
- real(kind=dp), dimension(:), allocatable [parameters::delta_p](#)
Proton pairing field.
- real(kind=dp), dimension(:), allocatable [parameters::rho_p_bcs](#)
Proton density (from wavefunctions)
- real(kind=dp), dimension(:), allocatable [parameters::rho_anom_p_bcs](#)
Anomalous density.
- real(kind=dp), dimension(:), allocatable [parameters::strength_bcs](#)
Interaction strength for protons.
- real(kind=dp), dimension(:), allocatable [parameters::occ_v](#)
Particle occupation probabilities.
- real(kind=dp), dimension(:), allocatable [parameters::occ_u](#)
Hole occupation probabilities.
- real(kind=dp), dimension(:), allocatable [parameters::e_qp_p](#)
Quasiparticle energies.
- real(kind=dp) [parameters::num_p_bcs](#)
Number of protons calculated from occupations.
- real(kind=dp) [parameters::e_pair_bcs](#)
BCS pairing energy.
- real(kind=dp), dimension(0:1) [parameters::e_pair_q](#)
Storage for pairing condensation energies.
- real(kind=dp) [parameters::e_pair_t](#)
Storage for total pairing condensation energy.
- real(kind=dp) [parameters::rho_e](#)
Electron density.
- real(kind=dp) [parameters::e_kinetic_e](#)
Total electron kinetic energy.
- real(kind=dp) [parameters::e_coulomb_e](#)
Total electron-electron potential energy from Coulomb interaction.
- real(kind=dp) [parameters::e_coulomb_pe](#)
Total proton-electron potential energy from Coulomb interaction.

6.5 srcs/routines.f90 File Reference

Modules

- module [routines](#)
Module to hold main routines: densities, energies.

Functions/Subroutines

- integer function [routines::str2int](#) (string)
- subroutine [routines::allocate_arrays](#) ()
Subroutine to allocate arrays for all densities and effective masses, and for their derivatives.
- subroutine [routines::initialise_uniform_mesh](#) ()
Subroutine to calculate number of mesh points for a uniform grid, allocate array for r , and then populate r with the mesh points.
- subroutine [routines::deallocate_mesh](#) ()
Subroutine to deallocate array for r .
- subroutine [routines::deallocate_arrays](#) ()
Subroutine to deallocate arrays for all densities and effective masses, and for their derivatives.
- real(kind=dp) function [routines::calc_rho_q](#) (rhoqgas, rhoqliq, rq, aq, gammaq, r)
Function to calculate the neutron or proton density at a given radius r , using supplied input parameters.
- subroutine [routines::calc_f_q](#) (rhot, rhoq, q, fq)
Subroutine to calculate the effective mass ratio $f_q = \frac{m}{m_q^*}$ for neutrons or protons, for standard Skyrme or for BSk forces.
- subroutine [routines::calc_w_q](#) (delrho, delrhoq, Wq)
Function to calculate the spin-orbit field W_q for neutrons or protons.
- subroutine [routines::calc_tau_tf](#) (rhoq, tauTF)
Function to calculate the zeroth-order contribution to the kinetic energy density for neutrons or protons.
- subroutine [routines::calc_tau_2_l](#) (rhoq, delrhoq, del2rhoq, tau2Lq, t2contq)
Function to calculate the local second-order contribution to the kinetic energy density for neutrons or protons.
- subroutine [routines::calc_tau_2_nl](#) (rhoq, delrhoq, fq, delfq, del2fq, Wq, q, tau2NLq, t2contq)
Subroutine to calculate the non-local second-order contribution to the kinetic energy density for neutrons or protons.
- subroutine [routines::calc_tau_4_no_spin](#) (rhoq, d1rhoq, d2rhoq, d3rhoq, d4rhoq, fq, d1fq, d2fq, d3fq, d4fq, tau4nospinq, t4contq)
Subroutine to calculate the fourth-order contributions to the kinetic energy density (without spin-orbit contributions) τ_{q_4} for neutrons or protons.
- subroutine [routines::calc_tau_4_so](#) (rhoq, d1rhoq, fq, d1fq, d2fq, d1Aq, d2Aq, d3Aq, q, tau_4_so_q, t4contq)
Subroutine to calculate the fourth-order contributions to the kinetic energy density (spin-orbit contributions) τ_{q_4} for neutrons or protons.
- subroutine [routines::calc_j_2_q](#) (rhoq, Wq, fq, q, J2q)
Subroutine to calculate the second-order contributions to the spin current density J_q for neutrons or protons.
- subroutine [routines::calc_j_4_q](#) (rhoq, d1rhoq, fq, d1fq, d2fq, d1Aq, d2Aq, d3Aq, q, J4q)
Subroutine to calculate the fourth-order contributions to the spin current density J_q for neutrons or protons.
- subroutine [routines::calc_div_j_2_q](#) (rhoq, d1rhoq, fq, d1fq, d1Aq, d2Aq, q, divJ2q)
Subroutine to calculate the second-order contributions to the divergence of the current density J_q for neutrons or protons.
- subroutine [routines::calc_div_j_4_q](#) (rhoq, d1rhoq, d2rhoq, fq, d1fq, d2fq, d3fq, d1Aq, d2Aq, d3Aq, d4Aq, q, divJ4q)
Subroutine to calculate the fourth-order contributions to the divergence of the current density J_q for neutrons or protons.

- real(kind=dp) function `routines::calc_u_q` (rhot, rhon, rhop, delrhot, delrhon, delrhon, del2rhot, del2rhon, del2rhon, taut, taun, taup, divJt, divJq, q)
Function to calculate the central potential U_q for neutrons or protons, for standard Skyrme or for BSk forces with extra terms.
- subroutine `routines::eval_fields` ()
Subroutine to evaluate matter density derivatives, and all fields: effective masses (and derivatives), spin-orbit fields, spin current densities (and derivatives)
- subroutine `routines::eval_tau_etf` ()
Subroutine to calculate the kinetic energy density τ_{ETF} with the (extended) Thomas-Fermi approximation at order specified by "etf_order".
- subroutine `routines::charge` (rho, Neutr, Nprot, Ngrid1, del1, rhoch)
Subroutine to calculate the charge density from the matter densities. Modified by M. Shelley.
- subroutine `routines::eval_coulomb` (prot_dens, calc_exchange)
Subroutine to evaluate the Coulomb potentials and energy densities, using either the proton or charge density.
- subroutine `routines::eval_electrons` ()
Subroutine to evaluate the proton-electron potential which contributes to U_q , all energy contributions, and the pressure, coming from a homogeneous electron gas. Also calculates the electron chemical potential.
- subroutine `routines::eval_u_q` ()
Subroutine to evaluate the central fields U_q .
- subroutine `routines::eval_skyrme_energy_density` ()
Subroutine to evaluate the Skyrme energy density, after first separately evaluating the field energy density.
- subroutine `routines::pressure` ()
Subroutine to calculate the nuclear contribution to the pressure, and then the total pressure.
- subroutine `routines::calc_particle_number` ()
Subroutine to calculate the number of particles in the Wigner-Seitz cell.
- subroutine `routines::eval_ws_quantities` ()
Subroutine to evaluate the various densities, fields, derivatives, energies, and particle numbers in the WS cell with densities ρ_{ho_q} .
- real(kind=dp) function `routines::calc_e_f_q` (rhoq, fq, q)
Subroutine to evaluate the Fermi energy at a given density and effective mass, for neutrons or protons.
- subroutine `routines::d1` (n, h, f, df)
This subroutine computes the first derivative of function f evaluated on the meshpoints 1,...,npt. The input is the function f with extrapolated values in -1, 0. Modified by M. Shelley.
- subroutine `routines::d2` (n, h, f, d2f)
This subroutine computes the second derivative of function f evaluated on the meshpoints 1,...,npt. The input is the function f with extrapolated values in -1, 0. Modified by M. Shelley.
- subroutine `routines::lap_sphe_symm` (f, df_dr, d2f_dr2)
Subroutine to carry out Laplacian in spherical symmetry.
- subroutine `routines::d3` (f, d3f_dr3)
Subroutine to calculate third derivative of function f evaluated on evenly-spaced meshpoints from 1 to n. Works with extrapolated values in -2,-1,0. Uses 7-point stencil.
- subroutine `routines::d4` (f, d4f_dr4)
Subroutine to calculate fourth derivative of function f evaluated on evenly-spaced meshpoints from 1 to n. Works with extrapolated values in -2,-1,0. Uses 7-point stencil.
- subroutine `routines::extrapolate_back_3` (grid)
Subroutine to extrapolate back 3 points on the r mesh, to facilitate the calculation of first and second derivatives using a 5-point stencil. Values in grid start at 4, extrapolated values are put in elements 3,2,1. Coefficients come from solving 4th-order polynomial, assuming that $f'(0) = 0$, and $f(-h) = f(h)$.
- real(kind=dp) function `routines::ws_integral` (quantity, n_max)
Function to integrate a density or field over the whole W-S cell, using Simpson's rule (+ Simpson's 3/8 rule if odd number of points)
- subroutine `routines::write_densities` ()
Subroutine to write neutron and proton densities to files.

- subroutine `routines::write_fields ()`
Subroutine to write neutron and proton fields to files.
- subroutine `routines::write_sp_states_p ()`
Subroutine to write proton single particle energies to file, and extra details if BCS has been performed.

Variables

- integer `routines::file_unit_0`
- integer `routines::file_unit_1`
Unit numbers for opening files.

6.6 srcs/strutinsky.f90 File Reference

Modules

- module `strutinsky`
Module to hold routines for carrying out Strutinsky correction.

Functions/Subroutines

- subroutine `strutinsky::sp_states_setup ()`
Subroutine to allocate + initialise variables needed in 'boundary' routine, and allocate arrays needed for BCS.
- subroutine `strutinsky::sp_states_deallocate ()`
Subroutine to deallocate arrays needed in 'boundary' routine, and deallocate arrays needed for BCS.
- subroutine `strutinsky::state_sort (sp_J, sp_L, sp_E, sp_wfns)`
Subroutine to sort single particle states by their energies. Use bubble sort.
- subroutine `strutinsky::calc_e_sc (q)`
Subroutine to calculate the Strutinsky shell correction energy for isospin q . First, sum over occupied states of s.p. energies. Second, integrate over W-S cell of smoothed ETF densities and fields.
- subroutine `strutinsky::calc_e_sc_pair ()`
Subroutine to calculate the Strutinsky shell correction energy for protons, using BCS occupation probabilities.

6.7 srcs/test.f90 File Reference

Modules

- module `test`
Module to hold test routines.

Functions/Subroutines

- subroutine `test::read_test_params` ()
Subroutine to read from "test_input.in" parameters used for test routines.
- subroutine `test::test_densities` ()
Subroutine to calculate the density profiles using sample parameters.
- subroutine `test::test_density_derivs` ()
Subroutine to test some derivatives of the total and individual densities.
- subroutine `test::test_eff_mass` ()
Subroutine to test effective masses $f m^{-3}$.
- subroutine `test::test_kinetic_densities` ()
Subroutine to test kinetic energy densities.
- subroutine `test::test_spin_current_densities` ()
Subroutine to write spin current densities and spin-orbit fields to file.
- subroutine `test::orders_kinetic_densities` ()
Subroutine to write order-by-order comparison of kinetic densities to file.
- subroutine `test::test_central_potentials` ()
Subroutine to write spin central potentials to file.
- subroutine `test::write_ws_cell_array_quantity` (quantity)
Utility subroutine that can be called anywhere at any time, for writing a quantity (evaluated over WS cell) to file.
- subroutine `test::test_calc_particle_number` ()
Subroutine to test calculation of neutron and proton particle numbers.
- subroutine `test::test_calc_coulomb_energy` ()
Subroutine to test calculation of Coulomb energy.
- subroutine `test::test_calc_skyrme_energy` ()
Subroutine to test calculation of Skyrme energy.
- subroutine `test::test_sp_energies` ()
Subroutine to test calculation of single particle energies.
- subroutine `test::test_si_correction` ()
Subroutine to test the Strutinsky integral correction.
- subroutine `test::bartel_bencheikh_benchmark` ()
Subroutine to check different contributions to τ_{ETF} at 2nd order, for comparison with results in Bartel and Bencheikh (2002)

Variables

- integer `test::file_unit`
- integer `test::file_unit_2`
Unit numbers for opening files.
- integer `test::sample_profile`
Test density profile to use.

Index

- a_q
 - parameters, [61](#)
- allocate_arrays
 - routines, [66](#)
- alpha
 - Extra parameters for BSk forces, [18](#)
- b1
 - Skyrme coefficients, [16](#)
- b10
 - Skyrme coefficients, [16](#)
- b11
 - Skyrme coefficients, [16](#)
- b12
 - Skyrme coefficients, [16](#)
- b13
 - Skyrme coefficients, [16](#)
- b2
 - Skyrme coefficients, [17](#)
- b3
 - Skyrme coefficients, [17](#)
- b4
 - Skyrme coefficients, [17](#)
- b5
 - Skyrme coefficients, [17](#)
- b6
 - Skyrme coefficients, [17](#)
- b7
 - Skyrme coefficients, [17](#)
- b8
 - Skyrme coefficients, [17](#)
- b9
 - Skyrme coefficients, [17](#)
- bartel_bencheikh_benchmark
 - test, [102](#)
- bcs_protons
 - pairing, [49](#)
- beta
 - Extra parameters for BSk forces, [18](#)
- boundary
 - rspace, [96](#)
- calc_bcs_num_p
 - pairing, [50](#)
- calc_chem_pots
 - Input parameters, [33](#)
- calc_div_j_2_q
 - routines, [66](#)
- calc_div_j_4_q
 - routines, [67](#)
- calc_e_f_q
 - routines, [68](#)
- calc_e_pair
 - pairing, [51](#)
- calc_e_sc
 - strutinsky, [98](#)
- calc_e_sc_pair
 - strutinsky, [98](#)
- calc_f_q
 - routines, [69](#)
- calc_gap
 - pairing, [51](#)
- calc_j_2_q
 - routines, [70](#)
- calc_j_4_q
 - routines, [70](#)
- calc_particle_number
 - routines, [71](#)
- calc_rho_q
 - routines, [72](#)
- calc_tau_2_l
 - routines, [73](#)
- calc_tau_2_nl
 - routines, [73](#)
- calc_tau_4_no_spin
 - routines, [74](#)
- calc_tau_4_so
 - routines, [75](#)
- calc_tau_tf
 - routines, [76](#)
- calc_u_q
 - routines, [77](#)
- calc_w_q
 - routines, [78](#)
- charge
 - routines, [78](#)
- coulomb_on
 - Input parameters, [33](#)
- d1
 - routines, [79](#)
- d1_a_q
 - parameters, [62](#)
- d2
 - routines, [80](#)
- d2_a_q
 - parameters, [62](#)
- d2_f_q
 - parameters, [62](#)
- d2_rho_q

- parameters, 62
- d2hmen
 - Variables for use in strutinsky module, 42
- d3
 - routines, 80
- d3_a_q
 - parameters, 62
- d3_f_q
 - parameters, 62
- d3_rho_q
 - parameters, 62
- d4
 - routines, 81
- d4_a_q
 - parameters, 62
- d4_f_q
 - parameters, 63
- d4_rho_q
 - parameters, 63
- deallocate_arrays
 - routines, 82
- deallocate_mesh
 - routines, 82
- del2_f_q
 - Global effective mass arrays, 22
- del2_rho_q
 - Global density arrays, 21
- del2_rho_t
 - Global density arrays, 21
- del_f_q
 - Global effective mass arrays, 22
- del_j_q
 - Global arrays for other densities and fields, 23
- del_j_t
 - Global arrays for other densities and fields, 23
- del_rho_q
 - Global density arrays, 21
- del_rho_t
 - Global density arrays, 21
- delta_mu
 - Variables for storing properties of Wigner-Seitz cell, 28
- delta_n
 - Variables for use in pairing module, 45
- delta_p
 - Variables for use in pairing module, 45
- delta_parametric
 - pairing, 52
- dhmen
 - Variables for use in strutinsky module, 42
- div_j_2_q
 - Global arrays for other densities and fields, 23
- div_j_4_q
 - parameters, 63
- div_j_q
 - Global arrays for other densities and fields, 24
- div_j_t
 - Global arrays for other densities and fields, 24
- div_w_q
 - Global arrays for other densities and fields, 24
- dp
 - parameters, 63
- dr
 - Input parameters, 33
- e2
 - Global parameters, 7
- e_coulomb
 - Variables for storing properties of Wigner-Seitz cell, 29
- e_coulomb_di
 - Variables for storing properties of Wigner-Seitz cell, 29
- e_coulomb_e
 - Variables for electron contributions, 48
- e_coulomb_ex
 - Variables for storing properties of Wigner-Seitz cell, 29
- e_coulomb_pe
 - Variables for electron contributions, 48
- e_density_c_di
 - Global arrays for other densities and fields, 24
- e_density_c_ex
 - Global arrays for other densities and fields, 24
- e_density_field
 - Global arrays for other densities and fields, 24
- e_density_sky
 - Global arrays for other densities and fields, 24
- e_field
 - Variables for storing properties of Wigner-Seitz cell, 29
- e_kinetic_e
 - Variables for electron contributions, 48
- e_kinetic_q
 - Variables for storing properties of Wigner-Seitz cell, 29
- e_kinetic_t
 - Variables for storing properties of Wigner-Seitz cell, 29
- e_pair_bcs
 - Variables for use in pairing module, 46
- e_pair_q
 - Variables for use in pairing module, 46
- e_pair_t
 - Variables for use in pairing module, 46
- e_qp_p
 - Variables for use in pairing module, 46
- e_sc_q
 - Variables for use in strutinsky module, 42
- e_sc_t
 - Variables for use in strutinsky module, 43
- e_skyme
 - Variables for storing properties of Wigner-Seitz cell, 30
- e_so
 - Variables for storing properties of Wigner-Seitz cell, 30

- e_total
 - Variables for storing properties of Wigner-Seitz cell, 30
- ecut
 - Variables for use in strutinsky module, 43
- electrons_on
 - Input parameters, 33
- emax0_mod
 - Input parameters, 34
- ep
 - Variables for use in strutinsky module, 43
- epsilon_lambda
 - Extra parameters for BSk forces, 18
- etf
 - etf.f90, 111
- etf.f90
 - etf, 111
 - total_energy_with_corrections, 113
- etf_order
 - Input parameters, 34
- eval_coulomb
 - routines, 82
- eval_electrons
 - routines, 83
- eval_fields
 - routines, 84
- eval_skyrme_energy_density
 - routines, 85
- eval_tau_etf
 - routines, 86
- eval_u_q
 - routines, 87
- eval_ws_quantities
 - routines, 88
- External profiles, 39
 - n_profile, 39
 - num_n, 39
 - num_p, 39
 - p_profile, 39
 - r_ws, 39
- Extra constants for calculating $\frac{\hbar^2}{2m}f$ for BSk forces, 9
 - mamuc2, 9
 - rydb, 9
 - xmh, 9
 - xmn, 9
 - xmp, 9
- Extra parameters for BSk forces, 18
 - alpha, 18
 - beta, 18
 - epsilon_lambda, 18
 - fn_neg, 18
 - fn_pos, 18
 - fp_neg, 19
 - fp_pos, 19
 - gamma, 19
 - t4, 19
 - t5, 19
 - x4, 19
 - x5, 19
- extrapolate_back_3
 - routines, 90
- f_q
 - Global effective mass arrays, 22
- file_unit
 - test, 110
- file_unit_0
 - routines, 95
- file_unit_1
 - routines, 95
- file_unit_2
 - test, 110
- fn_neg
 - Extra parameters for BSk forces, 18
- fn_pos
 - Extra parameters for BSk forces, 18
- force
 - Input parameters, 34
- force_initialise
 - parameters, 61
- force_string
 - Global strings for printing output, 41
- Format statements, 40
 - info_format, 40
 - n_floats, 40
- fp_neg
 - Extra parameters for BSk forces, 19
- fp_pos
 - Extra parameters for BSk forces, 19
- fr12
 - Numerical fractions, 10
- fr13
 - Numerical fractions, 10
- fr14
 - Numerical fractions, 10
- fr16
 - Numerical fractions, 10
- fr18
 - Numerical fractions, 11
- fr1_12
 - Numerical fractions, 11
- fr1_16
 - Numerical fractions, 11
- fr1_24
 - Numerical fractions, 11
- fr1_32
 - Numerical fractions, 11
- fr1_36
 - Numerical fractions, 11
- fr23
 - Numerical fractions, 11
- fr260_3
 - Numerical fractions, 11
- fr29
 - Numerical fractions, 12
- fr32

- Numerical fractions, [12](#)
- fr34
 - Numerical fractions, [12](#)
- fr35
 - Numerical fractions, [12](#)
- fr38
 - Numerical fractions, [12](#)
- fr3_10
 - Numerical fractions, [12](#)
- fr43
 - Numerical fractions, [12](#)
- fr53
 - Numerical fractions, [12](#)
- fr54
 - Numerical fractions, [13](#)
- fr83
 - Numerical fractions, [13](#)
- gamma
 - Extra parameters for BSk forces, [19](#)
- Global arrays for densities and fields for 4th-order terms, [27](#)
- Global arrays for other densities and fields, [23](#)
 - del_j_q, [23](#)
 - del_j_t, [23](#)
 - div_j_2_q, [23](#)
 - div_j_q, [24](#)
 - div_j_t, [24](#)
 - div_w_q, [24](#)
 - e_density_c_di, [24](#)
 - e_density_c_ex, [24](#)
 - e_density_field, [24](#)
 - e_density_sky, [24](#)
 - j_2_q, [24](#)
 - j_q, [25](#)
 - j_t, [25](#)
 - rho_ch, [25](#)
 - tau_2_cont_q, [25](#)
 - tau_2_l_q, [25](#)
 - tau_2_nl_q, [25](#)
 - tau_etf_q, [25](#)
 - tau_etf_t, [26](#)
 - tau_tf_q, [26](#)
 - u_q, [26](#)
 - v_c_di, [26](#)
 - v_c_ex, [26](#)
 - v_c_pe, [26](#)
 - w_q, [26](#)
- Global density arrays, [21](#)
 - del2_rho_q, [21](#)
 - del2_rho_t, [21](#)
 - del_rho_q, [21](#)
 - del_rho_t, [21](#)
 - rho_q, [21](#)
 - rho_t, [21](#)
- Global effective mass arrays, [22](#)
 - del2_f_q, [22](#)
 - del_f_q, [22](#)
 - f_q, [22](#)
- Global parameters, [7](#)
 - e2, [7](#)
 - hbar2_2m, [7](#)
 - hbarc, [8](#)
 - mec2, [8](#)
 - mnc2, [8](#)
 - mpc2, [8](#)
 - pi, [8](#)
 - rho0, [8](#)
- Global strings for printing output, [41](#)
 - force_string, [41](#)
 - iso_string, [41](#)
 - line_break, [41](#)
 - space4, [41](#)
 - table_string, [41](#)
- hb2m
 - Variables for use in strutinsky module, [43](#)
- hbar2_2m
 - Global parameters, [7](#)
- hbar2_2m_q
 - Skyrme parameters, [14](#)
- hbarc
 - Global parameters, [8](#)
- info_format
 - Format statements, [40](#)
- initialise_uniform_mesh
 - routines, [90](#)
- Input parameters, [32](#)
 - calc_chem_pots, [33](#)
 - coulomb_on, [33](#)
 - dr, [33](#)
 - electrons_on, [33](#)
 - emax0_mod, [34](#)
 - etf_order, [34](#)
 - force, [34](#)
 - lmax, [34](#)
 - n, [34](#)
 - neutron_pairing, [34](#)
 - nmaxstate, [35](#)
 - pair_alpha, [35](#)
 - pair_del_init, [35](#)
 - pair_dk, [35](#)
 - pair_eta, [35](#)
 - pair_k_max, [35](#)
 - pair_max_iters, [36](#)
 - pair_mix, [36](#)
 - pair_num_tol, [36](#)
 - pair_qp_cut, [36](#)
 - pair_tol, [36](#)
 - pair_v0, [36](#)
 - profile_r_max, [37](#)
 - proton_bcs, [37](#)
 - r_max, [37](#)
 - run_mode, [37](#)
 - smooth_qp_cut, [37](#)
 - specify_n, [37](#)
 - strut_r_max, [38](#)

- strutinsky_on, 38
 - verbose, 38
- interazchamelanalyt
 - pairing, 53
- iso_string
 - Global strings for printing output, 41
- j2_terms
 - Skyrme parameters, 14
- j_2_q
 - Global arrays for other densities and fields, 24
- j_4_q
 - parameters, 63
- j_q
 - Global arrays for other densities and fields, 25
- j_t
 - Global arrays for other densities and fields, 25
- jjp
 - Variables for use in strutinsky module, 43
- lap_sphe_symm
 - routines, 90
- line_break
 - Global strings for printing output, 41
- lmbda
 - pairing, 54
- lmax
 - Input parameters, 34
- lp
 - Variables for use in strutinsky module, 43
- mamuc2
 - Extra constants for calculating $\frac{\hbar^2}{2m}$ for BSk forces, 9
- mec2
 - Global parameters, 8
- Mesh variable, 20
 - r, 20
- mnc2
 - Global parameters, 8
- mpc2
 - Global parameters, 8
- mu_c
 - Variables for storing properties of Wigner-Seitz cell, 30
- mu_e
 - Variables for storing properties of Wigner-Seitz cell, 30
- mu_q
 - Variables for storing properties of Wigner-Seitz cell, 30
- n
 - Input parameters, 34
- n_floats
 - Format statements, 40
- n_profile
 - External profiles, 39
- n_q
 - Variables for storing properties of Wigner-Seitz cell, 31
- n_t
 - Variables for storing properties of Wigner-Seitz cell, 31
- neutron_pairing
 - Input parameters, 34
- nmaxstate
 - Input parameters, 35
- nnst
 - Variables for use in strutinsky module, 44
- num_n
 - External profiles, 39
- num_p
 - External profiles, 39
- num_p_bcs
 - Variables for use in pairing module, 46
- Numerical fractions, 10
 - fr12, 10
 - fr13, 10
 - fr14, 10
 - fr16, 10
 - fr18, 11
 - fr1_12, 11
 - fr1_16, 11
 - fr1_24, 11
 - fr1_32, 11
 - fr1_36, 11
 - fr23, 11
 - fr260_3, 11
 - fr29, 12
 - fr32, 12
 - fr34, 12
 - fr35, 12
 - fr38, 12
 - fr3_10, 12
 - fr43, 12
 - fr53, 12
 - fr54, 13
 - fr83, 13
- numerov
 - rspace, 96
- occ_u
 - Variables for use in pairing module, 46
- occ_v
 - Variables for use in pairing module, 47
- orders_kinetic_densities
 - test, 102
- p_profile
 - External profiles, 39
- pair_alpha
 - Input parameters, 35
- pair_del_init
 - Input parameters, 35
- pair_dk
 - Input parameters, 35
- pair_eta

- Input parameters, 35
- pair_gap_n
 - Variables for use in pairing module, 47
- pair_gap_p
 - Variables for use in pairing module, 47
- pair_k_max
 - Input parameters, 35
- pair_max_iters
 - Input parameters, 36
- pair_mix
 - Input parameters, 36
- pair_num_tol
 - Input parameters, 36
- pair_qp_cut
 - Input parameters, 36
- pair_tol
 - Input parameters, 36
- pair_v0
 - Input parameters, 36
- pairing, 49
 - bcs_protons, 49
 - calc_bcs_num_p, 50
 - calc_e_pair, 51
 - calc_gap, 51
 - delta_parametric, 52
 - interazchamelanalyt, 53
 - llambda, 54
- parameters, 55
 - a_q, 61
 - d1_a_q, 62
 - d2_a_q, 62
 - d2_f_q, 62
 - d2_rho_q, 62
 - d3_a_q, 62
 - d3_f_q, 62
 - d3_rho_q, 62
 - d4_a_q, 62
 - d4_f_q, 63
 - d4_rho_q, 63
 - div_j_4_q, 63
 - dp, 63
 - force_initialise, 61
 - j_4_q, 63
 - tau_4_cont_q, 63
 - tau_4_no_spin_q, 63
 - tau_4_so_q, 64
- pi
 - Global parameters, 8
- pressure
 - routines, 91
- pressure_e
 - Variables for storing properties of Wigner-Seitz cell, 31
- pressure_ex
 - Variables for storing properties of Wigner-Seitz cell, 31
- pressure_nucl
 - Variables for storing properties of Wigner-Seitz cell, 31
- pressure_t
 - Variables for storing properties of Wigner-Seitz cell, 31
- profile_r_max
 - Input parameters, 37
- proton_bcs
 - Input parameters, 37
- ps
 - Variables for use in strutinsky module, 44
- r
 - Mesh variable, 20
- r_max
 - Input parameters, 37
- r_ws
 - External profiles, 39
- read_test_params
 - test, 103
- rho0
 - Global parameters, 8
- rho_anom_p_bcs
 - Variables for use in pairing module, 47
- rho_ch
 - Global arrays for other densities and fields, 25
- rho_e
 - Variables for electron contributions, 48
- rho_p_bcs
 - Variables for use in pairing module, 47
- rho_q
 - Global density arrays, 21
- rho_t
 - Global density arrays, 21
- routines, 64
 - allocate_arrays, 66
 - calc_div_j_2_q, 66
 - calc_div_j_4_q, 67
 - calc_e_f_q, 68
 - calc_f_q, 69
 - calc_j_2_q, 70
 - calc_j_4_q, 70
 - calc_particle_number, 71
 - calc_rho_q, 72
 - calc_tau_2_l, 73
 - calc_tau_2_nl, 73
 - calc_tau_4_no_spin, 74
 - calc_tau_4_so, 75
 - calc_tau_tf, 76
 - calc_u_q, 77
 - calc_w_q, 78
 - charge, 78
 - d1, 79
 - d2, 80
 - d3, 80
 - d4, 81
 - deallocate_arrays, 82
 - deallocate_mesh, 82
 - eval_coulomb, 82

- eval_electrons, 83
- eval_fields, 84
- eval_skyrme_energy_density, 85
- eval_tau_etf, 86
- eval_u_q, 87
- eval_ws_quantities, 88
- extrapolate_back_3, 90
- file_unit_0, 95
- file_unit_1, 95
- initialise_uniform_mesh, 90
- lap_sphe_symm, 90
- pressure, 91
- str2int, 92
- write_densities, 92
- write_fields, 93
- write_sp_states_p, 93
- ws_integral, 94
- rspace, 95
 - boundary, 96
 - numerov, 96
- run_mode
 - Input parameters, 37
- rydb
 - Extra constants for calculating $\frac{\hbar^2}{2m}f$ for BSk forces, 9
- sample_profile
 - test, 110
- sigma
 - Skyrme parameters, 14
- Skyrme coefficients, 16
 - b1, 16
 - b10, 16
 - b11, 16
 - b12, 16
 - b13, 16
 - b2, 17
 - b3, 17
 - b4, 17
 - b5, 17
 - b6, 17
 - b7, 17
 - b8, 17
 - b9, 17
- Skyrme parameters, 14
 - $\hbar^2_{2m_q}$, 14
 - j2_terms, 14
 - sigma, 14
 - t0, 14
 - t1, 15
 - t2, 15
 - t3, 15
 - w0, 15
 - x0, 15
 - x1, 15
 - x2, 15
 - x3, 15
- smooth_qp_cut
 - Input parameters, 37
- sp_states_deallocate
 - strutinsky, 99
- sp_states_setup
 - strutinsky, 99
- space4
 - Global strings for printing output, 41
- specify_n
 - Input parameters, 37
- srcs/boundary.f90, 111
- srcs/etf.f90, 111
- srcs/pairing.f90, 115
- srcs/parameters.f90, 115
- srcs/routines.f90, 122
- srcs/strutinsky.f90, 124
- srcs/test.f90, 124
- state_sort
 - strutinsky, 100
- str2int
 - routines, 92
- strength_bcs
 - Variables for use in pairing module, 47
- strut_n
 - Variables for use in strutinsky module, 44
- strut_r_max
 - Input parameters, 38
- strutinsky, 97
 - calc_e_sc, 98
 - calc_e_sc_pair, 98
 - sp_states_deallocate, 99
 - sp_states_setup, 99
 - state_sort, 100
- strutinsky_on
 - Input parameters, 38
- t0
 - Skyrme parameters, 14
- t1
 - Skyrme parameters, 15
- t2
 - Skyrme parameters, 15
- t3
 - Skyrme parameters, 15
- t4
 - Extra parameters for BSk forces, 19
- t5
 - Extra parameters for BSk forces, 19
- table_string
 - Global strings for printing output, 41
- tau_2_cont_q
 - Global arrays for other densities and fields, 25
- tau_2_l_q
 - Global arrays for other densities and fields, 25
- tau_2_nl_q
 - Global arrays for other densities and fields, 25
- tau_4_cont_q
 - parameters, 63
- tau_4_no_spin_q
 - parameters, 63
- tau_4_so_q

- parameters, 64
- tau_eff_q
 - Global arrays for other densities and fields, 25
- tau_eff_t
 - Global arrays for other densities and fields, 26
- tau_tf_q
 - Global arrays for other densities and fields, 26
- test, 101
 - bartel_bencheikh_benchmark, 102
 - file_unit, 110
 - file_unit_2, 110
 - orders_kinetic_densities, 102
 - read_test_params, 103
 - sample_profile, 110
 - test_calc_coulomb_energy, 103
 - test_calc_particle_number, 104
 - test_calc_skymrme_energy, 104
 - test_central_potentials, 105
 - test_densities, 105
 - test_density_derivs, 106
 - test_eff_mass, 106
 - test_kinetic_densities, 107
 - test_si_correction, 107
 - test_sp_energies, 108
 - test_spin_current_densities, 109
 - write_ws_cell_array_quantity, 109
- test_calc_coulomb_energy
 - test, 103
- test_calc_particle_number
 - test, 104
- test_calc_skymrme_energy
 - test, 104
- test_central_potentials
 - test, 105
- test_densities
 - test, 105
- test_density_derivs
 - test, 106
- test_eff_mass
 - test, 106
- test_kinetic_densities
 - test, 107
- test_si_correction
 - test, 107
- test_sp_energies
 - test, 108
- test_spin_current_densities
 - test, 109
- total_energy_with_corrections
 - etf.f90, 113
- u_q
 - Global arrays for other densities and fields, 26
- v_c_di
 - Global arrays for other densities and fields, 26
- v_c_ex
 - Global arrays for other densities and fields, 26
- v_c_pe
 - Global arrays for other densities and fields, 26
- Variables for electron contributions, 48
 - e_coulomb_e, 48
 - e_coulomb_pe, 48
 - e_kinetic_e, 48
 - rho_e, 48
- Variables for storing properties of Wigner-Seitz cell, 28
 - delta_mu, 28
 - e_coulomb, 29
 - e_coulomb_di, 29
 - e_coulomb_ex, 29
 - e_field, 29
 - e_kinetic_q, 29
 - e_kinetic_t, 29
 - e_skymrme, 30
 - e_so, 30
 - e_total, 30
 - mu_c, 30
 - mu_e, 30
 - mu_q, 30
 - n_q, 31
 - n_t, 31
 - pressure_e, 31
 - pressure_ex, 31
 - pressure_nucl, 31
 - pressure_t, 31
- Variables for use in pairing module, 45
 - delta_n, 45
 - delta_p, 45
 - e_pair_bcs, 46
 - e_pair_q, 46
 - e_pair_t, 46
 - e_qp_p, 46
 - num_p_bcs, 46
 - occ_u, 46
 - occ_v, 47
 - pair_gap_n, 47
 - pair_gap_p, 47
 - rho_anom_p_bcs, 47
 - rho_p_bcs, 47
 - strength_bcs, 47
- Variables for use in strutinsky module, 42
 - d2hmen, 42
 - dhmen, 42
 - e_sc_q, 42
 - e_sc_t, 43
 - ecut, 43
 - ep, 43
 - hb2m, 43
 - jip, 43
 - lp, 43
 - nnst, 44
 - ps, 44
 - strut_n, 44
 - vpot, 44
 - vso, 44
- verbose
 - Input parameters, 38

- vpot
 - Variables for use in strutinsky module, [44](#)
- vso
 - Variables for use in strutinsky module, [44](#)
- w0
 - Skyrme parameters, [15](#)
- w_q
 - Global arrays for other densities and fields, [26](#)
- write_densities
 - routines, [92](#)
- write_fields
 - routines, [93](#)
- write_sp_states_p
 - routines, [93](#)
- write_ws_cell_array_quantity
 - test, [109](#)
- ws_integral
 - routines, [94](#)
- x0
 - Skyrme parameters, [15](#)
- x1
 - Skyrme parameters, [15](#)
- x2
 - Skyrme parameters, [15](#)
- x3
 - Skyrme parameters, [15](#)
- x4
 - Extra parameters for BSk forces, [19](#)
- x5
 - Extra parameters for BSk forces, [19](#)
- xmh
 - Extra constants for calculating $\frac{\hbar^2}{2m}$ for BSk forces, [9](#)
- xmn
 - Extra constants for calculating $\frac{\hbar^2}{2m}$ for BSk forces, [9](#)
- xmp
 - Extra constants for calculating $\frac{\hbar^2}{2m}$ for BSk forces, [9](#)