Supplementary Information

RadonPy: Automated Physical Property Calculation using All-atom Classical Molecular Dynamics Simulations for Polymer Informatics

Yoshihiro Hayashi,[1,2]* Junichiro Shiomi,[1,2,3] Junko Morikawa,[1,4] Ryo Yoshida[1,5,6]*

[1] The Institute of Statistical Mathematics (ISM), Research Organization of Information and Systems, 10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan

[2] Department of Mechanical Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo 113-8656, Japan

[3] Institute of Engineering Innovation, The University of Tokyo, 2-11 Yayoi, Bunkyo-ku, Tokyo 113-0032, Japan

[4] Department of Materials Science and Engineering, School of Materials and Chemical Technology, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan

[5] Research and Services Division of Materials Data and Integrated System (MaDIS), National Institute for Materials Science (NIMS), 1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan

[6] Department of Statistical Science, School of Multidisciplinary Science, The Graduate University of Advanced Studies (SOKENDAI), 10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan

E-mail: (Y.H.) yhayashi@ism.ac.jp, (R.Y.) yoshidar@ism.ac.jp

Table of Contents

Supplementary Figures	S2–S6
Supplementary Discussion	S7–S13
Supplementary Notes	S14–S16
Supplementary Methods	S17
Supplementary References	S18

Supplementary Figures

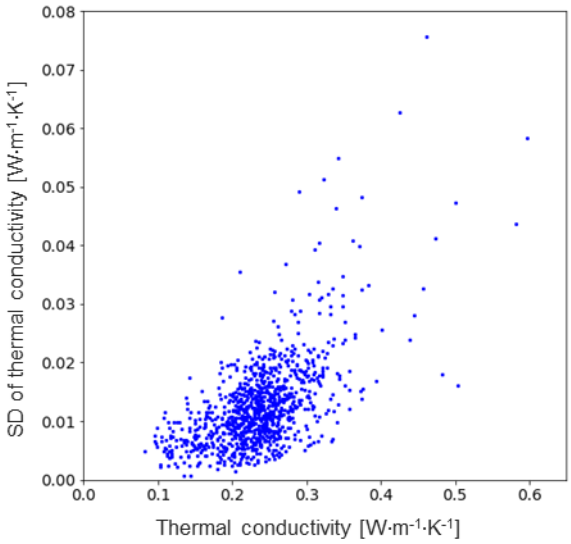
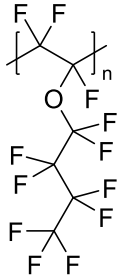
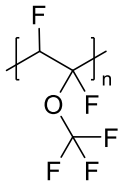


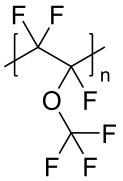
Figure 1. Relationship between thermal conductivity and its standard deviation (SD).



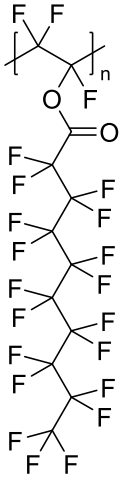
PI310 (1.780)



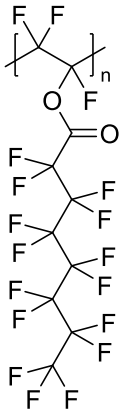
PI314 (1.750)



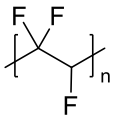
PI315 (1.816)



PI349 (1.803)



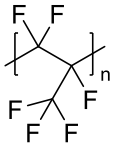
PI351 (1.766)



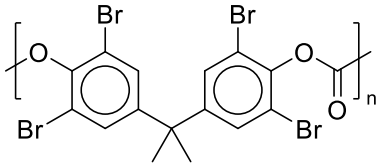
PI512 (1.799)

PI513 (1.914)

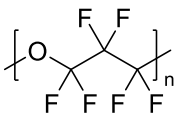
PI514 (1.892)



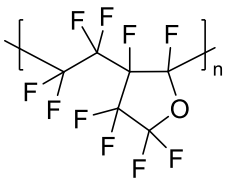
PI516 (1.791)



PI797 (1.816)

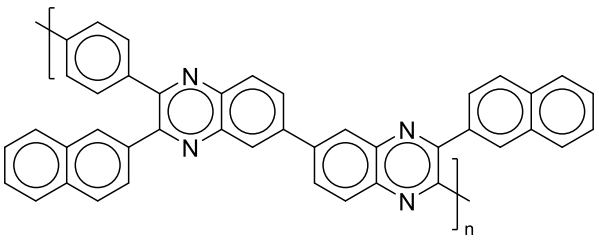


PI875 (1.851)

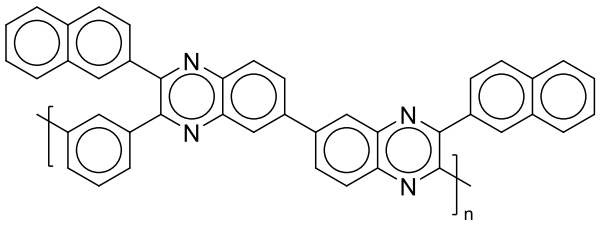


PI897 (1.822)

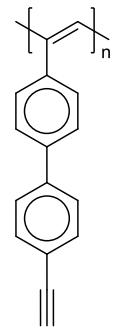
Figure 2. Chemical structure of high-density (>1.75 g·cm^{-3}) polymers and their calculated density (g cm^{-3}).



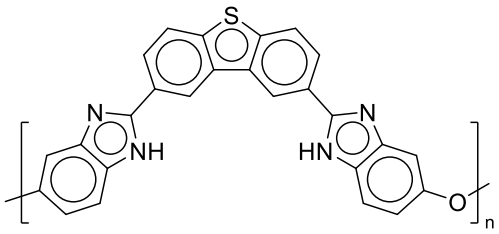
PI821 (1.776)



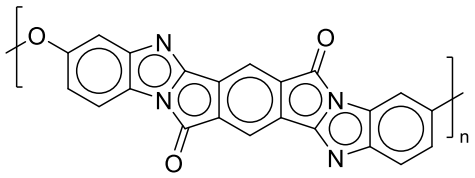
PI822 (1.799)



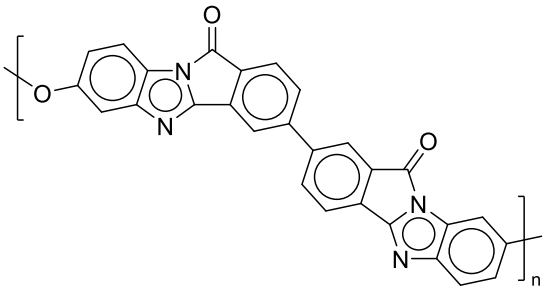
PI844 (1.765)



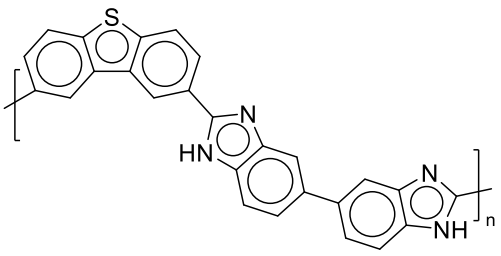
PI874 (1.820)



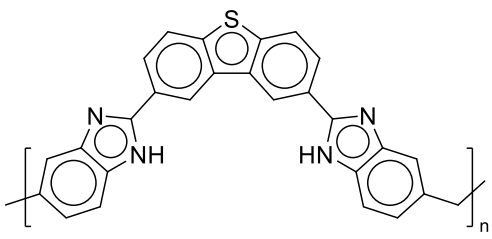
PI910 (1.804)



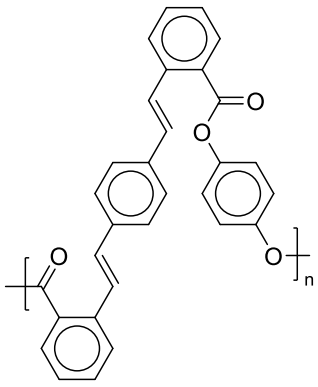
PI911 (1.839)



PI917 (1.805)



PI918 (1.808)



PI941 (1.764)

Figure 3. Chemical structure of high-refractive-index (>1.75) polymers and their calculated refractive index.

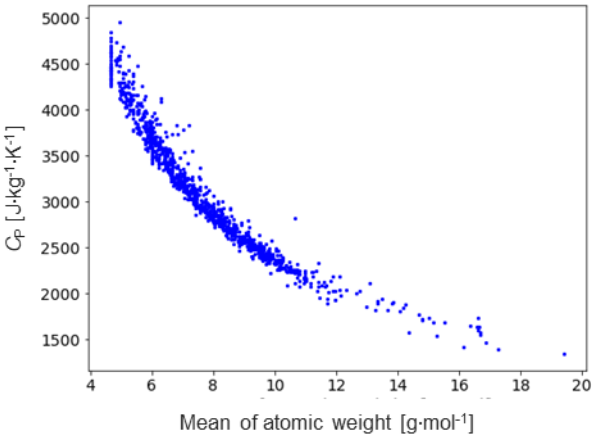


Figure 4. Correlation of mean atomic weights with specific heat capacity.

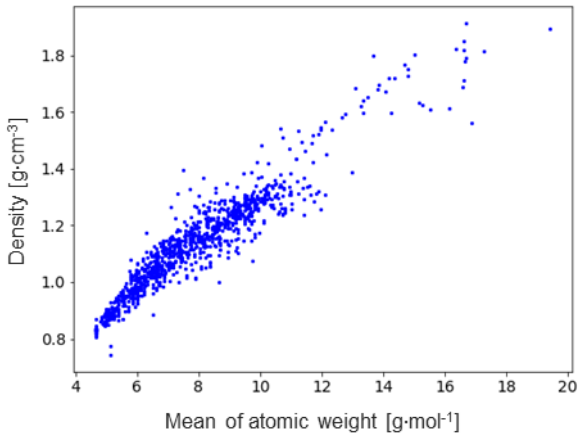


Figure 5. Correlation between mean atomic weight and density.

Supplementary Discussion

1. Examination of box size effects

We tested the effect of box size on several properties. For the evaluation, 28 different polymers were selected by taking structural variations into account. Supplementary Figures 6–10 below show the change of five properties (density, C_P, refractive index, linear expansion coefficient, and volume expansion coefficient) of the 28 polymers, of which chemical structures are shown in Supplementary Figure 11, when the number of polymer chains was varied from 10 to 50 with the number of atoms in each of a polymer chain set to 1,000, and when the number of atoms in each of a polymer chain was increased to 2,000 and the number of polymer chains was set to 25. According to the experimental results, the box size had no significant effect on the calculated properties. Therefore, we decided to perform the MD calculations with 10 polymer chains with approximately 1,000 atoms in each of a polymer chain.

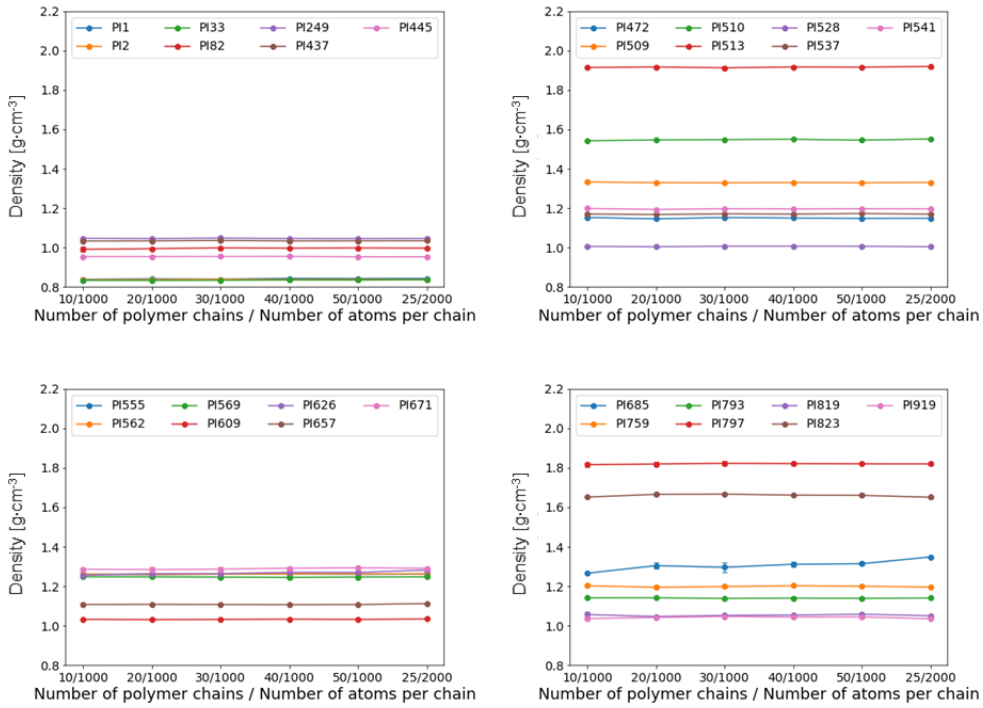


Figure 6. Box size effect on density.

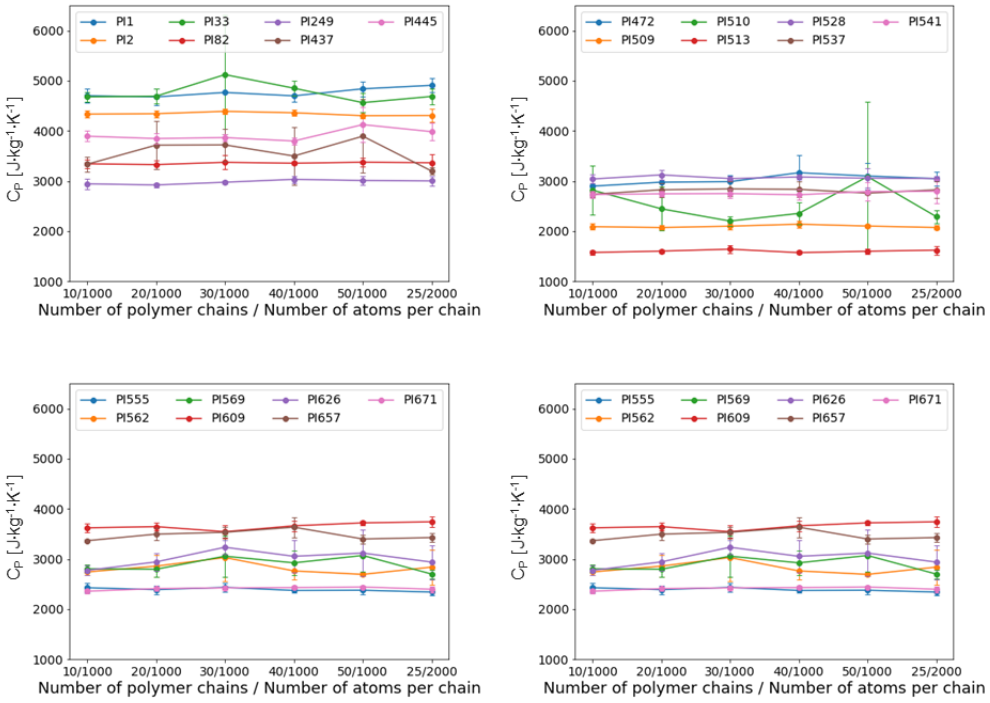


Figure 7. Box size effect on specific heat capacity (C_P).

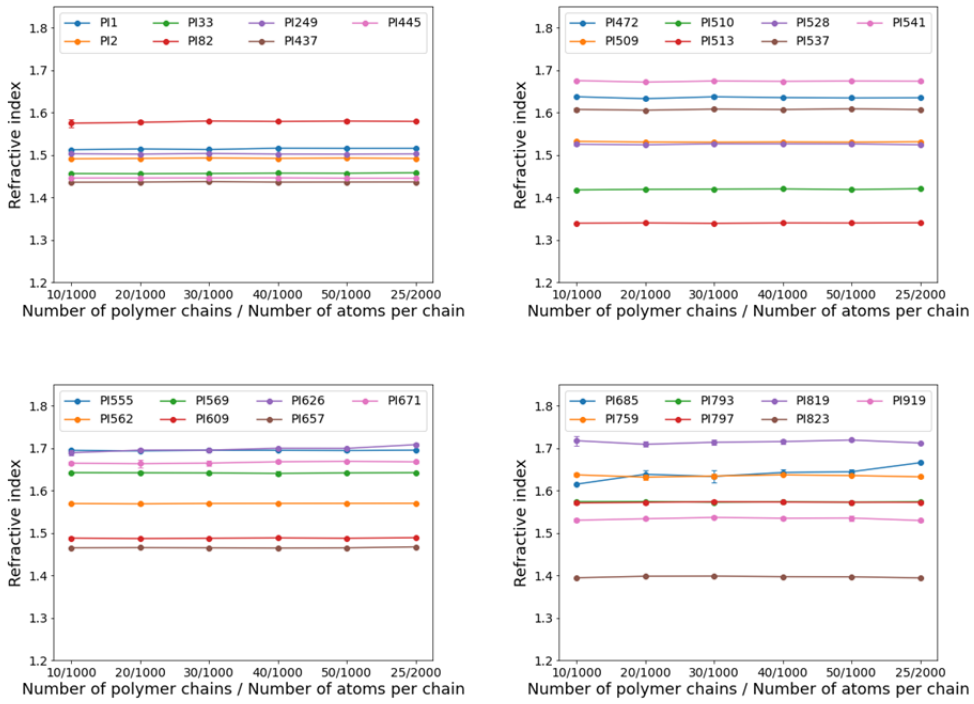


Figure 8. Box size effect on refractive index.

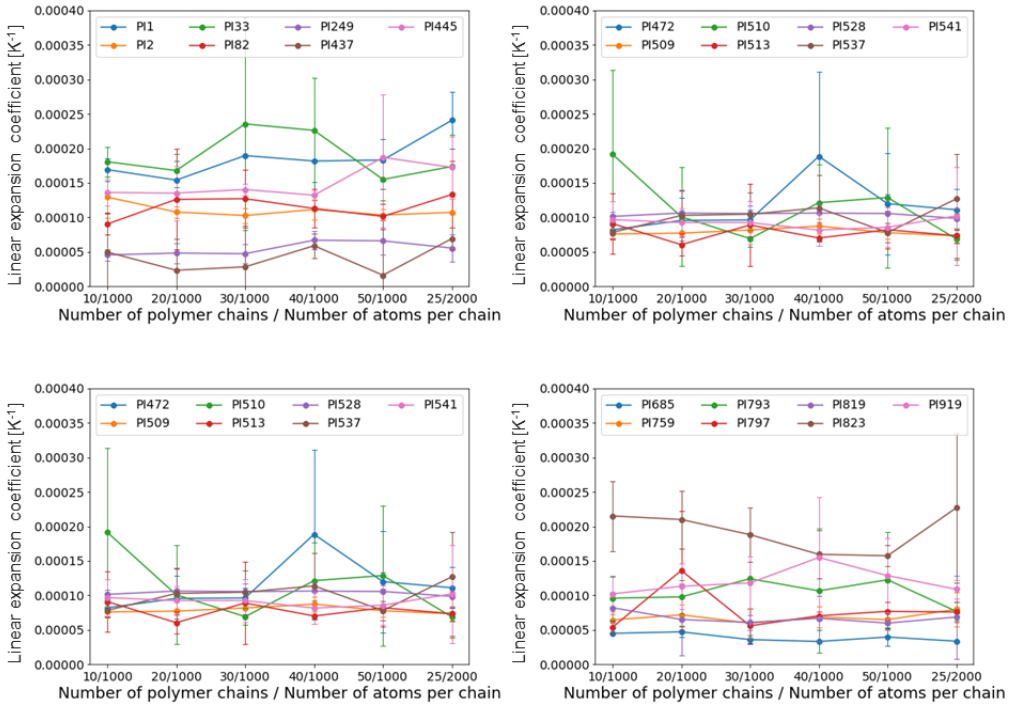


Figure 9. Box size effect on linear expansion coefficient.

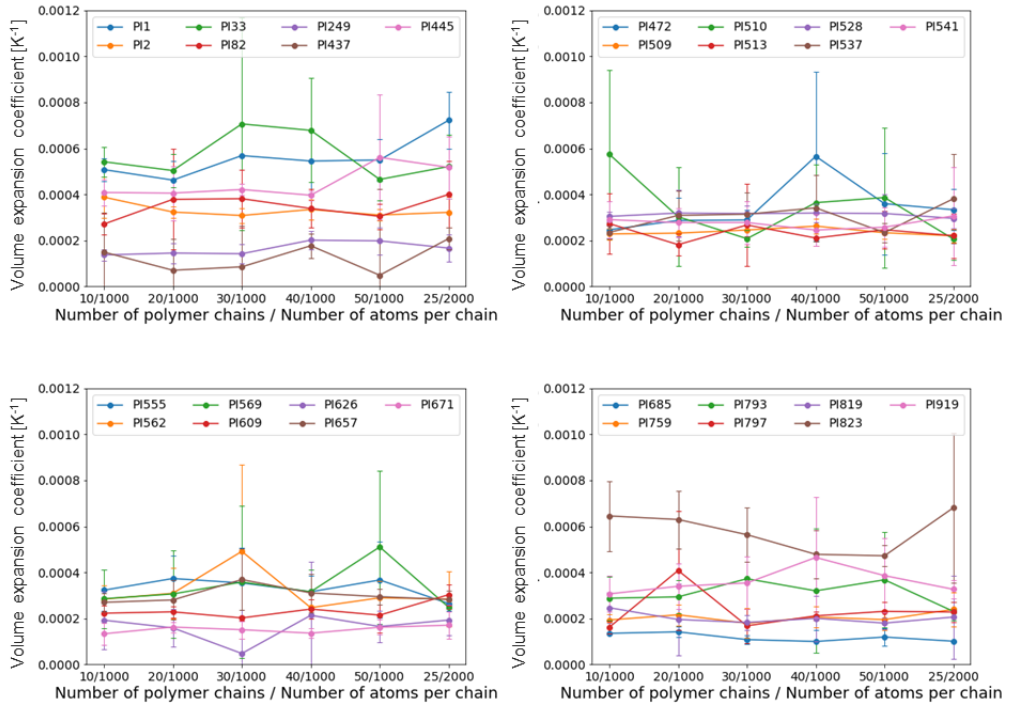


Figure 10. Box size effect on volume expansion coefficient.

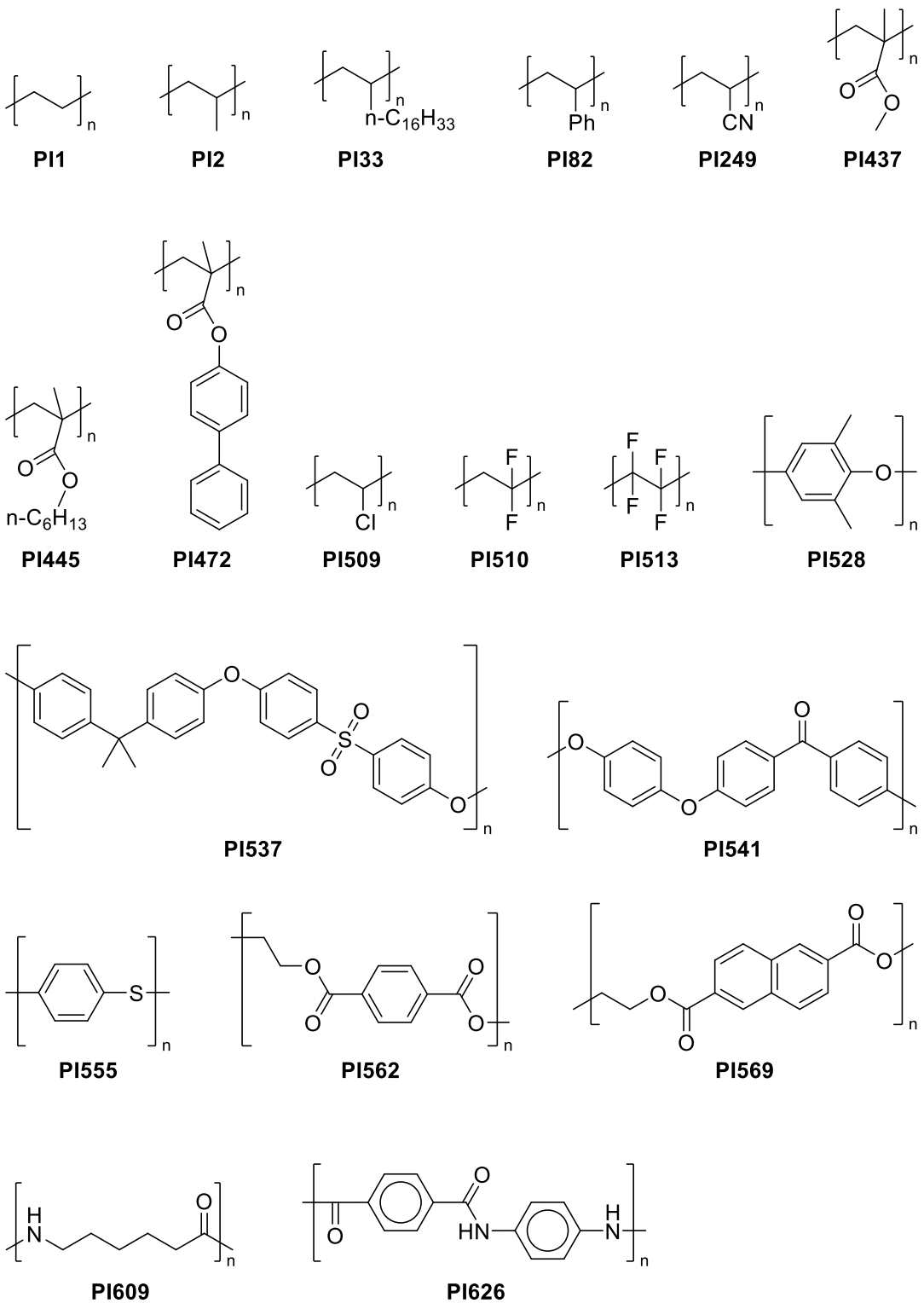


Figure 11. Chemical structure of 28 polymers in examination of box size effects.

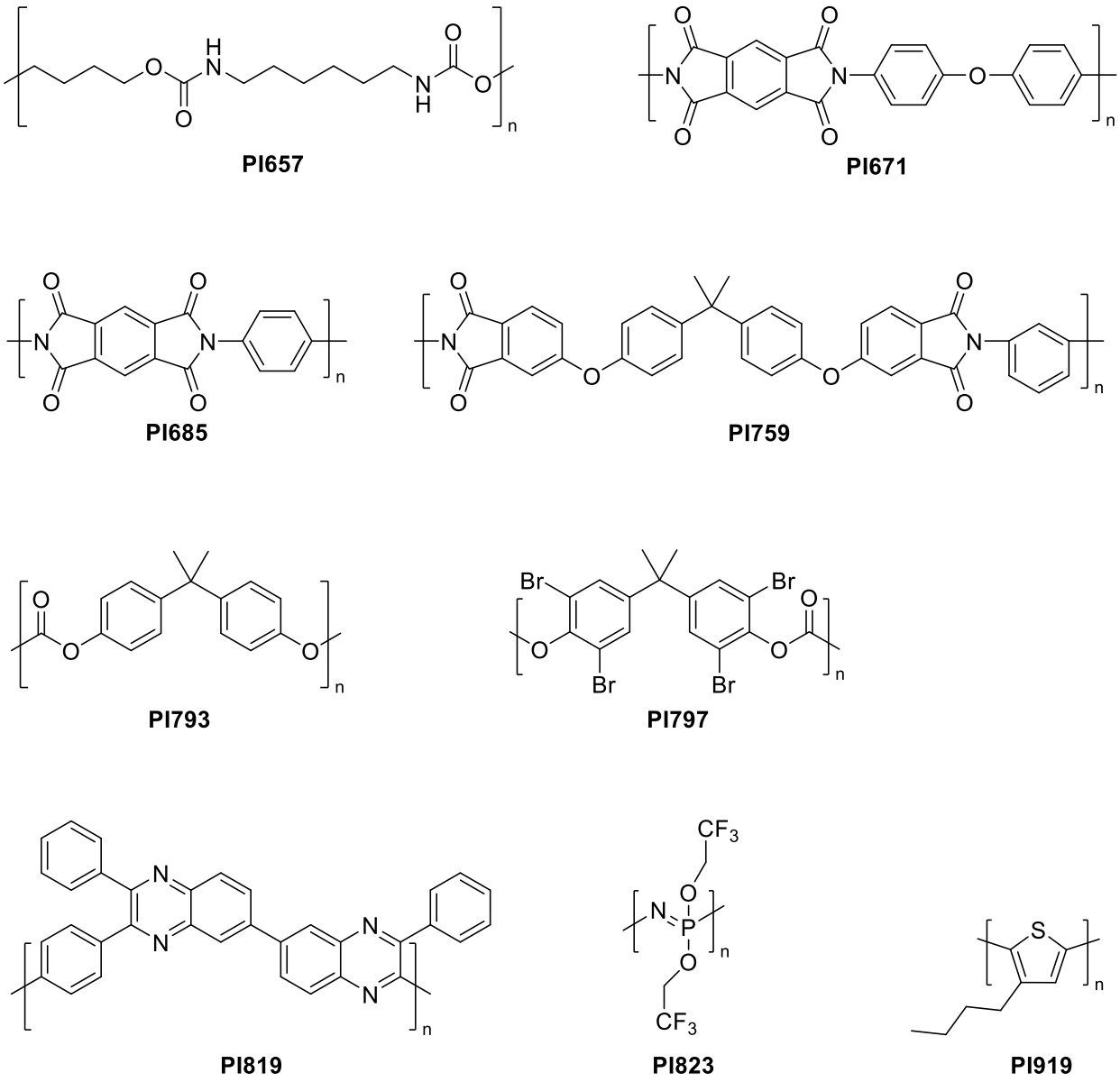


Figure 11. Continued.

2. Correlation between the $C_P^{PoLyInfo}/C_P^{MD}$ ratio and bond-stretching and -bending force constants

Einstein's formulas for the quantum-corrected ($C^{quantum}$) and classical ($C^{classical}$) heat capacities are respectively expressed as

$$C^{\mathrm{quantum}} = 3Nk_{\mathrm{B}}\left(\frac{\hbar\omega}{k_{\mathrm{B}}T}\right)^2 \frac{\exp\left(\frac{\hbar\omega}{k_{\mathrm{B}}T}\right)}{\left[\exp\left(\frac{\hbar\omega}{k_{\mathrm{B}}T}\right)-1\right]^2} \quad (1)$$

$$C^{\mathrm{classical}} = 3Nk_{\mathrm{B}} \quad (2)$$

where N is the number of atoms, k_{B} is the Boltzmann constant, $\hbar$ is the Planck constant, ω is the frequency, and T is the temperature. Hence, the $C^{quantum}/C^{classical}$ ratio is defined as

$$\frac{C^{\mathrm{quantum}}}{C^{\mathrm{classical}}} = \left(\frac{\hbar\omega}{k_{\mathrm{B}}T}\right)^2 \frac{\exp\left(\frac{\hbar\omega}{k_{\mathrm{B}}T}\right)}{\left[\exp\left(\frac{\hbar\omega}{k_{\mathrm{B}}T}\right)-1\right]^2} \quad (3)$$

This ratio decreases monotonically with increasing frequency ω. The MD value of C_P (C_P^{MD}) was computed using Eq 8 in the main text, the value of which is the classical C_P without quantum effects. On the other hand, the experimental value of C_P in PoLyInfo includes quantum effects. The frequency of the bond stretching and bending increases with increasing force constants. Therefore, the observed C_P in PoLyInfo ($C_P^{PoLyInfo}$) to MD-calculated C_P (C_P^{MD}) ratio should decrease with the increasing mean of the bond-stretching and -bending force constants. Such correlation can be seen in Supplementary Figure 12.

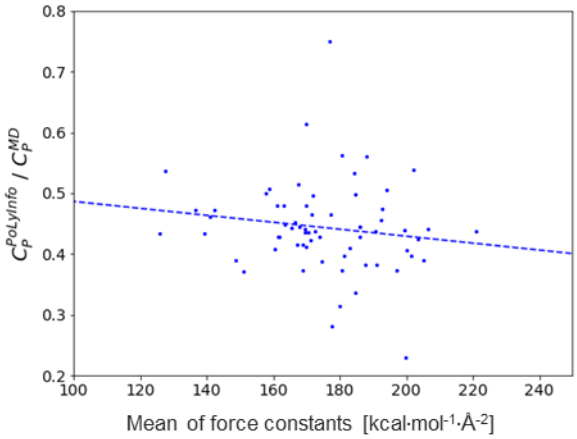


Figure 12. Correlation between the experimental C_P in PoLyInfo ($C_P^{PoLyInfo}$) to MD-calculated C_P (C_P^{MD}) ratio and the mean of the bond-stretching and -bending force constants. The GAFF2 force field parameters K_b and K_a of Eq 2 in the main text were used as the values of the force constants.

Supplementary Notes

1. Getting started with RadonPy

RadonPy is distributed at the GitHub (https://github.com/RadonPy/RadonPy/tree/main). Documentation, including installation guides, system requirements, and a detailed description of the module set and user interface, can be accessed through the tutorial on the GitHub site (https://github.com/RadonPy/RadonPy/blob/main/docs/RadonPy_tutorial_20220331.pdf).

Supplementary Figure 13 shows sample code to perform equilibrated and non-equilibrated MD calculations for a polystyrene (SMILES string: *C(C*)c1ccccc1). See the online tutorial for details on the code and function and options defined in each module.

```python
from radonpy.core import utils, poly
from radonpy.ff.gaff2_mod import GAFF2_mod
from radonpy.sim import qm
from radonpy.sim.preset import eq, tc

smiles = '*C(C*)c1ccccc1'
ter_smiles = '*C'
temp = 300
press = 1.0
omp_psi4 = 10
mpi = 10
omp = 1
gpu = 0
mem = 10000
work_dir = './work_dir'
ff = GAFF2_mod()

if __name__ == '__main__':
    # Conformation search
    mol = utils.mol_from_smiles(smiles)
    mol, energy = qm.conformation_search(mol, ff=ff, work_dir=work_dir,
                                         psi4_omp=omp_psi4, mpi=mpi, omp=omp, memory=mem, log_name='monomer1')

    # Electronic propety calculation
    qm.assign_charges(mol, charge='RESP', opt=False, work_dir=work_dir, omp=omp_psi4, memory=mem, log_name='monomer1')
    qm_data = qm.sp_prop(mol, opt=False, work_dir=work_dir, omp=omp_psi4, memory=mem, log_name='monomer1')
    polar_data = qm.polarizability(mol, opt=False, work_dir=work_dir, omp=omp_psi4, memory=mem, log_name='monomer1')

    # RESP charge calculation of a termination unit
    ter = utils.mol_from_smiles(ter_smiles)
    qm.assign_charges(ter, charge='RESP', opt=True, work_dir=work_dir, omp=omp_psi4, memory=mem, log_name='ter1')

    # Generate polymer chain
    dp = poly.calc_n_from_num_atoms(mol, 1000, terminal1=ter)
    homopoly = poly.polymerize_rw(mol, dp, tacticity='atactic')
    homopoly = poly.terminate_rw(homopoly, ter)
```

Figure 13. Sample code of performing equilibrated and non-equilibrated MD calculations for polystyrene using RadonPy.

```
# Force field assignment
result = ff.ff_assign(homopoly)
if not result:
    print('[ERROR: Can not assign force field parameters.]')

# Generate simulation cell
ac = poly.amorphous_cell(homopoly, 10, density=0.05)

# Equilibration MD
eqmd = eq.EQ21step(ac, work_dir=work_dir)
ac = eqmd.exec(temp=temp, press=1.0, mpi=mpi, omp=omp, gpu=gpu)

analy = eqmd.analyze()
prop_data = analy.get_all_prop(temp=temp, press=1.0, save=True)
result = analy.check_eq()

# Additional equilibration MD
for i in range(4):
    if result: break
    eqmd = eq.Additional(ac, work_dir=work_dir)
    ac = eqmd.exec(temp=temp, press=press, mpi=mpi, omp=omp, gpu=gpu)
    analy = eqmd.analyze()
    prop_data = analy.get_all_prop(temp=temp, press=press, save=True)
    result = analy.check_eq()

if not result:
    print('[ERROR: Did not reach an equilibrium state.]')

# Non-equilibrium MD for thermal condultivity
else:
    nemd = tc.NEMD_MP(ac, work_dir=work_dir)
    ac = nemd.exec(decomp=True, temp=temp, mpi=mpi, omp=omp, gpu=gpu)

    nemd_analy = nemd.analyze()
    TC = nemd_analy.calc_tc(decomp=True, save=True)
    if not nemd_analy.Tgrad_data['Tgrad_check']:
        print('[ERROR: Low linearity of temperature gradient.]')

    print('Thermal conductivity: %f' % TC)
```

Figure 13. Continued.

The specified SMILES string is converted into a Mol object by the "utils.mol_from_smiles" function. After converting the specified SMILES string into a Mol object, the quantum chemistry calculation module ("qm") is used to perform a conformation search ("qm.conformation_search") and charge assignment ("qm.assign_charges") for the input repeating unit. The sample code specifies the RESP charge model as an option. Other electronic properties such as HOMO and LUMO energies, dipole moments, and polarizability tensors are also calculated using this module. The "poly.calc_n_from_num_atoms", "poly.polymerize_rw", and "poly.terminate_rw" functions generate a polymer chain for a specified number of atoms (e.g., 1,000) with a self-avoiding random walk

algorithm. Although not shown in this paper, the current version also provides modules to generate alternating, random, and block copolymers, as described in the online tutorial. The "ff.ff_assign" method assigns the parameters of the GAFF2 force field. The "poly.amorhous_cell" function generates a simulation cell for amorphous polymers. In the sample code, 10 polymer chains are randomly arranged so that they do not overlap each other and have a density of 0.05 g·cm^{-3}. The user can also build simulation cells for polymer-polymer or polymer-solvent mixture systems. The user then specifies the temperature ("temp") and pressure ("press"), and performs an equilibrium MD calculation (the "exec" method of the "eq.EQ21step" class). The convergence of energy and density is monitored every 5 ns to determine if the system has reached equilibrium ("analy.check_eq"). If the equilibration is not reached, the equilibrium calculation is scheduled to restart from the saved previous state (the "exec" method of the "eq.Additional" class). If the equilibrium is successfully achieved, the NEMD simulation (the "exec" method of the "tc.NEMD_MP" class) is performed. When running a hybrid parallel computing with MPI (message passing interface), OpenMP (open multi-processing), and GPU (graphics processing unit), the number of MPI processes ("mpi"), OpenMP threads ("omp"), and GPUs ("gpu") can be specified according to the available computer architecture.

Supplementary Methods

1. Shotgun transfer learning

The objective is to predict experimental properties from the chemical structure of any given polymer repeating unit. For the model input, we used a subset of the 1006-dimensional descriptor vector that concatenated eight different fingerprints implemented in the RDKit package, including the extended-connectivity fingerprint (ECFP),[1] the functional-class fingerprint (FCFP),[1] topological torsion fingerprint,[2] atom pair fingerprint,[2] MACCS key, RDKit fingerprint,[3] pattern fingerprint,[3] and layered fingerprint.[3] For transfer learning, the source task was to predict the MD-calculated properties, and the target task was to predict the experimental properties in PoLyInfo.

From the MD dataset for the source task, all polymers in the experimental dataset to be predicted were excluded. Using the MD dataset, we trained 100 neural networks having randomly constructed different network structures. Each model was designed to form a fully connected pyramid-shaped layers in which the number of layers was randomly selected from three and four, and the number of neurons monotonically decreased from the input layer to the output one. All neurons in the hidden layers were activated by a rectified linear unit (ReLU),[4] and a linear transformation function form the output layer. For each model, we used a randomly chosen subset of the 400-dimensional descriptor.

We randomly selected 80%, 10%, and 10% of the experimental dataset for the training, validation, and testing for the target task, respectively. Using the training set, each pretrained source model was fine-tuned into a calibration model for the target task. Here, the pretrained model was used as a starting point for the model training. The given weights on the last output layers of the pretrained model were randomly initialized, while the learned parameters of the remaining layers were used as the initial values of training. All those parameters were then fine-tuned at a small learning rate.

The root mean square error (RMSE) of each transferred model with respect to the validation set was calculated, and the prediction performance on the test dataset was examined using the model showing the best transferability that achieved the lowest validation RMSE.

Supplementary References

1. Rogers, D. & Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**, 742–754 (2010).
2. Nilakantan, R., Bauman, N., Dixon, J. S. & Venkataraghavan, R. Topological torsion: a new molecular descriptor for SAR applications. comparison with other descriptors. *J. Chem. Inf. Comput. Sci.* **27**, 82–85 (1987).
3. Landrum, G. RDKit: open-source cheminformatics software. https://www.rdkit.org/ (2020).
4. Glorot, X., Bordes, A. & Bengio, Y. Deep sparse rectifier neural networks. *Proc. Fourteenth Int. Conf. Artif. Intell. Stat.* **15**, 315–323 (2011).