Corrfunc: Blazing fast correlation functions on the CPU

Dr. Manodeep Sinha
ASTRO 3D CoE, Swinburne

Repo: github.com/manodeep/Corrfunc/
Collaborators: Lehman Garrison
Contributors: Andrew Hearin, Nick Hand, Gillian Beltz-Mohrmann
Corrfunc: Blazing fast correlation functions on the CPU

Dr. Manodeep Sinha
ASTRO 3D CoE, Swinburne

Repo: github.com/manodeep/Corrfunc/
Collaborators: Lehman Garrison
Contributors: Andrew Hearin, Nick Hand, Gillian Beltz-Mohrmann
How do galaxies populate halos?
LCDM Picture: Galaxies live in Halos

- Dark Matter: 26.8%
- Ordinary Matter: 4.9%
- Dark Energy: 68.3%
LCDM Picture: Galaxies live in Halos

- Dark Matter: 26.8%
- Ordinary Matter: 4.9%
- Dark Energy: 68.3%
LCDDM Picture: Galaxies live in Halos

- Dark Matter: 26.8%
- Ordinary Matter: 4.9%
- Dark Energy: 68.3%
LCDM Picture: Galaxies live in Halos

Baryon Physics (messy)
LCDM Picture: Galaxies live in Halos

Baryon Physics (messy)
What do Galaxy Surveys See?
What do Galaxy Surveys See?

Sloan Digital Skies

"@GeertHub: How many pixels of imaging data did SDSS collect since 1998?"
About 8.75e+14 pixels, or as the SDSS PR said, 500,000 HDTVs worth
2:58 AM - 14 Oct 2014

Again, the NSF denied our request to purchase half a million HDTVs to display the entire image at once. Spoilsports.
2:59 AM - 14 Oct 2014
Can be simulated
Can be simulated

Can be observed
Can be simulated

Can be observed
What is a Correlation Function?

- Measures the excess probability of finding a pair at some separation.

Groth & Peebles, 1977
What is a Correlation Function?

- Measures the **excess probability** of finding a pair at some separation.

Amounts to measuring **pair-wise separations** and computing a **histogram**
Can be simulated

Can be observed
Correlation functions are fundamental for understanding how galaxies populate halos.
Simple Code for a Correlation Function

```c
for(int64_t i=0; i<N1; i++) {
    for(int64_t j=0; j<N2; j++) {
        double dist = distance_metric(i, j);
        if(dist < mindist || dist >= maxdist) continue;

        int ibin = dist_to_bin_index(dist);
        npairs[ibin]++;
    }
}
```
for(int64_t i=0;i<N1;i++) {
    for(int64_t j=0;j<N2;j++) {
        double dist = distance_metric(i, j);
        if(dist < mindist || dist >= maxdist) continue;

        int ibin = dist_to_bin_index(dist);
        npairs[ibin]++;
    }
}
Simple Code for a Correlation Function

```c
for(int64_t i=0;i<N1;i++) {
    for(int64_t j=0;j<N2;j++) {
        double dist = distance_metric(i, j);
        if(dist < mindist || dist >= maxdist) continue;

        int ibin = dist_to_bin_index(dist);
        npairs[ibin]++;
    }
}
```
How to design Software to suit Hardware?
Flops are free, data movement is expensive
Contiguous memory access is faster
Vectorised (SIMD) code is faster
Two Complementary Optimisations

Reduce total number of computations
(reduce data movement)
Two Complementary Optimisations

Reduce total number of computations
(reduce data movement)

Efficiently perform those computations
How Corrfunc works

- Grids the particle distribution into 3D cells of size $R_{\text{max}}$
- Stores particles contiguously within each cell
- Sorts particles within a cell in $z$
- Only associates pairs of cells that may contain pairs
- Uses vectorised kernels on cell-pairs
- Uses OpenMP at the cell-pair level
Reducing Computes: 3D Grid
Reducing Computes: 3D Grid

\[ \pi R_{\text{max}}^2 \approx 9 R_{\text{max}}^2 \]
Reducing Computes: 3D Grid
Reducing Computes: 3D Grid
Reducing Computes: 3D Grid
Reducing Computes: 3D Grid

Gonnet, 2007
Reducing Computes: Min. Separations between Cell-Pairs

Sinha & Garrison, 2019, accepted
Reducing Computes: Min. Separations between Cell-Pairs

\[ \sqrt{\Delta^2_X + \Delta^2_Y} \]
Reducing Computes: Min. Separations between Cell-Pairs

\[ \Delta^2_X + \Delta^2_Y \geq R_{\text{max}}^2 \]
Reducing Computes: Min. Separations between Cell-Pairs

Sinha & Garrison, 2019, accepted
Reducing Computes: Min. Separations between Cell-Pairs

Sinha & Garrison, 2019, accepted
Reducing Computes: Min. Separations between Cell-Pairs

\((x, y)\)

\(x - X_{\text{edge}}\)

\(y - Y_{\text{edge}}\)

\((|\Delta_X| + |x - X_{\text{edge}}|)^2 + (|\Delta_Y| + |y - Y_{\text{edge}}|)^2 \geq R_{\text{max}}^2\)
Reducing Computes: Sorting

Sinha & Garrison, 2019, accepted
Speeding Up Computes: SIMD Kernels

Sinha & Garrison, 2019, accepted
## Speeding Up Computes: SIMD Kernels

<table>
<thead>
<tr>
<th>$R_{\text{max}}$</th>
<th>$w_p(r_p)$</th>
<th>$\xi(r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AVX512F</td>
<td>AVX</td>
</tr>
<tr>
<td>10.0</td>
<td>1.1×</td>
<td>1.0×</td>
</tr>
<tr>
<td>20.0</td>
<td>2.7×</td>
<td>1.8×</td>
</tr>
<tr>
<td>40.0</td>
<td>3.0×</td>
<td>1.8×</td>
</tr>
<tr>
<td>80.0</td>
<td>3.9×</td>
<td>2.3×</td>
</tr>
<tr>
<td>100.0</td>
<td>3.8×</td>
<td>2.4×</td>
</tr>
</tbody>
</table>

*Sinha & Garrison, 2019, accepted*
# Speeding Up Computes: SIMD Kernels

<table>
<thead>
<tr>
<th>$R_{\text{max}}$</th>
<th>$w_p(r_p)$</th>
<th>$\xi(r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AVX512F</td>
<td>AVX</td>
</tr>
<tr>
<td>10.0</td>
<td>1.1×</td>
<td>1.0×</td>
</tr>
<tr>
<td>20.0</td>
<td>2.7×</td>
<td>1.8×</td>
</tr>
<tr>
<td>40.0</td>
<td>3.0×</td>
<td>1.8×</td>
</tr>
<tr>
<td>80.0</td>
<td>3.9×</td>
<td>2.3×</td>
</tr>
<tr>
<td>100.0</td>
<td>3.8×</td>
<td>2.4×</td>
</tr>
</tbody>
</table>

*Sinha & Garrison, 2019, accepted*
Corrfunc Performance: Single-core
Corrfunc Performance: Single-core

Corrfunc speed-up: \(\frac{t_{\text{other}}}{t_{\text{Corrfunc}}}\)

- halotools
- kdcount
- Treecorr
- CUTE_box
- scikit-learn KDTree
- SciPy cKDTree
- mlpack RangeSearch

\(N_{\text{particles}}\)

on github:
paper/scripts/generate_code_comparison.py
Why I wrote open-sourced Corrfunc
Why I wrote open-sourced Corrfunc

- Inherited codes took ~5 mins. MCMC would have exceeded the funding duration.

- Fast private version for my specific use-case

- Created custom code for experts with 6000x speedup (took < 24 hrs to create)
Why I wrote open-sourced Corrfunc

- Inherited codes took ~5 mins. MCMC would have exceeded the funding duration.
- Fast private version for my specific use-case
- Created custom code for experts with 6000x speedup (took < 24 hrs to create)
- Demonstrated the need for a fast, flexible, open-source package
Why I wrote open-sourced Corrfunc

- Inherited codes took ~5 mins. MCMC would have exceeded the funding duration.
  - fast private version for my specific use-case
- Created custom code for experts with 6000x speedup (took < 24 hrs to create)
- Demonstrated the need for a fast, flexible, open-source package
- That initial 5 min calc. now takes ~5 secs with Corrfunc
What can Corrfunc do for you?
What can Corrfunc do for you?

- Do you compute interactions out to a max. separation?
  - Molecular dynamics, neighbour searches, flocking behaviour
- Adapt the SIMD kernels and (manually) vectorise your code
What can Corrfunc do for you?

- Do you compute interactions out to a max. separation?
  - Molecular dynamics, neighbour searches, flocking behaviour
- Adapt the SIMD kernels and (manually) vectorise your code
- Compiling C code to python (both python2/3)
What can Corrfunc do for you?

- Do you compute interactions out to a max. separation?
  - Molecular dynamics, neighbour searches, flocking behaviour
- Adapt the SIMD kernels and (manually) vectorise your code
- Compiling C code to python (both python2/3)
- Happy to share experience ...
Conclusions

- **Corrfunc** is optimised using domain knowledge, good memory access pattern, vectorisation and OpenMP

- **Corrfunc** is “blazing fast” and
  - modular, user-friendly, documented, tested, OpenMP parallel, flexible API access, …

- GPU version coming - thanks to **ADACS**

- my highest cited bib-entry for last year ([ascl.net/1703.003](ascl.net/1703.003))
Corrfunc: Blazing fast correlation functions on the CPU

Sinha, Manodeep; Garrison, Lehman

Astrophysics Source Code Library, record ascl:1703.003

03/2017

ASCL

software

2017ascl.soft03003S
Title: Corrfunc: Blazing fast correlation functions on the CPU
Authors: Sinha, Manodeep; Garrison, Lehman
Publication: Astrophysics Source Code Library, record ascl:1703.003
Publication Date: 03/2017
Origin: ASCL
Keywords: Software
Bibliographic Code: 2017ascl.soft03003S
Conclusions

- **Corrfunc** is optimised using domain knowledge, good memory access pattern, vectorisation and OpenMP

- **Corrfunc** is “blazing fast” and
  - modular, user-friendly, documented, tested, OpenMP parallel, flexible API access, …

- GPU version coming - thanks to [ADACS](https://adacs.com/)

- my highest cited bib-entry for last year ([ascl.net/1703.003](https://ascl.net/1703.003))