Uniform-$\mathcal{H}$

Bridging the Gap between $\mathcal{H}$ and $\mathcal{H}^2$

R. Kriemann

SIAM PP22

MAX PLANCK INSTITUTE
FOR MATHEMATICS IN THE SCIENCES

Hierarchical Low-Rank Formats

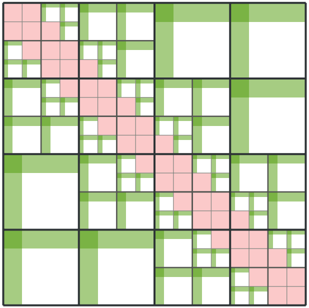
Hierarchical Low-Rank Formats

$\mathcal{H}$

- low-rank blocks represented as

$$A_{\tau,\sigma} = U_{\tau,\sigma} \cdot V_{\tau,\sigma}^H$$

- each low-rank block uses *own* row and column bases
- lax handling of admissibility
- Advantages:
  - *no dependency* between low-rank blocks due to data representation,
  - simple and efficient (parallel) arithmetic
- Disadvantages:
  - non-optimal storage costs for matrix ($\mathcal{O}(n \log n)$)

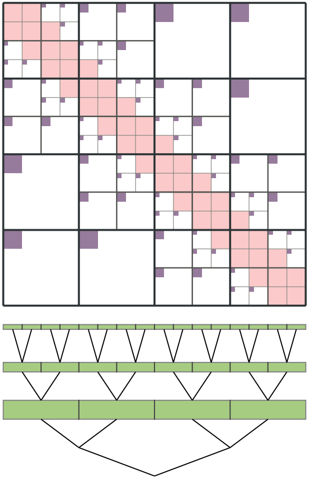


Uniform-$\mathcal{H}$

- low-rank blocks represented as

$$A_{\tau,\sigma} = \mathcal{U}_\tau \cdot S_{\tau,\sigma} \cdot \mathcal{V}_\sigma^H$$

- row/column bases *shared* by all blocks with *same* indexset
- *stricter* handling of admissibility
- Advantages:
  - optimal storage costs for matrix ($\mathcal{O}(n)$)
  - data dependency only per level per block row/column
  - *simple* and efficient arithmetic
- Disadvantages:
  - non-optimal storage cost for bases ($\mathcal{O}(n \log n)$)



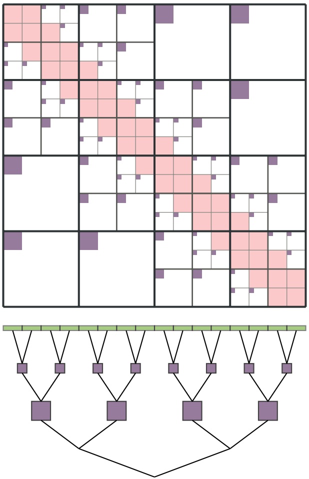
Hierarchical Low-Rank Formats

$\mathcal{H}^2$

- low-rank blocks represented as

$$A_{\tau,\sigma} = \mathcal{U}_\tau \cdot S_{\tau,\sigma} \cdot \mathcal{V}_\sigma^H$$

- row/column bases *shared* by all blocks with *non-disjoint* indexsets
- bases are *nested*
- *strict* handling of admissibility
- Advantages:
  - optimal storage costs for matrix and bases ($\mathcal{O}(n)$)
- Disadvantages:
  - *high dependency* between low-rank blocks,
  - only implicit block/basis data
  - much more complicated arithmetic



Uniform-$\mathcal{H}$ Arithmetic

Uniform-$\mathcal{H}$ Construction

Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Uniform-$\mathcal{H}$ Construction

Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

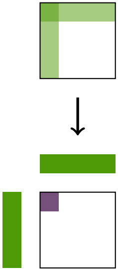
Algorithm (simplification of $\mathcal{H}^2$-construction[1])

first low-rank block $A_{\tau,\sigma} = U_{\tau,\sigma} \cdot V_{\tau,\sigma}^H$

1. QR-factorization:

$$\mathcal{W} R_w = U_{\tau,\sigma}$$
$$\mathcal{X} R_x = V_{\tau,\sigma}$$

2. $\mathcal{U}_\tau := \mathcal{W}; \quad \mathcal{V}_\sigma := \mathcal{X}$
3. $S_{\tau,\sigma} := R_w R_x^H$



[1] S. Börm: "Efficient numerical methods for non-local operators. $\mathcal{H}^2$-matrix compression, algorithms and analysis.", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Construction

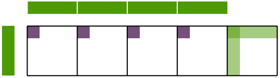
Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Algorithm (simplification of $\mathcal{H}^2$-construction[1])

Update row cluster basis for

$$\left(\mathcal{U}_\tau S_{\tau,\sigma_1} \mathcal{V}_{\sigma_1}^H \quad \cdots \quad \mathcal{U}_\tau S_{\tau,\sigma_i} \mathcal{V}_{\sigma_i}^H \quad \mathcal{W} T_{\tau,\sigma} \mathcal{X}^H \right)$$



[1] S. Börm: "Efficient numerical methods for non-local operators. $\mathcal{H}^2$-matrix compression, algorithms and analysis.", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Construction

Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Algorithm (simplification of $\mathcal{H}^2$-construction[1])

Update row cluster basis for

$$\begin{pmatrix} \mathcal{U}_\tau S_{\tau,\sigma_1} \mathcal{V}_{\sigma_1}^H & \cdots & \mathcal{U}_\tau S_{\tau,\sigma_i} \mathcal{V}_{\sigma_i}^H & \mathcal{W} T_{\tau,\sigma} \mathcal{X}^H \end{pmatrix}$$

$$= \begin{pmatrix} \mathcal{U}_\tau & \mathcal{W} \end{pmatrix} \begin{pmatrix} S_{\tau,\sigma_1} & \cdots & 0 \\ 0 & \cdots & T_{\tau,\sigma} \end{pmatrix} \begin{pmatrix} \mathcal{V}_{\sigma_1} & & \\ & \ddots & \\ & & \mathcal{X} \end{pmatrix}^H$$

[1] S. Börm: "Efficient numerical methods for non-local operators: $\mathcal{H}^2$-matrix compression, algorithms and analysis", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Construction

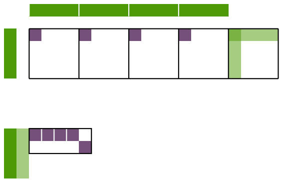
Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Algorithm (simplification of $\mathcal{H}^2$-construction[1])

Update row cluster basis for

$$\begin{aligned}&\left(\mathcal{U}_\tau S_{\tau,\sigma_1} \mathcal{V}_{\sigma_1}^H \quad \cdots \quad \mathcal{U}_\tau S_{\tau,\sigma_i} \mathcal{V}_{\sigma_i}^H \quad \mathcal{W} T_{\tau,\sigma} \mathcal{X}^H\right)\\ =&\left(\mathcal{U}_\tau \quad \mathcal{W}\right)\begin{pmatrix} S_{\tau,\sigma_1} & \cdots & 0 \\ 0 & \cdots & T_{\tau,\sigma}\end{pmatrix}\end{aligned}$$



[1] S. Börm: "Efficient numerical methods for non-local operators. $\mathcal{H}^2$-matrix compression, algorithms and analysis.", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Construction

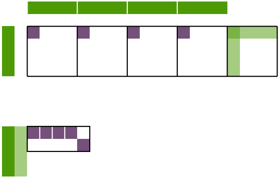
Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Algorithm (simplification of $\mathcal{H}^2$-construction[1])

Update row cluster basis for

$$
\begin{aligned}
&\left(\mathcal{U}_\tau S_{\tau,\sigma_1} \mathcal{V}_{\sigma_1}^H \quad \cdots \quad \mathcal{U}_\tau S_{\tau,\sigma_i} \mathcal{V}_{\sigma_i}^H \quad \mathcal{W} T_{\tau,\sigma} \mathcal{X}^H\right) \\
&= \left(\mathcal{U}_\tau \quad \mathcal{W}\right) \begin{pmatrix} S_{\tau,\sigma_1} & \cdots & 0 \\ 0 & \cdots & T_{\tau,\sigma} \end{pmatrix} \\
&= \left(\mathcal{U}_\tau \quad \mathcal{W}\right) R^H Q^H
\end{aligned}
$$



[1] S. Börm: "Efficient numerical methods for non-local operators. $\mathcal{H}^2$-matrix compression, algorithms and analysis.", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Construction

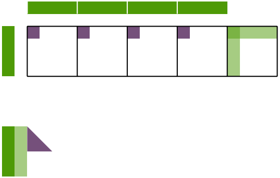
Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Algorithm (simplification of $\mathcal{H}^2$-construction[1])

Update row cluster basis for

$$
\begin{aligned}
&\left(\mathcal{U}_\tau S_{\tau,\sigma_1} \mathcal{V}_{\sigma_1}^H \quad \cdots \quad \mathcal{U}_\tau S_{\tau,\sigma_i} \mathcal{V}_{\sigma_i}^H \quad \mathcal{W} T_{\tau,\sigma} \mathcal{X}^H\right) \\
&= \left(\mathcal{U}_\tau \quad \mathcal{W}\right) \begin{pmatrix} S_{\tau,\sigma_1} & \cdots & 0 \\ 0 & \cdots & T_{\tau,\sigma} \end{pmatrix} \\
&= \left(\mathcal{U}_\tau \quad \mathcal{W}\right) R^H Q^H \\
&= \left(\mathcal{U}_\tau \quad \mathcal{W}\right) R^H
\end{aligned}
$$



[1] S. Börm: "Efficient numerical methods for non-local operators. $\mathcal{H}^2$-matrix compression, algorithms and analysis.", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Construction

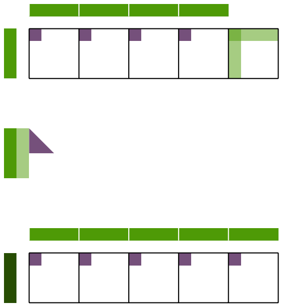
Goal

Use existing method to construct $\mathcal{H}$-matrices with standard low-rank blocks and convert on-the-fly to Uniform-$\mathcal{H}$-matrix.

Algorithm (simplification of $\mathcal{H}^2$-construction[1])

Update row cluster basis for

$$\begin{aligned}
&\left(\mathcal{U}_\tau S_{\tau,\sigma_1} \mathcal{V}_{\sigma_1}^H \quad \cdots \quad \mathcal{U}_\tau S_{\tau,\sigma_i} \mathcal{V}_{\sigma_i}^H \quad \mathcal{W} T_{\tau,\sigma} \mathcal{X}^H\right) \\
= &\left(\mathcal{U}_\tau \quad \mathcal{W}\right) \begin{pmatrix} S_{\tau,\sigma_1} & \cdots & 0 \\ 0 & \cdots & T_{\tau,\sigma} \end{pmatrix} \\
= &\left(\mathcal{U}_\tau \quad \mathcal{W}\right) R^H Q^H \\
= &\left(\mathcal{U}_\tau \quad \mathcal{W}\right) R^H \\
\approx &\, \tilde{\mathcal{U}}_\tau
\end{aligned}$$



with basis approximation defined by precision ε.

[1] S. Börm: "Efficient numerical methods for non-local operators. $\mathcal{H}^2$-matrix compression, algorithms and analysis.", EMS Tracts Math. 14, 2010

Uniform-$\mathcal{H}$ Arithmetic

Idea

Extend $\mathcal{H}$-arithmetic by updating bases when a matrix block is modified.

Use *accumulated* $\mathcal{H}$-arithmetic to reduce number of block updates.

Matrix multiplication $C = C + AB$

function HMUL(**inout**: C, **in**: $\mathcal{A}_C, \mathcal{P}_C$)
    for all pending upd. $(A_i, B_i) \in \mathcal{P}_C$ **do**
        if (A_i, B_i) is computable **then**
            $\mathcal{A}_C := \mathcal{A}_C + A_i \cdot B_i$

    if C has sub-blocks **then**
        for all sub-blocks C_{ij} **do**
            `Hmul`$(C_{ij}, \mathcal{A}_C|_{C_{ij}}, \mathcal{P}_C|_{C_{ij}})$;
    else
        $C := C + \mathcal{A}_C$;

Uniform-$\mathcal{H}$ Arithmetic

Idea

Extend $\mathcal{H}$-arithmetic by updating bases when a matrix block is modified.

Use *accumulated* $\mathcal{H}$-arithmetic to reduce number of block updates.

Matrix multiplication $C = C + AB$

function HMUL(**inout**: C, **in**: $\mathcal{A}_C, \mathcal{P}_C$)
  for all pending upd. $(A_i, B_i) \in \mathcal{P}_C$ **do**
    if (A_i, B_i) is computable **then**
      $\mathcal{A}_C := \mathcal{A}_C + A_i \cdot B_i$

  if C has sub-blocks **then**
    for all sub-blocks C_{ij} **do**
      Hmul($C_{ij}, \mathcal{A}_C|_{C_{ij}}, \mathcal{P}_C|_{C_{ij}}$);
  else
    $C := C + \mathcal{A}_C$;

function UNIHMUL(**inout**: C, **in**: $\mathcal{A}_C, \mathcal{P}_C$)
  for all pending upd. $(A_i, B_i) \in \mathcal{P}_C$ **do**
    if (A_i, B_i) is computable **then**
      $\mathcal{A}_C := \mathcal{A}_C + A_i \cdot B_i$

  if C has sub-blocks **then**
    for all sub-blocks C_{ij} **do**
      UniHmul($C_{ij}, \mathcal{A}_C|_{C_{ij}}, \mathcal{P}_C|_{C_{ij}}$);
  else
    $C := C + \mathcal{A}_C$;
    Update Bases;

Uniform-$\mathcal{H}$ Arithmetic

Idea

Extend $\mathcal{H}$-arithmetic by updating bases when a matrix block is modified.

Use *accumulated* $\mathcal{H}$-arithmetic to reduce number of block updates.

Matrix multiplication $C = C + AB$

```
function HMul(inout: C, in: A_C, P_C)
  for all pending upd. (A_i, B_i) ∈ P_C do
    if (A_i, B_i) is computable then
      A_C := A_C + A_i · B_i

  if C has sub-blocks then
    for all sub-blocks C_ij do
      Hmul(C_ij, A_C|C_ij, P_C|C_ij);
  else
    C := C + A_C;
```

```
function UniHMul(inout: C, in: A_C, P_C)
  for all pending upd. (A_i, B_i) ∈ P_C do
    if (A_i, B_i) is computable then
      A_C := A_C + A_i · B_i  (optimized)

  if C has sub-blocks then
    for all sub-blocks C_ij do
      UniHmul(C_ij, A_C|C_ij, P_C|C_ij);
  else
    C := C + A_C;
    Update Bases;
```

Uniform-$\mathcal{H}$ Arithmetic

Optimized evaluation of updates $A_i B_i$:

$$\sum_i A_i B_i = \sum_{ulr \times ulr} A_i B_i + \sum_{g \times ulr} A_i B_i + \sum_{ulr \times g} A_i B_i + \sum_{g \times g} A_i B_i$$

$$\sum_{\text{ulr} \times \text{ulr}} A_i B_i = \mathcal{U}_\tau \left(\sum_{\text{ulr} \times \text{ulr}} S_i^A \mathcal{V}_{\sigma_i}^H \mathcal{U}_{\tau_i} S_i^B \right) \mathcal{V}_\sigma^H$$

$$\sum_{\text{g} \times \text{ulr}} A_i B_i = \left(\sum_{\text{g} \times \text{ulr}} A_i \mathcal{U}_{\tau_i} S_i^B \right) \mathcal{V}_{cls}^H$$

$$\sum_{\text{ulr} \times \text{g}} A_i B_i = \mathcal{U}_\tau \left(\sum_{\text{ulr} \times \text{g}} S_i^A \mathcal{V}_{\tau_i}^H B_i \right)$$

$$\sum_{\text{g} \times \text{g}} A_i B_i = \sum_{\text{g} \times \text{g}} A_i B_i$$

Uniform-$\mathcal{H}$ Arithmetic

Optimized evaluation of updates $A_i B_i$:

$$\sum_i A_i B_i = \sum_{ulr \times ulr} A_i B_i + \sum_{\mathrm{g} \times ulr} A_i B_i + \sum_{ulr \times \mathrm{g}} A_i B_i + \sum_{\mathrm{g} \times \mathrm{g}} A_i B_i$$

$$\sum_{\mathrm{ulr} \times \mathrm{ulr}} A_i B_i = \mathcal{U}_\tau \left(\sum_{\mathrm{ulr} \times \mathrm{ulr}} S_i^A \mathcal{V}_{\sigma_i}^H \mathcal{U}_{\tau_i} S_i^B \right) \mathcal{V}_\sigma^H$$

$$\sum_{\mathrm{g} \times \mathrm{ulr}} A_i B_i = \left(\sum_{\mathrm{g} \times \mathrm{ulr}} A_i \mathcal{U}_{\tau_i} S_i^B \right) \mathcal{V}_{cls}^H$$

$$\sum_{\mathrm{ulr} \times \mathrm{g}} A_i B_i = \mathcal{U}_\tau \left(\sum_{\mathrm{ulr} \times \mathrm{g}} S_i^A \mathcal{V}_{\tau_i}^H B_i \right)$$

$$\sum_{\mathrm{g} \times \mathrm{g}} A_i B_i = \sum_{\mathrm{g} \times \mathrm{g}} A_i B_i$$

Red: dense matrices.

Numerical Results

Laplace SLP

Solve

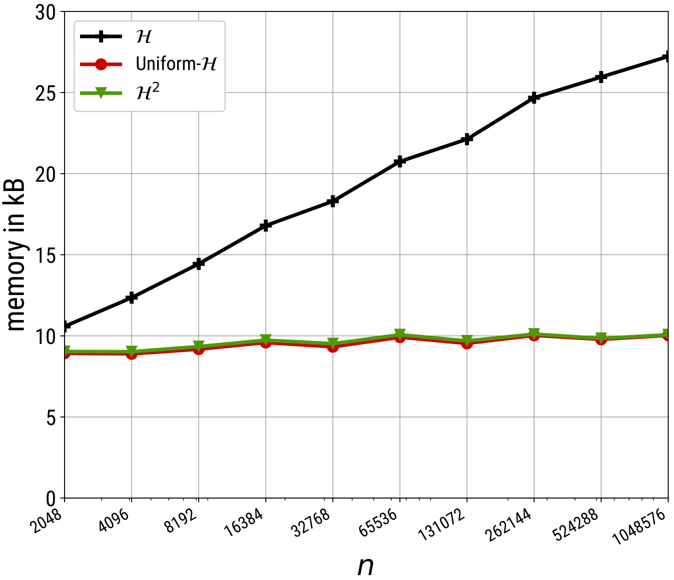
$$\int_\Gamma \frac{1}{|x-y|_2} u(x) dy = f(x), \quad x \in \Gamma = \partial\Omega \subset \mathbb{R}^3.$$

Matrix entries:

$$M_{ij} = \int_{t_i} \int_{t_j} \frac{1}{|x_i - x_j|_2} dx_i dx_j$$

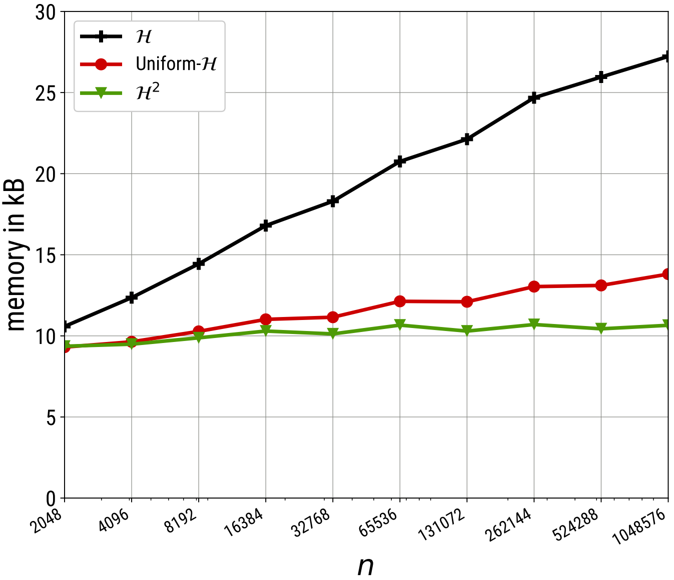
Laplace SLP

Matrix Memory (per DoF)



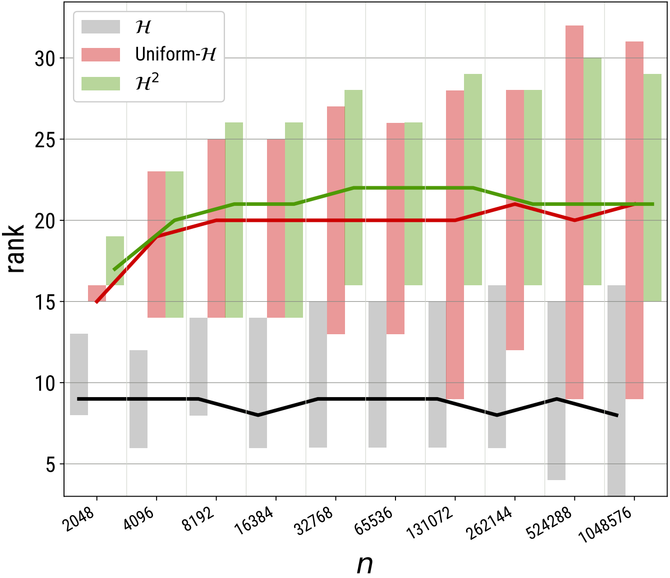
Laplace SLP

Total Memory (per DoF)



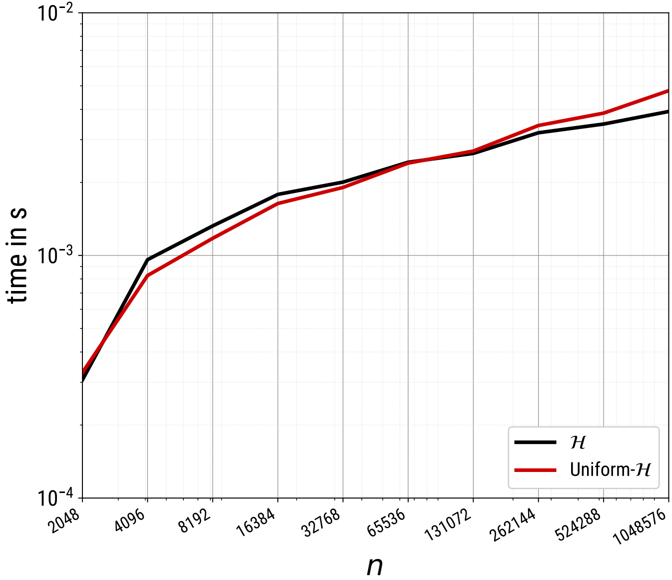
Laplace SLP

Ranks



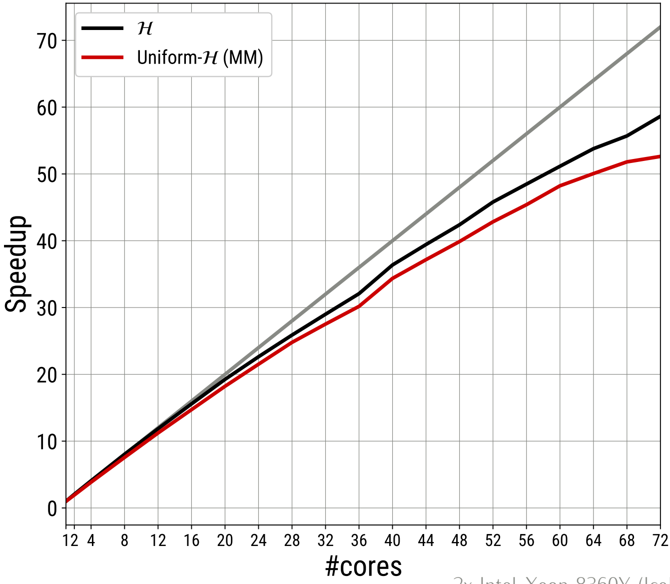
Laplace SLP

Runtime for Matrix Multiplication (per DoF)



Laplace SLP

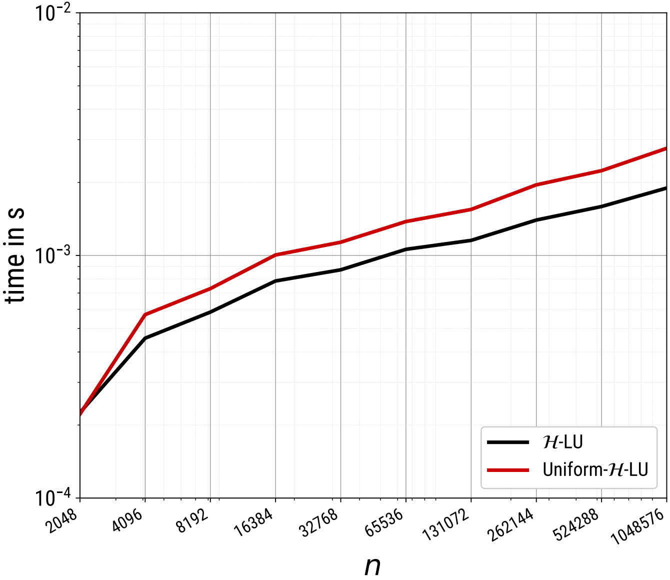
Parallel Speedup for Matrix Multiplication



2x Intel Xeon 8360Y (IceLake)

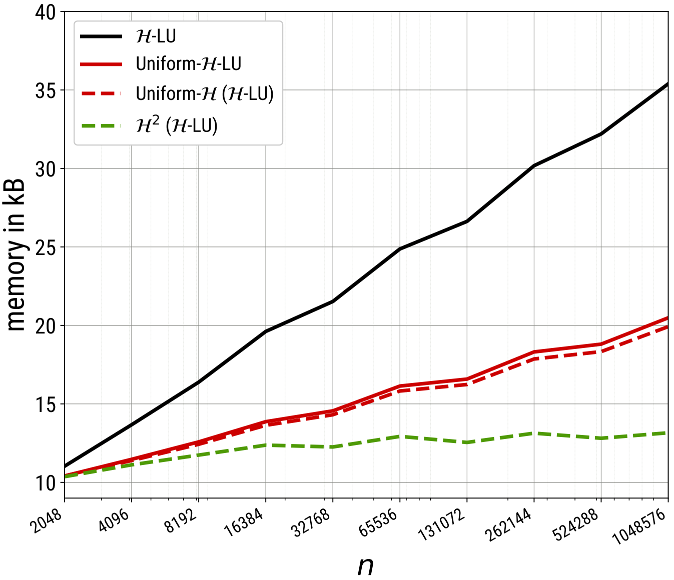
Laplace SLP

Runtime for LU factorization (per DoF)



Laplace SLP

Memory for LU factorization (per DoF)



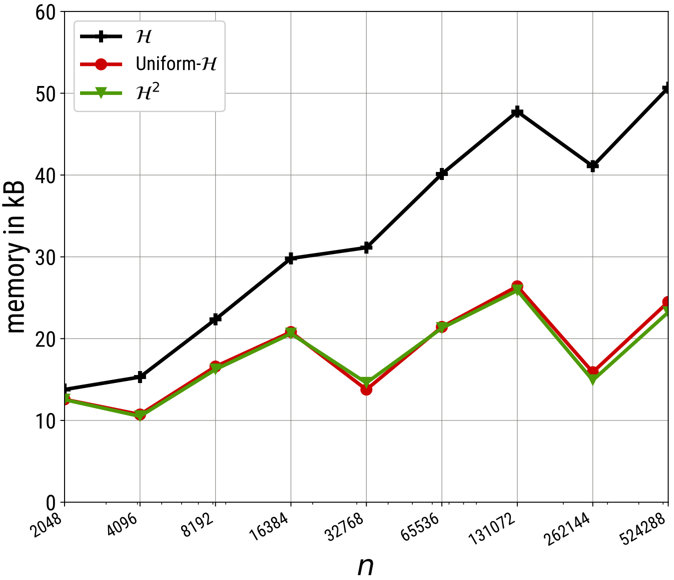
Gaussian Kernel

$$M_{ij} = e^{-\gamma|x_i - x_j|^2}$$

$$x_i \in [0,1]^3 \text{ (random)}$$

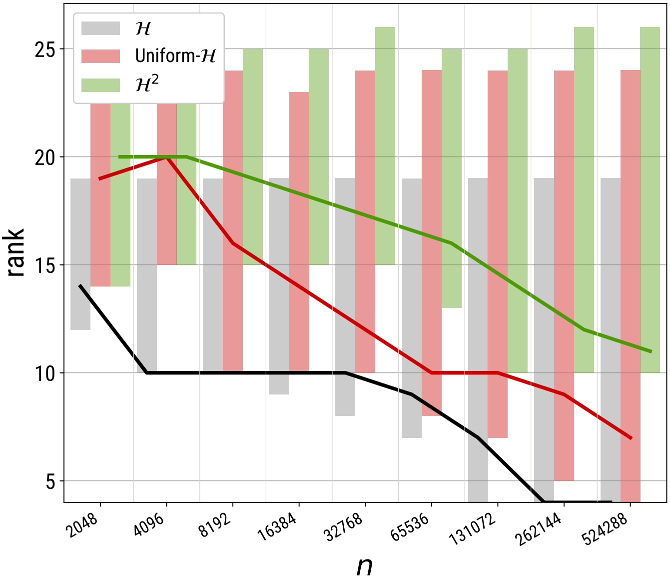
Gaussian Kernel

Total Memory (per DoF)



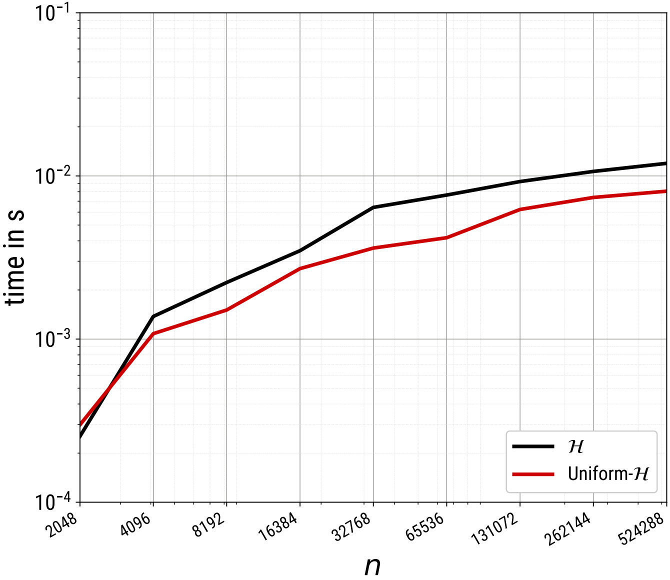
Gaussian Kernel

Ranks



Gaussian Kernel

Runtime for Matrix Multiplication (per DoF)



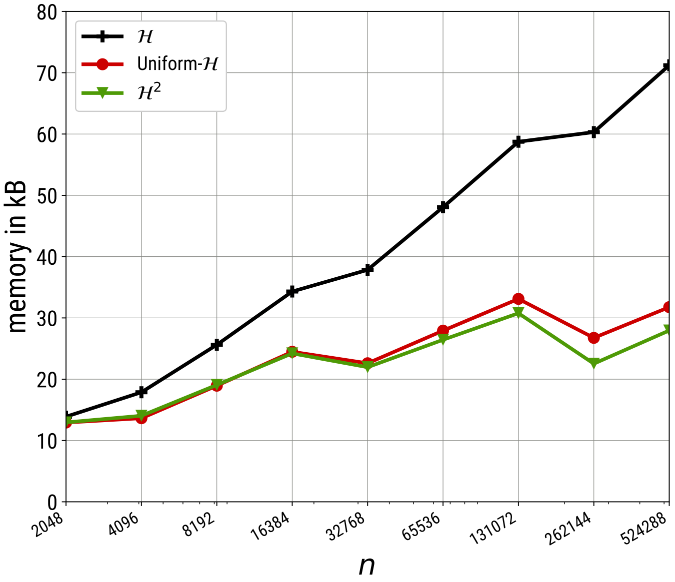
Matérn Covariance

$$M_{ij} = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\|x_i - x_j\|}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\|x_i - x_j\|}{\ell} \right)$$

$$x_i \in [0,1]^3 \text{ (random)}$$

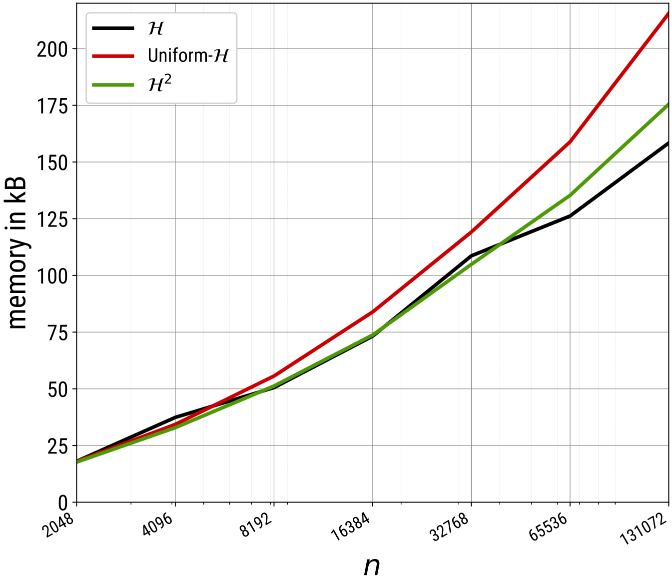
Matérn Covariance

Total Memory (per DoF)



Matérn Covariance

Memory for LU factorization (per DoF)



Conclusion

Conclusion

Compression

Uniform-$\mathcal{H}$ is (normally) much more efficient than $\mathcal{H}$ and close to $\mathcal{H}^2$.

Arithmetic

Uniform-$\mathcal{H}$ is comparable with $\mathcal{H}$.

But what about $\mathcal{H}^2$?

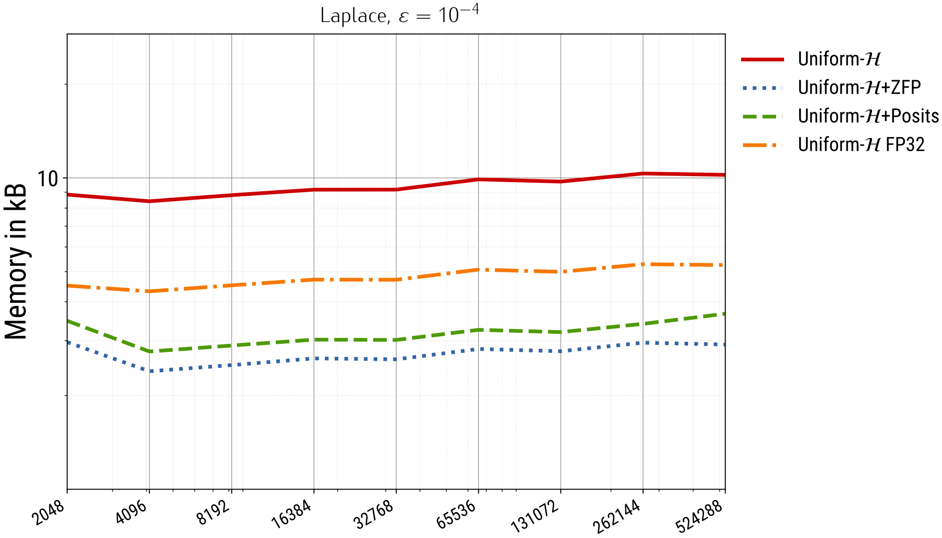
When to use?

If both compression *and* arithmetic are needed.

Can we do more?

Can we do more?

Use ZFP[1] or Posits[2] to further compress HLR data.

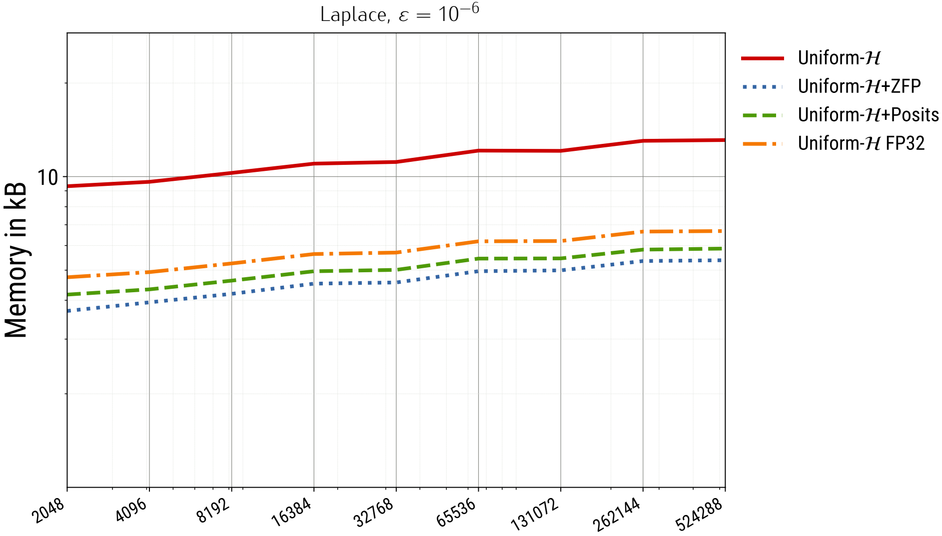


[1] Lindstrom: "Fixed-Rate Compressed Floating-Point Arrays.", IEEE Trans. on Vis. and Computer Graphics, 2014

[2] Gustafson, Yonemoto: "Beating Floating Point at its Own Game: Posit Arithmetic.", Supercomp. Frontiers and Innovations, 2017

Can we do more?

Use ZFP[1] or Posits[2] to further compress HLR data.

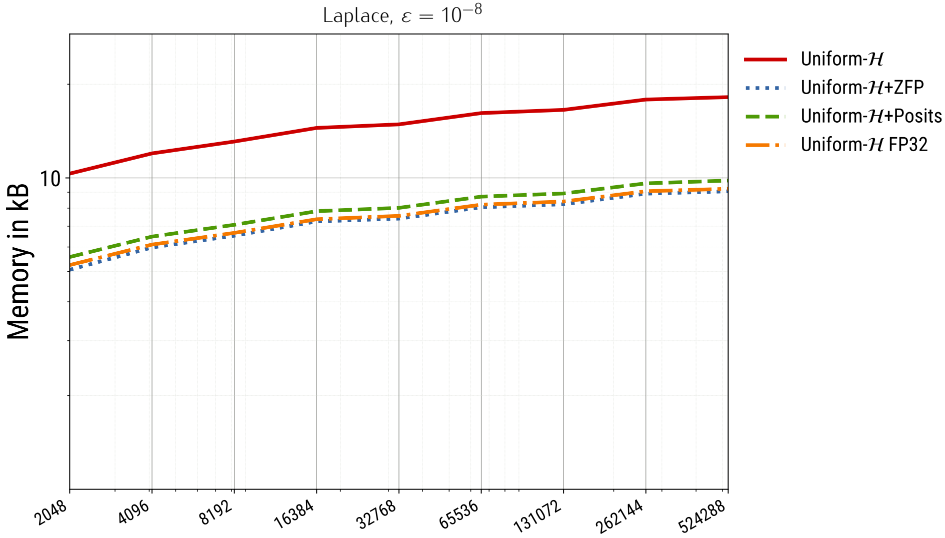


[1] Lindstrom: "Fixed-Rate Compressed Floating-Point Arrays.", IEEE Trans. on Vis. and Computer Graphics, 2014

[2] Gustafson, Yonemoto: "Beating Floating Point at its Own Game: Posit Arithmetic.", Supercomp. Frontiers and Innovations, 2017

Can we do more?

Use ZFP[1] or Posits[2] to further compress HLR data.

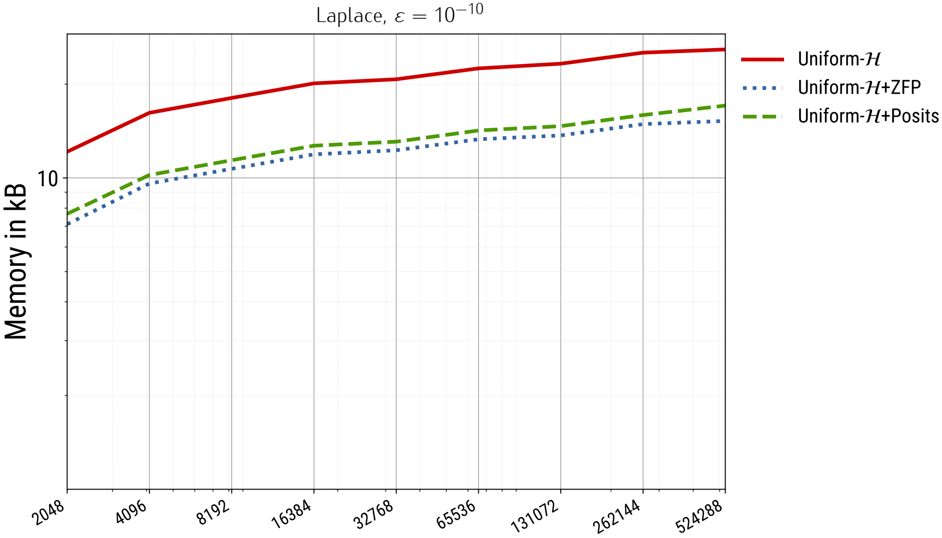


[1] Lindstrom: "Fixed-Rate Compressed Floating-Point Arrays.", IEEE Trans. on Vis. and Computer Graphics, 2014

[2] Gustafson, Yonemoto: "Beating Floating Point at its Own Game: Posit Arithmetic.", Supercomp. Frontiers and Innovations, 2017

Can we do more?

Use ZFP[1] or Posits[2] to further compress HLR data.



[1] Lindstrom: "Fixed-Rate Compressed Floating-Point Arrays.", IEEE Trans. on Vis. and Computer Graphics, 2014

[2] Gustafson, Yonemoto: "Beating Floating Point at its Own Game: Posit Arithmetic.", Supercomp. Frontiers and Innovations, 2017

Thank You

MAX PLANCK INSTITUTE
FOR MATHEMATICS IN THE SCIENCES