FAST JACOBIANS AND HESSIANS BY LEVERAGING SPARSITY

An Illustrated Guide to Automatic Sparse Differentiation

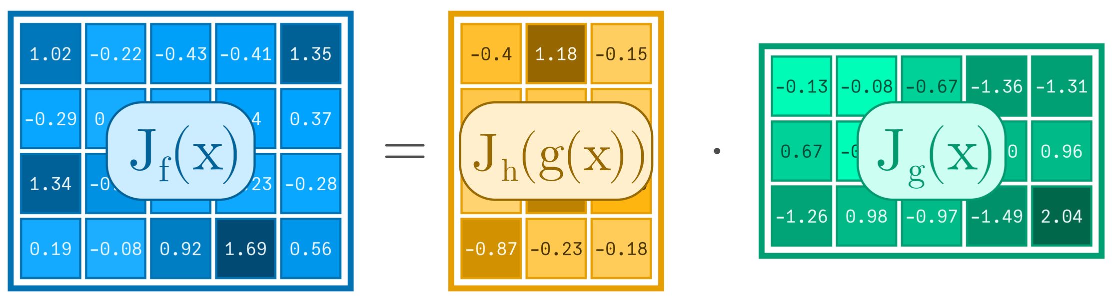
Adrian Hill[1,2], Guillaume Dalle[3] and Alexis Montoison[4]

[1]BIFOLD – Berlin Institute for the Foundations of Learning and Data, Berlin, Germany, [2]Machine Learning Group, Technical University of Berlin, Berlin, Germany,

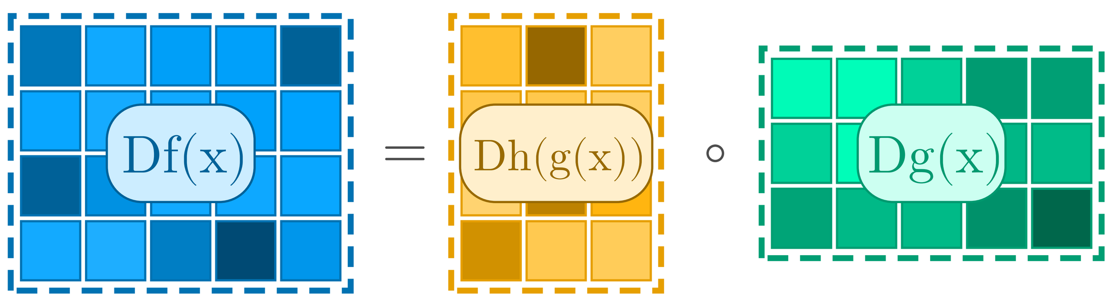
[3]LVMT, ENPC, Institut Polytechnique de Paris, Univ Gustave Eiffel, Marne-la-Vallée, France, [4]Argonne National Laboratory

Recap: Automatic Differentiation (AD)

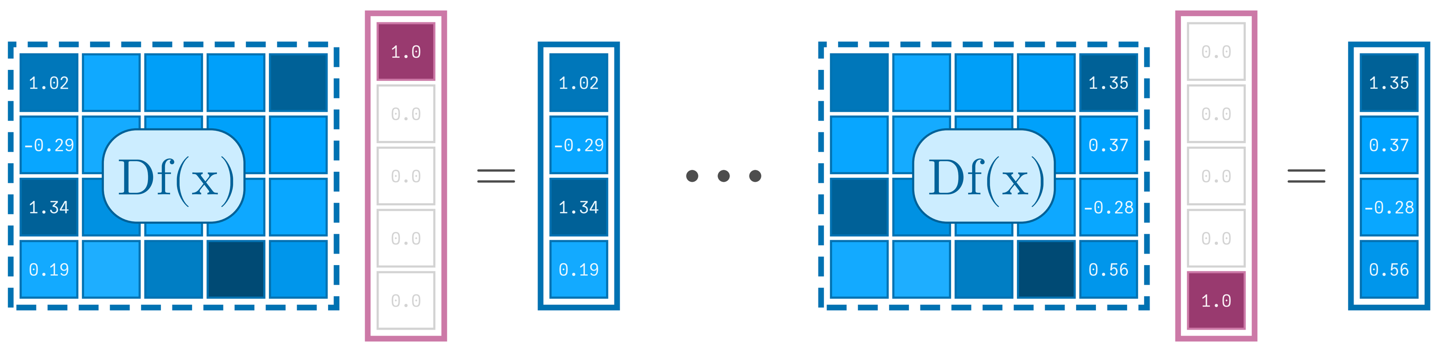
The chain rule tells us that the Jacobian of a composed function $f = h \circ g$ is obtained by multiplying the **Jacobian matrices** (solid) of h and g.



However, AD doesn't use Jacobian matrices, instead opting for matrix-free **Jacobian operators** (dashed). The chain rule now corresponds to a composition of operators.



To turn such (composed) **Jacobian operators** into **Jacobian matrices**, they are evaluated with all standard basis vectors.

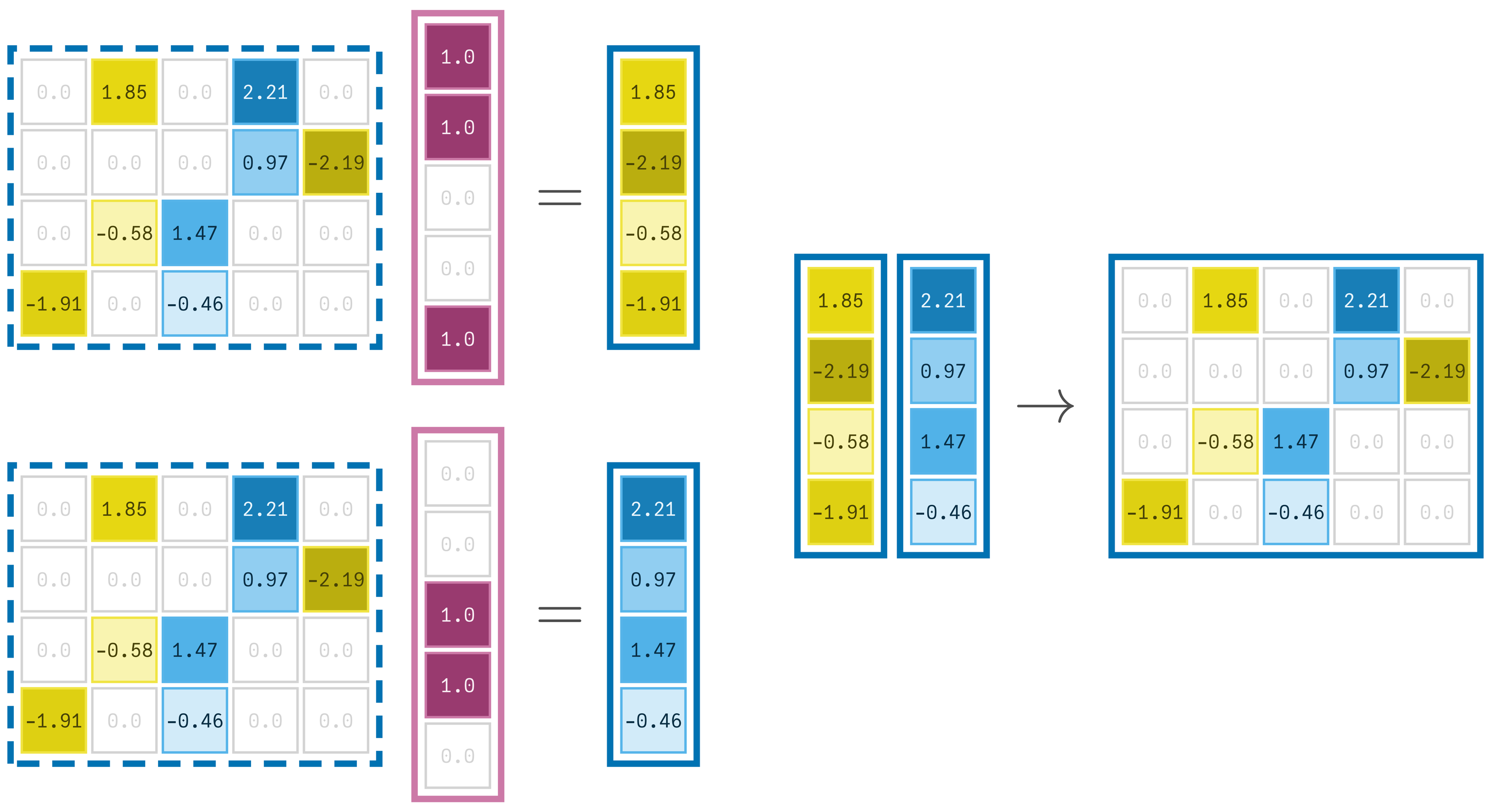


This either constructs Jacobian matrices column-by-column (forward mode, computing as many JVPs as there are inputs) or row-by-row (reverse mode, computing as many VJPs as there are outputs) .

Idea: Automatic Sparse Differentiation (ASD)

Since Jacobian operators are linear maps, we can:

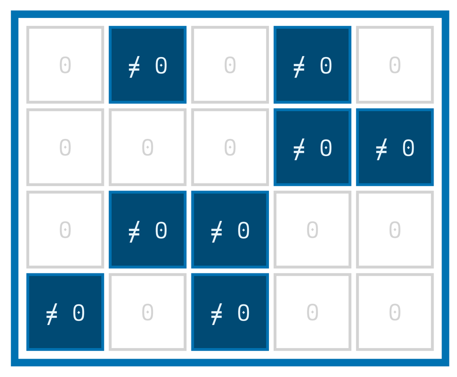
1. simultaneously compute the values of orthogonal columns/rows
2. decompress the resulting vectors into the Jacobian matrix.



Unfortunately, contrary to our illustrations, Jacobian operators (dashed) are black-box functions with unknown structure. Two preliminary steps are therefore required to determine orthogonal columns/rows.

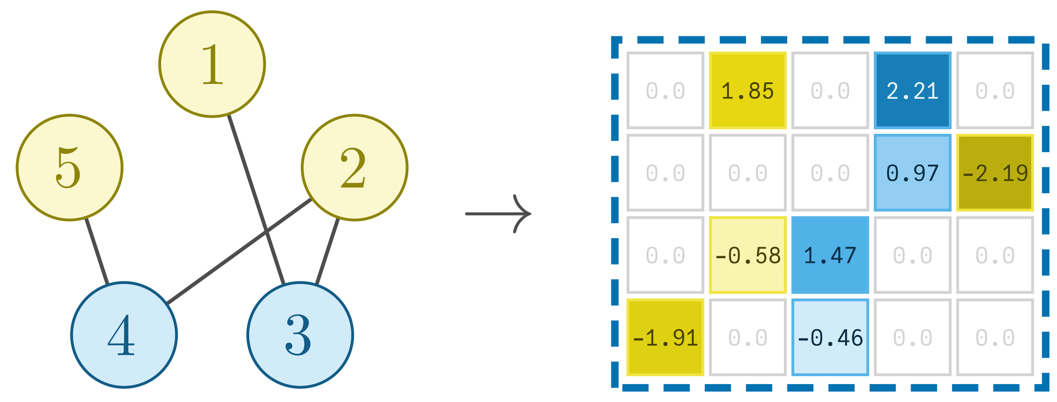
Step 1: Sparsity Pattern Detection

To find orthogonal colomns, the pattern of non-zero values in the Jacobian matrix has to be detected. This requires a fast binary AD system.



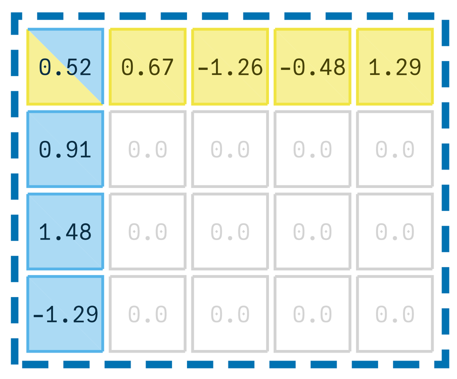
Step 2: Coloring

Graph coloring algorithms are applied to the sparsity pattern to detect orthogonal columns/rows.



Bicoloring

ASD can be accelerated even further by coloring both rows and columns and combining forward and reverse modes.



Demonstration

ASD is fully automatic, as can be seen in the following Julia code:

```julia
using DifferentiationInterface
using SparseConnectivityTracer, SparseMatrixColorings
import ForwardDiff

ad_backend = AutoForwardDiff()
asd_backend = AutoSparse(
    ad_backend;
    TracerSparsityDetector(),
    GreedyColoringAlgorithm()
)

jacobian(f, ad_backend,  x) # dense
jacobian(f, asd_backend, x) # sparse
```

References

Bibliography goes here

Check out our ICLR blog post for more information!