Computing 1152 cases?
You better use a computer!

Son Nguyen

under the guide of Dr. Martin Mohlenkamp
Ohio University, Mathematics Department

Winter 2011

Introduction

1. We need to compute integrals that include terms like:
$\sum_{a \neq b} w_1(|\gamma_a - \gamma_b|) \sum_{m \neq n} w_1(|\gamma_m - \gamma_n|) \sum_{u \neq v} w_1(|\gamma_u - \gamma_v|)$
2. Depending whether a, b, m, n, u, v are distinct or overlap, we have overall 1154 cases to compute.
3. When all indices are distinct we have the following case, which also serve as the general expression:

$$\sum_{a,b} \sum_{m,n} \sum_{u,v} \int \int \int \int \int \int$$
$$w_1(|\gamma_a - \gamma_b|) w_2(|\gamma_m - \gamma_n|)) w_3(|\gamma_u - \gamma_v|)$$
$$\phi_a(\gamma_a)\phi_b(\gamma_b)\phi_m(\gamma_m)\phi_n(\gamma_n)\phi_u(\gamma_u)\phi_v(\gamma_v)\Delta$$
$$\mathrm{d}\gamma_a d\gamma_b d\gamma_m d\gamma_n d\gamma_u d\gamma_v$$

- Where Δ is:

$$\begin{vmatrix} \theta_a(\gamma_a) & \theta_a(\gamma_b) & \theta_a(\gamma_m) & \theta_a(\gamma_n) & \theta_a(\gamma_u) & \theta_a(\gamma_v) \\ \theta_b(\gamma_a) & \theta_b(\gamma_b) & \theta_b(\gamma_m) & \theta_b(\gamma_n) & \theta_b(\gamma_u) & \theta_b(\gamma_v) \\ \theta_m(\gamma_a) & \theta_m(\gamma_b) & \theta_m(\gamma_m) & \theta_m(\gamma_n) & \theta_m(\gamma_u) & \theta_m(\gamma_v) \\ \theta_n(\gamma_a) & \theta_n(\gamma_b) & \theta_n(\gamma_m) & \theta_n(\gamma_n) & \theta_n(\gamma_u) & \theta_n(\gamma_v) \\ \theta_u(\gamma_a) & \theta_u(\gamma_b) & \theta_u(\gamma_m) & \theta_u(\gamma_n) & \theta_u(\gamma_u) & \theta_u(\gamma_v) \\ \theta_v(\gamma_a) & \theta_v(\gamma_b) & \theta_v(\gamma_m) & \theta_v(\gamma_n) & \theta_v(\gamma_u) & \theta_v(\gamma_v) \end{vmatrix}$$

Although the analysis can be done by hand, to actually implement these formulars requires automatic generation of all the cases.
$\rightarrow$ **It is hard to do by hand!**

So how to compute those things?

1. Generate all the cases, which is called by **geminal**, that (*) can take by the comparing the values of indices, then group them by their similarity. This is done using **Algorithm 1**
2. For each geminal: Expand the determinant Δ, and group them by their similarity. This task is done using **Algorithm 2**
3. The next step is to apply reductions to the resulting cases. This is a work in progress. See the other poster.

Notations

1. Geminals are presented by list of pair. E.g: $(0,1),(1,2),(2,3)$ means:
$$\sum_{a,b}\sum_{n}\int\int\int w_1(\gamma_a-\gamma_b)\ w_2(\gamma_a-\gamma_n)_3(\gamma_b-\gamma_n)$$
$$\phi_a(\gamma_a)\phi_b(\gamma_b)\phi_n(\gamma_n)$$
$$\begin{vmatrix}\theta_a(\gamma_a) & \theta_a(\gamma_b) & \theta_a(\gamma_n)\\ \theta_b(\gamma_a) & \theta_b(\gamma_b) & \theta_b(\gamma_n)\\ \theta_n(\gamma_a) & \theta_n(\gamma_b) & \theta_n(\gamma_n)\end{vmatrix}\mathrm{d}\gamma_a d\gamma_b d\gamma_n$$
2. **crea(x)**: Return a list containing$x$ and the length of x plus 1.
$$[0,1]\rightarrow[0,1,2]$$
3. **detg(x):** Return a list contains all elements of the determinant of x.
$$\begin{bmatrix}a & b\\ c & d\end{bmatrix}\rightarrow[(a,d,1),(b,c,-1)],$$
4. **order1(x)**: Order x by the first number in its elements.
$$[(0,1),(2,3),(1,2)]\rightarrow[(0,1),(1,2),(2,3)]$$

Notations

1. **order2(x)**: Order each elements in x.

$$[(0,1),(3,2),(2,1)] \rightarrow [(0,1),(2,3),(1,2)]$$

2. **Grouping duplicate**: If $\alpha = [(0,1),(1,2),(2,1),3]$, and $\beta = [(0,1),(1,2),(1,2),4]$, and α, β are duplicates, then grouping α and β means replacing α and β by $\gamma = [(0,1),(1,2),(2,1),7]$
3. **Corresponding Matrix**: Let α be $[(0,1),(1,2),(2,1),3]$. The corresponding matrix of α is:

$$\begin{bmatrix} (0,0) & (0,1) & (0,2) \\ (1,0) & (1,1) & (1,2) \\ (2,0) & (2,1) & (2,2) \end{bmatrix}$$

Algorithm 1

1. **Step 1**: Generate all elements in form of $[(0, 1), (m, n), (u, v)]$ that satisfies:
   - $m = crea(0, 1)$; $n = crea(0, 1, m)$; $u = crea(0, 1, m, n)$ $v = crea(0, 1, m, n, u)$
   - $m \neq n; u \neq v$
2. **Step 2**: Order the output in step 1 by function *order2*, and group duplicates.
3. **Step 3**: Interchange $0 \leftrightarrow 1$, and apply function *order2* to the output in step 2 to group duplicates
4. **Step 4**: Interchange $2 \leftrightarrow 3$ and then apply function *order2* for elements that contains 2, 3 in the output of step 3 to group duplicates.

Result

- There are 16 geminals:
  1. (0, 1), (2, 3), (0, 3), 8
  2. (0, 1), (2, 3), (2, 4), 4
  3. (0, 1), (0, 2), (1, 3), 8
  4. (0, 1), (0, 1), (2, 3), 2
  5. (0, 1), (2, 3), (2, 3), 2
  6. (0, 1), (0, 2), (0, 1), 8
  7. (0, 1), (0, 1), (1, 2), 8
  8. (0, 1), (2, 3), (0, 4), 4
  9. (0, 1), (1, 2), (0, 2), 8
  10. (0, 1), (1, 2), (1, 3), 8
  11. (0, 1), (1, 2), (3, 4), 4
  12. (0, 1), (0, 2), (2, 3), 8
  13. (0, 1), (2, 3), (4, 5), 1
  14. (0, 1), (2, 3), (0, 1), 2
  15. (0, 1), (0, 2), (0, 2), 8
  16. (0, 1), (0, 1), (0, 1), 4

Algorithm 2

1. **Step 1**. Create symmetry group for each geminals in the output of Part 1.1.
   - For a geminal α, generate a set G containing all possible permuatations can be a element of group symmetry of α.
   - Apply an elements, g, in G, to α, and order the outcome by *order2* to get β
   - If $\alpha \equiv \beta$, put g in to the group symmetry of α.
2. **Step 2**: Apply the group symmetry on each elements of the determinant.
   - For a geminal α, compute $detg(D_\alpha)$, where D_α is a *corresponding matrix* of α, to obtain the list D
   - Order D by *order1*, and *group duplicates* to get the list F
   - Apply the group symmetry of α to the first element, F_0, in F to obtain F_{01}. Order F_{01} by *order1*.
   - Compare F_{01} to the rest of F to *group duplicates*. Save the elements obtained in a list A
   - Delete F_{01} and its duplicates.
   - Repeat this procedure until the F empty. The final output is A.

Symmetry Group

- Eight geminals have identity symmetry group. Follows are symmetry group of the other 8 geminals, the identity was excluded for simplicity:

Geminals	Group Symmetry
$(0,1),(2,3),(2,4)$	$(1,0,2,3,4)$
$(0,1),(0,1),(2,3)$	$(0,1,3,2),(1,0,2,3),(1,0,3,2)$
$(0,1),(2,3),(2,3)$	$(0,1,3,2),(1,0,2,3),(1,0,3,2)$
$(0,1),(2,3),(0,4)$	$(0,1,3,2,4)$
$(0,1),(1,2),(3,4)$	$(0,1,2,4,3)$
$(0,1),(2,3),(4,5)$	$(0,1,2,3,5,4),(0,1,3,2,4,5),(1,0,3,2,5,4)$ $(0,1,3,2,5,4),(1,0,2,3,4,5),(1,0,2,3,5,4)$ $(1,0,3,2,4,5)$
$(0,1),(2,3),(0,1)$	$(0,1,3,2),(1,0,2,3),(1,0,3,2)$
$(0,1),(0,1),(0,1)$	$(1,0)$

Result

- Follows are the results for those geminals does not have identity symmetry.

.

Geminals	**Before**	**After**
$(0,1),(2,3),(2,4)$	120	66
$(0,1),(0,1),(2,3)$	24	10
$(0,1),(2,3),(2,3)$	24	10
$(0,1),(2,3),(0,4)$	120	66
$(0,1),(1,2),(3,4)$	120	66
$(0,1),(2,3),(4,5)$	720	120
$(0,1),(2,3),(0,1)$	24	10
$(0,1),(0,1),(0,1)$	2	2

What about this?

$$\sum_{a,b}\sum_{m,n}\sum_{u}\int\int\int\int\int w_1(|\gamma_a-\gamma_b|)w_2(|\gamma_m-\gamma_n|))V(r_{u)}$$
$$\phi_a(\gamma_a)\phi_b(\gamma_b)\phi_m(\gamma_m)\phi_n(\gamma_n)\phi_u(\gamma_u)\Delta d\gamma_a d\gamma_b d\gamma_m d\gamma_n d\gamma_u$$

- Where Δ is:

$$\begin{vmatrix} \theta_a(\gamma_a) & \theta_a(\gamma_b) & \theta_a(\gamma_m) & \theta_a(\gamma_n) & \theta_a(\gamma_u) \\ \theta_b(\gamma_a) & \theta_b(\gamma_b) & \theta_b(\gamma_m) & \theta_b(\gamma_n) & \theta_b(\gamma_u) \\ \theta_m(\gamma_a) & \theta_m(\gamma_b) & \theta_m(\gamma_m) & \theta_m(\gamma_n) & \theta_m(\gamma_u) \\ \theta_n(\gamma_a) & \theta_n(\gamma_b) & \theta_n(\gamma_m) & \theta_n(\gamma_n) & \theta_n(\gamma_u) \\ \theta_u(\gamma_a) & \theta_u(\gamma_b) & \theta_u(\gamma_m) & \theta_u(\gamma_n) & \theta_u(\gamma_u) \end{vmatrix}$$

Result

- The two algorithms also works!
- Nine distinct geminals has been found, and here are the detailed results:

.

Geminals	**Before**	**After**
$(0,1),(1,2),(0)$	6	6
$(0,1),(1,2),(1)$	6	6
$(0,1),(0,2),(3)$	24	24
$(0,1),(2,3),(1)$	24	14
$(0,1),(2,3),(4)$	120	38
$(0,1),(2,3),(3)$	24	14
$(0,1),(0,1),(1)$	2	2
$(0,1),(0,2),(1)$	6	2
$(0,1),(0,2),(2)$	6	6

Conclusion

- We have eliminated overall 840 cases from 1154 cases, and established sufficient data to move to the next step of computing those geminals.
- In the next step, all those cases will be reduced one more time down to where we can use the reduction properties to compute them.