The Jastrow Factor

Pablo López Ríos
26-Jul-2009
The Jastrow Factor

- Typical trial wave function – Slater-Jastrow:
  \[ \Psi_T(R) = e^{J(R)} \Psi_S(R) \]

- In VMC, Jastrow describes correlations

- DMC is like VMC with perfect Jastrow; actual Jastrow merely stabilizes method

- VMC is much simpler/cheaper than DMC – pushing VMC quality towards DMC is good
The Jastrow Factor

- Jastrow function: sum of terms, e.g.,

\[ J(R) = J_{e-e}(R) + J_{e-n}(R) + J_{e-e-n}(R) + \ldots \]

- Term: sum over particle groups of a function, e.g.,

\[ J_{e-e-n}(R) = \sum_i \sum_j \sum_I \Omega(r_{ij}, r_{iI}, r_{jI}) \]

- Function: expansion on basis of functions of single relative position vector, e.g.

\[ \Omega(r_{ij}, r_{iI}, r_{jI}) = \sum_{\nu_{ij}} \sum_{\mu_{iI}, \mu_{jI}} \sum_{p} \sum_{q} \lambda_{\nu_{ij}, \mu_{iI}, \mu_{jI}} \phi_{\nu_{ij}}(r_{ij}) \theta_{\mu_{iI}}(r_{iI}) \theta_{\mu_{jI}}(r_{jI}) \]
Why Generalize?

- All Jastrow terms constructed the same way
- No need to implement **new terms** (e.g., four-body)
- Easy to implement **new functional bases**
- Easy to add **anisotropies**
- Easy to add dependencies on **external potentials**
General Jastrow Term: definition

\[
J_{n,m}(R) = \frac{1}{n! m!} \sum_{i_1 \neq \ldots \neq i_n}^{N} \sum_{I_1 \neq \ldots \neq I_m}^{M} \prod_{\alpha < \beta}^{n} \Phi_{v_{\alpha \beta}}^{P_{\sigma(i_{\alpha}) \sigma(i_{\beta})}}(r_{i_{\alpha} i_{\beta}}) \prod_{\alpha, \gamma}^{n, m} \Theta_{\mu_{\alpha \gamma}, I_{\gamma}}^{S_{\sigma(i_{\alpha}) I_{\gamma}}}(r_{i_{\alpha} I_{\gamma}}) \times \\
\sum_{i_1 \neq \ldots \neq i_n}^{N} \sum_{I_1 \neq \ldots \neq I_m}^{M} \prod_{\alpha < \beta}^{n} \Phi_{v_{\alpha \beta}}^{P_{\sigma(i_{\alpha}) \sigma(i_{\beta})}}(r_{i_{\alpha} i_{\beta}}) \prod_{\alpha, \gamma}^{n, m} \Theta_{\mu_{\alpha \gamma}, I_{\gamma}}^{S_{\sigma(i_{\alpha}) I_{\gamma}}}(r_{i_{\alpha} I_{\gamma}})
\]
“One electron” Jastrow

\[ J_{n,m}^{i}(R) = \sum_{i_1=1}^{N} \sum_{i_2<...<i_n} \sum_{I_1<...<I_m} \sum_{\nu_{i_{a}i_{\beta}}}^{n} \sum_{\mu_{i_{a}I_y}}^{n,m} \lambda_{\nu,\mu} \times \]

\[ \prod_{\alpha<\beta}^{n} \Phi_{\nu_{i_{a}i_{\beta}}}^{p_{\sigma(i_{a})\sigma(i_{\beta})}}(r_{i_{\alpha}i_{\beta}}) \prod_{\alpha,\gamma}^{n,m} \Theta_{\mu_{i_{a}I_y}}^{S_{\sigma(i_{a})I_y}}(r_{i_{\alpha}I_y}) \]

Properties:
1. If two configurations only differ in the position of particle \( i \), change in total Jastrow is change in one-electron Jastrow

\[ J_{n,m}(R') - J_{n,m}(R) = J_{n,m}^{i}(R') - J_{n,m}^{i}(R) \]

2. Total Jastrow value can be computed from one-electron Jastrow for all electrons

\[ J_{n,m} = \sum_{i=1}^{N} \frac{1}{n} J_{n,m}^{i} \]
In this form, the gradient is calculated simply by multiplying the running accumulator for the value times a simple sum of gradients of basis functions.

Basis functions being exactly zero are an issue, but I don't expect this to ever happen in practice. The code currently ignores contributions to the gradient from zero-valued functions.
\[
\n\nabla^2 J_{n,m}(R) = \sum_{i_2 < \ldots < i_n} \sum_{I_1 < \ldots < I_m} \sum_{n,m} \sum_{\alpha < \beta} \sum_{\alpha, \gamma} \lambda_{P[S]V[\mu]} \times \\
\times \prod_{\alpha < \beta}^{n} \Phi_{\nu_i,\nu_{i_\beta}}^{P[\sigma(i), \sigma(i_\beta)]} \left( r_{i_\alpha i_{\beta}} \right) \prod_{\alpha, \gamma}^{n,m} \Theta_{\mu_i, \mu_{i_\gamma}}^{S[\sigma(i), \sigma(i_\gamma)]} \left( r_{i_\alpha i_{\gamma}} \right) \\
\times \left[ \sum_{\beta = 2}^{n} \nabla_{\nu_i,\nu_{i_\beta}}^{P[\sigma(i), \sigma(i_\beta)]} \left( r_{ii_\beta} \right) + \sum_{\gamma = 1}^{m} \nabla_{\mu_i, \mu_{i_\gamma}}^{S[\sigma(i), \sigma(i_\gamma)]} \left( r_{ii_\gamma} \right) \right]^2 \\
+ \sum_{\beta = 2}^{n} \left( \nabla_{\nu_i,\nu_{i_\beta}}^{P[\sigma(i), \sigma(i_\beta)]} \left( r_{ii_\beta} \right) - \nabla_{\nu_i,\nu_{i_\beta}}^{P[\sigma(i), \sigma(i_\beta)]} \left( r_{ii_\beta} \right) \right)^2 \\
+ \sum_{\gamma = 2}^{m} \left( \nabla_{\mu_i, \mu_{i_\gamma}}^{S[\sigma(i), \sigma(i_\gamma)]} \left( r_{ii_\gamma} \right) - \nabla_{\mu_i, \mu_{i_\gamma}}^{S[\sigma(i), \sigma(i_\gamma)]} \left( r_{ii_\gamma} \right) \right)^2 \right]
\]

Likewise for the Laplacian
The Linear Parameters

- The linear parameters have the following property by definition:

\[ \lambda \hat{P}_n \{P\} \hat{P}_m \{S\} = \lambda \{P\} \{S\} \]

- Thus, all sets of \( \{P\} \) and \( \{S\} \) which only differ in the ordering are equivalent

- The ordered set of values \( \{P\}, \{S\} \) is called the **signature** of a particle group

- Each signature corresponds to a unique set of parameters
\{P\} are particle-particle types
\{P\} can be written as a symmetric matrix
  - For a typical electronic calculation we would have:
    \[
P = \begin{pmatrix}
    1 & 2 \\
    2 & 1
\end{pmatrix}
\]
    - Rows/columns ordered as electron up, electron down
    - Distinguishes between parallel- and antiparallel-spin pairs
    - Does not distinguish between up-up and down-down
  
  - For an electron-hole system we could have:
    \[
P = \begin{pmatrix}
    1 & 2 & 3 & 3 \\
    2 & 1 & 3 & 3 \\
    3 & 3 & 4 & 5 \\
    3 & 3 & 5 & 4
\end{pmatrix}
\]
    - Rows/columns ordered as e-up, e-down, h-up, h-down
    - Like above within electrons, and within holes
    - Distinguishes between electron pairs and hole pairs
    - All electron-hole pairs indistinguishable regardless of spin
In a term with \( n=3 \), \( m=0 \) (three-electron), given:

\[
P = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}
\]

we can have the following combinations of particle types (second arrow means sorting):

- Types: \((1,1,1)\) → pairs: \((1,1,1)\) → signature \((1,1,1)\)
- Types: \((1,1,2)\) → pairs: \((1,2,2)\) → signature \((1,2,2)\)
- Types: \((1,2,2)\) → pairs: \((2,2,1)\) → signature \((1,2,2)\)
- Types: \((2,2,2)\) → pairs: \((1,1,1)\) → signature \((1,1,1)\)

Therefore we have two independent sets of linear parameters (**channels**).
{S} are particle-nucleus types

{S} can be written as a rectangular matrix

For a water molecule we could have:

\[ S = \begin{pmatrix} 1 & 1 \\ 2 & 3 \\ \checkmark & 3 \end{pmatrix} \]

- Rows ordered as O, H, H; columns as e-up, e-down
- Distinguishes between O and H
- Does not distinguish the two H's
- Does not distinguish up- and down-electrons in relation to O
- Distinguishes up- and down-electrons in relation to H

Full signatures are obtained from both \( \{P\} \) and \( \{S\} \) (ordering occurs within each set, though)

NB, basis functions depend on a single \( P_{ij} \) or \( S_{il} \): channel splitting is independent from that of linear parameters
Constraints on the Linear Parameters

\[
\lambda \hat{P}_n \{v\} \hat{P}_m \{\mu\} = \lambda \{v\} \{\mu\} \forall \hat{P}_n, \hat{P}_m : \hat{P}_n \{P\} = \{P\}, \hat{P}_m \{S\} = \{S\}
\]

Repeated signature indices imply symmetry with respect to exchange of bottom indices.

This is the spherical average of the gradient of \( J \) with respect to \( r_{ij} \) at \( r_{ij} = 0 \)

Cusp conditions

Constant that depends on the attributes of particles involved, dimensionality, etc.
Cusp Conditions

- Non-zero cusp only applicable to two-body terms:

\[
\left( \frac{\partial J_{2,0}(R)}{\partial r_{ij}} \right)_{r_{ij}=0} = \sum_{\nu_{ij}=1}^{p} \lambda_{\nu_{ij}}^{P_{ij}} \left( \frac{\partial \Phi_{\nu_{ij}}^{P_{ij}}(r_{ij})}{\partial r_{ij}} \right)_{r_{ij}=0} = \Gamma_{ij}
\]

\[
\left( \frac{\partial J_{1,1}(R)}{\partial r_{iI}} \right)_{r_{iI}=0} = \sum_{\mu_{iI}=1}^{p} \lambda_{\mu_{iI}}^{S_{iI}} \left( \frac{\partial \Theta_{\mu_{iI}}^{S_{iI}}(r_{iI})}{\partial r_{iI}} \right)_{r_{iI}=0} = \Gamma_{iI}
\]

- Of course, only one e-e term should have a non-zero cusp for a given particle-pair type. Same goes for e-n term.
Cusp Conditions

- The particle-particle no-cusp condition is:

\[
\sum_{\nu_{ii_{\beta}}, \nu_{ji_{\beta}}}^{p} \sum_{\mu_{il_{\gamma}}, \mu_{jl_{\gamma}}}^{q} \left[ \sum_{\nu_{ij}}^{p} \lambda_{\nu_{ij}}^{\{P\}} \{S\} \left( \frac{\partial \Phi_{ij}^{P_{\sigma(i)\sigma(j)}}(r_{ij})}{\partial r_{ij}} \right) \right] \times \\
\prod_{\beta=3}^{n} \Phi_{\nu_{ii_{\beta}}}^{P_{\sigma(i)\sigma(i_{\beta})}}(r_{ii_{\beta}}) \Phi_{\nu_{ji_{\beta}}}^{P_{\sigma(j)\sigma(i_{\beta})}}(r_{ij}) \prod_{\gamma=1}^{m} \Theta_{\mu_{il_{\gamma}}}^{S_{\sigma(i)I_{\gamma}}}(r_{iI_{\gamma}}) \Theta_{\mu_{jl_{\gamma}}}^{S_{\sigma(j)I_{\gamma}}}(r_{iI_{\gamma}}) = 0
\]

- The first sum is constrained to index pairs such that \( \Phi_{\nu_{ii_{\beta}}}^{P_{\sigma(i)\sigma(i_{\beta})}}(r_{ii_{\beta}}) \Phi_{\nu_{ji_{\beta}}}^{P_{\sigma(j)\sigma(i_{\beta})}}(r_{ij}) \) gives the same function.

- The second sum is constrained to index pairs such that \( \Theta_{\mu_{il_{\gamma}}}^{S_{\sigma(i)I_{\gamma}}}(r_{iI_{\gamma}}) \Theta_{\mu_{jl_{\gamma}}}^{S_{\sigma(j)I_{\gamma}}}(r_{iI_{\gamma}}) \) gives the same function.

- In many common cases, only the square bracket needs to be zero.
Cusp Conditions

- The particle-nucleus no-cusp condition is:

\[
\sum_{\nu_{ii}, \mu_{iJ}}^{p, q} \sum_{\mu_{iJ}}^{q} \left[ \lambda \left\{ P \right\} \left\{ S \right\} \left( \frac{\partial \Theta_{\mu_{iJ}}^{S_{\sigma(i)J}} (r_{iJ})}{\partial r_{iJ}} \right)_{r_{iJ}=0} \right] \times \prod_{\beta=2}^{n} \Phi_{\nu_{ii}}^{P_{\sigma(i)\sigma(i\beta)}} (r_{i\beta}) \Theta_{\mu_{i\beta J}}^{S_{\sigma(i\beta)J}} (r_{i\beta J}) = 0
\]

- The first sum is constrained to index pairs such that \( \Phi_{\nu_{ii}}^{P_{\sigma(i)\sigma(i\beta)}} (r_{i\beta}) \Theta_{\mu_{i\beta J}}^{S_{\sigma(i\beta)J}} (r_{i\beta J}) \) gives the same function.

- In many common cases, only the square bracket needs to be zero.
Get basis and cutoff functions
Loop over ion_vector(1:m)
  Loop over ispin_vector(1:n)
    Get signature and permutation
    Loop over ie_vector(1:n)
      Get cutoff functions (incremental)
      Loop over nu_vector(1:(n*(n-1))/2)
        Get product of e-e basis functions (permuted, incremental)
        Loop over mu_vector(1:n*m)
          Get product of e-n basis functions (permuted, incremental)
          Accumulate contribution
        End loop over mu_vector
      End loop over nu_vector
    End loop over ie_vector
  End loop over ispin_vector
End loop over ion_vector

It is convenient to write optimized versions of the loops, for performance.
Code Structure

- wfn_utils
- gjastrow
- gbackflow
- cdf
- gbasis

(for book-keeping functions only)

(not yet)
The gbasis module handles basis and cutoff functions.

Sharing of basis functions among several terms is allowed for functions without parameters in them.

Sharing of basis functions between Jastrow and backflow modules also possible.
The CASINO Data Format (CDF)

- Format for structured data resembling YAML
- Nodes in the structure may be:
  - scalars (keyword ↔ value, or value only)
  - blocks (keyword ↔ set of children nodes)
- Indentation used to indicate depth
- Alternatively, square brackets and commas can be used for children of a block
- Values are strings, converted to other types on request
- Keywords are case- and whitespace-insensitive
JASTROW:
Title: 2D HEG

TERM 1:
  Rank: [ 2, 0 ]
  Rules: [ 1-1 = 2-2 ]
  e-e basis: [ Type: polynomial, Order: 9 ]
  e-e cutoff:
    Type: polynomial
    Constants: [ C: 3 ]
  Parameters:
    Channel 1-1: [ L: [ 4.51888387, optimizable ] ]
    Channel 1-2: [ L: [ 4.51888387, optimizable ] ]

Linear parameters:
  Channel 1-1:
    c_1: [ 5.10744890784656D-003, optimizable ]
    c_2: [ -2.215745836891912E-004, fixed ]
    c_3: [ 2.6104991223271D-004, optimizable ]
    c_4: [-5.71501823949904D-003, optimizable ]
    c_5: [ 6.26629894651862D-003, optimizable ]
    c_6: [-3.69279305571898D-003, optimizable ]
    c_7: [ 1.21802169144595D-003, optimizable ]
    c_8: [-2.11189983328774D-004, optimizable ]
    c_9: [ 1.50406603295831D-005, optimizable ]
$H_2$ molecule
Current State of the Project

- CDF → done
- Channel splitting → done
- Basis set evaluation → done (basic ones)
- Basis set sharing → done
- Term evaluation → done
- Initial tests → done
- Speed → done
- Constraints → done
- Tests on interesting systems → in progress