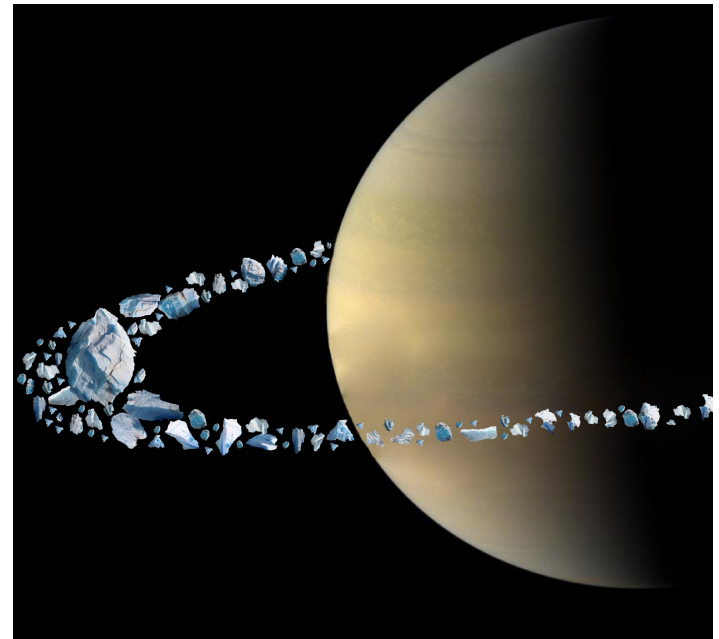
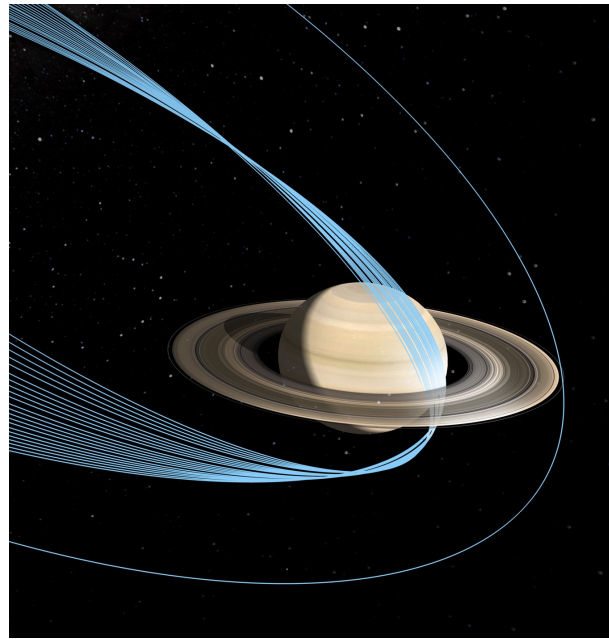
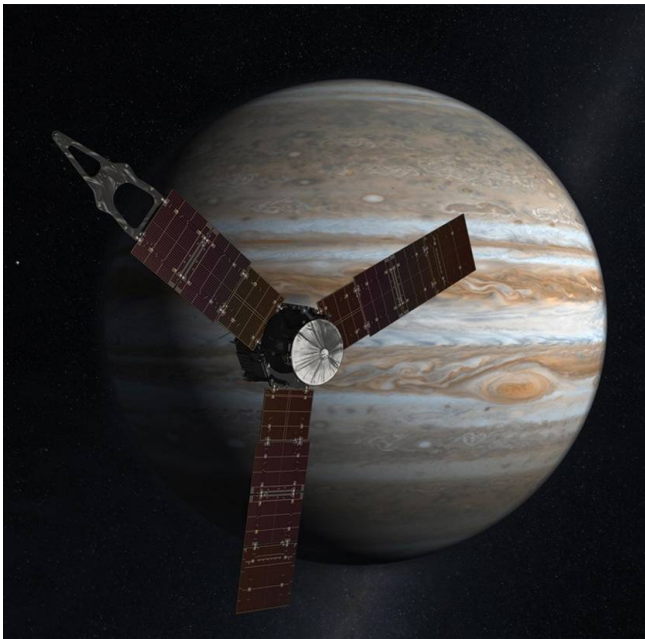


Quadratic Monte Carlo to Understand Jupiter's Interior Structure and the Origin of Saturn's Ring



B. Militzer, W. B. Hubbard, J. Wisdom, R. Dbouk,

UC Berkeley

U Arizona

MIT

MIT

F. Nimmo, B. Downey, R. French

UCSC

UCSC

Wellesley

Big Questions that my Group Helps Address


1. How did our solar system form?
2. What are giant planets made of?
3. How do material behave at high pressure?

Outline

1. Juno mission and Jupiter's **Dilute Core**
2. How did Saturn become the **Lord of the Rings?**
3. **Quadratic Monte Carlo** – a general-purpose sampling method



1. Juno Mission and Dilute Core



2. How did Saturn
become the Lord of
the Rings?

What is so Unusual About Planet Saturn?

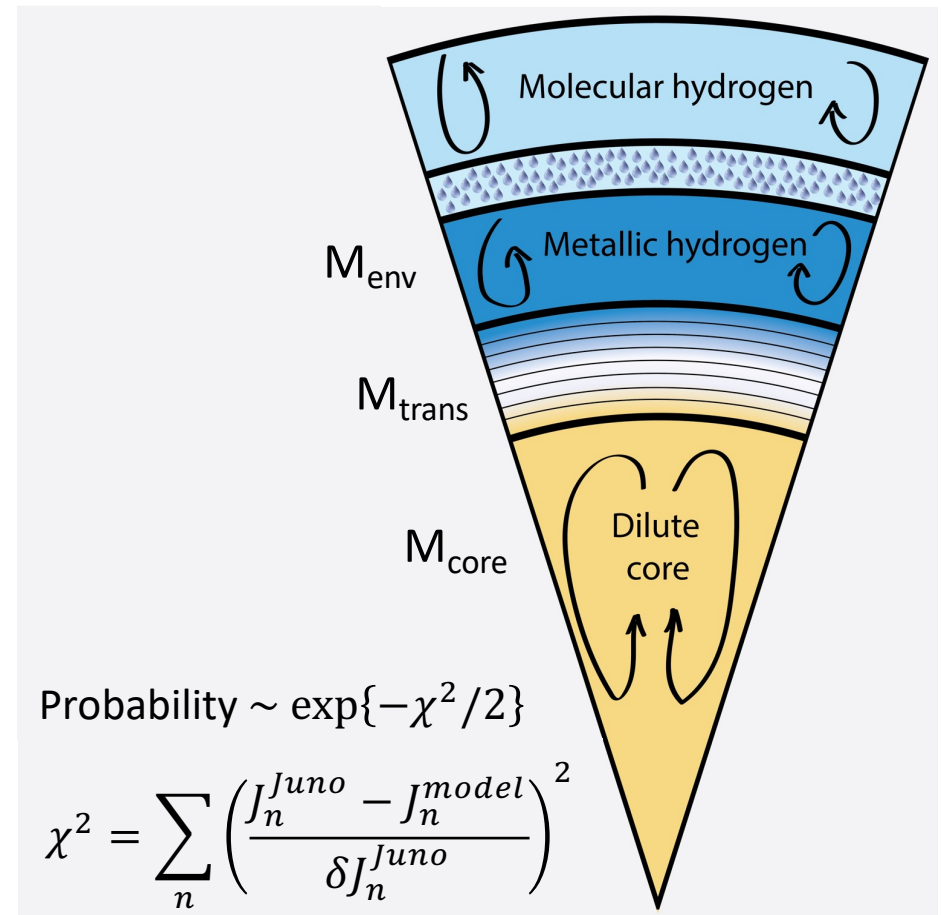
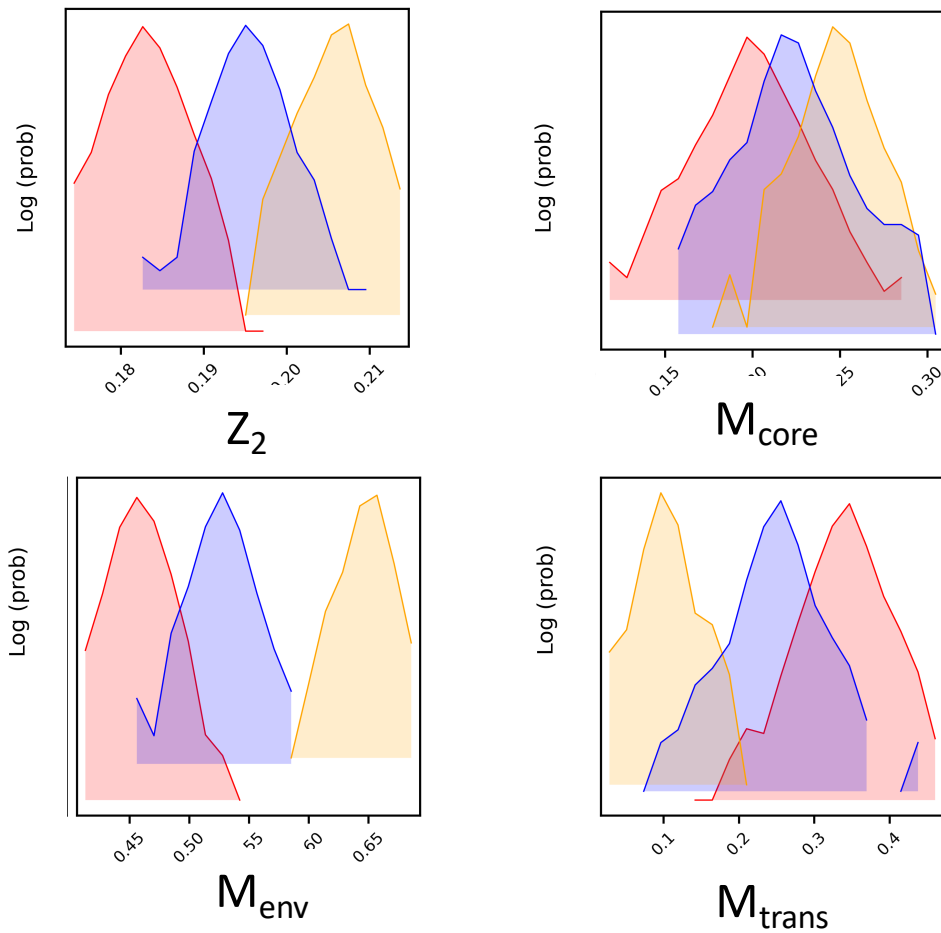
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune



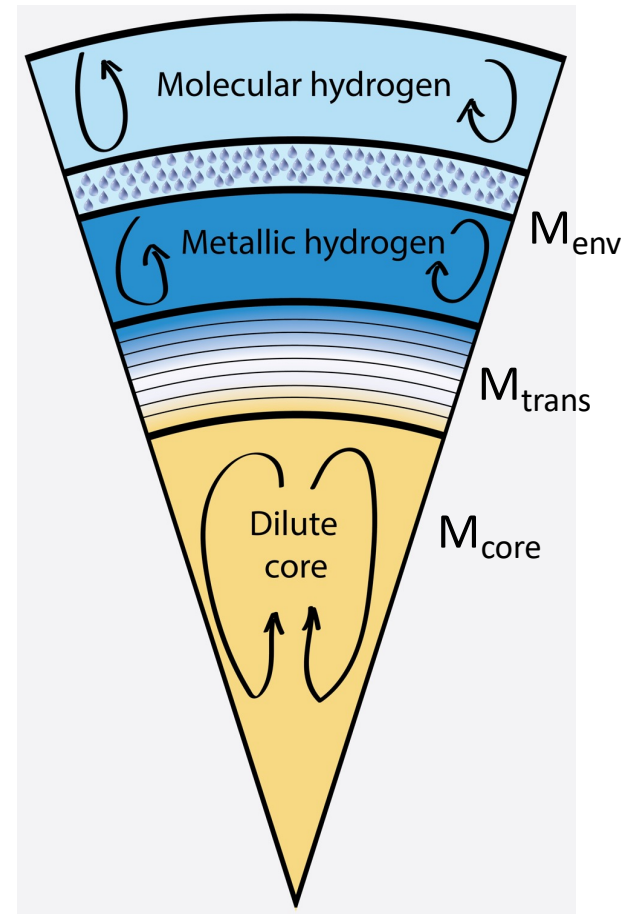
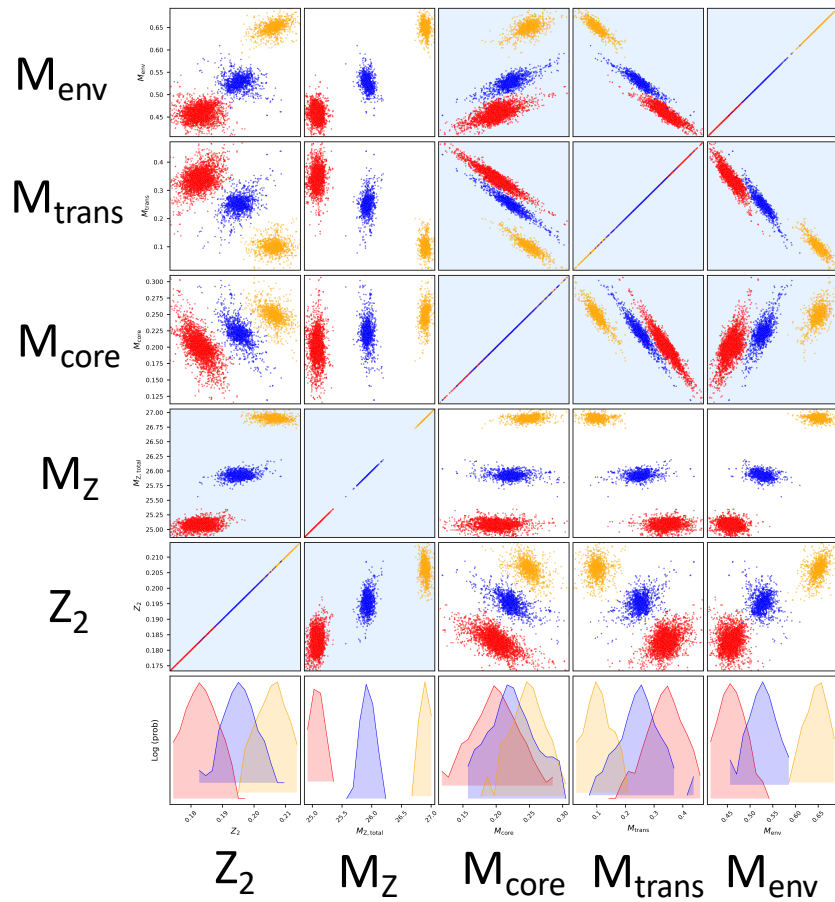


3. Quadratic Monte Carlo Method

Apply our QMC method Dilute Core Models

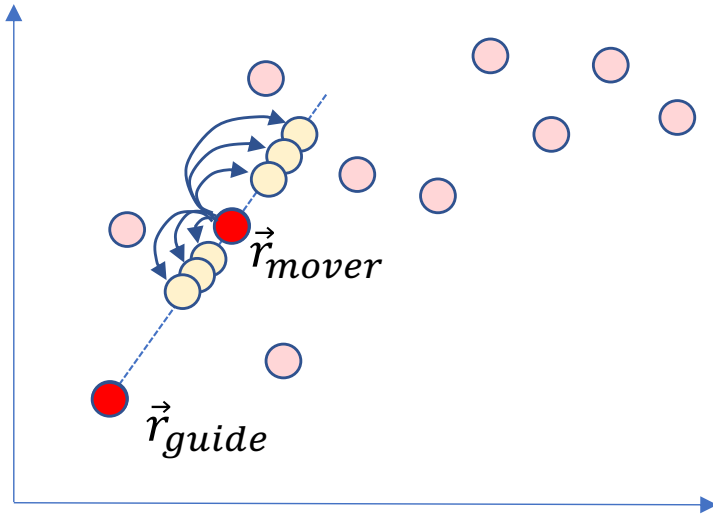


Apply our QMC method Dilute Core Models



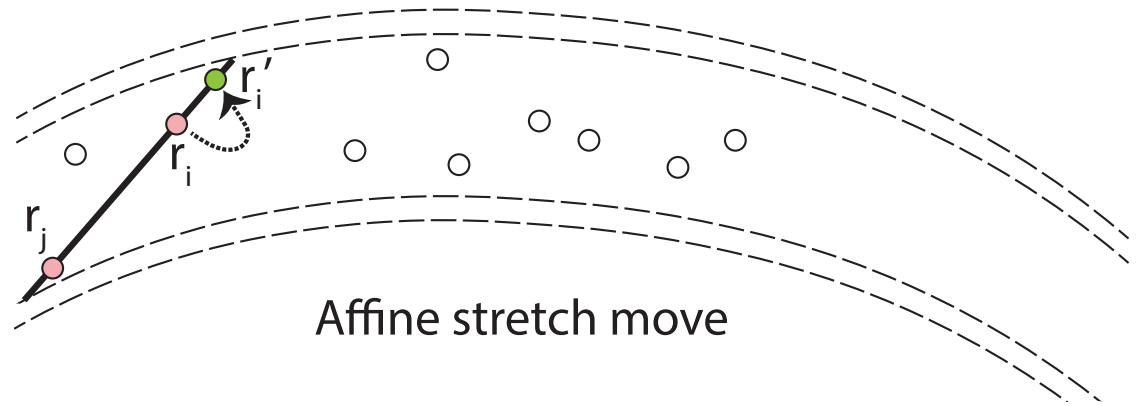
Affine Invariance MCMC by Goodman & Weare

JONATHAN GOODMAN AND JONATHAN WEARE

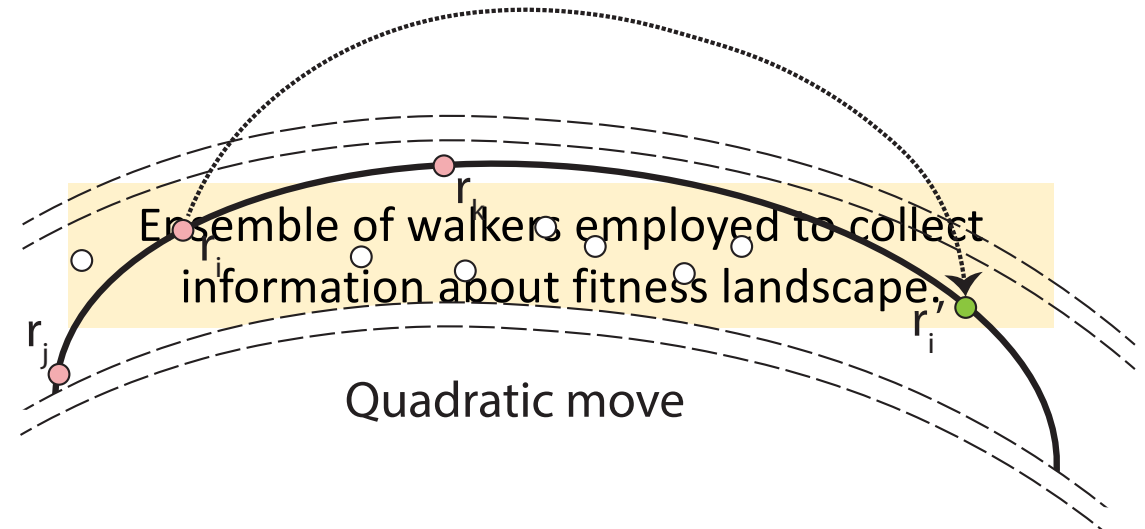


Affine Stretch move

$$g(z) \propto \begin{cases} \frac{1}{\sqrt{z}} & \text{if } z \in \left[\frac{1}{a}, a\right], \\ 0 & \text{otherwise.} \end{cases}$$



Affine stretch move



Ensemble of walkers employed to collect information about fitness landscape

Quadratic move

Our Quadratic Monte Carlo Method Explained at <http://miltzer.berkeley.edu/QMC>

```
Array1 <S> s = SetUpNStates(nWalkers);           // Set up nWalkers different states of type S
for(int iBlock=0; i<nBlocks; iBlock++) {         // loop over blocks
  for(int iStep=0; i<nStepsPerBlock; iStep++) { // loop over steps
    for(int i=0; i<nWalkers; i++) {             // try moving every walker once per step
      int j,k;
      SelectTwoOtherWalkersAtRandom(i,j,k);
      .....
      const double tj = -1.0;
      const double tk = +1.0;
      const double ti  = SampleT(a); // sample t space
      const double tNew = SampleT(a); // sample t space one more time

      const double wi = LagrangeInterpolation(tNew,ti,tj,tk);
      const double wj = LagrangeInterpolation(tNew,tj,tk,ti);
      const double wk = LagrangeInterpolation(tNew,tk,ti,tj);

      S sNew = s[0]; // Create new state by coping over an existing one.
      for(int d=0; d<nDim; d++) {
        sNew[d] = wi*s[i][d] + wj*s[j][d] + wk*s[k][d]; // set nDim parameters of new state
      }
      .....
      if (sNew.Valid()) { // check if state sNew is valid before calling Evaluate()
        sNew.Evaluate(); // Sets the energy sNew.y which defines the state's probability = exp(-y/temp)
        double dy = sNew.y - s[i].y; // difference in energy between new and old state
        double prob = pow(fabs(wi),nDim) * exp(-dy/temp); // Note |wi|^nDim and Boltzmann factors
        bool accept = (prob>Random()); // Random() returns a single random number between 0 and 1.

        if (accept) {
          for(int d=0; d<nDim; d++) {
            s[i][d] = sNew[d]; // copy over state sNew
          }
          s[i].y = sNew.y; // copy also its energy
        }
      }
    }
  } // look over all walkers
  ComputeDifferentEnsembleAverages(s);
} // end of loop over steps
PrintEndOfBlockStatement();
} // end of loop over blocks
PrintEndOfRunStatement();
```

New lines

Prefactor requires discussion

Sampling Details and Prefactors

Affine moves, type 1

$$\vec{r}'_i = \vec{r}_j + \lambda(\vec{r}_i - \vec{r}_j)$$

$$T_1(\lambda) = \frac{1}{\lambda} T_1\left(\frac{1}{\lambda}\right)$$

$$T_1(\lambda) \propto \frac{1}{\sqrt{\lambda}} \text{ if } \lambda \in \left[\frac{1}{a}, a\right]$$

$$\alpha = N - 1$$

$$A(\vec{r}_i \rightarrow \vec{r}'_i) = \min \left[1, \frac{\pi(\vec{r}'_i)}{\pi(\vec{r}_i)} \lambda^\alpha \right]$$

Affine moves, type 2

$$\vec{r}'_i = \vec{r}_j + \lambda(\vec{r}_i - \vec{r}_j)$$

$$T_2(\lambda) = \frac{a}{a^2 - 1} = \text{const}$$

$$\alpha = N - 2$$

Quadratic moves

$$\vec{r}'_i = w_i \vec{r}_i + w_j \vec{r}_j + w_k \vec{r}_k$$

$$w_i = L(t'_i; t_i, t_j, t_k),$$

$$w_j = L(t'_i; t_j, t_k, t_i),$$

$$w_k = L(t'_i; t_k, t_i, t_j),$$

$$L(x; x_0, x_1, x_2) \equiv \frac{x - x_1}{x_0 - x_1} \frac{x - x_2}{x_0 - x_2}$$

t_i and t'_i from the same distribution $\mathcal{P}(t)$

$$A(\vec{r}_i \rightarrow \vec{r}'_i) = \min \left[1, \frac{\pi(\vec{r}'_i)}{\pi(\vec{r}_i)} |w_i|^N \right]$$

BM, *Astrophys. J.* 953 (2023) 111

In Our Derivation of Prefactors, we follow Green and Mira

Regular detailed balance between two states $\pi(\vec{r})P(\vec{r} \rightarrow \vec{r}') = \pi(\vec{r}')P(\vec{r}' \rightarrow \vec{r})$

Green & Mira (2001) who formulated a generalized condition

$$\int \pi(d\vec{r})P(\vec{r} \rightarrow d\vec{r}') = \int \pi(d\vec{r}')P(\vec{r}' \rightarrow d\vec{r})$$

The notation $\int \pi(d\vec{r})$ refers to the integral,

$$\int \dots \pi(d\vec{r}) \equiv \int \dots p(\vec{r})d\vec{r} \quad ,$$

Green & Mira (2001) showed that the acceptance probability for a move from \vec{r} to \vec{r}' is given by,

$$A(\vec{r} \rightarrow \vec{r}') = \min \left\{ 1, \frac{\pi(\vec{r}')T'(\vec{\lambda}')}{\pi(\vec{r})T(\vec{\lambda})} \left| \frac{\partial(\vec{r}', \vec{\lambda}')}{\partial(\vec{r}, \vec{\lambda})} \right| \right\} \quad , \quad (\text{A4})$$

Jacobian determinant for the transformation from $(\vec{r}, \vec{\lambda})$ to $(\vec{r}', \vec{\lambda}')$

In Our Derivation of Prefactors, we follow Green and Mira

$$\frac{\partial \lambda'_1}{\partial \lambda_1} = 0 \quad , \quad \frac{\partial \lambda'_2}{\partial \lambda_2} = 0 \quad , \quad \frac{\partial \lambda'_1}{\partial \lambda_2} = 1 \quad \text{and} \quad \frac{\partial \lambda'_2}{\partial \lambda_1} = 1 \quad .$$

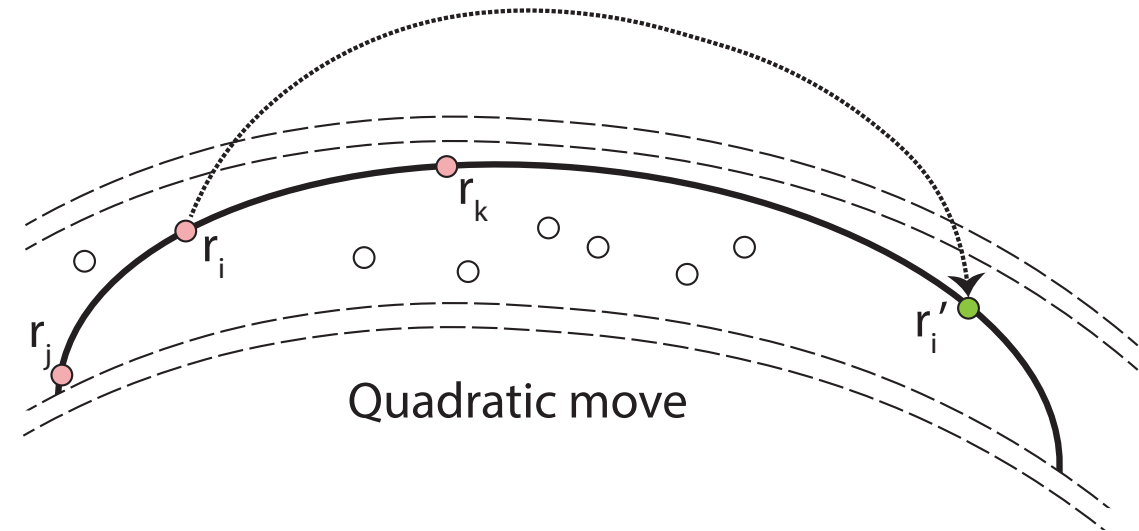
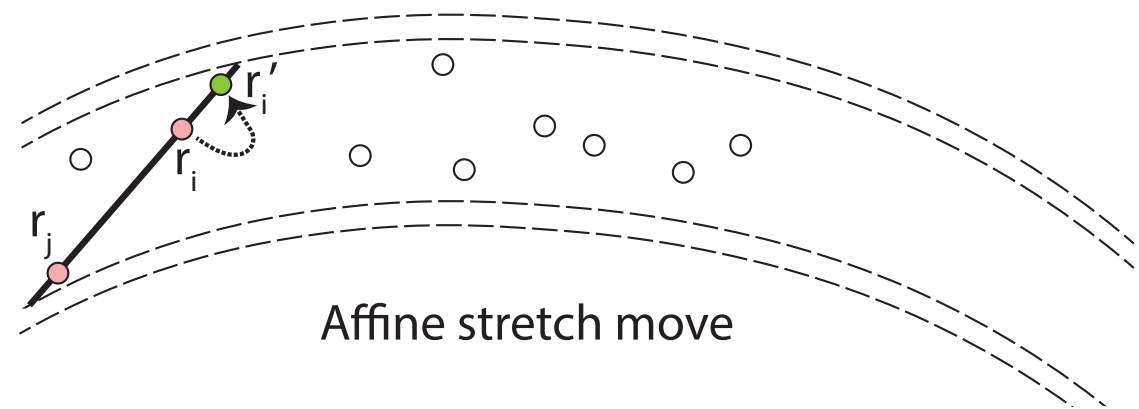
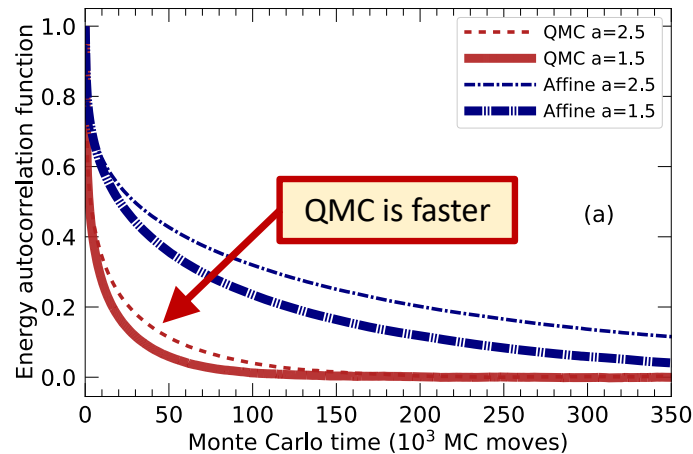
The Jacobian becomes a $(N + 2, N + 2)$ matrix:

$$J = \frac{\partial(\vec{r}'_i, \lambda'_1, \lambda'_2)}{\partial(\vec{r}_i, \lambda_1, \lambda_2)} = \begin{pmatrix} \frac{\partial r'_{ib}}{\partial r_{ia}} = w_i \delta_{ab} & \frac{\partial r'_{ib}}{\partial \lambda_1} & \frac{\partial r'_{ib}}{\partial \lambda_2} \\ \frac{\partial \lambda'_1}{\partial \lambda_1} = \frac{\partial \lambda_2}{\partial r_{ia}} & \frac{\partial \lambda'_1}{\partial \lambda_1} = 0 & \frac{\partial \lambda'_1}{\partial \lambda_2} = 1 \\ \frac{\partial \lambda'_2}{\partial r_{ia}} = \frac{\partial \lambda_1}{\partial r_{ia}} & \frac{\partial \lambda'_2}{\partial \lambda_1} = 1 & \frac{\partial \lambda'_2}{\partial \lambda_2} = 0 \end{pmatrix} \quad ,$$

and its determinant is given by a sum over permutations, σ_k ,

$$|J| = \sum_{\sigma_1 \cdots \sigma_N} \prod_{k=1}^N \frac{\partial r'_{i, \sigma_k}}{\partial \lambda_2} \frac{\partial \lambda_2}{\partial r_{i, k}} + \prod_{k=1}^N \frac{\partial r'_{i, \sigma_k}}{\partial \lambda_1} \frac{\partial \lambda_1}{\partial r_{i, k}} - \prod_{k=1}^N w_i \delta_{k, \sigma_k} = \sum_{\sigma_1 \cdots \sigma_N} \prod_{k=1}^N w_i \delta_{k, \sigma_k} = w_i^N$$

Our Quadratic Monte Carlo is more Efficient Than The Affine Invariant Sampler (emcee) that Moves Linearly

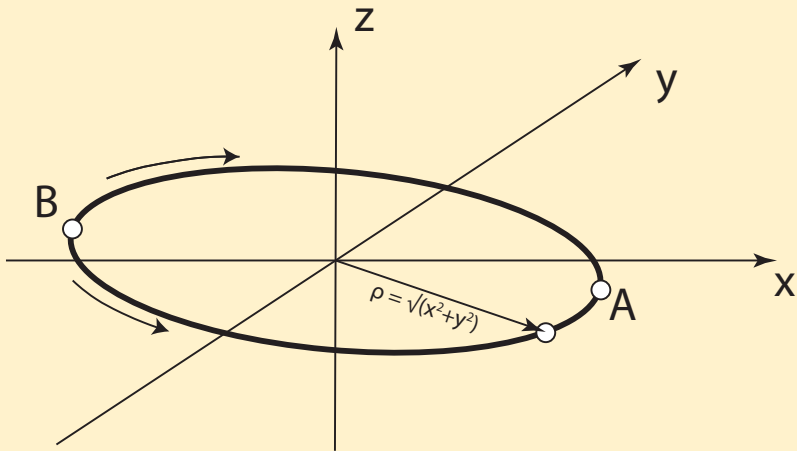


B. Militzer,
Astrophysical J. 953 (2023) 111

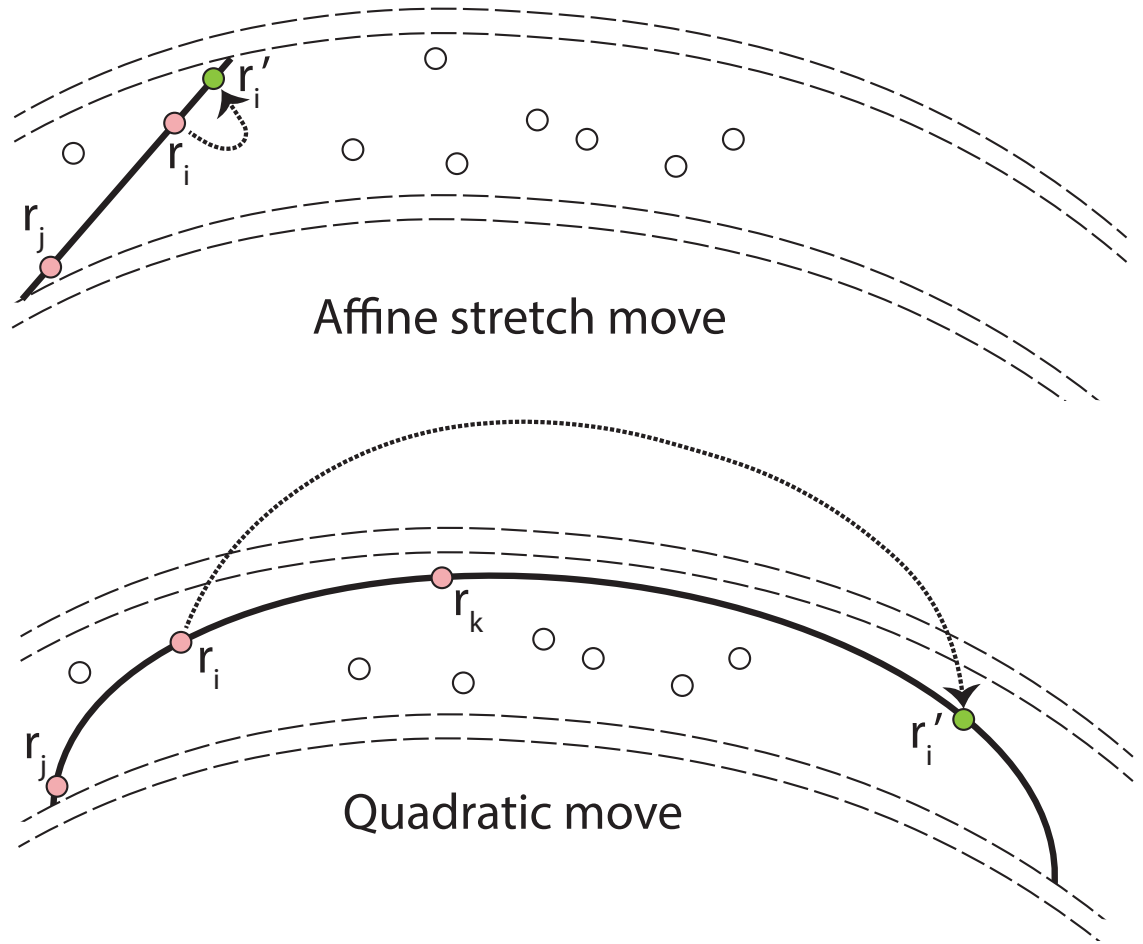
Open QMC source code
[10.5281/zenodo.8038144](https://doi.org/10.5281/zenodo.8038144)

Our Test Case: A Ring Potential

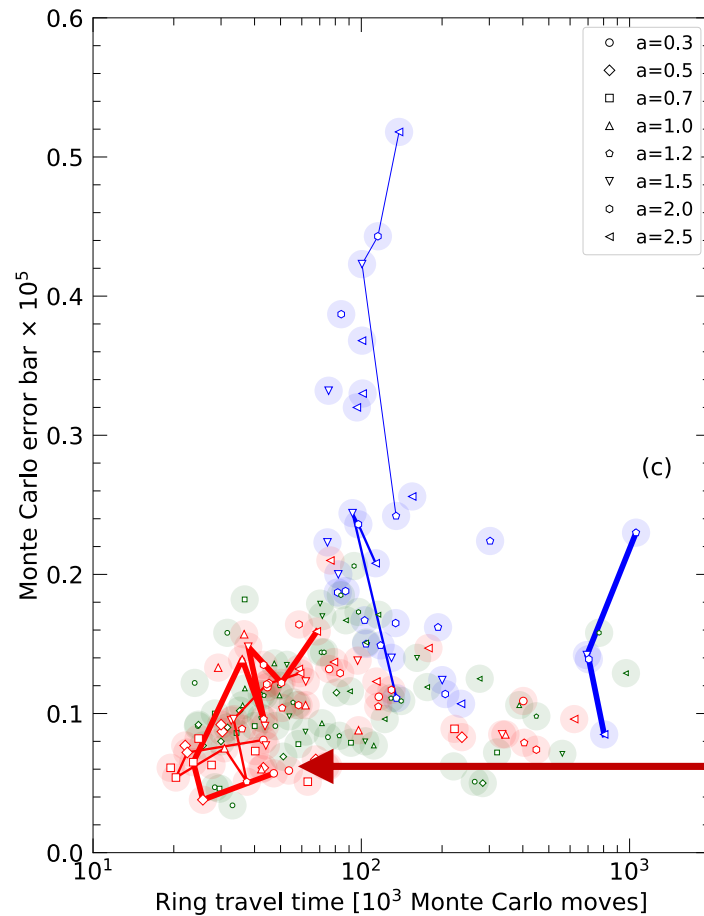
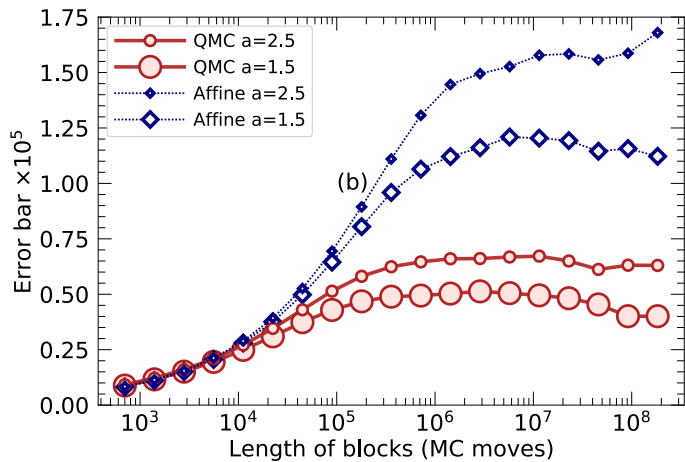
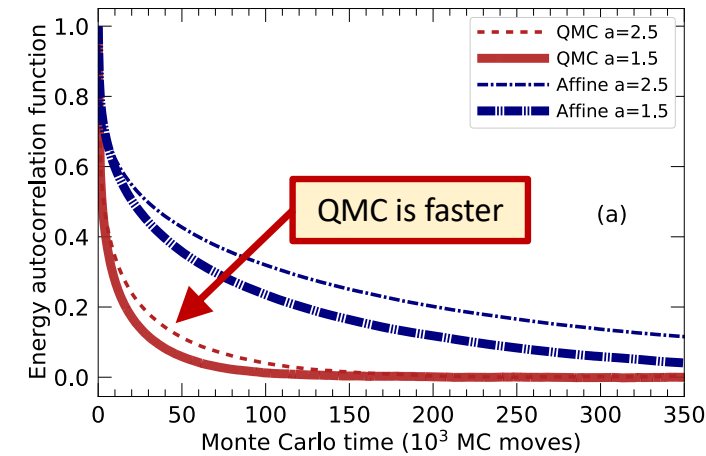
Test case: Ring potential



$$V(\vec{r}) = (2m)^{2m} \left[(\rho - R)^{2m} + \sum_{i=3}^N r_i^{2m} \right] - Cr_1$$

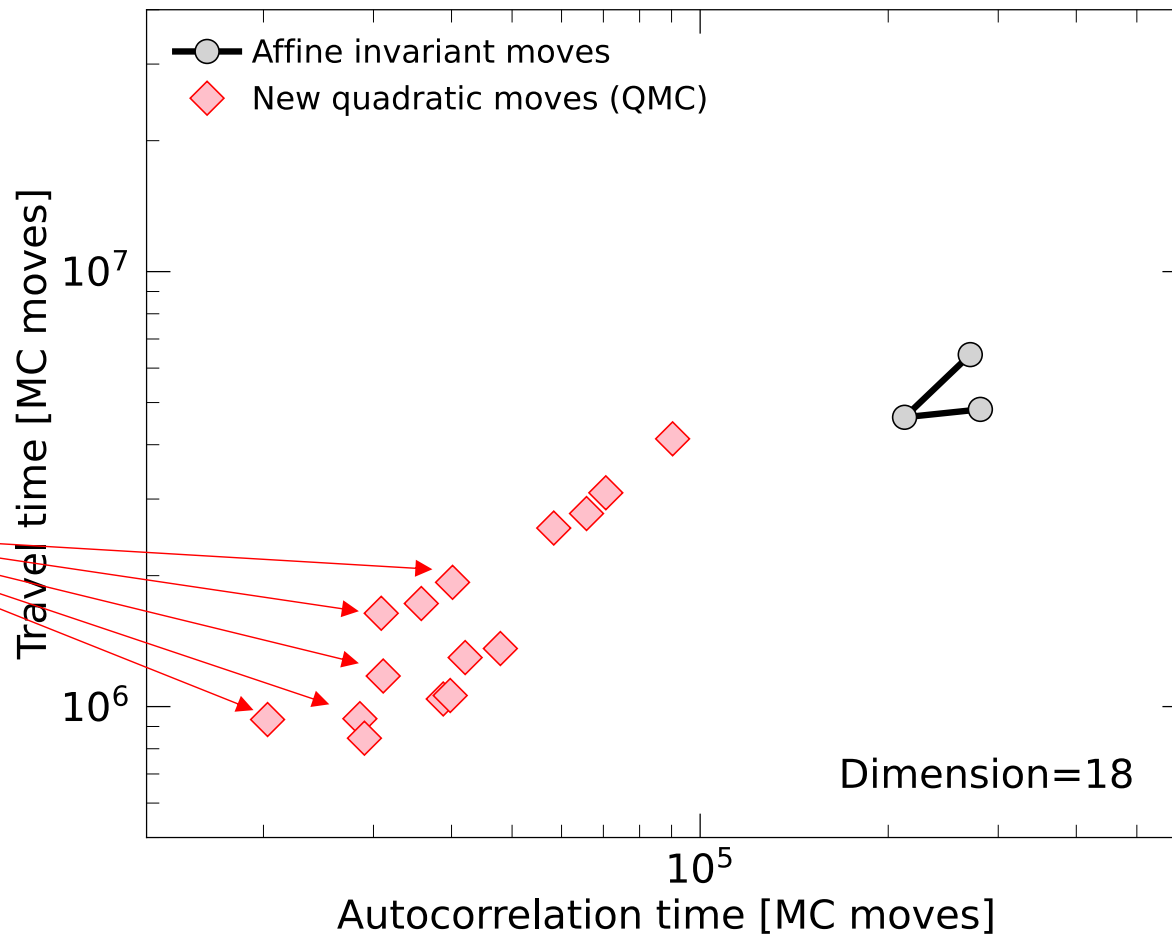


Two Performance Criteria: Autocorrelation & Travel Time



- Quadratic moves yield smaller error bars and shorter travel times.
- But don't use too many walkers.
- We suggest $N_W=2D...3D$

Comparison: QMC vs Affine Invariant Method

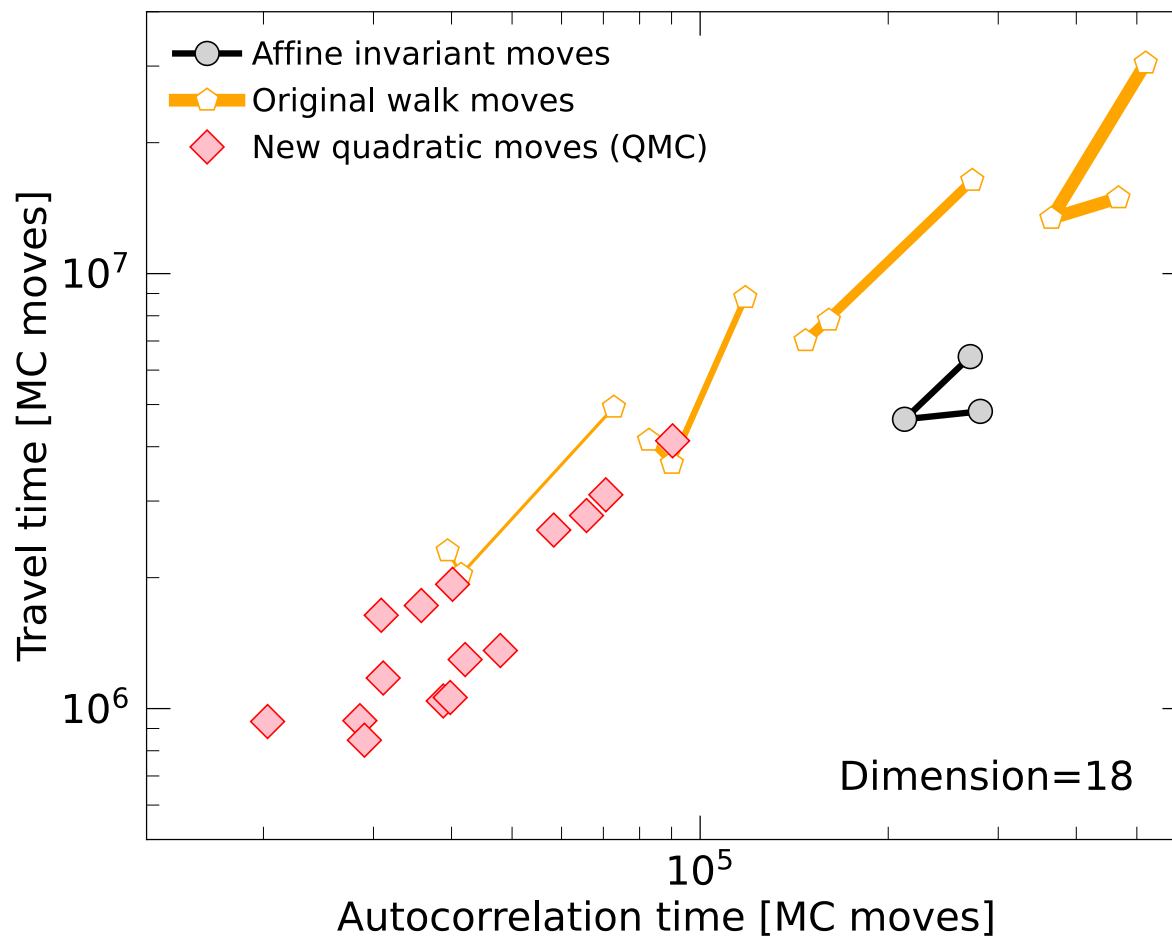


QMC moves are about four times as efficient

Test case:
Ring potential in 18 dimensions

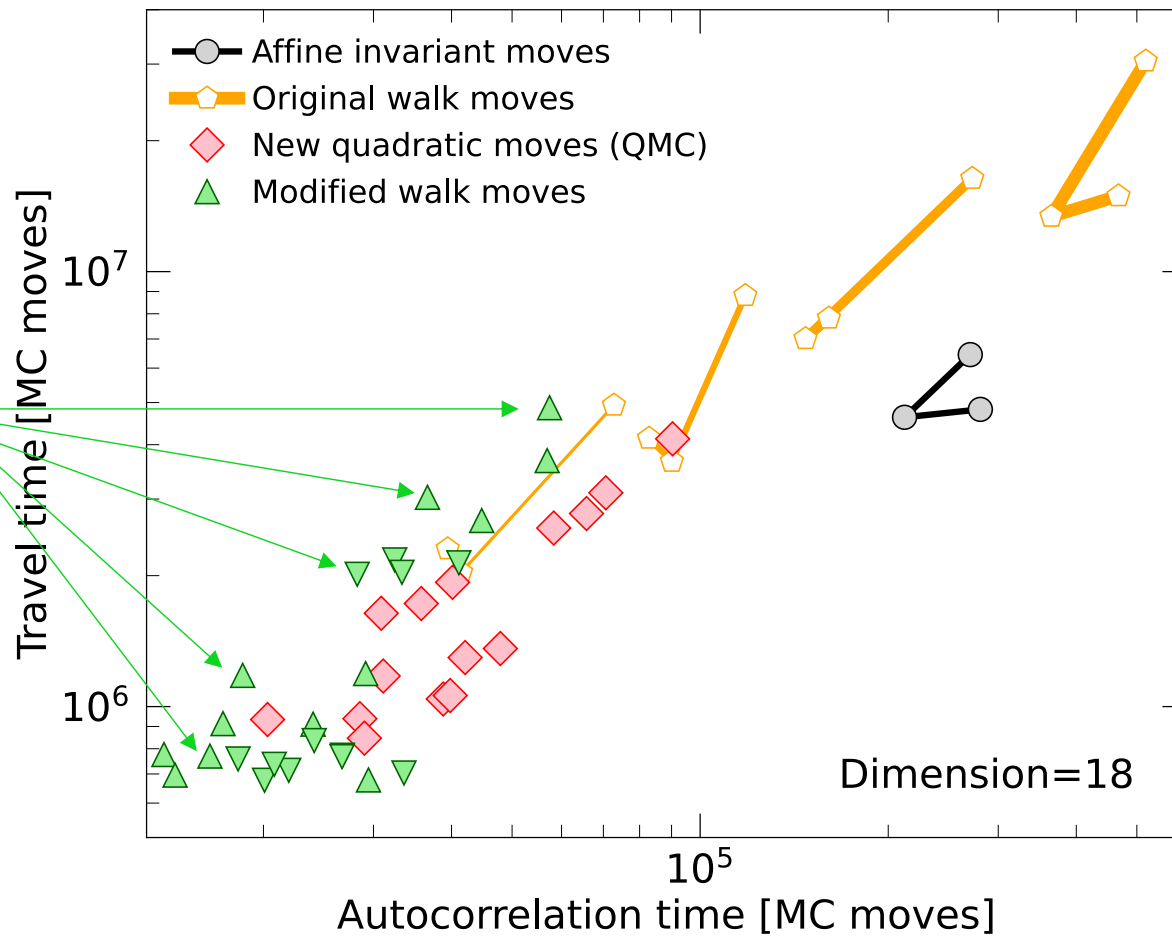
For tests in higher dimension, see: BM, *Astrophys. J* 953 (2023) 111

Comparison: QMC vs Affine Invariance Method



Test case:
Ring potential in
18 dimensions

Our Modified Walk moves



Modified "walk" moves also perform very well.

Test case:
Ring potential in
18 dimensions

Dimension=18

Our Modified Walk moves: Just Add a Scaling Factor

We follow Goodman & Weare (2010) in computing the average location all walkers in the subset,

$$\langle \vec{r} \rangle = \frac{1}{N_S} \sum_{j \in S} \vec{r}_j \quad .$$

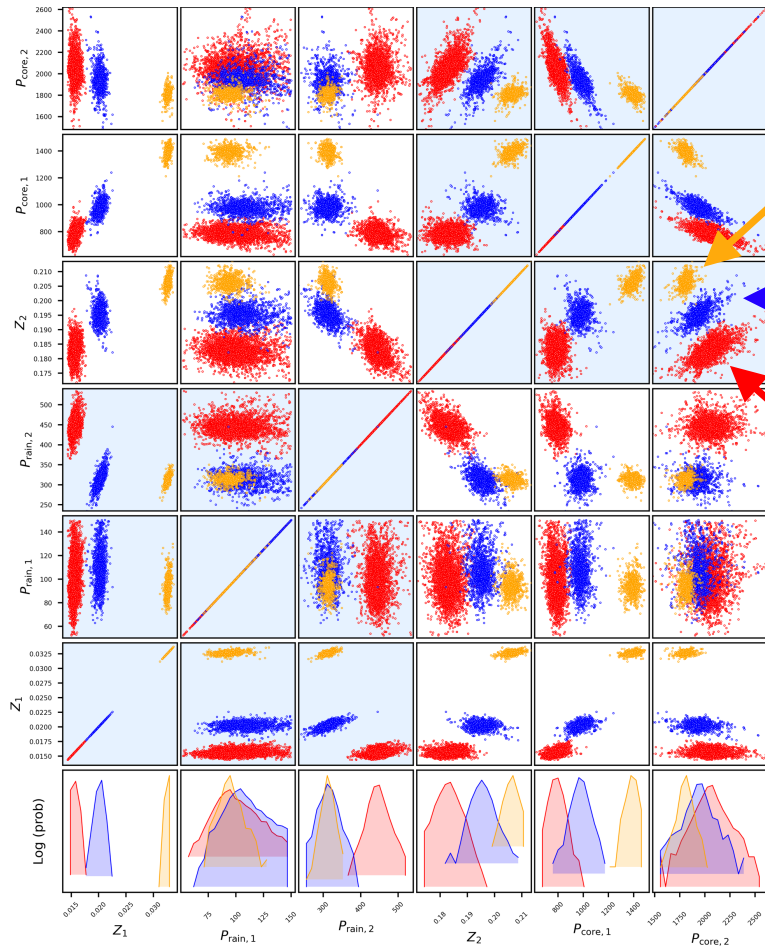
but we then modify their formula for computing the step size, W , by introducing a scaling factor a :

$$W = a \sum_{j \in S} Z_j (\vec{r}_j - \langle \vec{r} \rangle) \quad .$$

Z_j are univariate standard normal random numbers. By setting $a = 1$, one obtains the original walk moves,

Modified “walk” moves also perform very well.

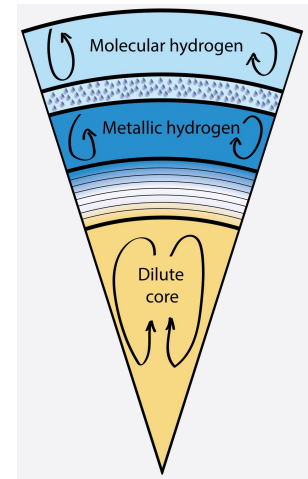
Apply our QMC method Dilute Core Models



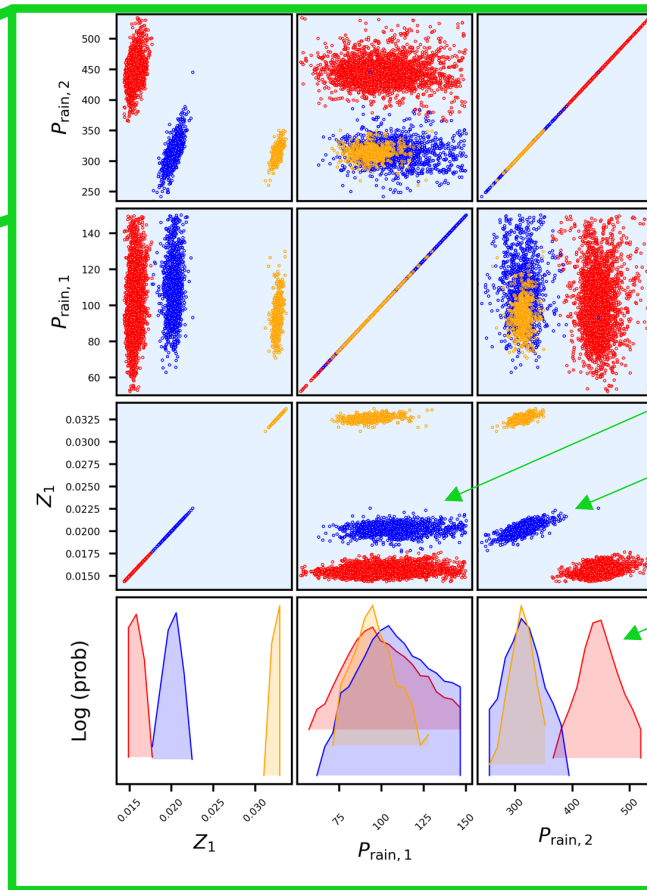
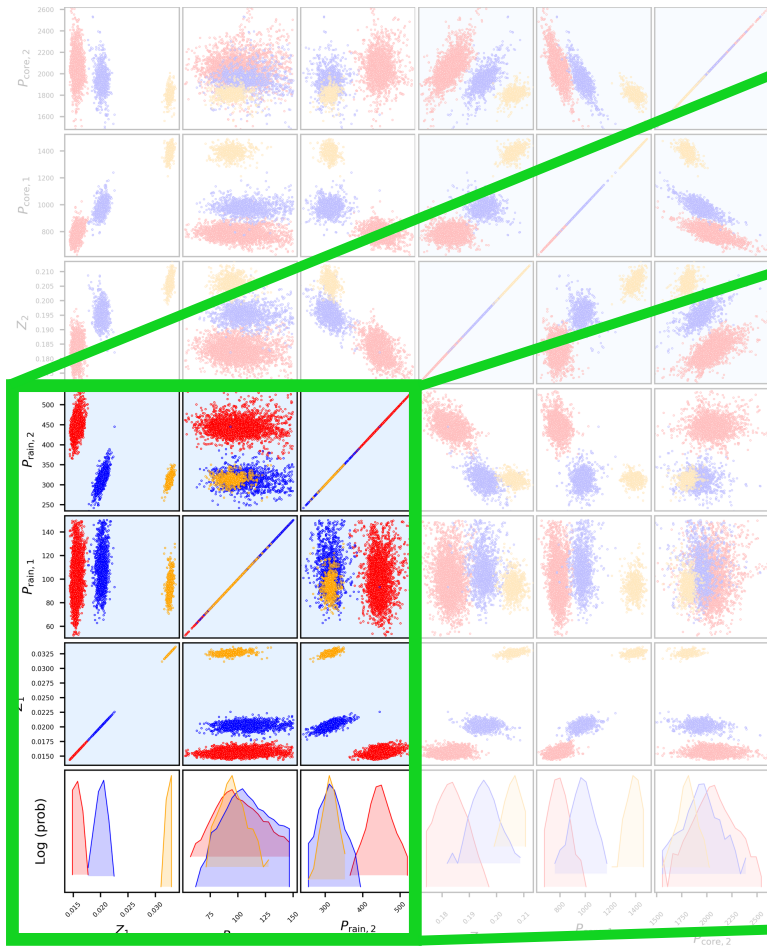
Dilute core model $T_{1\text{bar}}=166.1$ K
but EOS change by -3%

Dilute core model $T_{1\text{bar}}=170$ K

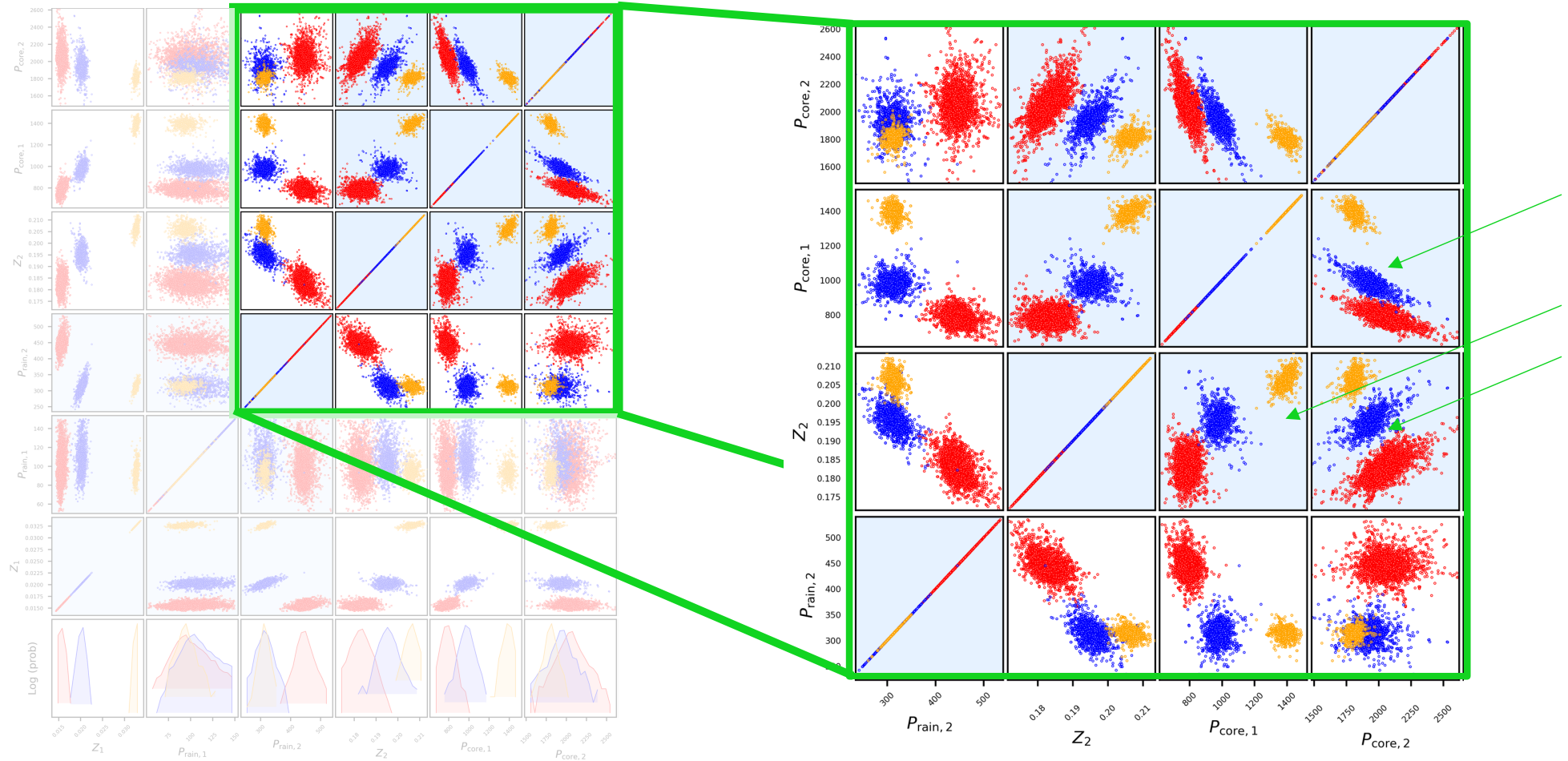
Our five-layer reference model
with a dilute core $T_{1\text{bar}}=166.1$ K



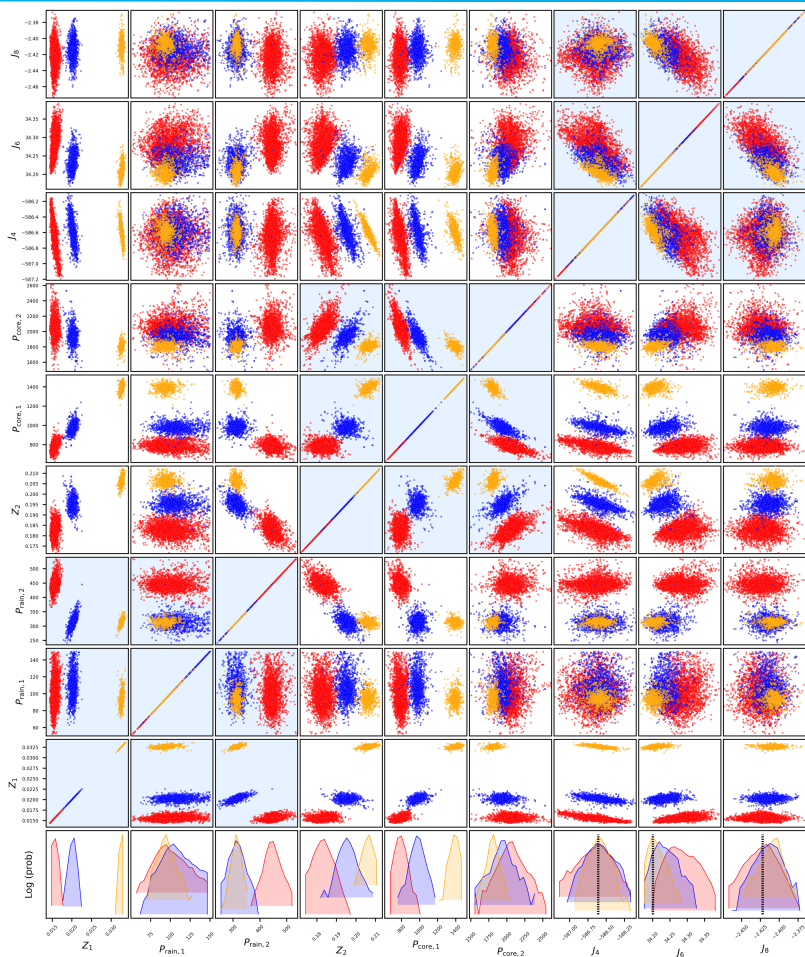
Apply our QMC method Dilute Core Models



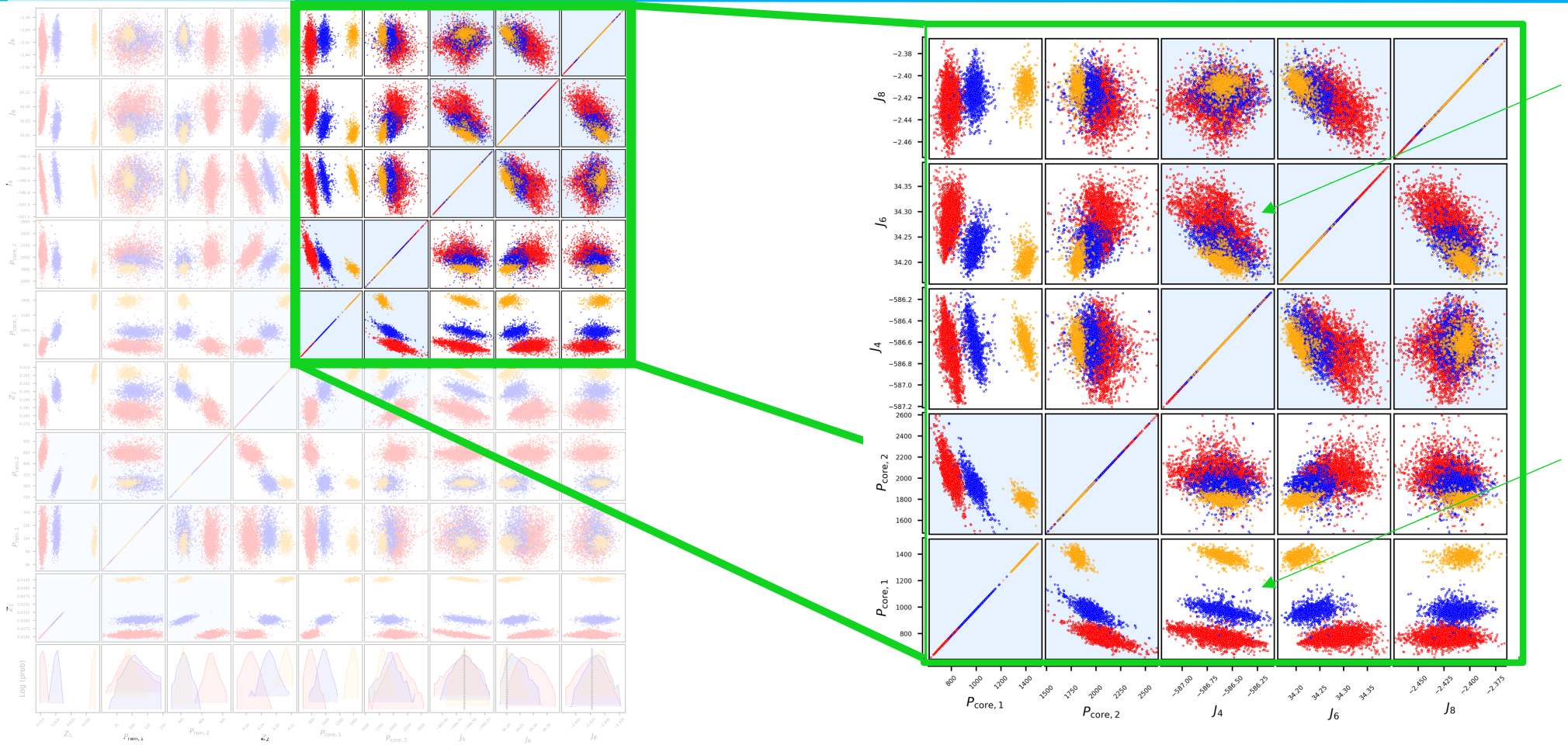
Apply our QMC method Dilute Core Models



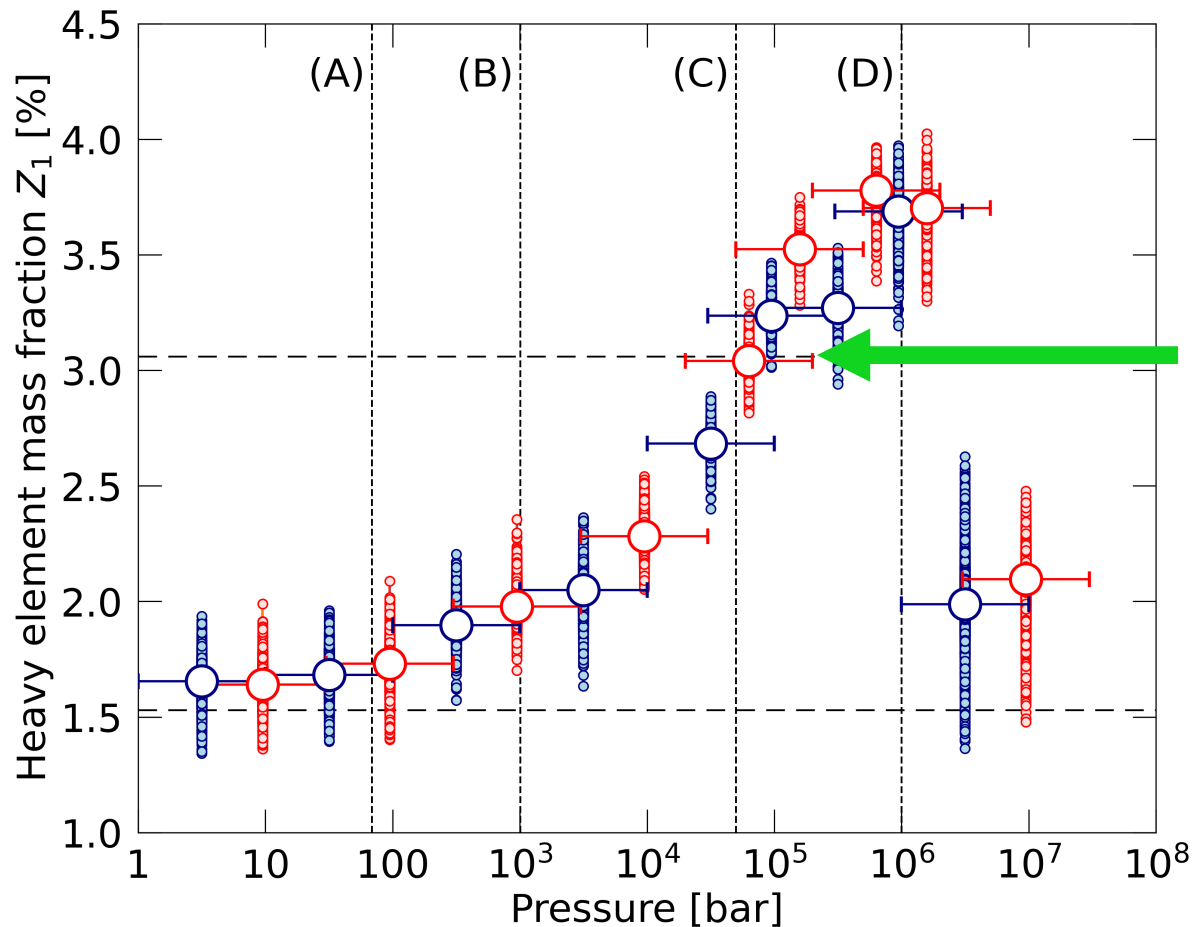
Apply our QMC method Dilute Core Models



Apply our QMC method Dilute Core Models



How does a 3% density reduction change Z_1 ?



In which pressure region are the models most sensitive?
Only from **10-100 GPa = 0.1-1 Mbar**.
Not the multi-megabar regime.

A **3% reduction** over a “decade” of pressure approximately **doubles Z_1** (given the typical assumptions in our dilute core models.)

Paper accepted for publication in ApJ

Conclusions for MC Methods

1. **Quadratic Monte Carlo** – a general-purpose sampling method
2. We recommend using a **modest number of walkers** only, between $2 \times D$ and $3 \times D$
3. We modified the walk moves by adding a scaling factor. There is **no universal scaling factor** that works for all applications.