First of all, to answer your question, the result output by Gromacs ***g_dos*** command is the result **delta_Cv_qm** that we are mentioned in the paper. The caution is that this result is molecule number based, thus we need to transfer it to system number based. The way of the transformation is a simple math calculation, which is done by the multiplication by the number of molecules in one system.

Apparently, this roughly answered your question but may not be the one you want, neither of me, let me explain it in detail, both from the theory and from the programming.

For the considerations of your convenience and the simplification of this document, I have attached three different literatures, they are;

1. *Thermodynamics and quantum corrections from molecular dynamics for liquid water*
2. *The two-phase model for calculating thermodynamic properties of liquids from molecular dynamics: Validation for the phase diagram of Lennard-Jones fluids*
3. *Statistical mechanics of fluids*

All the references in this writing will not be provided unless explicitly mentioned, because they either can be searched by names, or found in the three provided literatures or in their citations.

For the real-world molar heat capacity, it has two different definitions, C_P, heat capacity at constant pressure; C_V, heat capacity at constant volume, they have the following relation;

$$C_P - C_V = VT\frac{\alpha_P^2}{K_T}$$

where V is the molecular volume, T the is absolute temperature, α_P is the volumetric thermal expansion coefficient, and K_T is the isothermal compressibility. The last two terms can be determined experimentally, or they can be computed from the fluctuations in enthalpy H and volume;

$$< \delta V \delta H > = k_B T^2 < V > \alpha_P$$
$$< \delta V^2 > = k_B T < V > K_T$$

where k_B is the Boltzmann constant, <> means ensemble average.

In classical Molecular Dynamic (**MD**) or Monto Carlo (**MC**) simulation, either using polarizable or non-polarizable force field, as long as it doesn't take care of the infinite parametrization at the zero radius, in general, they are called the **hard-sphere** or **hard-core** simulation, in this sense, a very strong repulsion will be generated when atoms or the spherical molecules approach at a very close distance, which is a counterpart of **soft-sphere** or **soft-core** simulation.

In our paper;

> *we used the non-polarizable OPLS-AA force field formalism, the total energies for each ionic liquid system are evaluated as a sum of individual energies for the harmonic bond stretching and angle bending terms, a cosine series for torsional energetics, and Coulomb and 12-6 Lennard-Jones terms for the nonbonded interactions.*

Our simulation is the classical **MD** simulation. In this scenario, for thermodynamics properties C_V and C_P, they can be got from constant volume or constant pressure **MD** simulation through the fluctuations in energy and enthalpy;

$$C_V^{class} = \frac{< \delta U^2 >}{k_B T^2}$$

$$C_P^{class} = \frac{< \delta H^2 >}{k_B T^2}$$

However, this method will greatly overestimate C_P, due to the fact that a large part of C_P comes from vibrations that to some extent are quantum mechanical.

One way to address this issue is to use the two-phase thermodynamic method (**2PT**), which considers the distribution of normal modes, or in specific, density of states (**DoS**), as a function of normal-mode frequency *v*. **DoS** can be calculated from the Fourier Transform (**FT**) of the mass-weighted atomic velocity autocorrelation functions (**MVACF**).

By this chance, let me also clarify some different definitions where a lot of literatures seldom touch.

For a system, in **3D** space, for the velocity autocorrelation function (**VACF**), we have the definition;

$$c(t) = \sum_{j=1}^{N} \sum_{\beta=1}^{3} v_j(t)\, v_j(t+\tau)$$

where β enumerates all three dimensions, j goes through the total atom number N, v corresponds to atom velocity, t and τ means the time and time interval.

The prefix mass-weighted means the **normalization**, which is performed on the **VACF**.

Then the full formula of the **atomic-mass-weighted-velocity-autocorrelation-function** is;

$$C(t) = \frac{\sum_{j=1}^{N}\sum_{\beta=1}^{3} m_j v_j(t)\; v_j(t+\tau)}{\sum_{j=1}^{N}\sum_{\beta=1}^{3} v_j(t)\; v_j(t+\tau)}$$

In the same way, the **atomic-momentum-autocorrelation-function** is defined as;

$$C_m(t) = \sum_{j=1}^{N}\sum_{\beta=1}^{3} m_j v_j(t)\; m_j v_j(t+\tau)$$

Since the **ACF** definitions are on the atom level, similarly, the **molecular-velocity-autocorrelation-function** is defined as;

$$F(t) = \sum_{\alpha=1}^{M}\sum_{\beta=1}^{3} m_\alpha v_\alpha(t)\; v_\alpha(t+\tau)$$

Here m_α is the molecule weight;

$$v_\alpha(t) = \frac{\sum_{j=1}^{N}\sum_{\beta=1}^{3} m_j v_j(t)}{m_\alpha}$$

Then the **molecular-momentum-autocorrelation-function** is;

$$F_m(t) = \sum_{\alpha=1}^{M}\sum_{\beta=1}^{3} m_\alpha v_\alpha(t)\;\; m_\alpha v_\alpha(t+\tau)$$

One notation is that for the molecule-based **ACF** definition, the **periodic-boundary-condition** should be considered. For **ACF**-type definition, all the normalization should be based on the **VACF**.

The **Density of state** (**DoS**) function is defined as the distribution of vibrational normal modes;

$$S(v) = \frac{2}{k_B T}\sum_{j=1}^{N}\sum_{\beta=1}^{3} m_j s_j^{\beta}(v)$$

where, N is the total number of atoms, β is the different Cartesian coordinate, m_j is the mass of atom j; the spectral density $s_j^{\beta}(v)$ is determined from the square of the **FT** of the velocities;

$$s_j^{\beta}(v) = \lim_{\tau\to\infty}\frac{1}{2\tau}\left|\int_{-\tau}^{\tau} v_j^{\beta}(t) e^{-i2\pi v t} dt\right|^2$$

The mass-weighted velocity autocorrelation function is defined;

$$C(t) = \sum_{j=1}^{N} \sum_{\beta=1}^{3} m_j c(t)$$

where *c(t)* is the **VACF** which is already defined at top.

By using **Wiener-Khitchine theorem**, finally we can get;

$$s_j^k(v) = \frac{2}{k_B T} \lim_{\tau \to \infty} \int_{-\tau}^{\tau} c(t) e^{-i2\pi vt} dt$$

$$S(v) = \frac{2}{k_B T} \lim_{\tau \to \infty} \int_{-\tau}^{\tau} C(t) e^{-i2\pi vt} dt$$

Note that the velocity spectrum S(v) can be computed separately for different subsets of atoms (e. g., different elements, different chemical environments of the same element, or different molecules) and the velocity spectrum S(v) can then be computed as a sum of the effects from these different subsets of atoms. Thus, as we will see, the quantum corrections also can be partitioned among the different subsets of atoms. Even though once the dynamics, i. e., the set of velocities {(v/t)}, is determined, the quantum corrections may be computed separately for different subsets of atoms, it should be remembered that normally all atoms together contribute to determining the dynamics.

The **DoS** at zero frequency ***S(0)*** is related to the self-diffusion coefficient **D** in pure fluids.

$$\mathrm{D} = \frac{S(0) k_B T}{12 m N}$$

To compute the free energy of a given system, the **DoS** should be properly partitioned to different terms like an ideal gas, a hard sphere liquid, or a harmonic solid, especially for the fluid calculation, because a fluid of hard spheres will present a phase transition to a gas phase or to a solid phase, so it comes the theory called the two-phase thermodynamic (**2PT**) model.

Here I wouldn't go much detail, as a result, the **DoS** is divided into two different parts, the **gaslike** part and the **solidlike** part.

$$S(v) = S^g(v) + S^s(v)$$

Additionally, considering the contributions of **degree of freedom** of different parts, and in combination with different weighting functions, the quantum mechanical correction to the classical heat capacity can then be integrated by;

$$\delta C_V^{qm} = k_B \int (W(v) - 1) DoS(v) dv$$

It has another consideration, the hydrogen bond constrains are often employed during **MD** process, while the quantum correction is performed on the full freedom, so the harmonic bonded potential term $\mathbf{N_c k_B}$ should be added back to the C_P^{class}, as discussed in our paper, in equation 9.

As a conclusion, here is a list of all parameters defined in the **2PT** model for **DoS** calculation.

Zero frequency value;

$$S^{HS}(0) = s_0$$

Gas component;

$$S^g(v) = \frac{s_0}{1 + [\frac{\pi s_0 v}{6fN}]^2}$$

where f means the **fluidicity**, defined in the equation;

$$2\Delta^{-4.5} f^{7.5} - 6\Delta^{-3} f^5 - \Delta^{-1.5} f^{3.5} + 6\Delta^{-1.5} f^{2.5} + 2f - 2 = 0$$

(Note, the equation in Dr. Lin's paper has a typo, the equation in here is the correct one)

where the Δ means the normalized diffusivity constant;

$$\Delta = \frac{2s_0}{9N} (\frac{\pi k_B T}{m})^{1/2} \rho^{1/3} (\frac{6}{\pi})^{2/3}$$

where ρ is the density of system.

So,

$$D_0^{HS} = \frac{D}{f} = \frac{3}{8} \frac{1}{\rho(\sigma^{HS})^2} (\frac{k_B T}{\pi m})^{1/2}$$

where **D** is already defined in top, self-diffusivity; the term D_0^{HS} is so called the hard-sphere diffusivity; σ^{HS} is defined as the hard-sphere diameter.

Then the **hard-sphere packing fraction** is defined as;

$$y = \frac{\pi}{6} \rho (\sigma^{HS})^3$$

Then the **hard-sphere compressibility** is defined as;

$$z = \frac{1 + y + y^2 - y^3}{(1 - y)^3}$$

For the gas component, the diffusivity is defined as;

$$D_0^g = \frac{1}{3} \frac{k_B T s_0}{12 m f N}$$

For two different phases, the weighting functions are defined as;

$$W_E^g = 0.5$$

$$W_E^s = \frac{u^2 e^u}{(1 - e^u)^2}$$

where $u = hv/(k_B T)$.

Finally, we decompose **DoS** into gas phase part and solid phase part;

$$S^s = S - S^g$$

Separately applying them with different weighting functions and integrating them over all the **MD** simulation length, then we can get the term so called the **quantum-correction**.

$$\delta C_V^{qm} = k_B \int (S^g W_E^g + S^s W_E^s) dv$$

Cheers!

As a complement, in here I assume you know some basic **C** language grammars. I have provided a debugged ***gmx_dos.c*** module, because of some known issues for **DoS** calculation of the Gromacs version **5.0.6**, **5.0.7**, **5.1.2** and etc., you should use Gromacs version no later than **5.1.4**, here I used this version. Manually put this debugged file into the directory: *GROMACS/src/gromacs/gmxana/*; replace the old one, or to be safety make a backup and then replace it, recompile the Gromacs, use any prepared files to execute the **DoS** calculation routine command, *gmx dos –f *trr –s *tpr*, choose the residue you want to calculate, except specified output files, you will also get four additional files, and in those files, they are exactly showing the calculation process.

As an explicitly example, the following is a collection of this debugged ***gmx_dos.c*** module file outputs. The calculated system is the water molecule, **spc** model, the used *tpr* and *trr* files are also provided.

invNormalize = 0.252023 dt = 0.025 DIM = 3 beta = 0.403395

gnx = 90 bfac = 0.0403395 =8*dt*beta/2 XX = 0 YY = 1 ZZ = 2 Natom = 30

the_total_dos = square(dos_real) + square(dos_image) = 3.67E+06

	dos[MVACF]--fft								dos[DOS]*bfac
Frames	dos[VACF]	dos[MVACF]	dos[VACF]/dos[VACF][0]	Flag	dos[DOS]	dos[MVACF]/dos[VACF][0]	State-Choose	nu[j]	dos[DOS][j]
0	7.93577	373.375	1	real	240.514	47.04962467	Real	0	9.70222
1	-1.67148	7.47641	-0.210626064	image	0	0.942115258			
2	-1.23934	-53.0096	-0.156171361	real	287.033	-6.679830691	Real	0.8	11.5788
3	-0.534138	-42.8218	-0.067307646	image	50.8124	-5.396048525			
4	0.310458	2.29906	0.039121346	real	384.559	0.289708497	Real	1.6	15.5129
5	0.83817	26.6242	0.10561924	image	103.787	3.354961144			
6	-0.310505	-15.703	-0.039127268	real	370.743	-1.978761985	Real	2.4	14.9556
7	-0.614827	-27.4958	-0.077475406	image	82.1363	-3.464792956			
8	-0.124708	-11.6067	-0.015714669	real	385.114	-1.462580191	Real	3.2	15.5353
9	0.224516	-2.2241	0.028291647	image	135.402	-0.280262659			
10	-0.10422	-7.57953	-0.013132941	real	259.326	-0.955109586	Real	4	10.4611
11	-0.0221411	-0.685313	-0.002790038	image	36.2158	-0.086357468			
12	-0.0926096	-3.37633	-0.011669895	real	348.024	-0.425457139	Real	4.8	14.0391
13	-0.209344	-3.36084	-0.026379797	image	54.8701	-0.423505218			
14	0.138607	6.32088	0.017466106	real	324.543	0.796504939	Real	5.6	13.0919
15	0.189413	1.35074	0.023868257	image	98.8537	0.170209066			
16	-0.0239826	-8.04174	-0.003022089	real	426.186	-1.013353462	Real	6.4	17.1921
17	-0.18068	-8.15936	-0.022767797	image	53.0407	-1.02817496			
18	0.0451742	3.7611	0.005692478	real	396.287	0.473942667	Real	7.2	15.986
19	-0.161054	-2.79664	-0.020294691	image	48.9287	-0.352409407			
20	0.157421	5.7769	0.01983689	real	523.288	0.727957085	Real	8	21.1092
21	-0.0105057	-1.20237	-0.001323841	image	-30.9477	-0.151512708			
22	-0.00771176	-3.53071	-0.000971772	real	578.909	-0.444910828	Real	8.8	23.3529
23	-0.00147188	1.382	-0.000185474	image	-80.939	0.174148192			
24	0.131403	13.1206	0.016558318	real	457.245	1.653349328	Real	9.6	18.4451
25	-0.0465472	7.18861	-0.005865493	image	-40.892	0.905849086			
26	0.0491006	4.83195	0.006187251	real	399.649	0.608882314	Real	10.4	16.1217
27	0.0218023	0.103212	0.002747345	image	-111.568	0.013005921			
28	0.0113962	-2.94607	0.001436055	real	376.92	-0.371239338	Real	11.2	15.2048
29	-0.107559	-12.6045	-0.013553694	image	-77.2327	-1.588314681			
30	-0.053532	-12.0136	-0.006745659	real	309.078	-1.513854358	Real	12	12.4681
31	0.0246584	-0.790238	0.003107247	image	-90.299	-0.099579247			
32	0.118982	7.17031	0.014993126	real	313.504	0.903543071	Real	12.8	12.6466
33	0.0504584	1.31128	0.00635835	image	-23.3887	0.165236644			
34	-0.163657	-4.48661	-0.020622699	real	377.717	-0.565365428	Real	13.6	15.237
35	0.00559125	2.05253	0.000704563	image	-13.0099	0.258642829			
36	0.224852	10.0533	0.028333986	real	353.258	1.26683359	Real	14.4	14.2503
37	-0.0126913	-1.22345	-0.001599252	image	-20.2969	-0.154169035			
38	-0.0483637	-6.78465	-0.006094393	real	345.799	-0.854945393	Real	15.2	13.9494
39	-0.288808	-14.2405	-0.036393192	image	-22.1912	-1.79446985			
40	-0.148606	-8.57094	-0.018726097	real	354.66	-1.080038862	Real	16	14.3068
41	0.256388	4.59472	0.032307892	image	-54.3477	0.578988554			
42	-0.0308181	-6.47307	-0.003883442	real	370.429	-0.815682662	Real	16.8	14.9429
43	-0.0537113	-9.69744	-0.006768253	image	-42.4868	-1.221991061			
44	-0.0773951	-2.79579	-0.009752689	real	381.75	-0.352302297	Real	17.6	15.3996
45	-0.0620323	11.0781	-0.007816797	image	-24.4549	1.395970397			
46	0.22866	22.0319	0.028813839	real	369.486	2.776277538	Real	18.4	14.9049
47	0.00586226	5.19752	0.000738713	image	-42.3978	0.654948417			
48	0.304764	6.76359	0.038403835	real	332.28	0.852291586	Real	19.2	13.404
49	-0.174809	-9.12913	-0.022027982	image	-12.0401	-1.150377342			

wCdiff = 0.5 wSdiff = (Shs/(3*BOLTZ)) = 0.86235 wEdiff = 0.5 wAdiff = (wEdiff-wSdiff) = -0.36235 beta = 0.403395
AMU = 1.66E-27 NANO = 1.00E-09 M_PI = 3.14159 BOLTZ = 0.00831446 Delta = 0.645974 T = 298.15 K
cP = (BOLTZ*evaluate_integral(dos[DOS_CP]) = 1.77573 PLANCK = 0.399031 ideal gas entropy = 0.0246877 Natom = 30
recip_fac = 1 Nmol = 10 hard sphere entropy = 0.0215099 fluidicity = 0.453916 dt = 0.025 ps DoS0 = 9.70222 sigma_HS = 0.202842 nm
tmass = 180.15 amu Dos2 = 3.67E+06 V = 0.222562 nm^3 DoSTot = 289.796 rho = 1344.1 g/l beta = 0.403395 mol/kJ
Diffusion coefficient from VACF 6.79769 10^-5 cm^2/s Diffusion coefficient from DoS 11.1256 10^-5 cm^2/s
hard sphere packing fraction = 0.589034 hard sphere compressibility = 24.948 Heat capacity = 1000*cP/Nmol = 177.573 J/mol K

Equation

dos[DOS_DIFF][j] = DoS0/(1+sqr(DoS0*M_PI*nu[j]/(6*f*Natom)))
dos[DOS_SOLID][j] = dos[DOS][j]-dos[DOS_DIFF][j]
dos[DOS_CP][j] = (dos[DOS_DIFF][j]*wCdiff + dos[DOS_SOLID][j]*wCsolid(nu[j] beta))
dos[DOS_S][j] = (dos[DOS_DIFF][j]*wSdiff + dos[DOS_SOLID][j]*wSsolid(nu[j] beta))
dos[DOS_A][j] = (dos[DOS_DIFF][j]*wAdiff + dos[DOS_SOLID][j]*wAsolid(nu[j] beta))
dos[DOS_E][j] = (dos[DOS_DIFF][j]*wEdiff + dos[DOS_SOLID][j]*wEsolid(nu[j] beta))

j nu[j]*recip dos[DOS]/recip_fac dos[DOS_SOLID]/recip_fac dos[DOS_DIFF]/recip_fac dos[DOS_DIFF][j] dos[DOS_SOLID][j] wCsolid(nu[j],beta) wSsolid(nu[j],beta) wAsolid(nu[j],beta) wEsolid(nu[j],beta) dos[DOS_CP][j] dos[DOS_S][j] dos[DOS_A][j] dos[DOS_E][j]

j	nu[j]	dos[DOS]	dos[DOS_SOLID]	dos[DOS_DIFF]	dos[DOS_DIFF][j]	dos[DOS_SOLID][j]	wCsolid	wSsolid	wAsolid	wEsolid	dos[DOS_CP][j]	dos[DOS_S][j]	dos[DOS_A][j]	dos[DOS_E][j]
0	0	9.70222	0	9.70222	9.70222	0	1	1	0	1	4.85111	8.36671	-3.5156	4.85111
1	0.8	11.5788	2.67006	8.90873	8.90873	2.67006	0.998619	3.05039	0.000690851	0.00138151	7.12074	15.8272	-3.22623	4.45805
2	1.6	15.5129	8.35936	7.15357	7.15357	8.35936	0.994491	2.35931	0.00276226	0.00552147	11.8901	25.8912	-2.56901	3.62294
3	2.4	14.9556	9.57034	5.38527	5.38527	9.57034	0.987655	1.95728	0.00621081	0.0124062	12.1448	23.3758	-1.89191	2.81137
4	3.2	15.5353	11.5346	4.00074	4.00074	11.5346	0.97818	1.67438	0.0110308	0.0220131	13.2833	22.7633	-1.32243	2.25428
5	4	10.4611	7.45428	3.00683	3.00683	7.45428	0.966157	1.45736	0.0172144	0.034311	8.70542	13.4565	-0.961204	1.75918
6	4.8	14.0391	11.7326	2.30649	2.30649	11.7326	0.951702	1.28245	0.0247515	0.0492601	12.3192	17.0355	-0.545357	1.7312
7	5.6	13.0919	11.2833	1.80864	1.80864	11.2833	0.934952	1.13697	0.0336301	0.0668132	11.4536	14.3884	-0.275902	1.65819
8	6.4	17.1921	15.7441	1.448	1.448	15.7441	0.916061	1.01333	0.043836	0.0869153	15.1466	17.2028	0.165477	2.09241
9	7.2	15.986	14.8049	1.1811	1.1811	14.8049	0.895201	0.906624	0.0553533	0.109505	13.8439	14.441	0.391531	2.21176
10	8	21.1092	20.1299	0.979339	0.979339	20.1299	0.872556	0.813462	0.0681642	0.134515	18.0541	17.2194	1.01727	3.19744
11	8.8	23.3529	22.5291	0.823803	0.823803	22.5291	0.848319	0.731424	0.0822495	0.161872	19.5238	17.1888	1.55451	4.05873
12	9.6	18.4451	17.7433	0.70174	0.70174	17.7433	0.822689	0.6587	0.0975883	0.191498	14.9481	12.2927	1.47727	3.74868
13	10.4	16.1217	15.5173	0.604399	0.604399	15.5173	0.795872	0.593901	0.114158	0.223312	12.652	9.73692	1.55242	3.76739
14	11.2	15.2048	14.6791	0.525651	0.525651	14.6791	0.768072	0.535933	0.131936	0.257229	11.5375	8.32033	1.74624	4.03873
15	12	12.4681	12.007	0.46112	0.46112	12.007	0.739491	0.483912	0.150897	0.293162	9.10959	6.20796	1.64473	3.75055
16	12.8	12.6466	12.239	0.407627	0.407627	12.239	0.710326	0.437115	0.171016	0.331022	8.89748	5.70136	1.94535	4.25519
17	13.6	15.237	14.8741	0.362822	0.362822	14.8741	0.680769	0.394937	0.192266	0.370719	10.3073	6.18722	2.72832	5.69554
18	14.4	14.2503	13.9253	0.324941	0.324941	13.9253	0.651003	0.356867	0.214621	0.412163	9.22791	5.24971	2.87092	5.90198
19	15.2	13.9494	13.6567	0.292642	0.292642	13.6567	0.621198	0.322468	0.238052	0.455263	8.62985	4.65621	3.14498	6.36371
20	16	14.3068	14.0419	0.264888	0.264888	14.0419	0.591516	0.29136	0.262534	0.499929	8.43847	4.31969	3.5905	7.15242
21	16.8	14.9429	14.7021	0.240873	0.240873	14.7021	0.562102	0.263213	0.288036	0.546075	8.38449	4.07749	4.14744	8.14886
22	17.6	15.3996	15.1797	0.219958	0.219958	15.1797	0.533091	0.237736	0.314531	0.593613	8.20212	3.79842	4.69477	9.12081
23	18.4	14.9049	14.7033	0.201636	0.201636	14.7033	0.504603	0.214669	0.341991	0.642458	7.52014	3.33023	4.95533	9.54706
24	19.2	13.404	13.2185	0.185497	0.185497	13.2185	0.476744	0.193785	0.370388	0.69253	6.39459	2.72151	4.82876	9.24696