Open Performance portablE SeismiC Imaging - OPESCI

Gerard Gorman [1] Marcos de Aguiar [2] David Ham [1]
Felix Herrmann [3] Paul Kelly [1] Navjot Kukreja [2] Michael Lange [1]
Mathias Louboutin [3] Fabio Luporini [1] Paulius Velesko[1]
Vincenzo Pandolfo Felippe Vieira Zacarias [2]

[1]Imperial College London, UK

[2]SENAI CIMATEC, Brazil

[3]Seismic Lab. for Imaging and Modeling, The University of British Columbia, Canada

October 25, 2016

Imperial College London

OPESCI

Seismic imaging is a hard problem...

- The objective is to see clearly through km's of rock.
- The problem is ill posed - i.e. we know we know much less than we do know.
- Arguably the most data-intensive/compute-intensive problem in the private sector.
- Very far from being a solved problem - requirements:
  - Better physics model.
  - Fast, efficient, robust PDE solvers and their adjoint.
  - Inversion/optimisation techniques.
  - Ability to manage petabytes of data.

Code modernisation and innovation

- Innovation is being stifled by the sheer size and complexity of software.
- Slow and costly to modernise/optimise code.
- Skills gap: "there are not enough HPC developers in the world to modernise our code".
- Question: How can we be agile, innovative and productive?

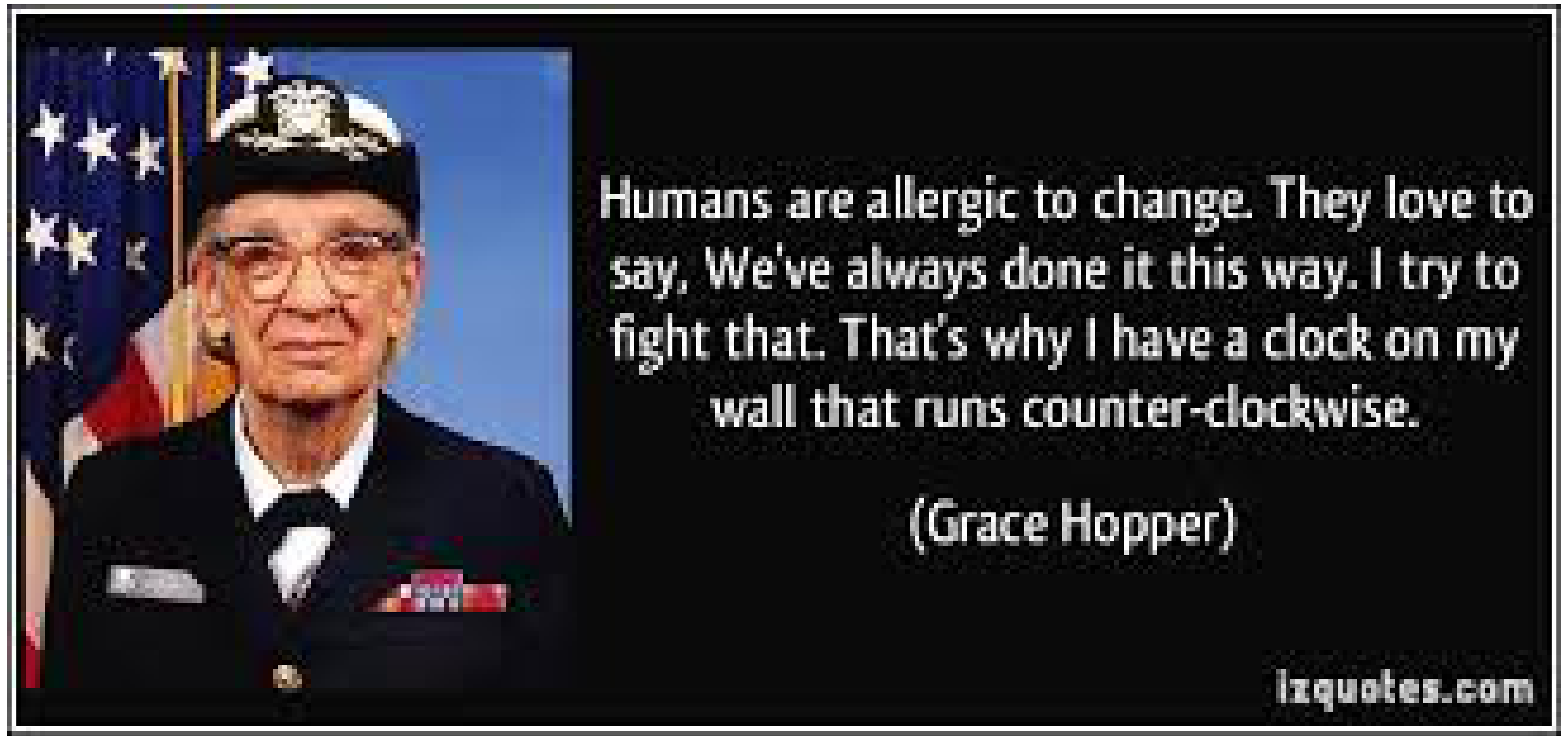
Options...

- New parallel programming languages like OpenCL can offer superior performance portability.
  - **BUT**: Part of the code modernisation challenge is the sheer quantity of legacy source code - OpenCL is far more verbose than what it is replacing.
- MATLAB highly successful high-level language.
  - **BUT**: Cannot get same performance as compiled codes; license model unattractive for HPC.
- Python (with Numpy etc) clear favorite high-level language among scientists and engineers, great at wrapping external libraries.
  - **BUT**: Pure (naive) Python code can be slow; and where do the external libraries come from?
- Julia fixes many of the performance issues present in pure Python.
  - **BUT**: Fast but still not HPC; and where do the external libraries come from?

Imperial College London

Lesson from history

- First ever compiler was the A-0 system (Arithmetic Language version 0), written by Grace Hopper (aka Amazing Grace) in 1951 for the UNIVAC I.
- First FORTRAN compiler delivered in April 1957. This was also the first optimizing compiler - customers were reluctant to use a *high-level programming language* unless its compiler could generate code with performance comparable to that of their hand-coded assembly.


Humans are allergic to change. They love to say, We've always done it this way. I try to fight that. That's why I have a clock on my wall that runs counter-clockwise.
(Grace Hopper)
izquotes.com

Imperial College London

Yesterdays high-level languages are todays low-level languages

- Goal: Rather than writing software to solve **one** set of equations for **one** computer architecture; write software that can read and manipulate equations, and write software in low-level languages for different computer architectures.
- Using code generation methods millions of lines of parallel code can be generated within seconds.
- Not just trying to match performance of hand-tuned code: through automation we perform aggressive optimisations impractical for human programmers to implement by hand — and get it right first time...

Imperial College
London

Open Performance portablE SeismiC Imaging (OPESCI) - http://opesci.org

- Funding: Intel Parallel Computing Center.
- Partners: Imperial College London, SENAI CIMATEC Brazil, University of British Columbia, BG Group
- Finite element:
  - Area where we have the strongest track record - particularly in CFD (shameless plug http://www.lcs-fast.com)
  - Big opportunities for seismic imaging - but impacts much more of the software stack therefore...
- Finite difference:
  - Brings together a number of different (open source) software technologies and communities to develop a code generation framework for seismic imaging.

Imperial College London

Symbolic computation is a powerful tool

Solving simple PDEs is (kind of) easy...

First-order diffusion equation:

```
for ti in range(timesteps):
    t0 = ti % 2
    t1 = (ti + 1) % 2
    for i in range(1, nx-1):
        for j in range(1, ny-1):
            uxx = (u[t0,i+1,j]-2*u[t0,i,j]+u[t0,i-1,j]) / dx2
            uyy = (u[t0,i,j+1]-2*u[t0,i,j]+u[t0,i,j-1]) / dy2
            u[t1, i, j] = u[t0, i, j] + dt * a * (uxx + uyy)
```

Symbolic computation is a powerful tool

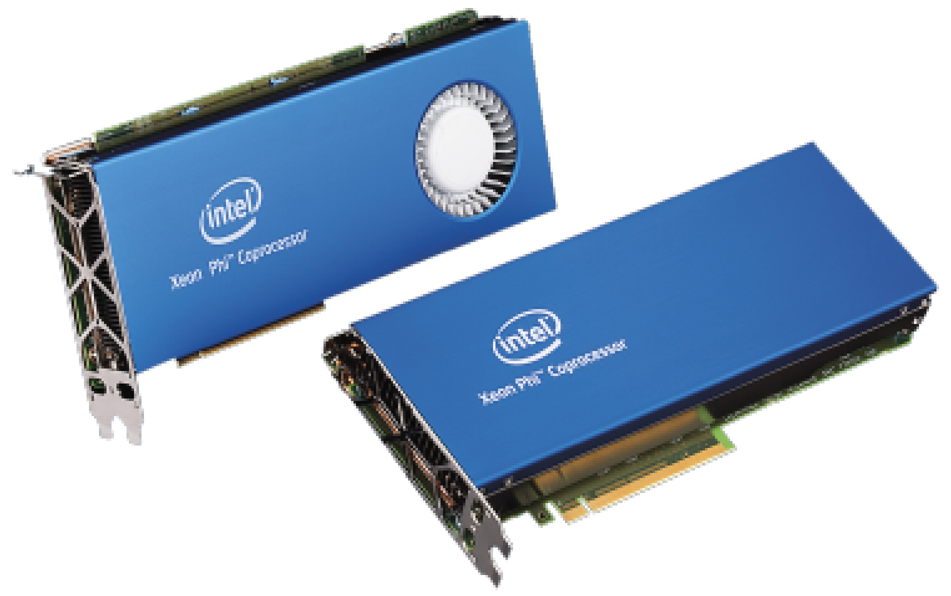
Solving complicated PDEs is not easy!

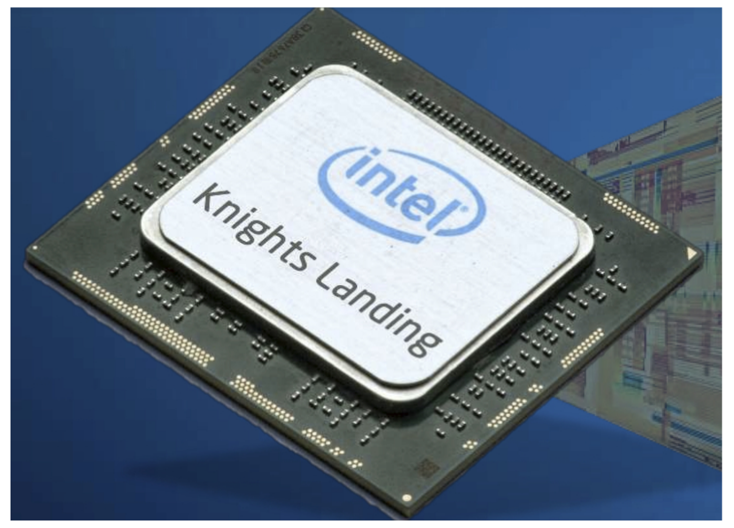
12th-order acoustic wave equation:

```
for (int i4 = 0; i4<149; i4+=1) {
  for (int i1 = 6; i1<64; i1++) {
    for (int i2 = 6; i2<64; i2++) {
      for (int i3 = 6; i3<64; i3++) {
        u[i4][i1][i2][i3] = 6.01250601250601e-9F*(2.80896e+8F*damp[i1][i2][i3]*u[i4-2][i1
              ][i2][i3]-3.3264e+8F*m[i1][i2][i3]*u[i4-2][i1][i2][i3]+6.6528e+8F*m[i1][i2
              ][i3]*u[i4-1][i1][i2][i3]-2.12255421155556e+7F*u[i4-1][i1][i2][i3
              ]-1.42617283950617e+2F*u[i4-1][i1][i2][i3-6]+2.46442666666667e+3F*u[i4-1][
              i1][i2][i3-5]-2.11786666666667e+4F*u[i4-1][i1][i2][i3-4]+1.25503209876543e
              +5F*u[i4-1][i1][i2][i3-3]-6.3536e+5F*u[i4-1][i1][i2][i3-2]+4.066304e+6F*u[
              i4-1][i1][i2][i3-1]+4.066304e+6F*u[i4-1][i1][i2][i3+1]-6.3536e+5F*u[i4-1][
              i1][i2][i3+2]+1.25503209876543e+5F*u[i4-1][i1][i2][i3+3]-2.11786666666667e
              +4F*u[i4-1][i1][i2][i3+4]+2.46442666666667e+3F*u[i4-1][i1][i2][i3
              +5]-1.42617283950617e+2F*u[i4-1][i1][i2][i3+6]-1.42617283950617e+2F*u[i4
              -1][i1][i2-6][i3]+2.46442666666667e+3F*u[i4-1][i1][i2-5][i3
              ]-2.11786666666667e+4F*u[i4-1][i1][i2-4][i3]+1.25503209876543e+5F*u[i4-1][
              i1][i2-3][i3]-6.3536e+5F*u[i4-1][i1][i2-2][i3]+4.066304e+6F*u[i4-1][i1][i2
              -1][i3]+4.066304e+6F*u[i4-1][i1][i2+1][i3]-6.3536e+5F*u[i4-1][i1][i2+2][i3
              ]+1.25503209876543e+5F*u[i4-1][i1][i2+3][i3]-2.11786666666667e+4F*u[i4-1][
              i1][i2+4][i3]+2.46442666666667e+3F*u[i4-1][i1][i2+5][i3]-1.42617283950617e
              +2F*u[i4-1][i1][i2+6][i3]-1.42617283950617e+2F*u[i4-1][i1-6][i2][i3
              ]+2.46442666666667e+3F*u[i4-1][i1-5][i2][i3]-2.11786666666667e+4F*u[i4-1][
              i1-4][i2][i3]+1.25503209876543e+5F*u[i4-1][i1-3][i2][i3]-6.3536e+5F*u[i4
              -1][i1-2][i2][i3]+4.066304e+6F*u[i4-1][i1-1][i2][i3]+4.066304e+6F*u[i4-1][
              i1+1][i2][i3]-6.3536e+5F*u[i4-1][i1+2][i2][i3]+1.25503209876543e+5F*u[i4
              -1][i1+3][i2][i3]-2.11786666666667e+4F*u[i4-1][i1+4][i2][i3
              ]+2.46442666666667e+3F*u[i4-1][i1+5][i2][i3]-1.42617283950617e+2F*u[i4-1][
              i1+6][i2][i3])/(1.68888888888889F*damp[i1][i2][i3]+2*m[i1][i2][i3]);
      }
    }
  }
```

Symbolic computation is a powerful tool

- **Getting performance on modern hardware is not easy!**
  - Functioning code exists but is not optimised for current hardware
  - **Evolution vs. revolution?**
- **Domain-specific languages (DSL) make revolution easy**
  - **Separate problem definition from implementation**
  - Creates a separate of concerns between scientists and computation experts
- **Performance portability through code-generation**
  - Code is auto-generated and optimised at run-time
  - Platform-spefic optimisation for target hardware





Symbolic computation is a powerful tool

- **Symbolic DSLs for solving PDEs have proven succesful**

FEniCS / Firedrake - Finite element DSL packages

Velocity-stress formulation of elastic wave equation, with isotropic stress:

$$\rho \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot \mathbb{T}$$

$$\frac{\partial \mathbb{T}}{\partial t} = \lambda \left(\nabla \cdot \mathbf{u} \right) \mathbb{I} + \mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}} \right)$$

Weak form of equations written in UFL[1]:

```
F_u = density*inner(w, (u - u0)/dt)*dx - inner(w, div(s0))*dx
solve(lhs(F_u) == rhs(F_u), u)
```

1

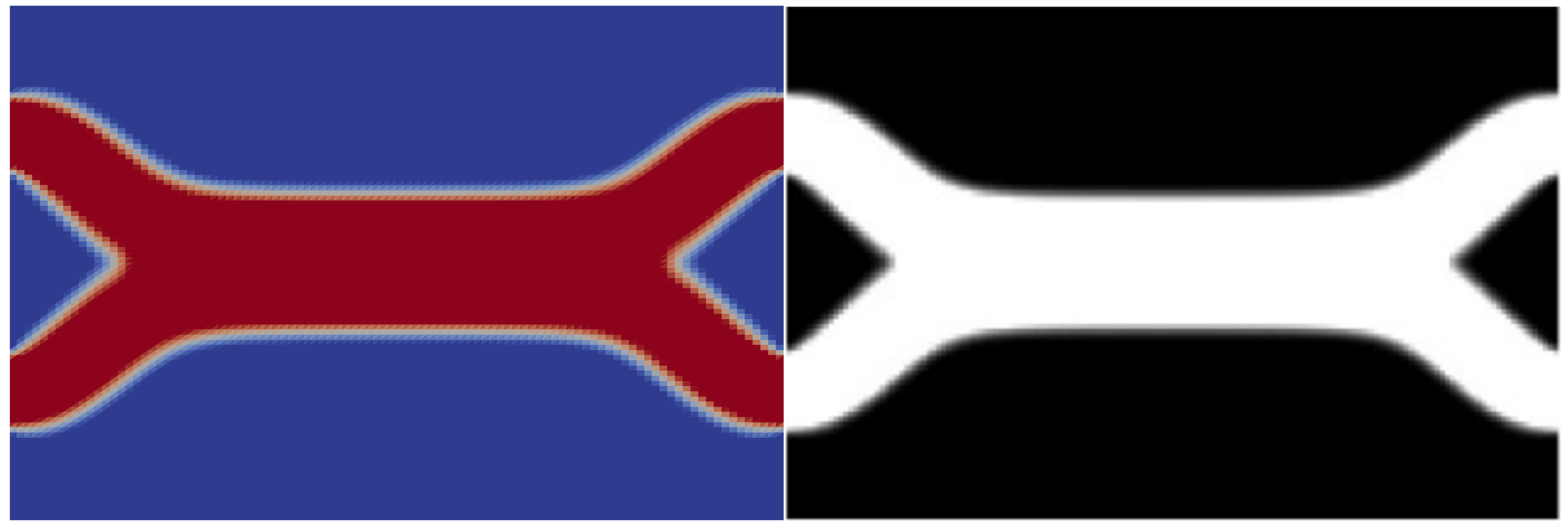
Symbolic computation is a powerful tool

- **Symbolic DSLs for solving PDEs have proven succesful**

Dolfin-Adjoint: Symbolic adjoints from symbolic PDEs[1]

- Solves complex optimisation problems
- 2015 Wilkinson prize winner

Below is the optimal design of a double pipe that minimises the dissipated power in the fluid.



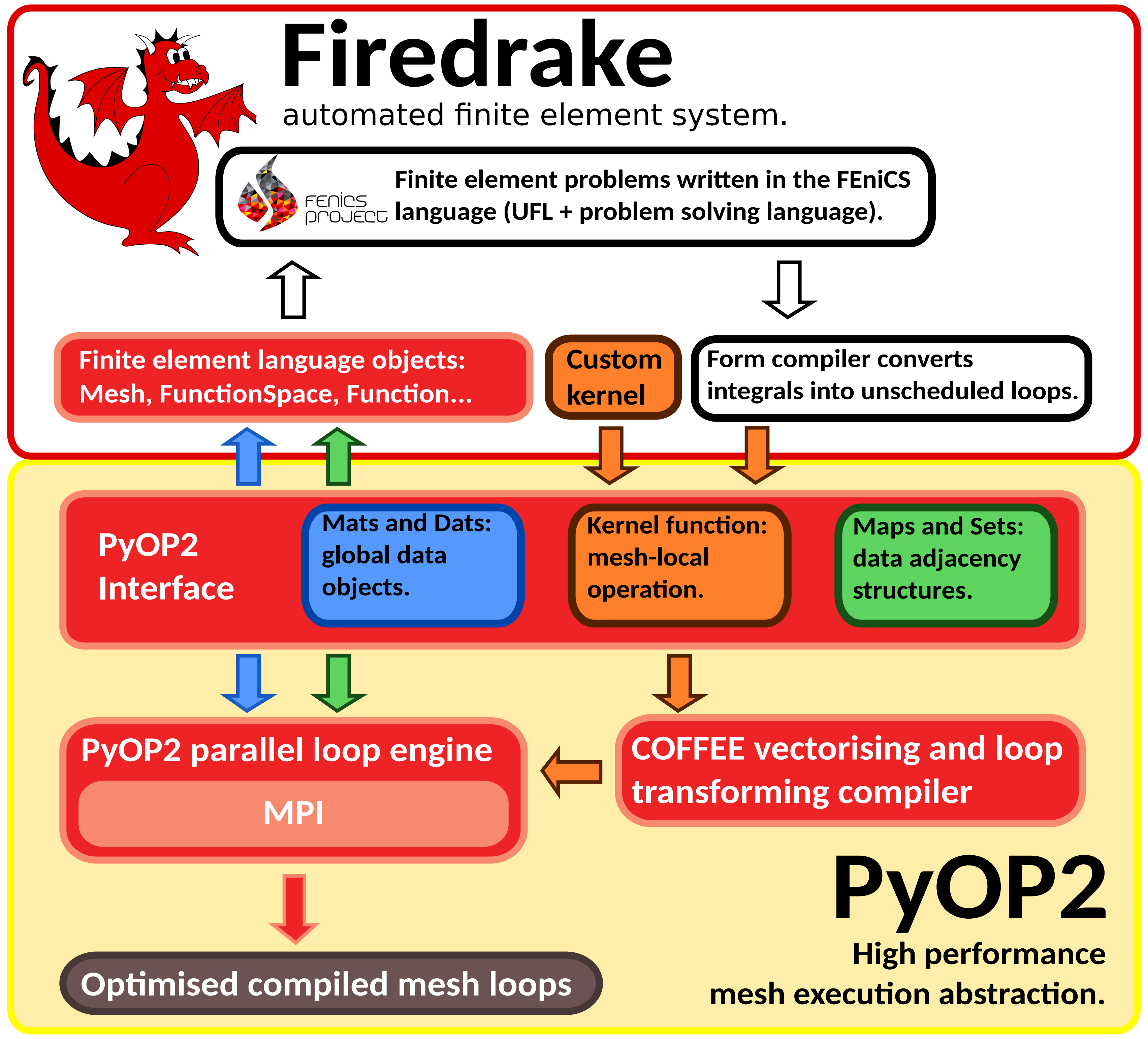
1

Automating computation for finite element

Imperial College London

Firedrake: Automated code generation for finite element

Application defines the discretised model equations in symbolic form through the Unified Form Language (UFL), a high-level domain-specific language for finite element problems that originated in the FEniCS project.
Multiple layers of abstraction allows Firedrake to utilise performance optimisations at various layers of its software stack, allowing application developers to describe the governing equations symbolically.


Firedrake
automated finite element system.
FEniCS project
Finite element problems written in the FEniCS language (UFL + problem solving language).
Finite element language objects: Mesh, FunctionSpace, Function...
Custom kernel
Form compiler converts integrals into unscheduled loops.
PyOP2 Interface
Mats and Dats: global data objects.
Kernel function: mesh-local operation.
Maps and Sets: data adjacency structures.
PyOP2 parallel loop engine
MPI
COFFEE vectorising and loop transforming compiler
Optimised compiled mesh loops
PyOP2
High performance mesh execution abstraction.

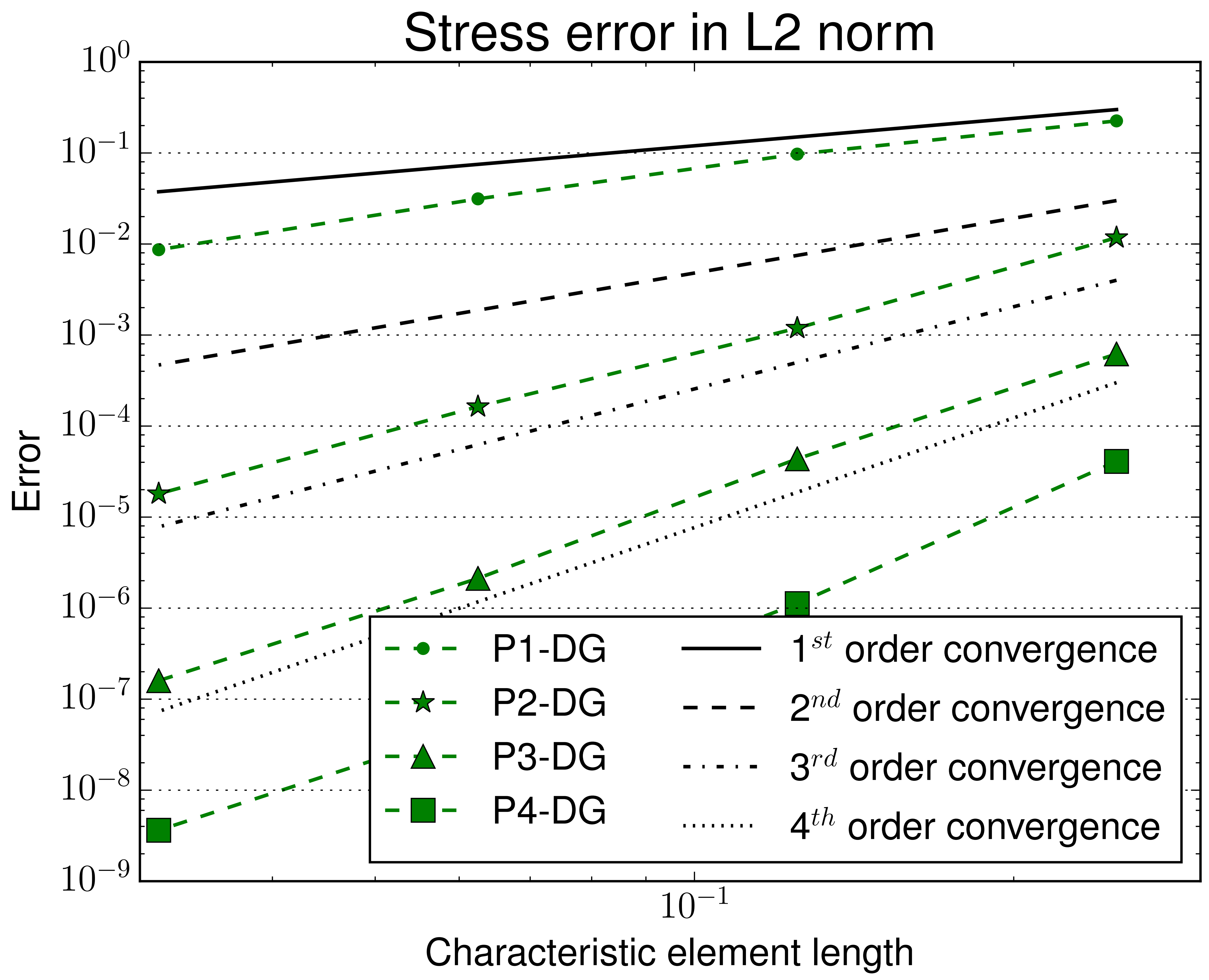
Imperial College London

Firedrake: automating computation for finite element

Using Firedrake we implemented **Seigen** - solves elastic wave equation on unstructured meshes using discontinuous Galerkin finite element (DG-FEM) using first-order upwinding and time integration via fourth-order explicit leapfrog method.

Code verification

In general, method of manufactured solutions used to verify implementation. The test case presented here is the propagation of an eigenmode on a triangular unit meshes.



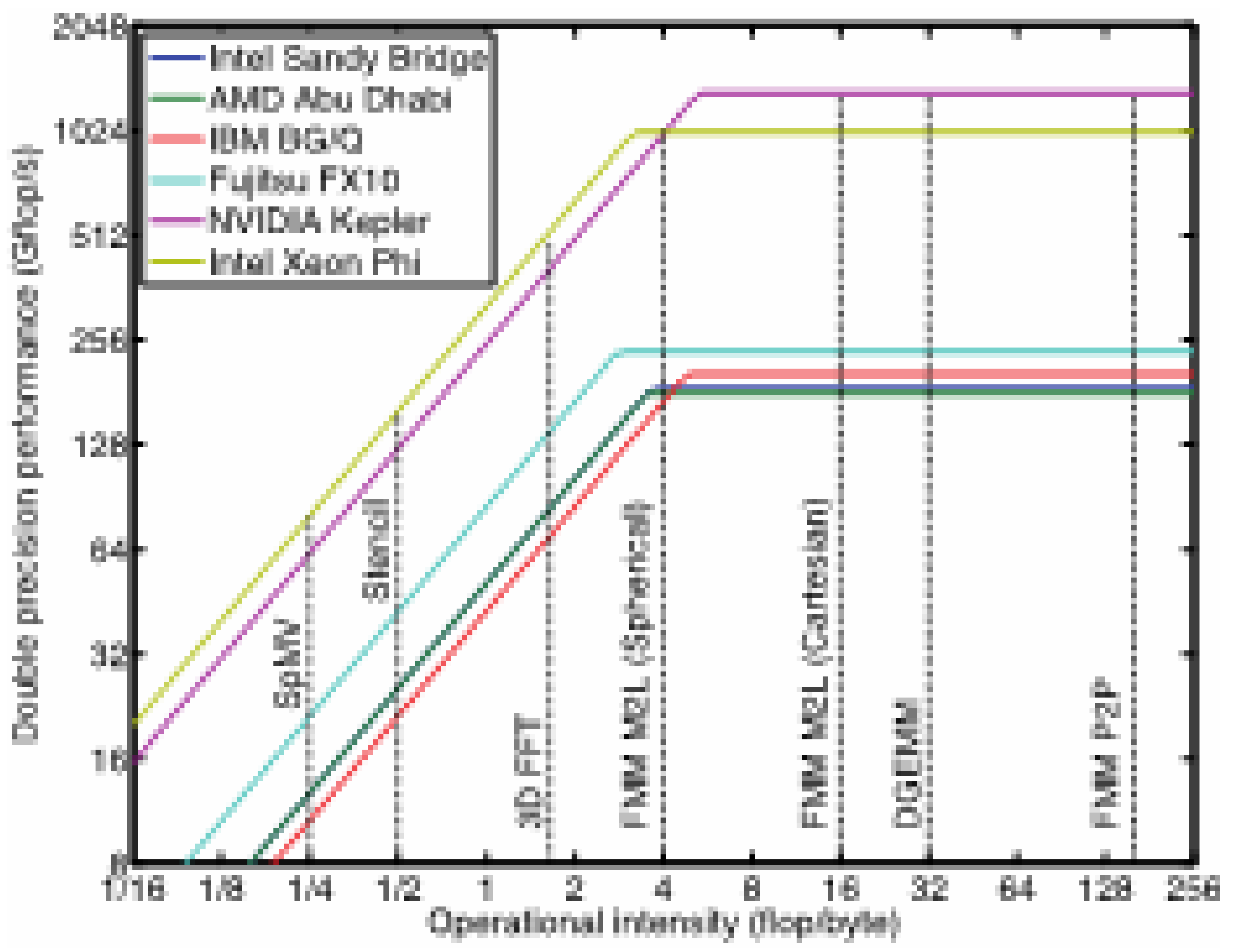
Imperial College London

Performance results

- Defining the model equations in symbolic form allows the exploration and utilisation of various numerical schemes and discretisations with significantly less programming effort than in monolithic codes, as well as allowing various performance enhancements on multiple levels.
- Implicit and explicit solution methods can be compared directly using the same formulations of the governing equations.
- Increasing the order of the spatial discretisation requires as little as changing a single parameter, which provides close control over the trade-off between model error and computational work.
- Automated low-level optimisation of assembly kernels through COFFEE, a loop transforming compiler that uses expression rewriting to minimise FLOPs and thus increase performance.

Quick revision: Roofline model - Rio Yokota's original slide

7 Dwarfs and Roofline Model



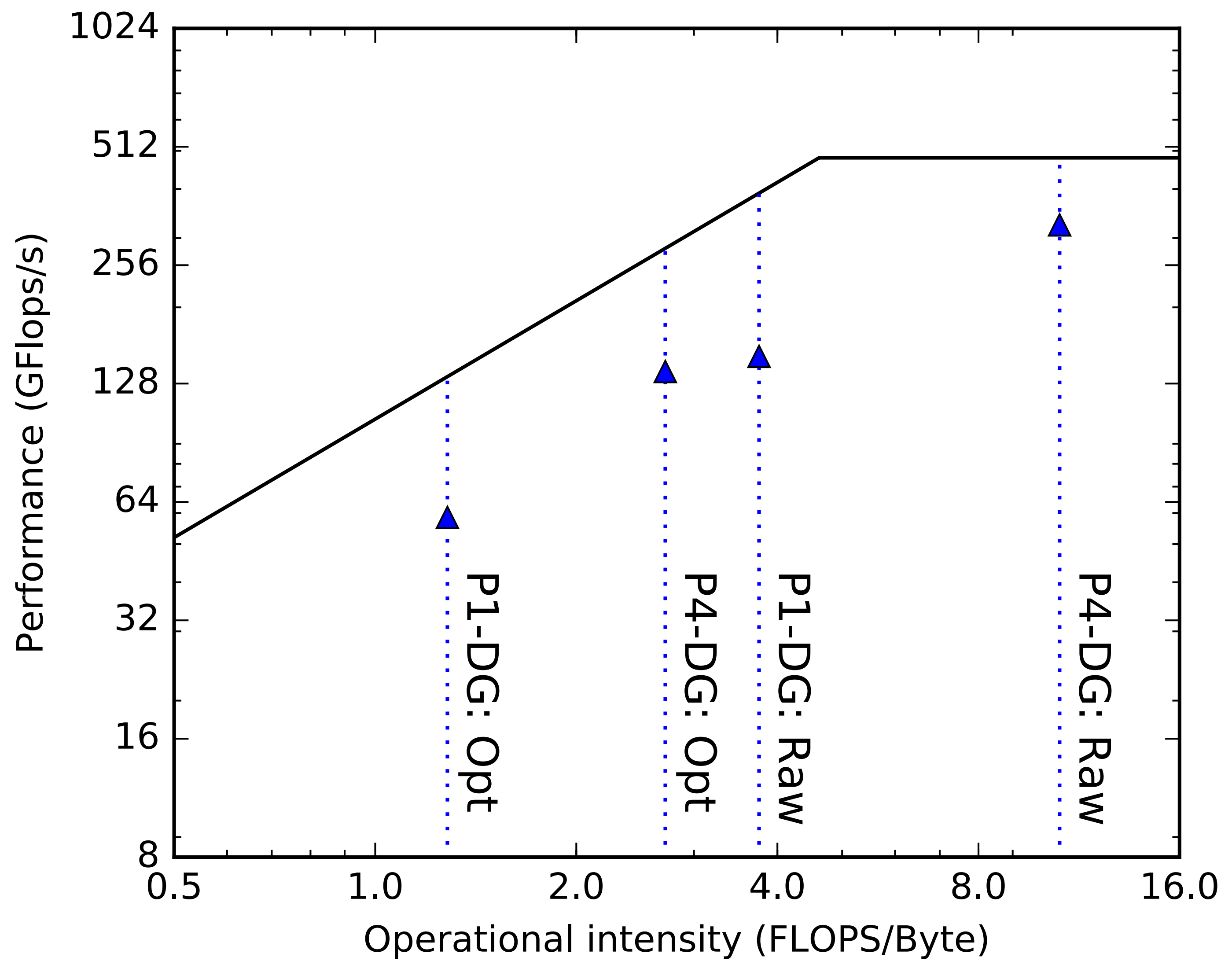
Rio Yokota

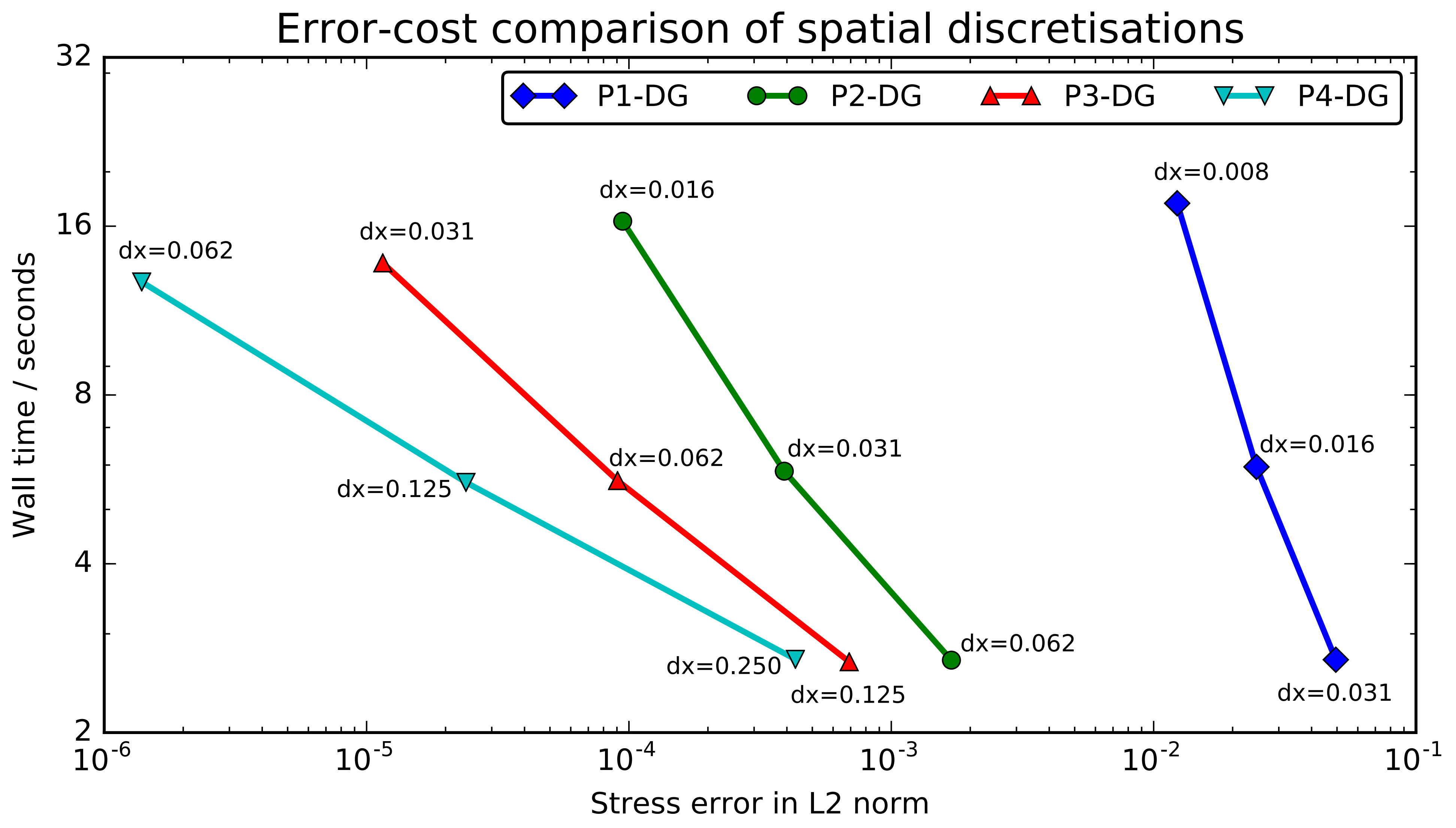
2

Imperial College London

Performance results

Performance results for an auto-generated interior facet assembly kernel on P1-DG and P4-DG discretisations. The increase in FLOPs from using a higher-order method is counter-acted by compiler optimisations that reduce the operational intensity until performance is again limited by memory bandwidth.





Imperial College London

Automating computation for finite difference

Imperial College
London

SymPy - Symbolic computation in Python

- Symbolic computer algebra system (CAS) written in pure Python[1]
- *Features:*
  - Enable rapid development - hours rather than months.
  - Greatly eases the development of adjoint models.
  - High degree of automation enables extensive test coverage.
  - Complex symbolic expressions as Python object trees.
  - Symbolic manipulation routines and interfaces.
  - Convert symbolic expressions to numeric functions:
    - Python of NumPy functions
    - C or Fortran kernels
  - For a great overview see A. Meuer's talk at SciPy 2016

For specialised domains generating C code is not enough!

1

Imperial College London

SymPy - vibrant community

```
# Mixed derivative bug reported 13 April 2016.
f = Function('f')(x, y)
d2fdxdy = f.diff(x).diff(y)
pprint(as_finite_diff(d2fdxdy, wrt=x))
-f(x - 1/2, y) + f(x + 1/2, y)

# Fixed committed 29 May 2016
f = Function('f')(x, y)
d2fdxdy = f.diff(x).diff(y)
pprint(as_finite_diff(d2fdxdy, wrt=x))
-Derivative(f(x - 1/2, y), y) + Derivative(f(x + 1/2, y), y)

# Enhancement committed 01 June 2016
pprint(as_finite_diff(as_finite_diff(d2fdxdy, wrt=x),wrt=y))
f(x - 1/2, y - 1/2) - f(x - 1/2, y + 1/2) - f(x + 1/2, y - 1/2) + f(x + 1/2, y +
 ↪   1/2)
dfgdx = (f/g).diff(x)
pprint(as_finite_diff(dfgdx, wrt=x))
(-f(x - 1/2, y) + f(x + 1/2, y))/g(x, y) - ((-g(x - 1/2, y) + g(x + 1/2, y)).f(x,
 ↪   y))/(g (x, y) * g (x, y))

# Merged into master on September 8th.
```

Todd Park's interpretation of Joy's Law: "Even if you get the best and the brightest to work for you, there will always be an infinite number of other, smarter people employed by others."

Imperial College
London

Devito - A Finite Difference DSL

Devito - A Finite Difference DSL for seismic imaging

- Aimed at creating fast high-order inversion kernels
- Diesign and development is driven by "real-world" problems

Devito is based on SymPy expressions

- Acoustic wave equation:

$$m\frac{\partial^2 u}{\partial t^2} + \eta\frac{\partial u}{\partial t} - \nabla^2 u = 0$$

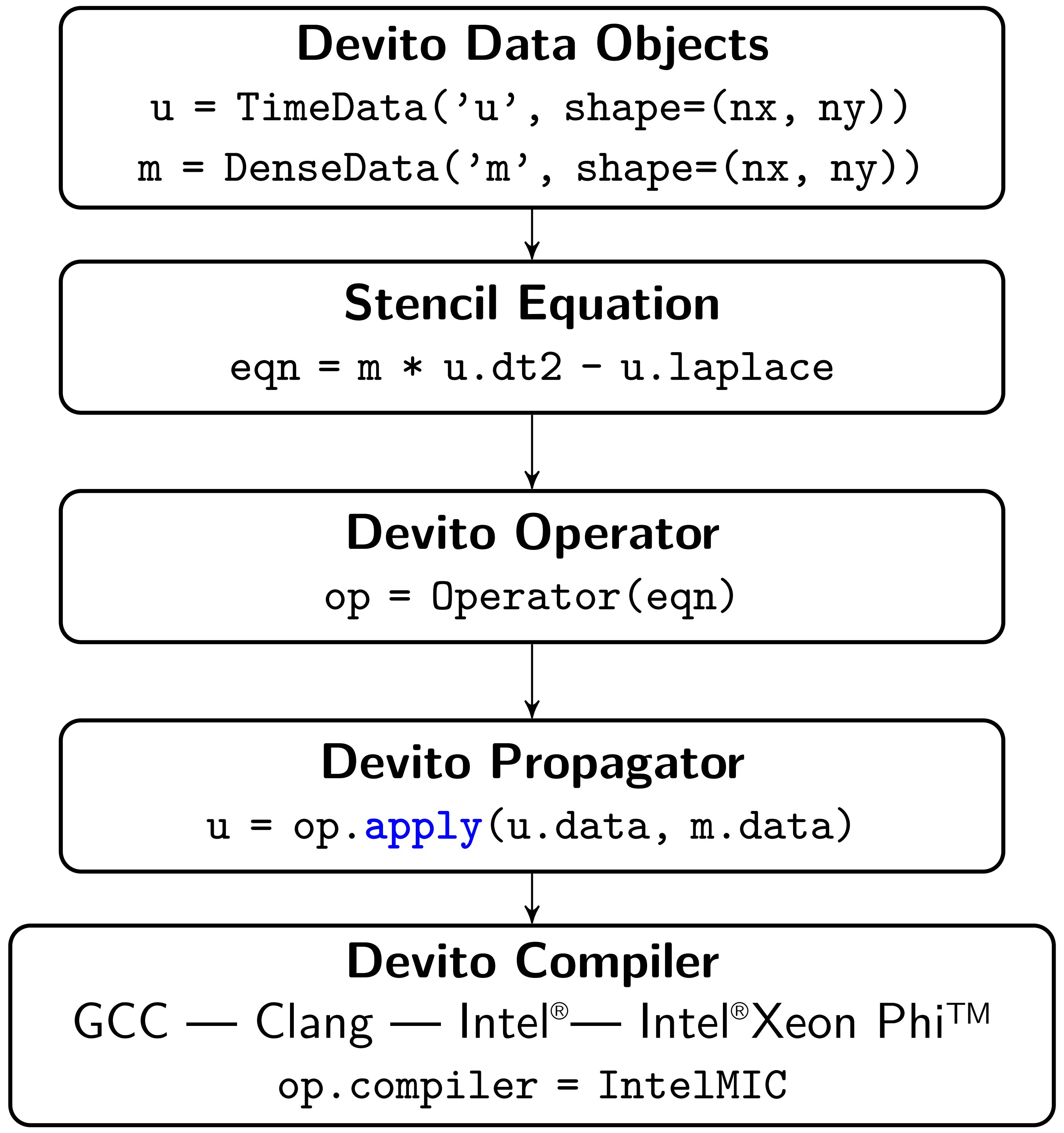
can be defined symbolically as

```
eqn = m * u.dt2 + eta * u.dt - u.laplace
```

Devito auto-generates optimised C kernel code

- OpenMP threading and vectorisation pragmas
- Cache blocking and auto-tuning
- Symbolic stencil optimisation (eg. CSE, hoisting)

Devito - A Finite Difference DSL



Devito Data Objects

```
u = TimeData('u', shape=(nx, ny))
m = DenseData('m', shape=(nx, ny))
```

High-level function symbols associated with user data

Stencil Equation

```
eqn = m * u.dt2 - u.laplace
```

Symbolic equations that expand Finite Difference stencils

Devito Operator

```
op = Operator(eqn)
```

Transform stencil expressions into explicit array accesses

Devito Propagator

```
u = op.apply(u.data, m.data)
```

Generate low-level optimized kernel code and apply to data

Devito Compiler

GCC — Clang — Intel® — Intel® Xeon Phi™

```
op.compiler = IntelMIC
```

Compiles and loads Platform specific executable function

Devito - A Finite Difference DSL

Real-world applications need more than PDE solves

- File I/O and support for large data sets
- Non-PDE kernel code, eg. sparse point interpolation

Devito follows the prinicple of Graceful Degradation

- Circumvent restrictions to the high-level API by customisation
- Devito translates high-level PDE-based stencils into “matrix index” format

```
# High-level expression equivalent to f.dx2
(-2*f(x, y) + f(x - h, y) + f(x + h, y)) / h**2

# Low-level expression with explicit indexing
(-2*f[x, y] + f[x - 1, y] + f[x + 1, y]) / h**2
```

- **Allows custom functionality in auto-generated kernels**

Devito - A Finite Difference DSL

Automated code optimisations:

- OpenMP and vectorisation pragmas
- Loop blocking and auto-tuning for block size
- Automated roofline plotting for performance analysis
- Just-In-Time (JIT) compilation aids aggressive compiler optimisation.
- Choice of API's - focus on Python because of potential to integrate with TensorFlow and other data science frameworks - could equally generate API's for C, Fortran or Julia.

Symbolic optimisations:

- Common sub-expression elemination:
  - Reduces compilation time from hours to seconds for large stencils
  - Enables further factorisation techniques to reduce flops

Imperial College London

Adjoint model in SymPy

The adjoint can often pose a major headache for developers - code generation immediately reduces the complexity.

```
# Derive stencil for adjoint model
wave_equationA = m * dtt - (dxx + dyy) - e * dt
stencilA = solve(wave_equationA, p(x, y, t-s))[0]
```

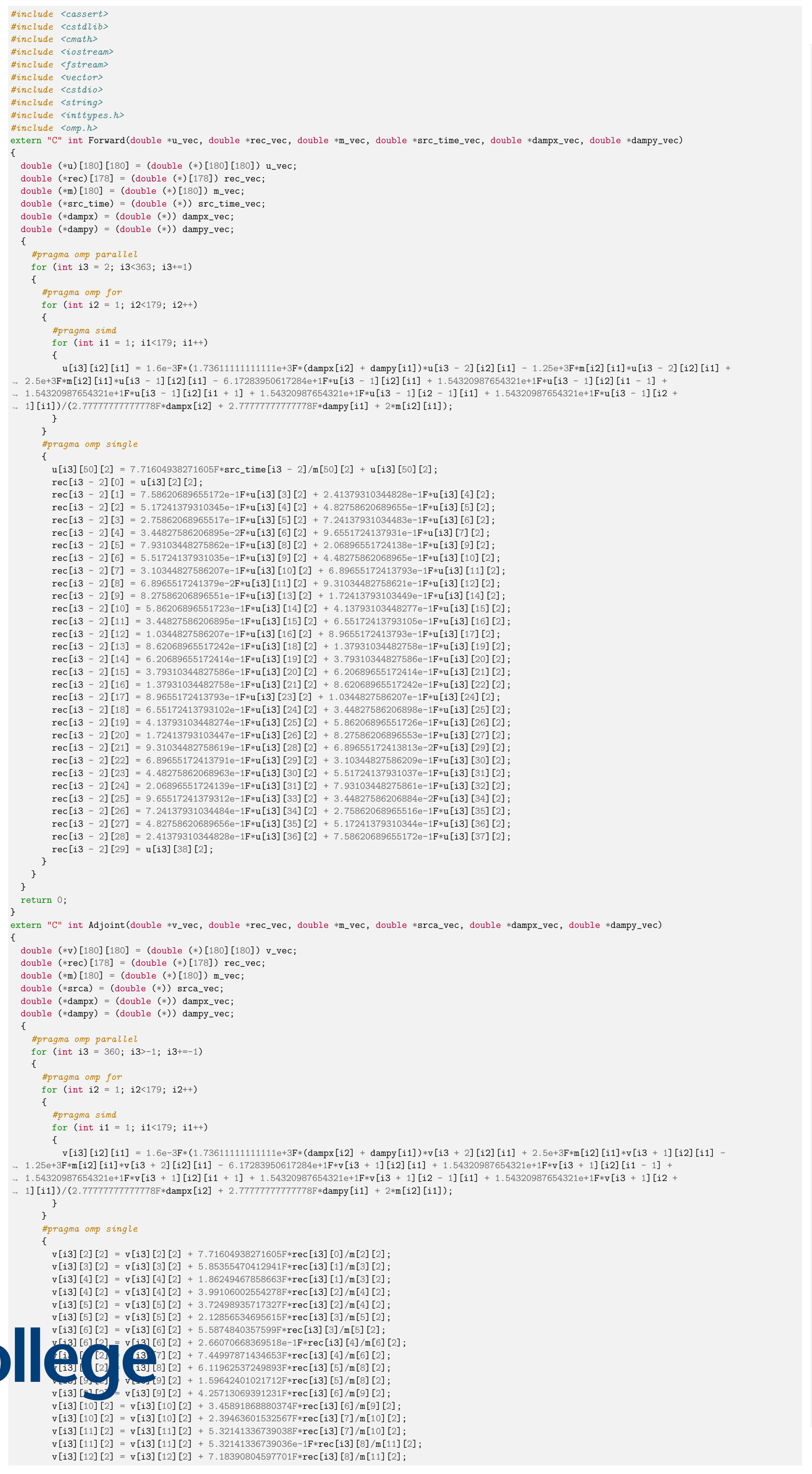
```
# Gradient calculation (performs the adjoint and cross correlation together.
def Gradient(self, rec, u):
    dt = self.dt
    grad = self._gradient_stencil.apply()[0]
    return (dt**-2)*grad

# Adjoint model as a separate call created purely for verification.
def Adjoint(self, rec):
    v = self._adjoint_stencil.apply()[0]
    return (self.srca.data, v)
```

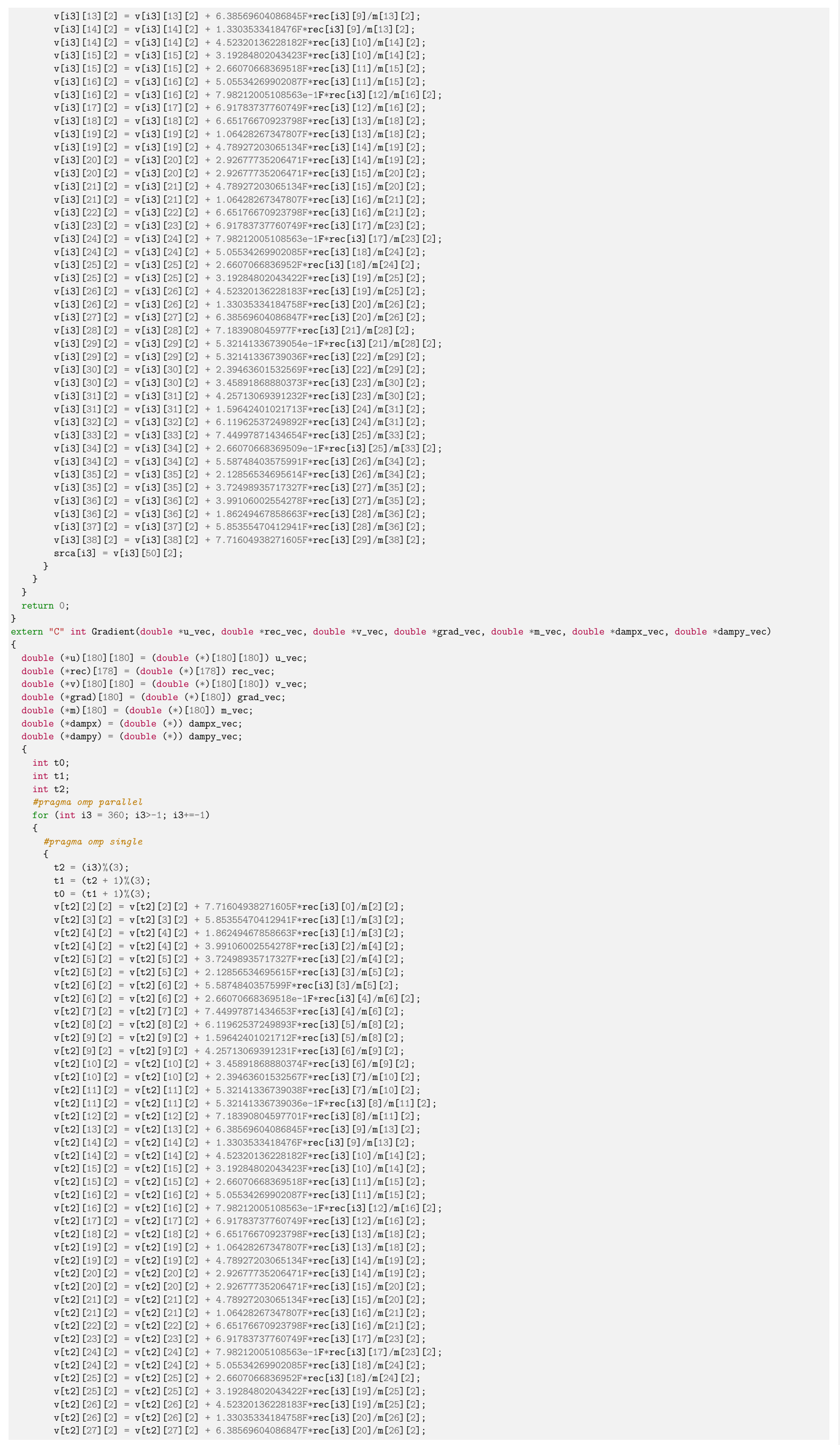
Can rigorously verify numerical discretisation - adjoint model and gradient defined at a high level.

Code generated for 2D acoustic propogator

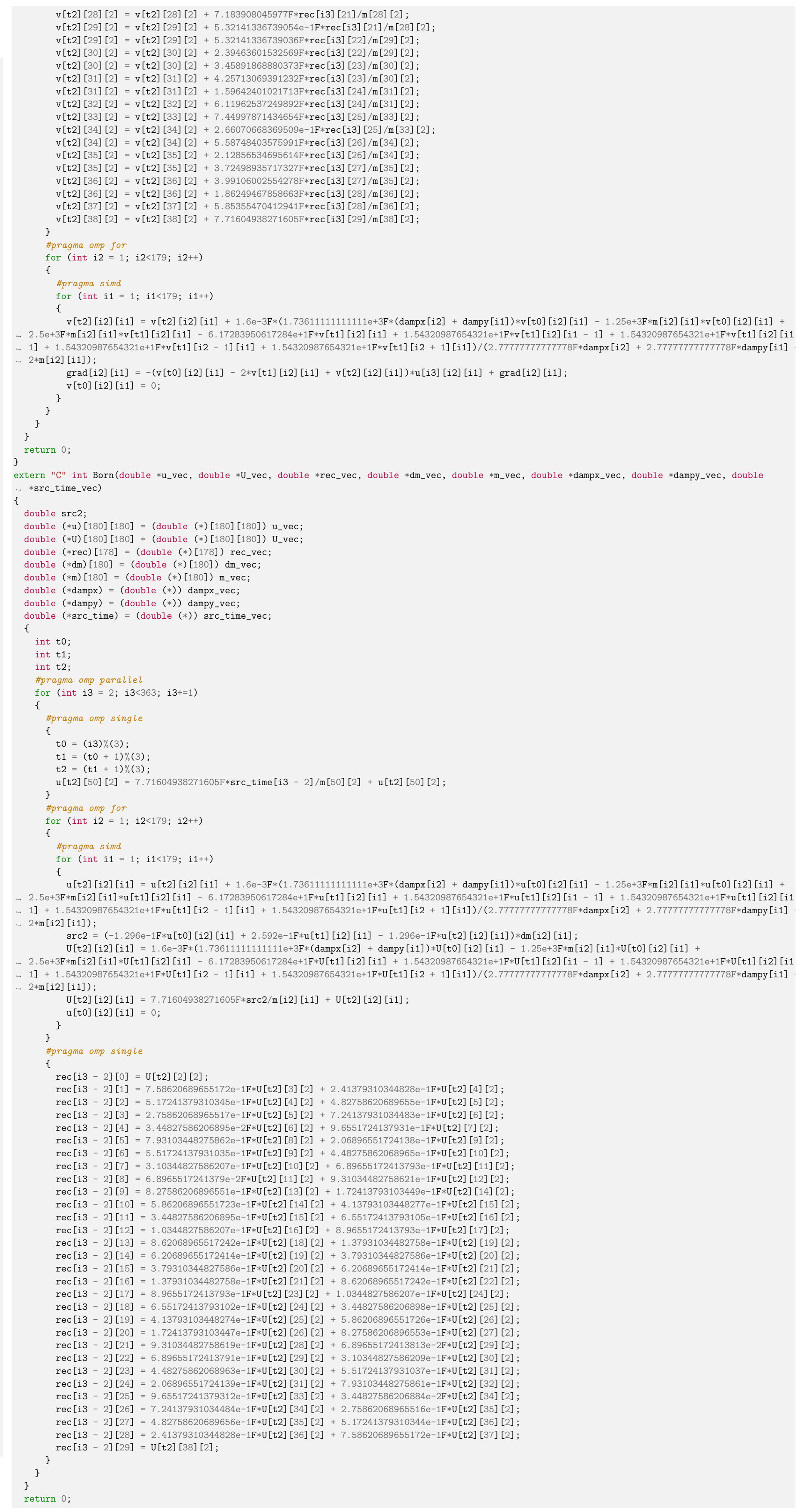
Automating generating C code with OpenMP and SIMD vectorisation: Just-in-time (JIT) compilation enables run time compilation when data parameters are known — enables more aggressive optimisation.



```
#include <cassert>
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <vector>
#include <cstdio>
#include <string>
#include <inttypes.h>
#include <omp.h>
extern "C" int Forward(double *u_vec, double *rec_vec, double *m_vec, double *src_time_vec, double *dampx_vec, double *dampy_vec)
{
  double (*u)[180][180] = (double (*)[180][180]) u_vec;
  double (*rec)[178] = (double (*)[178]) rec_vec;
  double (*m)[180] = (double (*)[180]) m_vec;
  double (*src_time) = (double (*)) src_time_vec;
  double (*dampx) = (double (*)) dampx_vec;
  double (*dampy) = (double (*)) dampy_vec;
  {
    #pragma omp parallel
    for (int i3 = 2; i3<363; i3+=1)
    {
      #pragma omp for
      for (int i2 = 1; i2<179; i2++)
      {
        #pragma simd
        for (int i1 = 1; i1<179; i1++)
        {
          u[i3][i2][i1] = 1.6e-3F*(1.73611111111111e+3F*(dampx[i2] + dampy[i1])*u[i3 - 2][i2][i1] - 1.25e+3F*m[i2][i1]*u[i3 - 2][i2][i1] +
→ 2.5e+3F*m[i2][i1]*u[i3 - 1][i2][i1] - 6.17283950617284e+1F*u[i3 - 1][i2][i1] + 1.54320987654321e+1F*u[i3 - 1][i2][i1 - 1] +
→ 1.54320987654321e+1F*u[i3 - 1][i2][i1 + 1] + 1.54320987654321e+1F*u[i3 - 1][i2 - 1][i1] + 1.54320987654321e+1F*u[i3 - 1][i2 +
→ 1][i1])/(2.77777777777778F*dampx[i2] + 2.77777777777778F*dampy[i1] + 2*m[i2][i1]);
        }
      }
      #pragma omp single
      {
        u[i3][50][2] = 7.71604938271605F*src_time[i3 - 2]/m[50][2] + u[i3][50][2];
        rec[i3 - 2][0] = u[i3][2][2];
        rec[i3 - 2][1] = 7.58620689655172e-1F*u[i3][3][2] + 2.41379310344828e-1F*u[i3][4][2];
        rec[i3 - 2][2] = 5.17241379310345e-1F*u[i3][4][2] + 4.82758620689655e-1F*u[i3][5][2];
        rec[i3 - 2][3] = 2.75862068965517e-1F*u[i3][5][2] + 7.24137931034483e-1F*u[i3][6][2];
        rec[i3 - 2][4] = 3.44827586206895e-2F*u[i3][6][2] + 9.6551724137931e-1F*u[i3][7][2];
        rec[i3 - 2][5] = 7.93103448275862e-1F*u[i3][8][2] + 2.06896551724138e-1F*u[i3][9][2];
        rec[i3 - 2][6] = 5.51724137931035e-1F*u[i3][9][2] + 4.48275862068965e-1F*u[i3][10][2];
        rec[i3 - 2][7] = 3.10344827586207e-1F*u[i3][10][2] + 6.89655172413793e-1F*u[i3][11][2];
        rec[i3 - 2][8] = 6.8965517241379e-2F*u[i3][11][2] + 9.31034482758621e-1F*u[i3][12][2];
        rec[i3 - 2][9] = 8.27586206896551e-1F*u[i3][13][2] + 1.72413793103449e-1F*u[i3][14][2];
        rec[i3 - 2][10] = 5.86206896551723e-1F*u[i3][14][2] + 4.13793103448277e-1F*u[i3][15][2];
        rec[i3 - 2][11] = 3.44827586206895e-1F*u[i3][15][2] + 6.55172413793105e-1F*u[i3][16][2];
        rec[i3 - 2][12] = 1.0344827586207e-1F*u[i3][16][2] + 8.9655172413793e-1F*u[i3][17][2];
        rec[i3 - 2][13] = 8.62068965517242e-1F*u[i3][18][2] + 1.37931034482758e-1F*u[i3][19][2];
        rec[i3 - 2][14] = 6.20689655172414e-1F*u[i3][19][2] + 3.79310344827586e-1F*u[i3][20][2];
        rec[i3 - 2][15] = 3.79310344827586e-1F*u[i3][20][2] + 6.20689655172414e-1F*u[i3][21][2];
        rec[i3 - 2][16] = 1.37931034482758e-1F*u[i3][21][2] + 8.62068965517242e-1F*u[i3][22][2];
        rec[i3 - 2][17] = 8.9655172413793e-1F*u[i3][23][2] + 1.0344827586207e-1F*u[i3][24][2];
        rec[i3 - 2][18] = 6.55172413793102e-1F*u[i3][24][2] + 3.44827586206898e-1F*u[i3][25][2];
        rec[i3 - 2][19] = 4.13793103448274e-1F*u[i3][25][2] + 5.86206896551726e-1F*u[i3][26][2];
        rec[i3 - 2][20] = 1.72413793103447e-1F*u[i3][26][2] + 8.27586206896553e-1F*u[i3][27][2];
        rec[i3 - 2][21] = 9.31034482758619e-1F*u[i3][28][2] + 6.89655172413813e-2F*u[i3][29][2];
        rec[i3 - 2][22] = 6.89655172413791e-1F*u[i3][29][2] + 3.10344827586209e-1F*u[i3][30][2];
        rec[i3 - 2][23] = 4.48275862068963e-1F*u[i3][30][2] + 5.51724137931037e-1F*u[i3][31][2];
        rec[i3 - 2][24] = 2.06896551724139e-1F*u[i3][31][2] + 7.93103448275861e-1F*u[i3][32][2];
        rec[i3 - 2][25] = 9.65517241379312e-1F*u[i3][33][2] + 3.44827586206884e-2F*u[i3][34][2];
        rec[i3 - 2][26] = 7.24137931034484e-1F*u[i3][34][2] + 2.75862068965516e-1F*u[i3][35][2];
        rec[i3 - 2][27] = 4.82758620689656e-1F*u[i3][35][2] + 5.17241379310344e-1F*u[i3][36][2];
        rec[i3 - 2][28] = 2.41379310344828e-1F*u[i3][36][2] + 7.58620689655172e-1F*u[i3][37][2];
        rec[i3 - 2][29] = u[i3][38][2];
      }
    }
  }
  return 0;
}
extern "C" int Adjoint(double *v_vec, double *rec_vec, double *m_vec, double *srca_vec, double *dampx_vec, double *dampy_vec)
{
  double (*v)[180][180] = (double (*)[180][180]) v_vec;
  double (*rec)[178] = (double (*)[178]) rec_vec;
  double (*m)[180] = (double (*)[180]) m_vec;
  double (*srca) = (double (*)) srca_vec;
  double (*dampx) = (double (*)) dampx_vec;
  double (*dampy) = (double (*)) dampy_vec;
  {
    #pragma omp parallel
    for (int i3 = 360; i3>-1; i3+=-1)
    {
      #pragma omp for
      for (int i2 = 1; i2<179; i2++)
      {
        #pragma simd
        for (int i1 = 1; i1<179; i1++)
        {
          v[i3][i2][i1] = 1.6e-3F*(1.73611111111111e+3F*(dampx[i2] + dampy[i1])*v[i3 + 2][i2][i1] + 2.5e+3F*m[i2][i1]*v[i3 + 1][i2][i1] -
→ 1.25e+3F*m[i2][i1]*v[i3 + 2][i2][i1] - 6.17283950617284e+1F*v[i3 + 1][i2][i1] + 1.54320987654321e+1F*v[i3 + 1][i2][i1 - 1] +
→ 1.54320987654321e+1F*v[i3 + 1][i2][i1 + 1] + 1.54320987654321e+1F*v[i3 + 1][i2 - 1][i1] + 1.54320987654321e+1F*v[i3 + 1][i2 +
→ 1][i1])/(2.77777777777778F*dampx[i2] + 2.77777777777778F*dampy[i1] + 2*m[i2][i1]);
        }
      }
      #pragma omp single
      {
        v[i3][2][2] = v[i3][2][2] + 7.71604938271605F*rec[i3][0]/m[2][2];
        v[i3][3][2] = v[i3][3][2] + 5.85355470412941F*rec[i3][1]/m[3][2];
        v[i3][4][2] = v[i3][4][2] + 1.86249467858663F*rec[i3][1]/m[3][2];
        v[i3][4][2] = v[i3][4][2] + 3.99106002554278F*rec[i3][2]/m[4][2];
        v[i3][5][2] = v[i3][5][2] + 3.72498935717327F*rec[i3][2]/m[4][2];
        v[i3][5][2] = v[i3][5][2] + 2.12856534695615F*rec[i3][3]/m[5][2];
        v[i3][6][2] = v[i3][6][2] + 5.5874840357599F*rec[i3][3]/m[5][2];
        v[i3][6][2] = v[i3][6][2] + 2.66070668369518e-1F*rec[i3][4]/m[6][2];
        v[i3][7][2] = v[i3][7][2] + 7.44997871434653F*rec[i3][4]/m[6][2];
        v[i3][8][2] = v[i3][8][2] + 6.11962537249893F*rec[i3][5]/m[8][2];
        v[i3][9][2] = v[i3][9][2] + 1.59642401021712F*rec[i3][5]/m[8][2];
        v[i3][9][2] = v[i3][9][2] + 4.25713069391231F*rec[i3][6]/m[9][2];
        v[i3][10][2] = v[i3][10][2] + 3.45891868880374F*rec[i3][6]/m[9][2];
        v[i3][10][2] = v[i3][10][2] + 2.39463601532567F*rec[i3][7]/m[10][2];
        v[i3][11][2] = v[i3][11][2] + 5.32141336739038F*rec[i3][7]/m[10][2];
        v[i3][11][2] = v[i3][11][2] + 5.32141336739036e-1F*rec[i3][8]/m[11][2];
        v[i3][12][2] = v[i3][12][2] + 7.18390804597701F*rec[i3][8]/m[11][2];
```



```
        v[i3][13][2] = v[i3][13][2] + 6.38569604086845F*rec[i3][9]/m[13][2];
        v[i3][14][2] = v[i3][14][2] + 1.3303533418476F*rec[i3][9]/m[13][2];
        v[i3][14][2] = v[i3][14][2] + 4.52320136228182F*rec[i3][10]/m[14][2];
        v[i3][15][2] = v[i3][15][2] + 3.19284802043423F*rec[i3][10]/m[14][2];
        v[i3][15][2] = v[i3][15][2] + 2.66070668369518F*rec[i3][11]/m[15][2];
        v[i3][16][2] = v[i3][16][2] + 5.05534269902087F*rec[i3][11]/m[15][2];
        v[i3][16][2] = v[i3][16][2] + 7.98212005108563e-1F*rec[i3][12]/m[16][2];
        v[i3][17][2] = v[i3][17][2] + 6.91783737760749F*rec[i3][12]/m[16][2];
        v[i3][18][2] = v[i3][18][2] + 6.65176670923798F*rec[i3][13]/m[18][2];
        v[i3][19][2] = v[i3][19][2] + 1.06428267347807F*rec[i3][13]/m[18][2];
        v[i3][19][2] = v[i3][19][2] + 4.78927203065134F*rec[i3][14]/m[19][2];
        v[i3][20][2] = v[i3][20][2] + 2.92677735206471F*rec[i3][14]/m[19][2];
        v[i3][20][2] = v[i3][20][2] + 2.92677735206471F*rec[i3][15]/m[20][2];
        v[i3][21][2] = v[i3][21][2] + 4.78927203065134F*rec[i3][15]/m[20][2];
        v[i3][21][2] = v[i3][21][2] + 1.06428267347807F*rec[i3][16]/m[21][2];
        v[i3][22][2] = v[i3][22][2] + 6.65176670923798F*rec[i3][16]/m[21][2];
        v[i3][23][2] = v[i3][23][2] + 6.91783737760749F*rec[i3][17]/m[23][2];
        v[i3][24][2] = v[i3][24][2] + 7.98212005108563e-1F*rec[i3][17]/m[23][2];
        v[i3][24][2] = v[i3][24][2] + 5.05534269902085F*rec[i3][18]/m[24][2];
        v[i3][25][2] = v[i3][25][2] + 2.6607066836952F*rec[i3][18]/m[24][2];
        v[i3][25][2] = v[i3][25][2] + 3.19284802043422F*rec[i3][19]/m[25][2];
        v[i3][26][2] = v[i3][26][2] + 4.52320136228183F*rec[i3][19]/m[25][2];
        v[i3][26][2] = v[i3][26][2] + 1.33035334184758F*rec[i3][20]/m[26][2];
        v[i3][27][2] = v[i3][27][2] + 6.38569604086847F*rec[i3][20]/m[26][2];
        v[i3][28][2] = v[i3][28][2] + 7.183908045977F*rec[i3][21]/m[28][2];
        v[i3][29][2] = v[i3][29][2] + 5.32141336739054e-1F*rec[i3][21]/m[28][2];
        v[i3][29][2] = v[i3][29][2] + 5.32141336739036F*rec[i3][22]/m[29][2];
        v[i3][30][2] = v[i3][30][2] + 2.39463601532569F*rec[i3][22]/m[29][2];
        v[i3][30][2] = v[i3][30][2] + 3.45891868880373F*rec[i3][23]/m[30][2];
        v[i3][31][2] = v[i3][31][2] + 4.25713069391232F*rec[i3][23]/m[30][2];
        v[i3][31][2] = v[i3][31][2] + 1.59642401021713F*rec[i3][24]/m[31][2];
        v[i3][32][2] = v[i3][32][2] + 6.11962537249892F*rec[i3][24]/m[31][2];
        v[i3][33][2] = v[i3][33][2] + 7.44997871434654F*rec[i3][25]/m[33][2];
        v[i3][34][2] = v[i3][34][2] + 2.66070668369509e-1F*rec[i3][25]/m[33][2];
        v[i3][34][2] = v[i3][34][2] + 5.58748403575991F*rec[i3][26]/m[34][2];
        v[i3][35][2] = v[i3][35][2] + 2.12856534695614F*rec[i3][26]/m[34][2];
        v[i3][35][2] = v[i3][35][2] + 3.72498935717327F*rec[i3][27]/m[35][2];
        v[i3][36][2] = v[i3][36][2] + 3.99106002554278F*rec[i3][27]/m[35][2];
        v[i3][36][2] = v[i3][36][2] + 1.86249467858663F*rec[i3][28]/m[36][2];
        v[i3][37][2] = v[i3][37][2] + 5.85355470412941F*rec[i3][28]/m[36][2];
        v[i3][38][2] = v[i3][38][2] + 7.71604938271605F*rec[i3][29]/m[38][2];
        srca[i3] = v[i3][50][2];
      }
    }
  }
  return 0;
}
extern "C" int Gradient(double *u_vec, double *rec_vec, double *v_vec, double *grad_vec, double *m_vec, double *dampx_vec, double *dampy_vec)
{
  double (*u)[180][180] = (double (*)[180][180]) u_vec;
  double (*rec)[178] = (double (*)[178]) rec_vec;
  double (*v)[180][180] = (double (*)[180][180]) v_vec;
  double (*grad)[180] = (double (*)[180]) grad_vec;
  double (*m)[180] = (double (*)[180]) m_vec;
  double (*dampx) = (double (*)) dampx_vec;
  double (*dampy) = (double (*)) dampy_vec;
  {
    int t0;
    int t1;
    int t2;
    #pragma omp parallel
    for (int i3 = 360; i3>-1; i3+=-1)
    {
      #pragma omp single
      {
        t2 = (i3)%(3);
        t1 = (t2 + 1)%(3);
        t0 = (t1 + 1)%(3);
        v[t2][2][2] = v[t2][2][2] + 7.71604938271605F*rec[i3][0]/m[2][2];
        v[t2][3][2] = v[t2][3][2] + 5.85355470412941F*rec[i3][1]/m[3][2];
        v[t2][4][2] = v[t2][4][2] + 1.86249467858663F*rec[i3][1]/m[3][2];
        v[t2][4][2] = v[t2][4][2] + 3.99106002554278F*rec[i3][2]/m[4][2];
        v[t2][5][2] = v[t2][5][2] + 3.72498935717327F*rec[i3][2]/m[4][2];
        v[t2][5][2] = v[t2][5][2] + 2.12856534695615F*rec[i3][3]/m[5][2];
        v[t2][6][2] = v[t2][6][2] + 5.5874840357599F*rec[i3][3]/m[5][2];
        v[t2][6][2] = v[t2][6][2] + 2.66070668369518e-1F*rec[i3][4]/m[6][2];
        v[t2][7][2] = v[t2][7][2] + 7.44997871434653F*rec[i3][4]/m[6][2];
        v[t2][8][2] = v[t2][8][2] + 6.11962537249893F*rec[i3][5]/m[8][2];
        v[t2][9][2] = v[t2][9][2] + 1.59642401021712F*rec[i3][5]/m[8][2];
        v[t2][9][2] = v[t2][9][2] + 4.25713069391231F*rec[i3][6]/m[9][2];
        v[t2][10][2] = v[t2][10][2] + 3.45891868880374F*rec[i3][6]/m[9][2];
        v[t2][10][2] = v[t2][10][2] + 2.39463601532567F*rec[i3][7]/m[10][2];
        v[t2][11][2] = v[t2][11][2] + 5.32141336739038F*rec[i3][7]/m[10][2];
        v[t2][11][2] = v[t2][11][2] + 5.32141336739036e-1F*rec[i3][8]/m[11][2];
        v[t2][12][2] = v[t2][12][2] + 7.18390804597701F*rec[i3][8]/m[11][2];
        v[t2][13][2] = v[t2][13][2] + 6.38569604086845F*rec[i3][9]/m[13][2];
        v[t2][14][2] = v[t2][14][2] + 1.3303533418476F*rec[i3][9]/m[13][2];
        v[t2][14][2] = v[t2][14][2] + 4.52320136228182F*rec[i3][10]/m[14][2];
        v[t2][15][2] = v[t2][15][2] + 3.19284802043423F*rec[i3][10]/m[14][2];
        v[t2][15][2] = v[t2][15][2] + 2.66070668369518F*rec[i3][11]/m[15][2];
        v[t2][16][2] = v[t2][16][2] + 5.05534269902087F*rec[i3][11]/m[15][2];
        v[t2][16][2] = v[t2][16][2] + 7.98212005108563e-1F*rec[i3][12]/m[16][2];
        v[t2][17][2] = v[t2][17][2] + 6.91783737760749F*rec[i3][12]/m[16][2];
        v[t2][18][2] = v[t2][18][2] + 6.65176670923798F*rec[i3][13]/m[18][2];
        v[t2][19][2] = v[t2][19][2] + 1.06428267347807F*rec[i3][13]/m[18][2];
        v[t2][19][2] = v[t2][19][2] + 4.78927203065134F*rec[i3][14]/m[19][2];
        v[t2][20][2] = v[t2][20][2] + 2.92677735206471F*rec[i3][14]/m[19][2];
        v[t2][20][2] = v[t2][20][2] + 2.92677735206471F*rec[i3][15]/m[20][2];
        v[t2][21][2] = v[t2][21][2] + 4.78927203065134F*rec[i3][15]/m[20][2];
        v[t2][21][2] = v[t2][21][2] + 1.06428267347807F*rec[i3][16]/m[21][2];
        v[t2][22][2] = v[t2][22][2] + 6.65176670923798F*rec[i3][16]/m[21][2];
        v[t2][23][2] = v[t2][23][2] + 6.91783737760749F*rec[i3][17]/m[23][2];
        v[t2][24][2] = v[t2][24][2] + 7.98212005108563e-1F*rec[i3][17]/m[23][2];
        v[t2][24][2] = v[t2][24][2] + 5.05534269902085F*rec[i3][18]/m[24][2];
        v[t2][25][2] = v[t2][25][2] + 2.6607066836952F*rec[i3][18]/m[24][2];
        v[t2][25][2] = v[t2][25][2] + 3.19284802043422F*rec[i3][19]/m[25][2];
        v[t2][26][2] = v[t2][26][2] + 4.52320136228183F*rec[i3][19]/m[25][2];
        v[t2][26][2] = v[t2][26][2] + 1.33035334184758F*rec[i3][20]/m[26][2];
        v[t2][27][2] = v[t2][27][2] + 6.38569604086847F*rec[i3][20]/m[26][2];
```



```
        v[t2][28][2] = v[t2][28][2] + 7.183908045977F*rec[i3][21]/m[28][2];
        v[t2][29][2] = v[t2][29][2] + 5.32141336739054e-1F*rec[i3][21]/m[28][2];
        v[t2][29][2] = v[t2][29][2] + 5.32141336739036F*rec[i3][22]/m[29][2];
        v[t2][30][2] = v[t2][30][2] + 2.39463601532569F*rec[i3][22]/m[29][2];
        v[t2][30][2] = v[t2][30][2] + 3.45891868880373F*rec[i3][23]/m[30][2];
        v[t2][31][2] = v[t2][31][2] + 4.25713069391232F*rec[i3][23]/m[30][2];
        v[t2][31][2] = v[t2][31][2] + 1.59642401021713F*rec[i3][24]/m[31][2];
        v[t2][32][2] = v[t2][32][2] + 6.11962537249892F*rec[i3][24]/m[31][2];
        v[t2][33][2] = v[t2][33][2] + 7.44997871434654F*rec[i3][25]/m[33][2];
        v[t2][34][2] = v[t2][34][2] + 2.66070668369509e-1F*rec[i3][25]/m[33][2];
        v[t2][34][2] = v[t2][34][2] + 5.58748403575991F*rec[i3][26]/m[34][2];
        v[t2][35][2] = v[t2][35][2] + 2.12856534695614F*rec[i3][26]/m[34][2];
        v[t2][35][2] = v[t2][35][2] + 3.72498935717327F*rec[i3][27]/m[35][2];
        v[t2][36][2] = v[t2][36][2] + 3.99106002554278F*rec[i3][27]/m[35][2];
        v[t2][36][2] = v[t2][36][2] + 1.86249467858663F*rec[i3][28]/m[36][2];
        v[t2][37][2] = v[t2][37][2] + 5.85355470412941F*rec[i3][28]/m[36][2];
        v[t2][38][2] = v[t2][38][2] + 7.71604938271605F*rec[i3][29]/m[38][2];
      }
      #pragma omp for
      for (int i2 = 1; i2<179; i2++)
      {
        #pragma simd
        for (int i1 = 1; i1<179; i1++)
        {
          v[t2][i2][i1] = v[t2][i2][i1] + 1.6e-3F*(1.73611111111111e+3F*(dampx[i2] + dampy[i1])*v[t0][i2][i1] - 1.25e+3F*m[i2][i1]*v[t0][i2][i1] +
→ 2.5e+3F*m[i2][i1]*v[t1][i2][i1] - 6.17283950617284e+1F*v[t1][i2][i1] + 1.54320987654321e+1F*v[t1][i2][i1 - 1] + 1.54320987654321e+1F*v[t1][i2][i1
→ 1] + 1.54320987654321e+1F*v[t1][i2 - 1][i1] + 1.54320987654321e+1F*v[t1][i2 + 1][i1])/(2.77777777777778F*dampx[i2] + 2.77777777777778F*dampy[i1]
→ 2*m[i2][i1]);
          grad[i2][i1] = -(v[t0][i2][i1] - 2*v[t1][i2][i1] + v[t2][i2][i1])*u[i3][i2][i1] + grad[i2][i1];
          v[t0][i2][i1] = 0;
        }
      }
    }
  }
  return 0;
}
extern "C" int Born(double *u_vec, double *U_vec, double *rec_vec, double *dm_vec, double *m_vec, double *dampx_vec, double *dampy_vec, double
→ *src_time_vec)
{
  double src2;
  double (*u)[180][180] = (double (*)[180][180]) u_vec;
  double (*U)[180][180] = (double (*)[180][180]) U_vec;
  double (*rec)[178] = (double (*)[178]) rec_vec;
  double (*dm)[180] = (double (*)[180]) dm_vec;
  double (*m)[180] = (double (*)[180]) m_vec;
  double (*dampx) = (double (*)) dampx_vec;
  double (*dampy) = (double (*)) dampy_vec;
  double (*src_time) = (double (*)) src_time_vec;
  {
    int t0;
    int t1;
    int t2;
    #pragma omp parallel
    for (int i3 = 2; i3<363; i3+=1)
    {
      #pragma omp single
      {
        t0 = (i3)%(3);
        t1 = (t0 + 1)%(3);
        t2 = (t1 + 1)%(3);
        u[t2][50][2] = 7.71604938271605F*src_time[i3 - 2]/m[50][2] + u[t2][50][2];
      }
      #pragma omp for
      for (int i2 = 1; i2<179; i2++)
      {
        #pragma simd
        for (int i1 = 1; i1<179; i1++)
        {
          u[t2][i2][i1] = u[t2][i2][i1] + 1.6e-3F*(1.73611111111111e+3F*(dampx[i2] + dampy[i1])*u[t0][i2][i1] - 1.25e+3F*m[i2][i1]*u[t0][i2][i1] +
→ 2.5e+3F*m[i2][i1]*u[t1][i2][i1] - 6.17283950617284e+1F*u[t1][i2][i1] + 1.54320987654321e+1F*u[t1][i2][i1 - 1] + 1.54320987654321e+1F*u[t1][i2][i1
→ 1] + 1.54320987654321e+1F*u[t1][i2 - 1][i1] + 1.54320987654321e+1F*u[t1][i2 + 1][i1])/(2.77777777777778F*dampx[i2] + 2.77777777777778F*dampy[i1]
→ 2*m[i2][i1]);
          src2 = (-1.296e-1F*u[t0][i2][i1] + 2.592e-1F*u[t1][i2][i1] - 1.296e-1F*u[t2][i2][i1])*dm[i2][i1];
          U[t2][i2][i1] = 1.6e-3F*(1.73611111111111e+3F*(dampx[i2] + dampy[i1])*U[t0][i2][i1] - 1.25e+3F*m[i2][i1]*U[t0][i2][i1] +
→ 2.5e+3F*m[i2][i1]*U[t1][i2][i1] - 6.17283950617284e+1F*U[t1][i2][i1] + 1.54320987654321e+1F*U[t1][i2][i1 - 1] + 1.54320987654321e+1F*U[t1][i2][i1
→ 1] + 1.54320987654321e+1F*U[t1][i2 - 1][i1] + 1.54320987654321e+1F*U[t1][i2 + 1][i1])/(2.77777777777778F*dampx[i2] + 2.77777777777778F*dampy[i1]
→ 2*m[i2][i1]);
          U[t2][i2][i1] = 7.71604938271605F*src2/m[i2][i1] + U[t2][i2][i1];
          u[t0][i2][i1] = 0;
        }
      }
      #pragma omp single
      {
        rec[i3 - 2][0] = U[t2][2][2];
        rec[i3 - 2][1] = 7.58620689655172e-1F*U[t2][3][2] + 2.41379310344828e-1F*U[t2][4][2];
        rec[i3 - 2][2] = 5.17241379310345e-1F*U[t2][4][2] + 4.82758620689655e-1F*U[t2][5][2];
        rec[i3 - 2][3] = 2.75862068965517e-1F*U[t2][5][2] + 7.24137931034483e-1F*U[t2][6][2];
        rec[i3 - 2][4] = 3.44827586206895e-2F*U[t2][6][2] + 9.6551724137931e-1F*U[t2][7][2];
        rec[i3 - 2][5] = 7.93103448275862e-1F*U[t2][8][2] + 2.06896551724138e-1F*U[t2][9][2];
        rec[i3 - 2][6] = 5.51724137931035e-1F*U[t2][9][2] + 4.48275862068965e-1F*U[t2][10][2];
        rec[i3 - 2][7] = 3.10344827586207e-1F*U[t2][10][2] + 6.89655172413793e-1F*U[t2][11][2];
        rec[i3 - 2][8] = 6.8965517241379e-2F*U[t2][11][2] + 9.31034482758621e-1F*U[t2][12][2];
        rec[i3 - 2][9] = 8.27586206896551e-1F*U[t2][13][2] + 1.72413793103449e-1F*U[t2][14][2];
        rec[i3 - 2][10] = 5.86206896551723e-1F*U[t2][14][2] + 4.13793103448277e-1F*U[t2][15][2];
        rec[i3 - 2][11] = 3.44827586206895e-1F*U[t2][15][2] + 6.55172413793105e-1F*U[t2][16][2];
        rec[i3 - 2][12] = 1.0344827586207e-1F*U[t2][16][2] + 8.9655172413793e-1F*U[t2][17][2];
        rec[i3 - 2][13] = 8.62068965517242e-1F*U[t2][18][2] + 1.37931034482758e-1F*U[t2][19][2];
        rec[i3 - 2][14] = 6.20689655172414e-1F*U[t2][19][2] + 3.79310344827586e-1F*U[t2][20][2];
        rec[i3 - 2][15] = 3.79310344827586e-1F*U[t2][20][2] + 6.20689655172414e-1F*U[t2][21][2];
        rec[i3 - 2][16] = 1.37931034482758e-1F*U[t2][21][2] + 8.62068965517242e-1F*U[t2][22][2];
        rec[i3 - 2][17] = 8.9655172413793e-1F*U[t2][23][2] + 1.0344827586207e-1F*U[t2][24][2];
        rec[i3 - 2][18] = 6.55172413793102e-1F*U[t2][24][2] + 3.44827586206898e-1F*U[t2][25][2];
        rec[i3 - 2][19] = 4.13793103448274e-1F*U[t2][25][2] + 5.86206896551726e-1F*U[t2][26][2];
        rec[i3 - 2][20] = 1.72413793103447e-1F*U[t2][26][2] + 8.27586206896553e-1F*U[t2][27][2];
        rec[i3 - 2][21] = 9.31034482758619e-1F*U[t2][28][2] + 6.89655172413813e-2F*U[t2][29][2];
        rec[i3 - 2][22] = 6.89655172413791e-1F*U[t2][29][2] + 3.10344827586209e-1F*U[t2][30][2];
        rec[i3 - 2][23] = 4.48275862068963e-1F*U[t2][30][2] + 5.51724137931037e-1F*U[t2][31][2];
        rec[i3 - 2][24] = 2.06896551724139e-1F*U[t2][31][2] + 7.93103448275861e-1F*U[t2][32][2];
        rec[i3 - 2][25] = 9.65517241379312e-1F*U[t2][33][2] + 3.44827586206884e-2F*U[t2][34][2];
        rec[i3 - 2][26] = 7.24137931034484e-1F*U[t2][34][2] + 2.75862068965516e-1F*U[t2][35][2];
        rec[i3 - 2][27] = 4.82758620689656e-1F*U[t2][35][2] + 5.17241379310344e-1F*U[t2][36][2];
        rec[i3 - 2][28] = 2.41379310344828e-1F*U[t2][36][2] + 7.58620689655172e-1F*U[t2][37][2];
        rec[i3 - 2][29] = U[t2][38][2];
      }
    }
  }
  return 0;
```

Example: 2D TTI wave model

Forward:

$$m\frac{d^2p(x,t)}{dt^2} - (1+2\epsilon)G_{\bar{x}\bar{x}}p(x,t) - \sqrt{(1+2\delta)}G_{\bar{z}\bar{z}}r(x,t) = q$$
$$m\frac{d^2r(x,t)}{dt^2} - \sqrt{(1+2\delta)}G_{\bar{x}\bar{x}}p(x,t) - G_{\bar{z}\bar{z}}r(x,t) = q$$

where

$$\begin{aligned} G_{\bar{x}\bar{x}} &= cos(\theta)^2\frac{d^2}{dx^2} + sin(\theta)^2\frac{d^2}{dz^2} - sin(2\theta)\frac{d^2}{dxdz} \\ G_{\bar{z}\bar{z}} &= sin(\theta)^2\frac{d^2}{dx^2} + cos(\theta)^2\frac{d^2}{dz^2} + sin(2\theta)\frac{d^2}{dxdz} \end{aligned}$$

Imperial College
London

Example: 2D TTI wave model

Adjoint:

$$m\frac{d^2 p(x,t)}{dt^2} - G^T_{\bar{x}\bar{x}}((1+2\epsilon)p(x,t)) - G^T_{\bar{z}\bar{z}}(\sqrt{(1+2\delta)}r(x,t)) = q$$
$$m\frac{d^2 r(x,t)}{dt^2} - G^T_{\bar{x}\bar{x}}(\sqrt{(1+2\delta)}p(x,t)) - G^T_{\bar{z}\bar{z}}r(x,t) = q$$

SymPy implementation

```
dttp=as_finite_diff(p(z,x,t).diff(t,t), [t-s,t, t+s])
dttr=as_finite_diff(r(z,x,t).diff(t,t), [t-s,t, t+s])
dtp=as_finite_diff(p(z,x,t).diff(t), [t-s,t])
dtr=as_finite_diff(r(z,x,t).diff(t), [t-s,t])

dxxp=as_finite_diff(p(z,x,t).diff(x,x), [x-h, x, x+h])
dzzp=as_finite_diff(p(z,x,t).diff(z,z), [z-h, z, z+h])
dxxr=as_finite_diff(r(z,x,t).diff(x,x), [x-h, x, x+h])
dzzr=as_finite_diff(r(z,x,t).diff(z,z), [z-h, z, z+h])

dxzp = .5/(h**2)*(-2*p(z,x,t) + p(z+h,x,t) + p(z-h,x,t) - p(z-h,x+h,t) + p(z,x-h,t) - p(z+h,x-h,t) + p(z,x+h,t))
dxzr = .5/(h**2)*(-2*r(z,x,t) + r(z+h,x,t) + r(z-h,x,t) - r(z-h,x+h,t) + r(z,x-h,t) - r(z+h,x-h,t) + r(z,x+h,t))

Gxxp=cos(Th)**2 * dxxp + sin(Th)**2 * dzzp - sin(2*Th)*dxzp
Gzzr=sin(Th)**2 * dxxr + cos(Th)**2 * dzzr + sin(2*Th)*dxzr

# Forward wave equations for p and r
wavep = m * dttp - A * Gxxp - B * Gzzr - q
waver = m * dttr - B * Gxxp - Gzzr - q

stencilp =solve(wavep,p(z,x,t+s),simplify=False)[0]
tsp=lambdify(arglambp,  stencilp,  "numpy")

stencilr = solve(waver,r(z,x,t+s),simplify=False)[0]
tsr=lambdify(arglambr, stencilr,  "numpy")
```

Imperial College
London

Automatically generated code for 3D TTI forward model

...just another afternoon for Mathias...

Imperial College
London

```
#include <cassert>
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <vector>
#include <cstdio>
#include <string>
#include <inttypes.h>
#include <sys/time.h>
#include <math.h>
#include <omp.h>
struct profiler
{
  double post_stencils0;
  double post_stencils1;
  double post_stencils2;
  double loop_body;
} ;
extern "C" int ForwardOperator(float *m_vec, float *damp_vec, float *u_vec, flo
at *v_vec, float *epsilon_vec, float *delta_vec, float *theta_vec, float *phi_ve
c, float *src_vec, float *src_coords_vec, float *rec_vec, float *rec_coords_vec,
 long i1block, long i2block, struct profiler *timings)
{
  float (*m)[70][70] = (float (*)[70][70]) m_vec;
  float (*damp)[70][70] = (float (*)[70][70]) damp_vec;
  float (*u)[70][70][70] = (float (*)[70][70][70]) u_vec;
  float (*v)[70][70][70] = (float (*)[70][70][70]) v_vec;
  float (*epsilon)[70][70] = (float (*)[70][70]) epsilon_vec;
  float (*delta)[70][70] = (float (*)[70][70]) delta_vec;
  float (*theta)[70][70] = (float (*)[70][70]) theta_vec;
```

```
  float (*phi)[70][70] = (float (*)[70][70]) phi_vec;
  float (*src)[1] = (float (*)[1]) src_vec;
  float (*src_coords)[3] = (float (*)[3]) src_coords_vec;
  float (*rec)[101] = (float (*)[101]) rec_vec;
  float (*rec_coords)[3] = (float (*)[3]) rec_coords_vec;
  {
    int t0;
    int t1;
    int t2;
    #pragma omp parallel
    for (int i4 = 0; i4<3; i4+=1)
    {
      #pragma omp single
      {
        t0 = (i4)%(3);
        t1 = (t0 + 1)%(3);
        t2 = (t1 + 1)%(3);
      }
      struct timeval start_loop_body, end_loop_body;
      #pragma omp master
      gettimeofday(&start_loop_body, NULL);
      {
        #pragma omp for schedule(static)
        for (int i1b = 6; i1b<64 - (58 % i1block); i1b+=i1block)
          for (int i2b = 6; i2b<64 - (58 % i2block); i2b+=i2block)
            for (int i1 = i1b; i1<i1b+i1block; i1++)
              for (int i2 = i2b; i2<i2b+i2block; i2++)
              {
                #pragma omp simd aligned(m, theta, delta, phi, u, damp, v, epsil
on:64)
                for (int i3 = 6; i3<64; i3++)
```

```
                {
                  float temp1 = 8.85879567828298e-1F*damp[i1][i2][i3];
                  float temp3 = 2.0F*m[i1][i2][i3];
                  float temp4 = 1.0F/(temp1 + temp3);
                  float temp5 = 4.0F*m[i1][i2][i3];
                  float temp9 = temp1 - temp3;
                  float temp10 = 1.56956521739130F;
                  float temp12 = 5.00000000000000e-2F;
                  float temp13 = 2.91666666666667e-1F*temp12;
                  float temp16 = sin(theta[i1][i2][i3]);
                  float temp19 = cos(phi[i1][i2][i3]);
                  float temp20 = cos(theta[i1][i2][i3]);
                  float temp21 = (1.0F/60.0F)*temp12;
                  float temp27 = temp12*v[t1][i1][i2][i3 - 2];
                  float temp30 = temp12*v[t1][i1][i2][i3 - 1];
                  float temp31 = (3.0F/4.0F)*temp30;
                  float temp34 = temp12*v[t1][i1][i2][i3 + 1];
                  float temp35 = (3.0F/4.0F)*temp34;
                  float temp38 = temp12*v[t1][i1][i2][i3 + 2];
                  float temp41 = temp21*v[t1][i1][i2][i3 + 3];
                  float temp42 = sin(phi[i1][i2][i3]);
                  float temp43 = temp16*temp42;
                  float temp48 = temp12*v[t1][i1][i2 - 2][i3];
                  float temp51 = temp12*v[t1][i1][i2 - 1][i3];
                  float temp52 = (3.0F/4.0F)*temp51;
                  float temp55 = temp12*v[t1][i1][i2 + 1][i3];
                  float temp56 = (3.0F/4.0F)*temp55;
                  float temp59 = temp12*v[t1][i1][i2 + 2][i3];
                  float temp62 = temp21*v[t1][i1][i2 + 3][i3];
                  float temp63 = temp16*temp19;
                  float temp65 = temp12*v[t1][i1][i2][i3];
```

```
                  float temp66 = (7.0F/12.0F)*temp65;
                  float temp67 = (1.0F/30.0F)*temp12;
                  float temp72 = temp12*v[t1][i1 - 1][i2][i3];
                  float temp75 = temp12*v[t1][i1 + 1][i2][i3];
                  float temp78 = temp12*v[t1][i1 + 2][i2][i3];
                  float temp81 = temp12*v[t1][i1 + 3][i2][i3];
                  float temp84 = temp21*v[t1][i1 + 4][i2][i3];
                  float temp87 = temp12*v[t1][i1 - 2][i2][i3];
                  float temp88 = (3.0F/4.0F)*temp72;
                  float temp89 = (3.0F/4.0F)*temp75;
                  float temp90 = temp21*v[t1][i1 + 3][i2][i3];
                  float temp91 = temp12*v[t1][i1][i2][i3 + 3];
                  float temp94 = temp21*v[t1][i1][i2][i3 + 4];
                  float temp95 = temp12*v[t1][i1][i2 + 3][i3];
                  float temp98 = temp21*v[t1][i1][i2 + 4][i3];
                  float temp99 = 2.91666666666667e-1F*temp12*(temp20*((2.0F/5.0F
)*temp30 - 4.0F/3.0F*temp34 + (1.0F/2.0F)*temp38 + temp66 - temp67*v[t1][i1][i2]
[i3 - 2] - 2.0F/15.0F*temp91 + temp94) + temp43*((2.0F/5.0F)*temp51 - 4.0F/3.0F*
temp55 + (1.0F/2.0F)*temp59 + temp66 - temp67*v[t1][i1][i2 - 2][i3] - 2.0F/15.0F
*temp95 + temp98) + temp63*(temp21*v[t1][i1 - 3][i2][i3] + (3.0F/20.0F)*temp78 -
 3.0F/20.0F*temp87 + temp88 - temp89 - temp90));
                  float temp100 = 3.75e-1F*temp12;
                  float temp102 = cos(theta[i1][i2][i3 - 1]);
                  float temp103 = (3.0F/4.0F)*temp65;
                  float temp104 = -temp103;
                  float temp107 = temp12*v[t1][i1][i2][i3 - 3];
                  float temp108 = (3.0F/4.0F)*temp27;
                  float temp109 = (3.0F/20.0F)*temp34;
                  float temp110 = temp21*v[t1][i1][i2][i3 + 2];
                  float temp112 = sin(phi[i1][i2][i3 - 1]);
                  float temp113 = sin(theta[i1][i2][i3 - 1]);
```

```
                  float temp114 = temp112*temp113;
                  float temp116 = (3.0F/20.0F)*temp12;
                  float temp118 = (3.0F/4.0F)*temp12;
                  float temp120 = temp118*v[t1][i1][i2 - 1][i3 - 1];
                  float temp122 = temp118*v[t1][i1][i2 + 1][i3 - 1];
                  float temp125 = cos(phi[i1][i2][i3 - 1]);
                  float temp126 = temp113*temp125;
                  float temp127 = (7.0F/12.0F)*temp30;
                  float temp129 = (2.0F/5.0F)*temp12;
                  float temp131 = temp129*v[t1][i1 - 1][i2][i3 - 1];
                  float temp132 = (4.0F/3.0F)*temp12;
                  float temp134 = (1.0F/2.0F)*temp12;
                  float temp136 = (2.0F/15.0F)*temp12;
                  float temp138 = 7.5e-2F*temp12;
                  float temp140 = cos(theta[i1][i2][i3 + 2]);
                  float temp141 = (3.0F/20.0F)*temp65;
                  float temp142 = -temp141;
                  float temp143 = temp12*v[t1][i1][i2][i3 + 4];
                  float temp146 = temp21*v[t1][i1][i2][i3 + 5];
                  float temp148 = sin(phi[i1][i2][i3 + 2]);
                  float temp149 = sin(theta[i1][i2][i3 + 2]);
                  float temp150 = temp148*temp149;
                  float temp153 = temp116*v[t1][i1][i2 - 2][i3 + 2];
                  float temp157 = temp116*v[t1][i1][i2 + 2][i3 + 2];
                  float temp159 = cos(phi[i1][i2][i3 + 2]);
                  float temp160 = temp149*temp159;
                  float temp161 = (7.0F/12.0F)*temp38;
                  float temp166 = temp134*v[t1][i1 + 2][i2][i3 + 2];
                  float temp168 = 8.33333333333333e-3F*temp12;
                  float temp170 = cos(theta[i1][i2][i3 - 3]);
                  float temp171 = (1.0F/60.0F)*temp65;
```

```
                  float temp172 = -temp171;
                  float temp177 = temp12*v[t1][i1][i2][i3 - 5];
                  float temp178 = temp12*v[t1][i1][i2][i3 - 4];
                  float temp179 = (3.0F/20.0F)*temp30;
                  float temp181 = sin(phi[i1][i2][i3 - 3]);
                  float temp182 = sin(theta[i1][i2][i3 - 3]);
                  float temp183 = temp181*temp182;
                  float temp184 = temp21*v[t1][i1][i2 - 3][i3 - 3];
                  float temp190 = temp21*v[t1][i1][i2 + 3][i3 - 3];
                  float temp191 = cos(phi[i1][i2][i3 - 3]);
                  float temp192 = temp182*temp191;
                  float temp193 = (7.0F/12.0F)*temp107;
                  float temp200 = cos(theta[i1][i2][i3 + 3]);
                  float temp201 = (3.0F/4.0F)*temp38;
                  float temp202 = temp12*v[t1][i1][i2][i3 + 5];
                  float temp204 = temp21*v[t1][i1][i2][i3 + 6];
                  float temp206 = sin(phi[i1][i2][i3 + 3]);
                  float temp207 = sin(theta[i1][i2][i3 + 3]);
                  float temp209 = temp21*v[t1][i1][i2 - 3][i3 + 3];
                  float temp214 = -temp21*v[t1][i1][i2 + 3][i3 + 3];
                  float temp215 = cos(phi[i1][i2][i3 + 3]);
                  float temp220 = -temp136*v[t1][i1 + 3][i2][i3 + 3];
                  float temp222 = cos(theta[i1][i2][i3 - 2]);
                  float temp223 = temp21*v[t1][i1][i2][i3 + 1];
                  float temp225 = sin(phi[i1][i2][i3 - 2]);
                  float temp226 = sin(theta[i1][i2][i3 - 2]);
                  float temp227 = temp225*temp226;
                  float temp230 = -temp116*v[t1][i1][i2 - 2][i3 - 2];
                  float temp234 = temp116*v[t1][i1][i2 + 2][i3 - 2];
                  float temp236 = cos(phi[i1][i2][i3 - 2]);
                  float temp237 = temp226*temp236;
```

```
                  float temp238 = (7.0F/12.0F)*temp27;
                  float temp240 = -temp67*v[t1][i1 - 2][i2][i3 - 2];
                  float temp246 = cos(theta[i1][i2][i3 + 1]);
                  float temp248 = sin(phi[i1][i2][i3 + 1]);
                  float temp249 = sin(theta[i1][i2][i3 + 1]);
                  float temp250 = temp248*temp249;
                  float temp254 = temp118*v[t1][i1][i2 - 1][i3 + 1];
                  float temp256 = -temp118*v[t1][i1][i2 + 1][i3 + 1];
                  float temp259 = cos(phi[i1][i2][i3 + 1]);
                  float temp260 = temp249*temp259;
                  float temp261 = (7.0F/12.0F)*temp34;
                  float temp265 = -temp132*v[t1][i1 + 1][i2][i3 + 1];
                  float temp270 = sin(theta[i1 - 4][i2][i3]);
                  float temp272 = cos(phi[i1 - 4][i2][i3]);
                  float temp273 = cos(theta[i1 - 4][i2][i3]);
                  float temp274 = sin(phi[i1 - 4][i2][i3]);
                  float temp279 = temp12*v[t1][i1 - 5][i2][i3];
                  float temp281 = temp12*v[t1][i1 - 4][i2][i3];
                  float temp282 = temp12*v[t1][i1 - 3][i2][i3];
                  float temp283 = 6.66666666666667e-2F*temp12;
                  float temp285 = sin(theta[i1 - 3][i2][i3]);
                  float temp287 = cos(phi[i1 - 3][i2][i3]);
                  float temp288 = cos(theta[i1 - 3][i2][i3]);
                  float temp289 = temp21*v[t1][i1 - 3][i2][i3 - 3];
                  float temp295 = sin(phi[i1 - 3][i2][i3]);
                  float temp296 = temp285*temp295;
                  float temp297 = temp21*v[t1][i1 - 3][i2 - 3][i3];
                  float temp303 = temp285*temp287;
                  float temp304 = -2.0F/15.0F*temp65;
                  float temp305 = (7.0F/12.0F)*temp282;
                  float temp306 = temp21*v[t1][i1 + 1][i2][i3];
```

```
                  float temp307 = 2.5e-1F*temp12;
                  float temp309 = sin(theta[i1 - 2][i2][i3]);
                  float temp311 = cos(phi[i1 - 2][i2][i3]);
                  float temp312 = cos(theta[i1 - 2][i2][i3]);
                  float temp313 = -temp116*v[t1][i1 - 2][i2][i3 - 2];
                  float temp314 = temp116*v[t1][i1 - 2][i2][i3 + 2];
                  float temp315 = sin(phi[i1 - 2][i2][i3]);
                  float temp316 = temp309*temp315;
                  float temp319 = -temp116*v[t1][i1 - 2][i2 - 2][i3];
                  float temp323 = temp116*v[t1][i1 - 2][i2 + 2][i3];
                  float temp325 = temp309*temp311;
                  float temp326 = (1.0F/2.0F)*temp65;
                  float temp327 = (7.0F/12.0F)*temp87;
                  float temp328 = temp21*v[t1][i1 + 2][i2][i3];
                  float temp329 = 6.66666666666667e-1F*temp12;
                  float temp331 = sin(theta[i1 - 1][i2][i3]);
                  float temp333 = cos(phi[i1 - 1][i2][i3]);
                  float temp334 = cos(theta[i1 - 1][i2][i3]);
                  float temp335 = temp118*v[t1][i1 - 1][i2][i3 - 1];
                  float temp336 = temp118*v[t1][i1 - 1][i2][i3 + 1];
                  float temp337 = sin(phi[i1 - 1][i2][i3]);
                  float temp338 = temp331*temp337;
                  float temp342 = temp118*v[t1][i1 - 1][i2 - 1][i3];
                  float temp344 = temp118*v[t1][i1 - 1][i2 + 1][i3];
                  float temp347 = temp331*temp333;
                  float temp348 = -4.0F/3.0F*temp65;
                  float temp349 = (7.0F/12.0F)*temp72;
                  float temp350 = 2.0e-1F*temp12;
                  float temp352 = sin(theta[i1 + 1][i2][i3]);
                  float temp354 = cos(phi[i1 + 1][i2][i3]);
                  float temp355 = cos(theta[i1 + 1][i2][i3]);
```

```
                  float temp356 = temp118*v[t1][i1 + 1][i2][i3 - 1];
                  float temp357 = -temp118*v[t1][i1 + 1][i2][i3 + 1];
                  float temp358 = sin(phi[i1 + 1][i2][i3]);
                  float temp359 = temp352*temp358;
                  float temp363 = temp118*v[t1][i1 + 1][i2 - 1][i3];
                  float temp365 = -temp118*v[t1][i1 + 1][i2 + 1][i3];
                  float temp368 = temp352*temp354;
                  float temp369 = (2.0F/5.0F)*temp65;
                  float temp370 = (7.0F/12.0F)*temp75;
                  float temp371 = temp12*v[t1][i1 + 4][i2][i3];
                  float temp374 = temp21*v[t1][i1 + 5][i2][i3];
                  float temp375 = 1.66666666666667e-2F*temp12;
                  float temp377 = sin(theta[i1 + 2][i2][i3]);
                  float temp379 = cos(phi[i1 + 2][i2][i3]);
                  float temp380 = cos(theta[i1 + 2][i2][i3]);
                  float temp381 = temp116*v[t1][i1 + 2][i2][i3 - 2];
                  float temp382 = temp116*v[t1][i1 + 2][i2][i3 + 2];
                  float temp383 = sin(phi[i1 + 2][i2][i3]);
                  float temp384 = temp377*temp383;
                  float temp387 = temp116*v[t1][i1 + 2][i2 - 2][i3];
                  float temp391 = temp116*v[t1][i1 + 2][i2 + 2][i3];
                  float temp393 = temp377*temp379;
                  float temp394 = -1.0F/30.0F*temp65;
                  float temp395 = (7.0F/12.0F)*temp78;
                  float temp396 = temp12*v[t1][i1 + 5][i2][i3];
                  float temp398 = temp21*v[t1][i1 + 6][i2][i3];
                  float temp400 = sin(phi[i1][i2 - 1][i3]);
                  float temp402 = sin(theta[i1][i2 - 1][i3]);
                  float temp403 = temp400*temp402;
                  float temp406 = temp12*v[t1][i1][i2 - 3][i3];
                  float temp407 = (3.0F/4.0F)*temp48;
                  float temp408 = (3.0F/20.0F)*temp55;
                  float temp409 = temp21*v[t1][i1][i2 + 2][i3];
                  float temp410 = cos(theta[i1][i2 - 1][i3]);
                  float temp411 = cos(phi[i1][i2 - 1][i3]);
                  float temp412 = temp402*temp411;
                  float temp413 = (7.0F/12.0F)*temp51;
                  float temp414 = temp129*v[t1][i1 - 1][i2 - 1][i3];
                  float temp417 = sin(phi[i1][i2 + 2][i3]);
                  float temp419 = sin(theta[i1][i2 + 2][i3]);
                  float temp420 = temp417*temp419;
                  float temp421 = temp12*v[t1][i1][i2 + 4][i3];
                  float temp424 = temp21*v[t1][i1][i2 + 5][i3];
                  float temp425 = cos(theta[i1][i2 + 2][i3]);
                  float temp426 = cos(phi[i1][i2 + 2][i3]);
                  float temp427 = temp419*temp426;
                  float temp428 = (7.0F/12.0F)*temp59;
                  float temp429 = temp134*v[t1][i1 + 2][i2 + 2][i3];
                  float temp432 = sin(phi[i1][i2 - 3][i3]);
                  float temp434 = sin(theta[i1][i2 - 3][i3]);
                  float temp435 = temp432*temp434;
                  float temp440 = temp12*v[t1][i1][i2 - 5][i3];
                  float temp441 = temp12*v[t1][i1][i2 - 4][i3];
                  float temp442 = (3.0F/20.0F)*temp51;
                  float temp443 = cos(theta[i1][i2 - 3][i3]);
                  float temp444 = cos(phi[i1][i2 - 3][i3]);
                  float temp445 = temp434*temp444;
                  float temp446 = (7.0F/12.0F)*temp406;
                  float temp449 = sin(phi[i1][i2 + 3][i3]);
                  float temp451 = sin(theta[i1][i2 + 3][i3]);
                  float temp452 = (3.0F/4.0F)*temp59;
                  float temp453 = temp12*v[t1][i1][i2 + 5][i3];
```

```
                  float temp455 = temp21*v[t1][i1][i2 + 6][i3];
                  float temp456 = cos(theta[i1][i2 + 3][i3]);
                  float temp457 = cos(phi[i1][i2 + 3][i3]);
                  float temp458 = -temp136*v[t1][i1 + 3][i2 + 3][i3];
                  float temp460 = sin(phi[i1][i2 - 2][i3]);
                  float temp462 = sin(theta[i1][i2 - 2][i3]);
                  float temp463 = temp460*temp462;
                  float temp464 = temp21*v[t1][i1][i2 + 1][i3];
                  float temp465 = cos(theta[i1][i2 - 2][i3]);
                  float temp466 = cos(phi[i1][i2 - 2][i3]);
                  float temp467 = temp462*temp466;
                  float temp468 = (7.0F/12.0F)*temp48;
                  float temp469 = -temp67*v[t1][i1 - 2][i2 - 2][i3];
                  float temp472 = sin(phi[i1][i2 + 1][i3]);
                  float temp474 = sin(theta[i1][i2 + 1][i3]);
                  float temp475 = temp472*temp474;
                  float temp476 = cos(theta[i1][i2 + 1][i3]);
                  float temp477 = cos(phi[i1][i2 + 1][i3]);
                  float temp478 = temp474*temp477;
                  float temp479 = (7.0F/12.0F)*temp55;
                  float temp480 = -temp132*v[t1][i1 + 1][i2 + 1][i3];
                  float temp483 = cos(theta[i1][i2][i3 - 4]);
                  float temp484 = (7.0F/12.0F)*temp178;
                  float temp485 = sin(theta[i1][i2][i3 - 4]);
                  float temp487 = cos(phi[i1][i2][i3 - 4]);
                  float temp488 = sin(phi[i1][i2][i3 - 4]);
                  float temp489 = -temp67*v[t1][i1][i2 - 2][i3 - 2];
                  float temp490 = temp129*v[t1][i1][i2 - 1][i3 - 1];
                  float temp491 = -temp132*v[t1][i1][i2 + 1][i3 + 1];
                  float temp492 = temp134*v[t1][i1][i2 + 2][i3 + 2];
                  float temp493 = (3.0F/4.0F)*temp87;
```

```
                  float temp494 = (3.0F/20.0F)*temp75;
                  float temp495 = (3.0F/20.0F)*temp72;
                  float temp497 = sin(theta[i1 + 3][i2][i3]);
                  float temp499 = cos(phi[i1 + 3][i2][i3]);
                  float temp500 = (3.0F/4.0F)*temp78;
                  float temp501 = cos(theta[i1 + 3][i2][i3]);
                  float temp502 = (7.0F/12.0F)*temp81;
                  float temp503 = sin(phi[i1 + 3][i2][i3]);
                  float temp505 = sin(phi[i1][i2 - 4][i3]);
                  float temp507 = sin(theta[i1][i2 - 4][i3]);
                  float temp508 = cos(phi[i1][i2 - 4][i3]);
                  float temp509 = (7.0F/12.0F)*temp441;
                  float temp510 = cos(theta[i1][i2 - 4][i3]);
                  float temp511 = temp100*temp102*(temp102*(temp104 - 3.0F/20.0F
*temp107 + temp108 + temp109 - temp110 + temp21*v[t1][i1][i2][i3 - 4]) + temp114
*(-temp116*v[t1][i1][i2 - 2][i3 - 1] + temp116*v[t1][i1][i2 + 2][i3 - 1] + temp1
20 - temp122 + temp21*v[t1][i1][i2 - 3][i3 - 1] - temp21*v[t1][i1][i2 + 3][i3 -
1]) + temp126*(temp127 + temp131 - temp132*v[t1][i1 + 1][i2][i3 - 1] + temp134*v
[t1][i1 + 2][i2][i3 - 1] - temp136*v[t1][i1 + 3][i2][i3 - 1] + temp21*v[t1][i1 +
 4][i2][i3 - 1] - temp67*v[t1][i1 - 2][i2][i3 - 1])) - temp100*temp246*(temp246*
(temp103 - temp179 - temp201 + temp21*v[t1][i1][i2][i3 - 2] + (3.0F/20.0F)*temp9
1 - temp94) + temp250*(-temp116*v[t1][i1][i2 - 2][i3 + 1] + temp116*v[t1][i1][i2
 + 2][i3 + 1] + temp21*v[t1][i1][i2 - 3][i3 + 1] - temp21*v[t1][i1][i2 + 3][i3 +
 1] + temp254 + temp256) + temp260*(temp129*v[t1][i1 - 1][i2][i3 + 1] + temp134*
v[t1][i1 + 2][i2][i3 + 1] - temp136*v[t1][i1 + 3][i2][i3 + 1] + temp21*v[t1][i1
+ 4][i2][i3 + 1] + temp261 + temp265 - temp67*v[t1][i1 - 2][i2][i3 + 1])) + temp
100*temp331*temp333*(temp334*(temp131 - temp132*v[t1][i1 - 1][i2][i3 + 1] + temp
134*v[t1][i1 - 1][i2][i3 + 2] - temp136*v[t1][i1 - 1][i2][i3 + 3] + temp21*v[t1]
[i1 - 1][i2][i3 + 4] + temp349 - temp67*v[t1][i1 - 1][i2][i3 - 2]) + temp338*(-t
emp132*v[t1][i1 - 1][i2 + 1][i3] + temp134*v[t1][i1 - 1][i2 + 2][i3] - temp136*v
[t1][i1 - 1][i2 + 3][i3] + temp21*v[t1][i1 - 1][i2 + 4][i3] + temp349 + temp414
```

```
- temp67*v[t1][i1 - 1][i2 - 2][i3]) + temp347*(temp104 + temp21*v[t1][i1 - 4][i2
][i3] - 3.0F/20.0F*temp282 - temp328 + temp493 + temp494)) - temp100*temp352*tem
p354*(temp355*(temp129*v[t1][i1 + 1][i2][i3 - 1] + temp134*v[t1][i1 + 1][i2][i3
+ 2] - temp136*v[t1][i1 + 1][i2][i3 + 3] + temp21*v[t1][i1 + 1][i2][i3 + 4] + te
mp265 + temp370 - temp67*v[t1][i1 + 1][i2][i3 - 2]) + temp359*(temp129*v[t1][i1
+ 1][i2 - 1][i3] + temp134*v[t1][i1 + 1][i2 + 2][i3] - temp136*v[t1][i1 + 1][i2
+ 3][i3] + temp21*v[t1][i1 + 1][i2 + 4][i3] + temp370 + temp480 - temp67*v[t1][i
1 + 1][i2 - 2][i3]) + temp368*(temp103 + temp21*v[t1][i1 - 2][i2][i3] - temp495
- temp500 + (3.0F/20.0F)*temp81 - temp84)) + temp100*temp400*temp402*(temp403*(t
emp104 + temp21*v[t1][i1][i2 - 4][i3] - 3.0F/20.0F*temp406 + temp407 + temp408 -
 temp409) + temp410*(-temp116*v[t1][i1][i2 - 1][i3 - 2] + temp116*v[t1][i1][i2 -
 1][i3 + 2] + temp120 + temp21*v[t1][i1][i2 - 1][i3 - 3] - temp21*v[t1][i1][i2 -
 1][i3 + 3] - temp254) + temp412*(-temp132*v[t1][i1 + 1][i2 - 1][i3] + temp134*v
[t1][i1 + 2][i2 - 1][i3] - temp136*v[t1][i1 + 3][i2 - 1][i3] + temp21*v[t1][i1 +
 4][i2 - 1][i3] + temp413 + temp414 - temp67*v[t1][i1 - 2][i2 - 1][i3])) - temp1
00*temp472*temp474*(temp475*(temp103 + temp21*v[t1][i1][i2 - 2][i3] - temp442 -
temp452 + (3.0F/20.0F)*temp95 - temp98) + temp476*(-temp116*v[t1][i1][i2 + 1][i3
 - 2] + temp116*v[t1][i1][i2 + 1][i3 + 2] + temp122 + temp21*v[t1][i1][i2 + 1][i
3 - 3] - temp21*v[t1][i1][i2 + 1][i3 + 3] + temp256) + temp478*(temp129*v[t1][i1
 - 1][i2 + 1][i3] + temp134*v[t1][i1 + 2][i2 + 1][i3] - temp136*v[t1][i1 + 3][i2
 + 1][i3] + temp21*v[t1][i1 + 4][i2 + 1][i3] + temp479 + temp480 - temp67*v[t1][
i1 - 2][i2 + 1][i3])) + temp102*temp329*(temp102*(temp127 + (2.0F/5.0F)*temp27 +
 (1.0F/2.0F)*temp34 + temp348 - 2.0F/15.0F*temp38 + temp41 - temp67*v[t1][i1][i2
][i3 - 3]) + temp114*(temp127 - temp132*v[t1][i1][i2 + 1][i3 - 1] + temp134*v[t1
][i1][i2 + 2][i3 - 1] - temp136*v[t1][i1][i2 + 3][i3 - 1] + temp21*v[t1][i1][i2
+ 4][i3 - 1] + temp490 - temp67*v[t1][i1][i2 - 2][i3 - 1]) + temp126*(-temp116*v
[t1][i1 - 2][i2][i3 - 1] + temp116*v[t1][i1 + 2][i2][i3 - 1] + temp21*v[t1][i1 -
 3][i2][i3 - 1] - temp21*v[t1][i1 + 3][i2][i3 - 1] + temp335 - temp356)) - temp1
3*temp16*temp19*(temp20*(temp21*v[t1][i1][i2][i3 - 3] - 3.0F/20.0F*temp27 + temp
31 - temp35 + (3.0F/20.0F)*temp38 - temp41) + temp43*(temp21*v[t1][i1][i2 - 3][i
3] - 3.0F/20.0F*temp48 + temp52 - temp56 + (3.0F/20.0F)*temp59 - temp62) + temp6
```

```
3*(temp66 - temp67*v[t1][i1 - 2][i2][i3] + (2.0F/5.0F)*temp72 - 4.0F/3.0F*temp75
 + (1.0F/2.0F)*temp78 - 2.0F/15.0F*temp81 + temp84)) + temp138*temp140*(temp140*
(temp142 + (3.0F/20.0F)*temp143 - temp146 + temp21*v[t1][i1][i2][i3 - 1] + temp3
5 - 3.0F/4.0F*temp91) + temp150*(temp118*v[t1][i1][i2 - 1][i3 + 2] - temp118*v[t
1][i1][i2 + 1][i3 + 2] - temp153 + temp157 + temp21*v[t1][i1][i2 - 3][i3 + 2] -
temp21*v[t1][i1][i2 + 3][i3 + 2]) + temp160*(temp129*v[t1][i1 - 1][i2][i3 + 2] -
 temp132*v[t1][i1 + 1][i2][i3 + 2] - temp136*v[t1][i1 + 3][i2][i3 + 2] + temp161
 + temp166 + temp21*v[t1][i1 + 4][i2][i3 + 2] - temp67*v[t1][i1 - 2][i2][i3 + 2]
)) - temp138*temp222*(temp222*((3.0F/4.0F)*temp107 + temp141 - 3.0F/20.0F*temp17
8 + temp21*v[t1][i1][i2][i3 - 5] - temp223 - temp31) + temp227*(temp118*v[t1][i1
][i2 - 1][i3 - 2] - temp118*v[t1][i1][i2 + 1][i3 - 2] + temp21*v[t1][i1][i2 - 3]
[i3 - 2] - temp21*v[t1][i1][i2 + 3][i3 - 2] + temp230 + temp234) + temp237*(temp
129*v[t1][i1 - 1][i2][i3 - 2] - temp132*v[t1][i1 + 1][i2][i3 - 2] + temp134*v[t1
][i1 + 2][i2][i3 - 2] - temp136*v[t1][i1 + 3][i2][i3 - 2] + temp21*v[t1][i1 + 4]
[i2][i3 - 2] + temp238 + temp240)) - temp138*temp309*temp311*(temp312*(temp129*v
[t1][i1 - 2][i2][i3 - 1] - temp132*v[t1][i1 - 2][i2][i3 + 1] + temp134*v[t1][i1
- 2][i2][i3 + 2] - temp136*v[t1][i1 - 2][i2][i3 + 3] + temp21*v[t1][i1 - 2][i2][
i3 + 4] + temp240 + temp327) + temp316*(temp129*v[t1][i1 - 2][i2 - 1][i3] - temp
132*v[t1][i1 - 2][i2 + 1][i3] + temp134*v[t1][i1 - 2][i2 + 2][i3] - temp136*v[t1
][i1 - 2][i2 + 3][i3] + temp21*v[t1][i1 - 2][i2 + 4][i3] + temp327 + temp469) +
temp325*(temp141 + temp21*v[t1][i1 - 5][i2][i3] - 3.0F/20.0F*temp281 + (3.0F/4.0
F)*temp282 - temp306 - temp88)) + temp138*temp377*temp379*(temp380*(temp129*v[t1
][i1 + 2][i2][i3 - 1] - temp132*v[t1][i1 + 2][i2][i3 + 1] - temp136*v[t1][i1 + 2
][i2][i3 + 3] + temp166 + temp21*v[t1][i1 + 2][i2][i3 + 4] + temp395 - temp67*v[
t1][i1 + 2][i2][i3 - 2]) + temp384*(temp129*v[t1][i1 + 2][i2 - 1][i3] - temp132*
v[t1][i1 + 2][i2 + 1][i3] - temp136*v[t1][i1 + 2][i2 + 3][i3] + temp21*v[t1][i1
+ 2][i2 + 4][i3] + temp395 + temp429 - temp67*v[t1][i1 + 2][i2 - 2][i3]) + temp3
93*(temp142 + temp21*v[t1][i1 - 1][i2][i3] + (3.0F/20.0F)*temp371 - temp374 - 3.
0F/4.0F*temp81 + temp89)) + temp138*temp417*temp419*(temp420*(temp142 + temp21*v
[t1][i1][i2 - 1][i3] + (3.0F/20.0F)*temp421 - temp424 + temp56 - 3.0F/4.0F*temp9
5) + temp425*(temp118*v[t1][i1][i2 + 2][i3 - 1] - temp118*v[t1][i1][i2 + 2][i3 +
```

```
 1] + temp157 + temp21*v[t1][i1][i2 + 2][i3 - 3] - temp21*v[t1][i1][i2 + 2][i3 +
 3] - temp234) + temp427*(temp129*v[t1][i1 - 1][i2 + 2][i3] - temp132*v[t1][i1 +
 1][i2 + 2][i3] - temp136*v[t1][i1 + 3][i2 + 2][i3] + temp21*v[t1][i1 + 4][i2 +
2][i3] + temp428 + temp429 - temp67*v[t1][i1 - 2][i2 + 2][i3])) - temp138*temp46
0*temp462*(temp463*(temp141 + temp21*v[t1][i1][i2 - 5][i3] + (3.0F/4.0F)*temp406
 - 3.0F/20.0F*temp441 - temp464 - temp52) + temp465*(temp118*v[t1][i1][i2 - 2][i
3 - 1] - temp118*v[t1][i1][i2 - 2][i3 + 1] + temp153 + temp21*v[t1][i1][i2 - 2][
i3 - 3] - temp21*v[t1][i1][i2 - 2][i3 + 3] + temp230) + temp467*(temp129*v[t1][i
1 - 1][i2 - 2][i3] - temp132*v[t1][i1 + 1][i2 - 2][i3] + temp134*v[t1][i1 + 2][i
2 - 2][i3] - temp136*v[t1][i1 + 3][i2 - 2][i3] + temp21*v[t1][i1 + 4][i2 - 2][i3
] + temp468 + temp469)) + temp140*temp375*(temp140*((1.0F/2.0F)*temp143 + temp16
1 - 2.0F/15.0F*temp202 + temp204 + (2.0F/5.0F)*temp34 + temp394 - 4.0F/3.0F*temp
91) + temp150*(temp129*v[t1][i1][i2 - 1][i3 + 2] - temp132*v[t1][i1][i2 + 1][i3
+ 2] - temp136*v[t1][i1][i2 + 3][i3 + 2] + temp161 + temp21*v[t1][i1][i2 + 4][i3
 + 2] + temp492 - temp67*v[t1][i1][i2 - 2][i3 + 2]) + temp160*(temp118*v[t1][i1
- 1][i2][i3 + 2] - temp118*v[t1][i1 + 1][i2][i3 + 2] + temp21*v[t1][i1 - 3][i2][
i3 + 2] - temp21*v[t1][i1 + 3][i2][i3 + 2] - temp314 + temp382)) + temp168*temp1
70*(temp170*(-temp108 + temp172 - 3.0F/20.0F*temp177 + (3.0F/4.0F)*temp178 + tem
p179 + temp21*v[t1][i1][i2][i3 - 6]) + temp183*(-temp116*v[t1][i1][i2 - 2][i3 -
3] + temp116*v[t1][i1][i2 + 2][i3 - 3] + temp118*v[t1][i1][i2 - 1][i3 - 3] - tem
p118*v[t1][i1][i2 + 1][i3 - 3] + temp184 - temp190) + temp192*(temp129*v[t1][i1
- 1][i2][i3 - 3] - temp132*v[t1][i1 + 1][i2][i3 - 3] + temp134*v[t1][i1 + 2][i2]
[i3 - 3] - temp136*v[t1][i1 + 3][i2][i3 - 3] + temp193 + temp21*v[t1][i1 + 4][i2
][i3 - 3] - temp67*v[t1][i1 - 2][i2][i3 - 3])) - temp168*temp200*(temp200*(-temp
109 - 3.0F/4.0F*temp143 + temp171 + temp201 + (3.0F/20.0F)*temp202 - temp204) +
temp206*temp207*(-temp116*v[t1][i1][i2 - 2][i3 + 3] + temp116*v[t1][i1][i2 + 2][
i3 + 3] + temp118*v[t1][i1][i2 - 1][i3 + 3] - temp118*v[t1][i1][i2 + 1][i3 + 3]
+ temp209 + temp214) + temp207*temp215*(temp129*v[t1][i1 - 1][i2][i3 + 3] - temp
132*v[t1][i1 + 1][i2][i3 + 3] + temp134*v[t1][i1 + 2][i2][i3 + 3] + temp21*v[t1]
[i1 + 4][i2][i3 + 3] + temp220 - temp67*v[t1][i1 - 2][i2][i3 + 3] + (7.0F/12.0F)
*temp91)) - temp168*temp270*temp272*(temp270*temp272*(temp171 + (2.0F/5.0F)*temp
```

```
279 + (7.0F/12.0F)*temp281 - 4.0F/3.0F*temp282 - temp67*v[t1][i1 - 6][i2][i3] -
2.0F/15.0F*temp72 + (1.0F/2.0F)*temp87) + temp270*temp274*(-temp116*v[t1][i1 - 4
][i2 - 2][i3] + temp116*v[t1][i1 - 4][i2 + 2][i3] + temp118*v[t1][i1 - 4][i2 - 1
][i3] - temp118*v[t1][i1 - 4][i2 + 1][i3] + temp21*v[t1][i1 - 4][i2 - 3][i3] - t
emp21*v[t1][i1 - 4][i2 + 3][i3]) + temp273*(-temp116*v[t1][i1 - 4][i2][i3 - 2] +
 temp116*v[t1][i1 - 4][i2][i3 + 2] + temp118*v[t1][i1 - 4][i2][i3 - 1] - temp118
*v[t1][i1 - 4][i2][i3 + 1] + temp21*v[t1][i1 - 4][i2][i3 - 3] - temp21*v[t1][i1
- 4][i2][i3 + 3])) + temp168*temp285*temp287*(temp288*(temp129*v[t1][i1 - 3][i2]
[i3 - 1] - temp132*v[t1][i1 - 3][i2][i3 + 1] + temp134*v[t1][i1 - 3][i2][i3 + 2]
 - temp136*v[t1][i1 - 3][i2][i3 + 3] + temp21*v[t1][i1 - 3][i2][i3 + 4] + temp30
5 - temp67*v[t1][i1 - 3][i2][i3 - 2]) + temp296*(temp129*v[t1][i1 - 3][i2 - 1][i
3] - temp132*v[t1][i1 - 3][i2 + 1][i3] + temp134*v[t1][i1 - 3][i2 + 2][i3] - tem
p136*v[t1][i1 - 3][i2 + 3][i3] + temp21*v[t1][i1 - 3][i2 + 4][i3] + temp305 - te
mp67*v[t1][i1 - 3][i2 - 2][i3]) + temp303*(temp172 + temp21*v[t1][i1 - 6][i2][i3
] - 3.0F/20.0F*temp279 + (3.0F/4.0F)*temp281 - temp493 + temp495)) + temp168*tem
p432*temp434*(temp435*(temp172 + temp21*v[t1][i1][i2 - 6][i3] - temp407 - 3.0F/2
0.0F*temp440 + (3.0F/4.0F)*temp441 + temp442) + temp443*(-temp116*v[t1][i1][i2 -
 3][i3 - 2] + temp116*v[t1][i1][i2 - 3][i3 + 2] + temp118*v[t1][i1][i2 - 3][i3 -
 1] - temp118*v[t1][i1][i2 - 3][i3 + 1] + temp184 - temp209) + temp445*(temp129*
v[t1][i1 - 1][i2 - 3][i3] - temp132*v[t1][i1 + 1][i2 - 3][i3] + temp134*v[t1][i1
 + 2][i2 - 3][i3] - temp136*v[t1][i1 + 3][i2 - 3][i3] + temp21*v[t1][i1 + 4][i2
- 3][i3] + temp446 - temp67*v[t1][i1 - 2][i2 - 3][i3])) - temp168*temp449*temp45
1*(temp449*temp451*(temp171 - temp408 - 3.0F/4.0F*temp421 + temp452 + (3.0F/20.0
F)*temp453 - temp455) + temp451*temp457*(temp129*v[t1][i1 - 1][i2 + 3][i3] - tem
p132*v[t1][i1 + 1][i2 + 3][i3] + temp134*v[t1][i1 + 2][i2 + 3][i3] + temp21*v[t1
][i1 + 4][i2 + 3][i3] + temp458 - temp67*v[t1][i1 - 2][i2 + 3][i3] + (7.0F/12.0F
)*temp95) + temp456*(-temp116*v[t1][i1][i2 + 3][i3 - 2] + temp116*v[t1][i1][i2 +
 3][i3 + 2] + temp118*v[t1][i1][i2 + 3][i3 - 1] - temp118*v[t1][i1][i2 + 3][i3 +
 1] + temp190 + temp214)) - temp168*temp483*(temp483*(-4.0F/3.0F*temp107 + temp1
71 + (2.0F/5.0F)*temp177 + (1.0F/2.0F)*temp27 - 2.0F/15.0F*temp30 + temp484 - te
mp67*v[t1][i1][i2][i3 - 6]) + temp485*temp487*(-temp116*v[t1][i1 - 2][i2][i3 - 4
```

```
] + temp116*v[t1][i1 + 2][i2][i3 - 4] + temp118*v[t1][i1 - 1][i2][i3 - 4] - temp
118*v[t1][i1 + 1][i2][i3 - 4] + temp21*v[t1][i1 - 3][i2][i3 - 4] - temp21*v[t1][
i1 + 3][i2][i3 - 4]) + temp485*temp488*(temp129*v[t1][i1][i2 - 1][i3 - 4] - temp
132*v[t1][i1][i2 + 1][i3 - 4] + temp134*v[t1][i1][i2 + 2][i3 - 4] - temp136*v[t1
][i1][i2 + 3][i3 - 4] + temp21*v[t1][i1][i2 + 4][i3 - 4] + temp484 - temp67*v[t1
][i1][i2 - 2][i3 - 4])) - temp168*temp497*temp499*(temp497*temp499*(temp171 - 3.
0F/4.0F*temp371 + (3.0F/20.0F)*temp396 - temp398 - temp494 + temp500) + temp497*
temp503*(temp129*v[t1][i1 + 3][i2 - 1][i3] - temp132*v[t1][i1 + 3][i2 + 1][i3] +
 temp134*v[t1][i1 + 3][i2 + 2][i3] + temp21*v[t1][i1 + 3][i2 + 4][i3] + temp458
+ temp502 - temp67*v[t1][i1 + 3][i2 - 2][i3]) + temp501*(temp129*v[t1][i1 + 3][i
2][i3 - 1] - temp132*v[t1][i1 + 3][i2][i3 + 1] + temp134*v[t1][i1 + 3][i2][i3 +
2] + temp21*v[t1][i1 + 3][i2][i3 + 4] + temp220 + temp502 - temp67*v[t1][i1 + 3]
[i2][i3 - 2])) - temp168*temp505*temp507*(temp505*temp507*(temp171 - 4.0F/3.0F*t
emp406 + (2.0F/5.0F)*temp440 + (1.0F/2.0F)*temp48 + temp509 - 2.0F/15.0F*temp51
- temp67*v[t1][i1][i2 - 6][i3]) + temp507*temp508*(-temp116*v[t1][i1 - 2][i2 - 4
][i3] + temp116*v[t1][i1 + 2][i2 - 4][i3] + temp118*v[t1][i1 - 1][i2 - 4][i3] -
temp118*v[t1][i1 + 1][i2 - 4][i3] + temp21*v[t1][i1 - 3][i2 - 4][i3] - temp21*v[
t1][i1 + 3][i2 - 4][i3]) + temp510*(temp129*v[t1][i1][i2 - 4][i3 - 1] - temp132*
v[t1][i1][i2 - 4][i3 + 1] + temp134*v[t1][i1][i2 - 4][i3 + 2] - temp136*v[t1][i1
][i2 - 4][i3 + 3] + temp21*v[t1][i1][i2 - 4][i3 + 4] + temp509 - temp67*v[t1][i1
][i2 - 4][i3 - 2])) + temp170*temp283*(temp170*((2.0F/5.0F)*temp178 + temp193 +
temp223 - 4.0F/3.0F*temp27 + (1.0F/2.0F)*temp30 + temp304 - temp67*v[t1][i1][i2]
[i3 - 5]) + temp183*(temp129*v[t1][i1][i2 - 1][i3 - 3] - temp132*v[t1][i1][i2 +
1][i3 - 3] + temp134*v[t1][i1][i2 + 2][i3 - 3] - temp136*v[t1][i1][i2 + 3][i3 -
3] + temp193 + temp21*v[t1][i1][i2 + 4][i3 - 3] - temp67*v[t1][i1][i2 - 2][i3 -
3]) + temp192*(-temp116*v[t1][i1 - 2][i2][i3 - 3] + temp116*v[t1][i1 + 2][i2][i3
 - 3] + temp118*v[t1][i1 - 1][i2][i3 - 3] - temp118*v[t1][i1 + 1][i2][i3 - 3] -
temp21*v[t1][i1 + 3][i2][i3 - 3] + temp289)) - temp20*temp99 - temp222*temp307*(
temp222*((2.0F/5.0F)*temp107 + temp110 + temp238 - 4.0F/3.0F*temp30 + temp326 -
2.0F/15.0F*temp34 - temp67*v[t1][i1][i2][i3 - 4]) + temp227*(temp129*v[t1][i1][i
2 - 1][i3 - 2] - temp132*v[t1][i1][i2 + 1][i3 - 2] + temp134*v[t1][i1][i2 + 2][i
```

```
3 - 2] - temp136*v[t1][i1][i2 + 3][i3 - 2] + temp21*v[t1][i1][i2 + 4][i3 - 2] +
temp238 + temp489) + temp237*(temp118*v[t1][i1 - 1][i2][i3 - 2] - temp118*v[t1][
i1 + 1][i2][i3 - 2] + temp21*v[t1][i1 - 3][i2][i3 - 2] - temp21*v[t1][i1 + 3][i2
][i3 - 2] + temp313 + temp381)) - temp246*temp350*(temp246*(-2.0F/15.0F*temp143
+ temp146 + temp261 + temp369 - 4.0F/3.0F*temp38 - temp67*v[t1][i1][i2][i3 - 1]
+ (1.0F/2.0F)*temp91) + temp250*(temp129*v[t1][i1][i2 - 1][i3 + 1] + temp134*v[t
1][i1][i2 + 2][i3 + 1] - temp136*v[t1][i1][i2 + 3][i3 + 1] + temp21*v[t1][i1][i2
 + 4][i3 + 1] + temp261 + temp491 - temp67*v[t1][i1][i2 - 2][i3 + 1]) + temp260*
(-temp116*v[t1][i1 - 2][i2][i3 + 1] + temp116*v[t1][i1 + 2][i2][i3 + 1] + temp21
*v[t1][i1 - 3][i2][i3 + 1] - temp21*v[t1][i1 + 3][i2][i3 + 1] + temp336 + temp35
7)) + temp283*temp285*temp287*(temp288*(-temp116*v[t1][i1 - 3][i2][i3 - 2] + tem
p116*v[t1][i1 - 3][i2][i3 + 2] + temp118*v[t1][i1 - 3][i2][i3 - 1] - temp118*v[t
1][i1 - 3][i2][i3 + 1] - temp21*v[t1][i1 - 3][i2][i3 + 3] + temp289) + temp296*(
-temp116*v[t1][i1 - 3][i2 - 2][i3] + temp116*v[t1][i1 - 3][i2 + 2][i3] + temp118
*v[t1][i1 - 3][i2 - 1][i3] - temp118*v[t1][i1 - 3][i2 + 1][i3] - temp21*v[t1][i1
 - 3][i2 + 3][i3] + temp297) + temp303*((2.0F/5.0F)*temp281 + temp304 + temp305
+ temp306 - temp67*v[t1][i1 - 5][i2][i3] + (1.0F/2.0F)*temp72 - 4.0F/3.0F*temp87
)) + temp283*temp432*temp434*(temp435*(temp304 + (2.0F/5.0F)*temp441 + temp446 +
 temp464 - 4.0F/3.0F*temp48 + (1.0F/2.0F)*temp51 - temp67*v[t1][i1][i2 - 5][i3])
 + temp443*(temp129*v[t1][i1][i2 - 3][i3 - 1] - temp132*v[t1][i1][i2 - 3][i3 + 1
] + temp134*v[t1][i1][i2 - 3][i3 + 2] - temp136*v[t1][i1][i2 - 3][i3 + 3] + temp
21*v[t1][i1][i2 - 3][i3 + 4] + temp446 - temp67*v[t1][i1][i2 - 3][i3 - 2]) + tem
p445*(-temp116*v[t1][i1 - 2][i2 - 3][i3] + temp116*v[t1][i1 + 2][i2 - 3][i3] + t
emp118*v[t1][i1 - 1][i2 - 3][i3] - temp118*v[t1][i1 + 1][i2 - 3][i3] - temp21*v[
t1][i1 + 3][i2 - 3][i3] + temp297)) - temp307*temp309*temp311*(temp312*(temp118*
v[t1][i1 - 2][i2][i3 - 1] - temp118*v[t1][i1 - 2][i2][i3 + 1] + temp21*v[t1][i1
- 2][i2][i3 - 3] - temp21*v[t1][i1 - 2][i2][i3 + 3] + temp313 + temp314) + temp3
16*(temp118*v[t1][i1 - 2][i2 - 1][i3] - temp118*v[t1][i1 - 2][i2 + 1][i3] + temp
21*v[t1][i1 - 2][i2 - 3][i3] - temp21*v[t1][i1 - 2][i2 + 3][i3] + temp319 + temp
323) + temp325*((2.0F/5.0F)*temp282 + temp326 + temp327 + temp328 - temp67*v[t1]
[i1 - 4][i2][i3] - 4.0F/3.0F*temp72 - 2.0F/15.0F*temp75)) - temp307*temp460*temp
```

```
462*(temp463*(temp326 + (2.0F/5.0F)*temp406 + temp409 + temp468 - 4.0F/3.0F*temp
51 - 2.0F/15.0F*temp55 - temp67*v[t1][i1][i2 - 4][i3]) + temp465*(temp129*v[t1][
i1][i2 - 2][i3 - 1] - temp132*v[t1][i1][i2 - 2][i3 + 1] + temp134*v[t1][i1][i2 -
 2][i3 + 2] - temp136*v[t1][i1][i2 - 2][i3 + 3] + temp21*v[t1][i1][i2 - 2][i3 +
4] + temp468 + temp489) + temp467*(temp118*v[t1][i1 - 1][i2 - 2][i3] - temp118*v
[t1][i1 + 1][i2 - 2][i3] + temp21*v[t1][i1 - 3][i2 - 2][i3] - temp21*v[t1][i1 +
3][i2 - 2][i3] + temp319 + temp387)) + temp329*temp331*temp333*(temp334*(-temp11
6*v[t1][i1 - 1][i2][i3 - 2] + temp116*v[t1][i1 - 1][i2][i3 + 2] + temp21*v[t1][i
1 - 1][i2][i3 - 3] - temp21*v[t1][i1 - 1][i2][i3 + 3] + temp335 - temp336) + tem
p338*(-temp116*v[t1][i1 - 1][i2 - 2][i3] + temp116*v[t1][i1 - 1][i2 + 2][i3] + t
emp21*v[t1][i1 - 1][i2 - 3][i3] - temp21*v[t1][i1 - 1][i2 + 3][i3] + temp342 - t
emp344) + temp347*(temp348 + temp349 - temp67*v[t1][i1 - 3][i2][i3] + (1.0F/2.0F
)*temp75 - 2.0F/15.0F*temp78 + (2.0F/5.0F)*temp87 + temp90)) + temp329*temp400*t
emp402*(temp403*(temp348 + temp413 + (2.0F/5.0F)*temp48 + (1.0F/2.0F)*temp55 - 2
.0F/15.0F*temp59 + temp62 - temp67*v[t1][i1][i2 - 3][i3]) + temp410*(-temp132*v[
t1][i1][i2 - 1][i3 + 1] + temp134*v[t1][i1][i2 - 1][i3 + 2] - temp136*v[t1][i1][
i2 - 1][i3 + 3] + temp21*v[t1][i1][i2 - 1][i3 + 4] + temp413 + temp490 - temp67*
v[t1][i1][i2 - 1][i3 - 2]) + temp412*(-temp116*v[t1][i1 - 2][i2 - 1][i3] + temp1
16*v[t1][i1 + 2][i2 - 1][i3] + temp21*v[t1][i1 - 3][i2 - 1][i3] - temp21*v[t1][i
1 + 3][i2 - 1][i3] + temp342 - temp363)) - temp350*temp352*temp354*(temp355*(-te
mp116*v[t1][i1 + 1][i2][i3 - 2] + temp116*v[t1][i1 + 1][i2][i3 + 2] + temp21*v[t
1][i1 + 1][i2][i3 - 3] - temp21*v[t1][i1 + 1][i2][i3 + 3] + temp356 + temp357) +
 temp359*(-temp116*v[t1][i1 + 1][i2 - 2][i3] + temp116*v[t1][i1 + 1][i2 + 2][i3]
 + temp21*v[t1][i1 + 1][i2 - 3][i3] - temp21*v[t1][i1 + 1][i2 + 3][i3] + temp363
 + temp365) + temp368*(temp369 + temp370 - 2.0F/15.0F*temp371 + temp374 - temp67
*v[t1][i1 - 1][i2][i3] - 4.0F/3.0F*temp78 + (1.0F/2.0F)*temp81)) - temp350*temp4
72*temp474*(temp475*(temp369 - 2.0F/15.0F*temp421 + temp424 + temp479 - 4.0F/3.0
F*temp59 - temp67*v[t1][i1][i2 - 1][i3] + (1.0F/2.0F)*temp95) + temp476*(temp129
*v[t1][i1][i2 + 1][i3 - 1] + temp134*v[t1][i1][i2 + 1][i3 + 2] - temp136*v[t1][i
1][i2 + 1][i3 + 3] + temp21*v[t1][i1][i2 + 1][i3 + 4] + temp479 + temp491 - temp
67*v[t1][i1][i2 + 1][i3 - 2]) + temp478*(-temp116*v[t1][i1 - 2][i2 + 1][i3] + te
```

```
mp116*v[t1][i1 + 2][i2 + 1][i3] + temp21*v[t1][i1 - 3][i2 + 1][i3] - temp21*v[t1
][i1 + 3][i2 + 1][i3] + temp344 + temp365)) + temp375*temp377*temp379*(temp380*(
temp118*v[t1][i1 + 2][i2][i3 - 1] - temp118*v[t1][i1 + 2][i2][i3 + 1] + temp21*v
[t1][i1 + 2][i2][i3 - 3] - temp21*v[t1][i1 + 2][i2][i3 + 3] - temp381 + temp382)
 + temp384*(temp118*v[t1][i1 + 2][i2 - 1][i3] - temp118*v[t1][i1 + 2][i2 + 1][i3
] + temp21*v[t1][i1 + 2][i2 - 3][i3] - temp21*v[t1][i1 + 2][i2 + 3][i3] - temp38
7 + temp391) + temp393*((1.0F/2.0F)*temp371 + temp394 + temp395 - 2.0F/15.0F*tem
p396 + temp398 + (2.0F/5.0F)*temp75 - 4.0F/3.0F*temp81)) + temp375*temp417*temp4
19*(temp420*(temp394 + (1.0F/2.0F)*temp421 + temp428 - 2.0F/15.0F*temp453 + temp
455 + (2.0F/5.0F)*temp55 - 4.0F/3.0F*temp95) + temp425*(temp129*v[t1][i1][i2 + 2
][i3 - 1] - temp132*v[t1][i1][i2 + 2][i3 + 1] - temp136*v[t1][i1][i2 + 2][i3 + 3
] + temp21*v[t1][i1][i2 + 2][i3 + 4] + temp428 + temp492 - temp67*v[t1][i1][i2 +
 2][i3 - 2]) + temp427*(temp118*v[t1][i1 - 1][i2 + 2][i3] - temp118*v[t1][i1 + 1
][i2 + 2][i3] + temp21*v[t1][i1 - 3][i2 + 2][i3] - temp21*v[t1][i1 + 3][i2 + 2][
i3] - temp323 + temp391)) - temp43*temp99;
                  float temp514 = temp12*u[t1][i1][i2 - 2][i3];
                  float temp516 = temp12*u[t1][i1][i2 - 1][i3];
                  float temp517 = (3.0F/4.0F)*temp516;
                  float temp519 = temp12*u[t1][i1][i2 + 1][i3];
                  float temp520 = (3.0F/4.0F)*temp519;
                  float temp522 = temp12*u[t1][i1][i2 + 2][i3];
                  float temp524 = temp21*u[t1][i1][i2 + 3][i3];
                  float temp525 = temp21*u[t1][i1][i2 - 3][i3] - 3.0F/20.0F*temp
514 + temp517 - temp520 + (3.0F/20.0F)*temp522 - temp524;
                  float temp526 = temp12*u[t1][i1][i2][i3];
                  float temp527 = (7.0F/12.0F)*temp526;
                  float temp530 = temp12*u[t1][i1 - 1][i2][i3];
                  float temp532 = temp12*u[t1][i1 + 1][i2][i3];
                  float temp534 = temp12*u[t1][i1 + 2][i2][i3];
                  float temp536 = temp12*u[t1][i1 + 3][i2][i3];
                  float temp538 = temp21*u[t1][i1 + 4][i2][i3];
```

```
                  float temp539 = temp527 + (2.0F/5.0F)*temp530 - 4.0F/3.0F*temp
532 + (1.0F/2.0F)*temp534 - 2.0F/15.0F*temp536 + temp538 - temp67*u[t1][i1 - 2][
i2][i3];
                  float temp541 = temp12*u[t1][i1 - 2][i2][i3];
                  float temp542 = (3.0F/4.0F)*temp530;
                  float temp543 = (3.0F/4.0F)*temp532;
                  float temp544 = temp21*u[t1][i1 + 3][i2][i3];
                  float temp545 = temp21*u[t1][i1 - 3][i2][i3] + (3.0F/20.0F)*te
mp534 - 3.0F/20.0F*temp541 + temp542 - temp543 - temp544;
                  float temp546 = temp12*u[t1][i1][i2 + 3][i3];
                  float temp548 = temp21*u[t1][i1][i2 + 4][i3];
                  float temp549 = (2.0F/5.0F)*temp516 - 4.0F/3.0F*temp519 + (1.0
F/2.0F)*temp522 + temp527 - 2.0F/15.0F*temp546 + temp548 - temp67*u[t1][i1][i2 -
 2][i3];
                  float temp550 = (3.0F/4.0F)*temp526;
                  float temp551 = -temp550;
                  float temp553 = temp12*u[t1][i1][i2 - 3][i3];
                  float temp554 = (3.0F/4.0F)*temp514;
                  float temp555 = (3.0F/20.0F)*temp519;
                  float temp556 = temp21*u[t1][i1][i2 + 2][i3];
                  float temp557 = temp21*u[t1][i1][i2 - 4][i3] + temp551 - 3.0F/
20.0F*temp553 + temp554 + temp555 - temp556;
                  float temp558 = (7.0F/12.0F)*temp516;
                  float temp561 = temp129*u[t1][i1 - 1][i2 - 1][i3];
                  float temp565 = -temp132*u[t1][i1 + 1][i2 - 1][i3] + temp134*u
[t1][i1 + 2][i2 - 1][i3] - temp136*u[t1][i1 + 3][i2 - 1][i3] + temp21*u[t1][i1 +
 4][i2 - 1][i3] + temp558 + temp561 - temp67*u[t1][i1 - 2][i2 - 1][i3];
                  float temp567 = temp12*u[t1][i1 - 3][i2][i3];
                  float temp568 = (3.0F/4.0F)*temp541;
                  float temp569 = (3.0F/20.0F)*temp532;
                  float temp570 = temp21*u[t1][i1 + 2][i2][i3];
```

```
                  float temp571 = temp21*u[t1][i1 - 4][i2][i3] + temp551 - 3.0F/
20.0F*temp567 + temp568 + temp569 - temp570;
                  float temp572 = (7.0F/12.0F)*temp530;
                  float temp577 = -temp132*u[t1][i1 - 1][i2 + 1][i3] + temp134*u
[t1][i1 - 1][i2 + 2][i3] - temp136*u[t1][i1 - 1][i2 + 3][i3] + temp21*u[t1][i1 -
 1][i2 + 4][i3] + temp561 + temp572 - temp67*u[t1][i1 - 1][i2 - 2][i3];
                  float temp578 = (3.0F/20.0F)*temp526;
                  float temp579 = -temp578;
                  float temp580 = temp12*u[t1][i1][i2 + 4][i3];
                  float temp582 = temp21*u[t1][i1][i2 + 5][i3];
                  float temp583 = temp21*u[t1][i1][i2 - 1][i3] + temp520 - 3.0F/
4.0F*temp546 + temp579 + (3.0F/20.0F)*temp580 - temp582;
                  float temp584 = (7.0F/12.0F)*temp522;
                  float temp588 = temp134*u[t1][i1 + 2][i2 + 2][i3];
                  float temp590 = temp129*u[t1][i1 - 1][i2 + 2][i3] - temp132*u[
t1][i1 + 1][i2 + 2][i3] - temp136*u[t1][i1 + 3][i2 + 2][i3] + temp21*u[t1][i1 +
4][i2 + 2][i3] + temp584 + temp588 - temp67*u[t1][i1 - 2][i2 + 2][i3];
                  float temp591 = temp12*u[t1][i1 + 4][i2][i3];
                  float temp593 = temp21*u[t1][i1 + 5][i2][i3];
                  float temp594 = temp21*u[t1][i1 - 1][i2][i3] - 3.0F/4.0F*temp5
36 + temp543 + temp579 + (3.0F/20.0F)*temp591 - temp593;
                  float temp595 = (7.0F/12.0F)*temp534;
                  float temp599 = temp129*u[t1][i1 + 2][i2 - 1][i3] - temp132*u[
t1][i1 + 2][i2 + 1][i3] - temp136*u[t1][i1 + 2][i2 + 3][i3] + temp21*u[t1][i1 +
2][i2 + 4][i3] + temp588 + temp595 - temp67*u[t1][i1 + 2][i2 - 2][i3];
                  float temp600 = (1.0F/60.0F)*temp526;
                  float temp601 = -temp600;
                  float temp604 = temp12*u[t1][i1][i2 - 5][i3];
                  float temp605 = temp12*u[t1][i1][i2 - 4][i3];
                  float temp606 = (3.0F/20.0F)*temp516;
                  float temp607 = temp21*u[t1][i1][i2 - 6][i3] - temp554 + temp6
```

```
01 - 3.0F/20.0F*temp604 + (3.0F/4.0F)*temp605 + temp606;
                  float temp608 = (7.0F/12.0F)*temp553;
                  float temp614 = temp129*u[t1][i1 - 1][i2 - 3][i3] - temp132*u[
t1][i1 + 1][i2 - 3][i3] + temp134*u[t1][i1 + 2][i2 - 3][i3] - temp136*u[t1][i1 +
 3][i2 - 3][i3] + temp21*u[t1][i1 + 4][i2 - 3][i3] + temp608 - temp67*u[t1][i1 -
 2][i2 - 3][i3];
                  float temp617 = temp12*u[t1][i1 - 5][i2][i3];
                  float temp618 = temp12*u[t1][i1 - 4][i2][i3];
                  float temp619 = (3.0F/20.0F)*temp530;
                  float temp620 = temp21*u[t1][i1 - 6][i2][i3] - temp568 + temp6
01 - 3.0F/20.0F*temp617 + (3.0F/4.0F)*temp618 + temp619;
                  float temp621 = (7.0F/12.0F)*temp567;
                  float temp627 = temp129*u[t1][i1 - 3][i2 - 1][i3] - temp132*u[
t1][i1 - 3][i2 + 1][i3] + temp134*u[t1][i1 - 3][i2 + 2][i3] - temp136*u[t1][i1 -
 3][i2 + 3][i3] + temp21*u[t1][i1 - 3][i2 + 4][i3] + temp621 - temp67*u[t1][i1 -
 3][i2 - 2][i3];
                  float temp628 = (3.0F/4.0F)*temp522;
                  float temp629 = temp12*u[t1][i1][i2 + 5][i3];
                  float temp630 = temp21*u[t1][i1][i2 + 6][i3];
                  float temp631 = -temp555 - 3.0F/4.0F*temp580 + temp600 + temp6
28 + (3.0F/20.0F)*temp629 - temp630;
                  float temp634 = -temp136*u[t1][i1 + 3][i2 + 3][i3];
                  float temp635 = temp129*u[t1][i1 - 1][i2 + 3][i3] - temp132*u[
t1][i1 + 1][i2 + 3][i3] + temp134*u[t1][i1 + 2][i2 + 3][i3] + temp21*u[t1][i1 +
4][i2 + 3][i3] + (7.0F/12.0F)*temp546 + temp634 - temp67*u[t1][i1 - 2][i2 + 3][i
3];
                  float temp636 = (3.0F/4.0F)*temp534;
                  float temp637 = temp12*u[t1][i1 + 5][i2][i3];
                  float temp638 = temp21*u[t1][i1 + 6][i2][i3];
                  float temp639 = -temp569 - 3.0F/4.0F*temp591 + temp600 + temp6
36 + (3.0F/20.0F)*temp637 - temp638;
```

```
                  float temp640 = (7.0F/12.0F)*temp536;
                  float temp643 = temp129*u[t1][i1 + 3][i2 - 1][i3] - temp132*u[
t1][i1 + 3][i2 + 1][i3] + temp134*u[t1][i1 + 3][i2 + 2][i3] + temp21*u[t1][i1 +
3][i2 + 4][i3] + temp634 + temp640 - temp67*u[t1][i1 + 3][i2 - 2][i3];
                  float temp644 = temp21*u[t1][i1][i2 + 1][i3];
                  float temp645 = temp21*u[t1][i1][i2 - 5][i3] - temp517 + (3.0F
/4.0F)*temp553 + temp578 - 3.0F/20.0F*temp605 - temp644;
                  float temp646 = (7.0F/12.0F)*temp514;
                  float temp648 = -temp67*u[t1][i1 - 2][i2 - 2][i3];
                  float temp650 = temp129*u[t1][i1 - 1][i2 - 2][i3] - temp132*u[
t1][i1 + 1][i2 - 2][i3] + temp134*u[t1][i1 + 2][i2 - 2][i3] - temp136*u[t1][i1 +
 3][i2 - 2][i3] + temp21*u[t1][i1 + 4][i2 - 2][i3] + temp646 + temp648;
                  float temp651 = temp21*u[t1][i1 + 1][i2][i3];
                  float temp652 = temp21*u[t1][i1 - 5][i2][i3] - temp542 + (3.0F
/4.0F)*temp567 + temp578 - 3.0F/20.0F*temp618 - temp651;
                  float temp653 = (7.0F/12.0F)*temp541;
                  float temp655 = temp129*u[t1][i1 - 2][i2 - 1][i3] - temp132*u[
t1][i1 - 2][i2 + 1][i3] + temp134*u[t1][i1 - 2][i2 + 2][i3] - temp136*u[t1][i1 -
 2][i2 + 3][i3] + temp21*u[t1][i1 - 2][i2 + 4][i3] + temp648 + temp653;
                  float temp656 = temp21*u[t1][i1][i2 - 2][i3] + (3.0F/20.0F)*te
mp546 - temp548 + temp550 - temp606 - temp628;
                  float temp657 = (7.0F/12.0F)*temp519;
                  float temp659 = -temp132*u[t1][i1 + 1][i2 + 1][i3];
                  float temp660 = temp129*u[t1][i1 - 1][i2 + 1][i3] + temp134*u[
t1][i1 + 2][i2 + 1][i3] - temp136*u[t1][i1 + 3][i2 + 1][i3] + temp21*u[t1][i1 +
4][i2 + 1][i3] + temp657 + temp659 - temp67*u[t1][i1 - 2][i2 + 1][i3];
                  float temp661 = temp21*u[t1][i1 - 2][i2][i3] + (3.0F/20.0F)*te
mp536 - temp538 + temp550 - temp619 - temp636;
                  float temp662 = (7.0F/12.0F)*temp532;
                  float temp663 = temp129*u[t1][i1 + 1][i2 - 1][i3] + temp134*u[
t1][i1 + 1][i2 + 2][i3] - temp136*u[t1][i1 + 1][i2 + 3][i3] + temp21*u[t1][i1 +
```

```
1][i2 + 4][i3] + temp659 + temp662 - temp67*u[t1][i1 + 1][i2 - 2][i3];
                  float temp664 = -temp116*u[t1][i1 - 4][i2 - 2][i3] + temp116*u
[t1][i1 - 4][i2 + 2][i3] + temp118*u[t1][i1 - 4][i2 - 1][i3] - temp118*u[t1][i1
- 4][i2 + 1][i3] + temp21*u[t1][i1 - 4][i2 - 3][i3] - temp21*u[t1][i1 - 4][i2 +
3][i3];
                  float temp665 = -2.0F/15.0F*temp530 + (1.0F/2.0F)*temp541 - 4.
0F/3.0F*temp567 + temp600 + (2.0F/5.0F)*temp617 + (7.0F/12.0F)*temp618 - temp67*
u[t1][i1 - 6][i2][i3];
                  float temp666 = -temp116*u[t1][i1 - 2][i2 - 4][i3] + temp116*u
[t1][i1 + 2][i2 - 4][i3] + temp118*u[t1][i1 - 1][i2 - 4][i3] - temp118*u[t1][i1
+ 1][i2 - 4][i3] + temp21*u[t1][i1 - 3][i2 - 4][i3] - temp21*u[t1][i1 + 3][i2 -
4][i3];
                  float temp667 = (7.0F/12.0F)*temp605;
                  float temp668 = (1.0F/2.0F)*temp514 - 2.0F/15.0F*temp516 - 4.0
F/3.0F*temp553 + temp600 + (2.0F/5.0F)*temp604 + temp667 - temp67*u[t1][i1][i2 -
 6][i3];
                  float temp669 = temp21*u[t1][i1 - 3][i2 - 3][i3];
                  float temp670 = -temp116*u[t1][i1 - 3][i2 - 2][i3] + temp116*u
[t1][i1 - 3][i2 + 2][i3] + temp118*u[t1][i1 - 3][i2 - 1][i3] - temp118*u[t1][i1
- 3][i2 + 1][i3] - temp21*u[t1][i1 - 3][i2 + 3][i3] + temp669;
                  float temp671 = -2.0F/15.0F*temp526;
                  float temp672 = (1.0F/2.0F)*temp530 - 4.0F/3.0F*temp541 + (2.0
F/5.0F)*temp618 + temp621 + temp651 - temp67*u[t1][i1 - 5][i2][i3] + temp671;
                  float temp673 = -temp116*u[t1][i1 - 2][i2 - 3][i3] + temp116*u
[t1][i1 + 2][i2 - 3][i3] + temp118*u[t1][i1 - 1][i2 - 3][i3] - temp118*u[t1][i1
+ 1][i2 - 3][i3] - temp21*u[t1][i1 + 3][i2 - 3][i3] + temp669;
                  float temp674 = -4.0F/3.0F*temp514 + (1.0F/2.0F)*temp516 + (2.
0F/5.0F)*temp605 + temp608 + temp644 - temp67*u[t1][i1][i2 - 5][i3] + temp671;
                  float temp675 = -temp116*u[t1][i1 - 2][i2 - 2][i3];
                  float temp676 = temp116*u[t1][i1 + 2][i2 - 2][i3];
                  float temp677 = temp118*u[t1][i1 - 1][i2 - 2][i3] - temp118*u[
```

```
t1][i1 + 1][i2 - 2][i3] + temp21*u[t1][i1 - 3][i2 - 2][i3] - temp21*u[t1][i1 + 3
][i2 - 2][i3] + temp675 + temp676;
                  float temp678 = (1.0F/2.0F)*temp526;
                  float temp679 = -4.0F/3.0F*temp516 - 2.0F/15.0F*temp519 + (2.0
F/5.0F)*temp553 + temp556 + temp646 - temp67*u[t1][i1][i2 - 4][i3] + temp678;
                  float temp680 = temp118*u[t1][i1 - 1][i2 - 1][i3];
                  float temp681 = temp118*u[t1][i1 + 1][i2 - 1][i3];
                  float temp682 = -temp116*u[t1][i1 - 2][i2 - 1][i3] + temp116*u
[t1][i1 + 2][i2 - 1][i3] + temp21*u[t1][i1 - 3][i2 - 1][i3] - temp21*u[t1][i1 +
3][i2 - 1][i3] + temp680 - temp681;
                  float temp683 = -4.0F/3.0F*temp526;
                  float temp684 = (2.0F/5.0F)*temp514 + (1.0F/2.0F)*temp519 - 2.
0F/15.0F*temp522 + temp524 + temp558 - temp67*u[t1][i1][i2 - 3][i3] + temp683;
                  float temp685 = temp118*u[t1][i1 - 1][i2 + 1][i3];
                  float temp686 = -temp118*u[t1][i1 + 1][i2 + 1][i3];
                  float temp687 = -temp116*u[t1][i1 - 2][i2 + 1][i3] + temp116*u
[t1][i1 + 2][i2 + 1][i3] + temp21*u[t1][i1 - 3][i2 + 1][i3] - temp21*u[t1][i1 +
3][i2 + 1][i3] + temp685 + temp686;
                  float temp688 = (2.0F/5.0F)*temp526;
                  float temp689 = -4.0F/3.0F*temp522 + (1.0F/2.0F)*temp546 - 2.0
F/15.0F*temp580 + temp582 + temp657 - temp67*u[t1][i1][i2 - 1][i3] + temp688;
                  float temp690 = temp116*u[t1][i1 - 2][i2 + 2][i3];
                  float temp691 = temp116*u[t1][i1 + 2][i2 + 2][i3];
                  float temp692 = temp118*u[t1][i1 - 1][i2 + 2][i3] - temp118*u[
t1][i1 + 1][i2 + 2][i3] + temp21*u[t1][i1 - 3][i2 + 2][i3] - temp21*u[t1][i1 + 3
][i2 + 2][i3] - temp690 + temp691;
                  float temp693 = -1.0F/30.0F*temp526;
                  float temp694 = (2.0F/5.0F)*temp519 - 4.0F/3.0F*temp546 + (1.0
F/2.0F)*temp580 + temp584 - 2.0F/15.0F*temp629 + temp630 + temp693;
                  float temp695 = temp118*u[t1][i1 - 2][i2 - 1][i3] - temp118*u[
t1][i1 - 2][i2 + 1][i3] + temp21*u[t1][i1 - 2][i2 - 3][i3] - temp21*u[t1][i1 - 2
```

```
][i2 + 3][i3] + temp675 + temp690;
                  float temp696 = -4.0F/3.0F*temp530 - 2.0F/15.0F*temp532 + (2.0
F/5.0F)*temp567 + temp570 + temp653 - temp67*u[t1][i1 - 4][i2][i3] + temp678;
                  float temp697 = -temp116*u[t1][i1 - 1][i2 - 2][i3] + temp116*u
[t1][i1 - 1][i2 + 2][i3] + temp21*u[t1][i1 - 1][i2 - 3][i3] - temp21*u[t1][i1 -
1][i2 + 3][i3] + temp680 - temp685;
                  float temp698 = (1.0F/2.0F)*temp532 - 2.0F/15.0F*temp534 + (2.
0F/5.0F)*temp541 + temp544 + temp572 - temp67*u[t1][i1 - 3][i2][i3] + temp683;
                  float temp699 = -temp116*u[t1][i1 + 1][i2 - 2][i3] + temp116*u
[t1][i1 + 1][i2 + 2][i3] + temp21*u[t1][i1 + 1][i2 - 3][i3] - temp21*u[t1][i1 +
1][i2 + 3][i3] + temp681 + temp686;
                  float temp700 = -4.0F/3.0F*temp534 + (1.0F/2.0F)*temp536 - 2.0
F/15.0F*temp591 + temp593 + temp662 - temp67*u[t1][i1 - 1][i2][i3] + temp688;
                  float temp701 = temp118*u[t1][i1 + 2][i2 - 1][i3] - temp118*u[
t1][i1 + 2][i2 + 1][i3] + temp21*u[t1][i1 + 2][i2 - 3][i3] - temp21*u[t1][i1 + 2
][i2 + 3][i3] - temp676 + temp691;
                  float temp702 = (2.0F/5.0F)*temp532 - 4.0F/3.0F*temp536 + (1.0
F/2.0F)*temp591 + temp595 - 2.0F/15.0F*temp637 + temp638 + temp693;
                  float temp703 = temp12*u[t1][i1][i2][i3 - 3];
                  float temp704 = temp12*u[t1][i1][i2][i3 - 2];
                  float temp705 = temp12*u[t1][i1][i2][i3 - 1];
                  float temp706 = (3.0F/4.0F)*temp705;
                  float temp707 = temp12*u[t1][i1][i2][i3 + 1];
                  float temp708 = (3.0F/4.0F)*temp707;
                  float temp709 = temp12*u[t1][i1][i2][i3 + 2];
                  float temp710 = temp12*u[t1][i1][i2][i3 + 3];
                  float temp711 = (1.0F/60.0F)*temp710;
                  float temp712 = temp20*temp42;
                  float temp713 = temp19*temp20;
                  float temp714 = temp12*u[t1][i1][i2][i3 + 4];
                  float temp715 = (1.0F/60.0F)*temp714;
                  float temp716 = 2.91666666666667e-1F*temp12*(-temp16*(temp527
- 1.0F/30.0F*temp704 + (2.0F/5.0F)*temp705 - 4.0F/3.0F*temp707 + (1.0F/2.0F)*tem
p709 - 2.0F/15.0F*temp710 + temp715) + temp545*temp713 + temp549*temp712);
                  float temp717 = temp12*u[t1][i1][i2][i3 - 4];
                  float temp718 = (3.0F/4.0F)*temp704;
                  float temp719 = (3.0F/20.0F)*temp707;
                  float temp720 = (1.0F/60.0F)*temp709;
                  float temp721 = temp102*temp112;
                  float temp725 = temp118*u[t1][i1][i2 - 1][i3 - 1];
                  float temp727 = temp118*u[t1][i1][i2 + 1][i3 - 1];
                  float temp730 = temp102*temp125;
                  float temp731 = (7.0F/12.0F)*temp705;
                  float temp734 = temp129*u[t1][i1 - 1][i2][i3 - 1];
                  float temp738 = temp12*u[t1][i1][i2][i3 + 5];
                  float temp739 = (1.0F/60.0F)*temp738;
                  float temp740 = temp140*temp148;
                  float temp743 = temp116*u[t1][i1][i2 - 2][i3 + 2];
                  float temp747 = temp116*u[t1][i1][i2 + 2][i3 + 2];
                  float temp749 = temp140*temp159;
                  float temp750 = (7.0F/12.0F)*temp709;
                  float temp755 = temp134*u[t1][i1 + 2][i2][i3 + 2];
                  float temp757 = temp12*u[t1][i1][i2][i3 - 6];
                  float temp758 = temp12*u[t1][i1][i2][i3 - 5];
                  float temp759 = (3.0F/20.0F)*temp705;
                  float temp760 = temp170*temp181;
                  float temp761 = temp21*u[t1][i1][i2 - 3][i3 - 3];
                  float temp767 = temp21*u[t1][i1][i2 + 3][i3 - 3];
                  float temp768 = temp170*temp191;
                  float temp769 = (7.0F/12.0F)*temp703;
                  float temp775 = (3.0F/4.0F)*temp709;
                  float temp776 = temp21*u[t1][i1][i2][i3 + 6];
```

```
                  float temp778 = temp21*u[t1][i1][i2 - 3][i3 + 3];
                  float temp783 = -temp21*u[t1][i1][i2 + 3][i3 + 3];
                  float temp788 = -temp136*u[t1][i1 + 3][i2][i3 + 3];
                  float temp789 = (1.0F/60.0F)*temp707;
                  float temp790 = temp222*temp225;
                  float temp793 = -temp116*u[t1][i1][i2 - 2][i3 - 2];
                  float temp797 = temp116*u[t1][i1][i2 + 2][i3 - 2];
                  float temp799 = temp222*temp236;
                  float temp800 = (7.0F/12.0F)*temp704;
                  float temp802 = -temp67*u[t1][i1 - 2][i2][i3 - 2];
                  float temp807 = temp246*temp248;
                  float temp811 = temp118*u[t1][i1][i2 - 1][i3 + 1];
                  float temp813 = -temp118*u[t1][i1][i2 + 1][i3 + 1];
                  float temp816 = temp246*temp259;
                  float temp817 = (7.0F/12.0F)*temp707;
                  float temp821 = -temp132*u[t1][i1 + 1][i2][i3 + 1];
                  float temp824 = temp21*u[t1][i1 - 3][i2][i3 - 3];
                  float temp830 = temp288*temp295;
                  float temp831 = temp287*temp288;
                  float temp832 = -temp116*u[t1][i1 - 2][i2][i3 - 2];
                  float temp833 = temp116*u[t1][i1 - 2][i2][i3 + 2];
                  float temp834 = temp312*temp315;
                  float temp835 = temp311*temp312;
                  float temp836 = temp118*u[t1][i1 - 1][i2][i3 - 1];
                  float temp837 = temp118*u[t1][i1 - 1][i2][i3 + 1];
                  float temp838 = temp334*temp337;
                  float temp839 = temp333*temp334;
                  float temp840 = temp118*u[t1][i1 + 1][i2][i3 - 1];
                  float temp841 = -temp118*u[t1][i1 + 1][i2][i3 + 1];
                  float temp842 = temp355*temp358;
                  float temp843 = temp354*temp355;
```

```
                  float temp844 = temp116*u[t1][i1 + 2][i2][i3 - 2];
                  float temp845 = temp116*u[t1][i1 + 2][i2][i3 + 2];
                  float temp846 = temp380*temp383;
                  float temp847 = temp379*temp380;
                  float temp848 = temp400*temp410;
                  float temp849 = temp410*temp411;
                  float temp850 = temp417*temp425;
                  float temp851 = temp425*temp426;
                  float temp852 = temp432*temp443;
                  float temp853 = temp443*temp444;
                  float temp854 = temp460*temp465;
                  float temp855 = temp465*temp466;
                  float temp856 = temp472*temp476;
                  float temp857 = temp476*temp477;
                  float temp858 = (7.0F/12.0F)*temp717;
                  float temp859 = -temp67*u[t1][i1][i2 - 2][i3 - 2];
                  float temp860 = temp129*u[t1][i1][i2 - 1][i3 - 1];
                  float temp861 = -temp132*u[t1][i1][i2 + 1][i3 + 1];
                  float temp862 = temp134*u[t1][i1][i2 + 2][i3 + 2];
                  float temp863 = -temp100*temp113*(-temp113*(temp551 - 3.0F/20.
0F*temp703 + (1.0F/60.0F)*temp717 + temp718 + temp719 - temp720) + temp721*(-tem
p116*u[t1][i1][i2 - 2][i3 - 1] + temp116*u[t1][i1][i2 + 2][i3 - 1] + temp21*u[t1
][i1][i2 - 3][i3 - 1] - temp21*u[t1][i1][i2 + 3][i3 - 1] + temp725 - temp727) +
temp730*(-temp132*u[t1][i1 + 1][i2][i3 - 1] + temp134*u[t1][i1 + 2][i2][i3 - 1]
- temp136*u[t1][i1 + 3][i2][i3 - 1] + temp21*u[t1][i1 + 4][i2][i3 - 1] - temp67*
u[t1][i1 - 2][i2][i3 - 1] + temp731 + temp734)) + temp100*temp249*(-temp249*(tem
p550 + (1.0F/60.0F)*temp704 + (3.0F/20.0F)*temp710 - temp715 - temp759 - temp775
) + temp807*(-temp116*u[t1][i1][i2 - 2][i3 + 1] + temp116*u[t1][i1][i2 + 2][i3 +
 1] + temp21*u[t1][i1][i2 - 3][i3 + 1] - temp21*u[t1][i1][i2 + 3][i3 + 1] + temp
811 + temp813) + temp816*(temp129*u[t1][i1 - 1][i2][i3 + 1] + temp134*u[t1][i1 +
 2][i2][i3 + 1] - temp136*u[t1][i1 + 3][i2][i3 + 1] + temp21*u[t1][i1 + 4][i2][i
```

```
3 + 1] - temp67*u[t1][i1 - 2][i2][i3 + 1] + temp817 + temp821)) + temp100*temp33
3*temp334*(-temp331*(-temp132*u[t1][i1 - 1][i2][i3 + 1] + temp134*u[t1][i1 - 1][
i2][i3 + 2] - temp136*u[t1][i1 - 1][i2][i3 + 3] + temp21*u[t1][i1 - 1][i2][i3 +
4] + temp572 - temp67*u[t1][i1 - 1][i2][i3 - 2] + temp734) + temp571*temp839 + t
emp577*temp838) + temp100*temp337*(-temp333*temp577 + temp337*temp571) - temp100
*temp354*temp355*(-temp352*(temp129*u[t1][i1 + 1][i2][i3 - 1] + temp134*u[t1][i1
 + 1][i2][i3 + 2] - temp136*u[t1][i1 + 1][i2][i3 + 3] + temp21*u[t1][i1 + 1][i2]
[i3 + 4] + temp662 - temp67*u[t1][i1 + 1][i2][i3 - 2] + temp821) + temp661*temp8
43 + temp663*temp842) - temp100*temp358*(-temp354*temp663 + temp358*temp661) + t
emp100*temp400*temp410*(-temp402*(-temp116*u[t1][i1][i2 - 1][i3 - 2] + temp116*u
[t1][i1][i2 - 1][i3 + 2] + temp21*u[t1][i1][i2 - 1][i3 - 3] - temp21*u[t1][i1][i
2 - 1][i3 + 3] + temp725 - temp811) + temp557*temp848 + temp565*temp849) - temp1
00*temp411*(temp400*temp565 - temp411*temp557) - temp100*temp472*temp476*(-temp4
74*(-temp116*u[t1][i1][i2 + 1][i3 - 2] + temp116*u[t1][i1][i2 + 1][i3 + 2] + tem
p21*u[t1][i1][i2 + 1][i3 - 3] - temp21*u[t1][i1][i2 + 1][i3 + 3] + temp727 + tem
p813) + temp656*temp856 + temp660*temp857) + temp100*temp477*(temp472*temp660 -
temp477*temp656) - temp113*temp329*(-temp113*(temp683 - 1.0F/30.0F*temp703 + (2.
0F/5.0F)*temp704 + (1.0F/2.0F)*temp707 - 2.0F/15.0F*temp709 + temp711 + temp731)
 + temp721*(-temp132*u[t1][i1][i2 + 1][i3 - 1] + temp134*u[t1][i1][i2 + 2][i3 -
1] - temp136*u[t1][i1][i2 + 3][i3 - 1] + temp21*u[t1][i1][i2 + 4][i3 - 1] - temp
67*u[t1][i1][i2 - 2][i3 - 1] + temp731 + temp860) + temp730*(-temp116*u[t1][i1 -
 2][i2][i3 - 1] + temp116*u[t1][i1 + 2][i2][i3 - 1] + temp21*u[t1][i1 - 3][i2][i
3 - 1] - temp21*u[t1][i1 + 3][i2][i3 - 1] + temp836 - temp840)) - temp13*temp19*
temp20*(-temp16*((1.0F/60.0F)*temp703 - 3.0F/20.0F*temp704 + temp706 - temp708 +
 (3.0F/20.0F)*temp709 - temp711) + temp525*temp712 + temp539*temp713) + temp13*t
emp19*(-temp19*temp549 + temp42*temp545) - temp13*temp42*(-temp19*temp525 + temp
42*temp539) - temp138*temp149*(-temp149*(temp579 + (1.0F/60.0F)*temp705 + temp70
8 - 3.0F/4.0F*temp710 + (3.0F/20.0F)*temp714 - temp739) + temp740*(temp118*u[t1]
[i1][i2 - 1][i3 + 2] - temp118*u[t1][i1][i2 + 1][i3 + 2] + temp21*u[t1][i1][i2 -
 3][i3 + 2] - temp21*u[t1][i1][i2 + 3][i3 + 2] - temp743 + temp747) + temp749*(t
emp129*u[t1][i1 - 1][i2][i3 + 2] - temp132*u[t1][i1 + 1][i2][i3 + 2] - temp136*u
```

```
[t1][i1 + 3][i2][i3 + 2] + temp21*u[t1][i1 + 4][i2][i3 + 2] - temp67*u[t1][i1 -
2][i2][i3 + 2] + temp750 + temp755)) + temp138*temp226*(-temp226*(temp578 + (3.0
F/4.0F)*temp703 - temp706 - 3.0F/20.0F*temp717 + (1.0F/60.0F)*temp758 - temp789)
 + temp790*(temp118*u[t1][i1][i2 - 1][i3 - 2] - temp118*u[t1][i1][i2 + 1][i3 - 2
] + temp21*u[t1][i1][i2 - 3][i3 - 2] - temp21*u[t1][i1][i2 + 3][i3 - 2] + temp79
3 + temp797) + temp799*(temp129*u[t1][i1 - 1][i2][i3 - 2] - temp132*u[t1][i1 + 1
][i2][i3 - 2] + temp134*u[t1][i1 + 2][i2][i3 - 2] - temp136*u[t1][i1 + 3][i2][i3
 - 2] + temp21*u[t1][i1 + 4][i2][i3 - 2] + temp800 + temp802)) - temp138*temp311
*temp312*(-temp309*(temp129*u[t1][i1 - 2][i2][i3 - 1] - temp132*u[t1][i1 - 2][i2
][i3 + 1] + temp134*u[t1][i1 - 2][i2][i3 + 2] - temp136*u[t1][i1 - 2][i2][i3 + 3
] + temp21*u[t1][i1 - 2][i2][i3 + 4] + temp653 + temp802) + temp652*temp835 + te
mp655*temp834) - temp138*temp315*(-temp311*temp655 + temp315*temp652) + temp138*
temp379*temp380*(-temp377*(temp129*u[t1][i1 + 2][i2][i3 - 1] - temp132*u[t1][i1
+ 2][i2][i3 + 1] - temp136*u[t1][i1 + 2][i2][i3 + 3] + temp21*u[t1][i1 + 2][i2][
i3 + 4] + temp595 - temp67*u[t1][i1 + 2][i2][i3 - 2] + temp755) + temp594*temp84
7 + temp599*temp846) + temp138*temp383*(-temp379*temp599 + temp383*temp594) + te
mp138*temp417*temp425*(-temp419*(temp118*u[t1][i1][i2 + 2][i3 - 1] - temp118*u[t
1][i1][i2 + 2][i3 + 1] + temp21*u[t1][i1][i2 + 2][i3 - 3] - temp21*u[t1][i1][i2
+ 2][i3 + 3] + temp747 - temp797) + temp583*temp850 + temp590*temp851) - temp138
*temp426*(temp417*temp590 - temp426*temp583) - temp138*temp460*temp465*(-temp462
*(temp118*u[t1][i1][i2 - 2][i3 - 1] - temp118*u[t1][i1][i2 - 2][i3 + 1] + temp21
*u[t1][i1][i2 - 2][i3 - 3] - temp21*u[t1][i1][i2 - 2][i3 + 3] + temp743 + temp79
3) + temp645*temp854 + temp650*temp855) + temp138*temp466*(temp460*temp650 - tem
p466*temp645) - temp149*temp375*(-temp149*(temp693 + (2.0F/5.0F)*temp707 - 4.0F/
3.0F*temp710 + (1.0F/2.0F)*temp714 - 2.0F/15.0F*temp738 + temp750 + temp776) + t
emp740*(temp129*u[t1][i1][i2 - 1][i3 + 2] - temp132*u[t1][i1][i2 + 1][i3 + 2] -
temp136*u[t1][i1][i2 + 3][i3 + 2] + temp21*u[t1][i1][i2 + 4][i3 + 2] - temp67*u[
t1][i1][i2 - 2][i3 + 2] + temp750 + temp862) + temp749*(temp118*u[t1][i1 - 1][i2
][i3 + 2] - temp118*u[t1][i1 + 1][i2][i3 + 2] + temp21*u[t1][i1 - 3][i2][i3 + 2]
 - temp21*u[t1][i1 + 3][i2][i3 + 2] - temp833 + temp845)) + temp16*temp716 - tem
p168*temp182*(-temp182*(temp601 + (3.0F/4.0F)*temp717 - temp718 + (1.0F/60.0F)*t
```

```
emp757 - 3.0F/20.0F*temp758 + temp759) + temp760*(-temp116*u[t1][i1][i2 - 2][i3
- 3] + temp116*u[t1][i1][i2 + 2][i3 - 3] + temp118*u[t1][i1][i2 - 1][i3 - 3] - t
emp118*u[t1][i1][i2 + 1][i3 - 3] + temp761 - temp767) + temp768*(temp129*u[t1][i
1 - 1][i2][i3 - 3] - temp132*u[t1][i1 + 1][i2][i3 - 3] + temp134*u[t1][i1 + 2][i
2][i3 - 3] - temp136*u[t1][i1 + 3][i2][i3 - 3] + temp21*u[t1][i1 + 4][i2][i3 - 3
] - temp67*u[t1][i1 - 2][i2][i3 - 3] + temp769)) + temp168*temp207*(temp200*temp
206*(-temp116*u[t1][i1][i2 - 2][i3 + 3] + temp116*u[t1][i1][i2 + 2][i3 + 3] + te
mp118*u[t1][i1][i2 - 1][i3 + 3] - temp118*u[t1][i1][i2 + 1][i3 + 3] + temp778 +
temp783) + temp200*temp215*(temp129*u[t1][i1 - 1][i2][i3 + 3] - temp132*u[t1][i1
 + 1][i2][i3 + 3] + temp134*u[t1][i1 + 2][i2][i3 + 3] + temp21*u[t1][i1 + 4][i2]
[i3 + 3] - temp67*u[t1][i1 - 2][i2][i3 + 3] + (7.0F/12.0F)*temp710 + temp788) -
temp207*(temp600 - 3.0F/4.0F*temp714 - temp719 + (3.0F/20.0F)*temp738 + temp775
- temp776)) - temp168*temp272*temp273*(-temp270*(-temp116*u[t1][i1 - 4][i2][i3 -
 2] + temp116*u[t1][i1 - 4][i2][i3 + 2] + temp118*u[t1][i1 - 4][i2][i3 - 1] - te
mp118*u[t1][i1 - 4][i2][i3 + 1] + temp21*u[t1][i1 - 4][i2][i3 - 3] - temp21*u[t1
][i1 - 4][i2][i3 + 3]) + temp272*temp273*temp665 + temp273*temp274*temp664) - te
mp168*temp274*(-temp272*temp664 + temp274*temp665) + temp168*temp287*temp288*(-t
emp285*(temp129*u[t1][i1 - 3][i2][i3 - 1] - temp132*u[t1][i1 - 3][i2][i3 + 1] +
temp134*u[t1][i1 - 3][i2][i3 + 2] - temp136*u[t1][i1 - 3][i2][i3 + 3] + temp21*u
[t1][i1 - 3][i2][i3 + 4] + temp621 - temp67*u[t1][i1 - 3][i2][i3 - 2]) + temp620
*temp831 + temp627*temp830) + temp168*temp295*(-temp287*temp627 + temp295*temp62
0) + temp168*temp432*temp443*(-temp434*(-temp116*u[t1][i1][i2 - 3][i3 - 2] + tem
p116*u[t1][i1][i2 - 3][i3 + 2] + temp118*u[t1][i1][i2 - 3][i3 - 1] - temp118*u[t
1][i1][i2 - 3][i3 + 1] + temp761 - temp778) + temp607*temp852 + temp614*temp853)
 - temp168*temp444*(temp432*temp614 - temp444*temp607) - temp168*temp449*temp456
*(temp449*temp456*temp631 - temp451*(-temp116*u[t1][i1][i2 + 3][i3 - 2] + temp11
6*u[t1][i1][i2 + 3][i3 + 2] + temp118*u[t1][i1][i2 + 3][i3 - 1] - temp118*u[t1][
i1][i2 + 3][i3 + 1] + temp767 + temp783) + temp456*temp457*temp635) + temp168*te
mp457*(temp449*temp635 - temp457*temp631) + temp168*temp485*(temp483*temp487*(-t
emp116*u[t1][i1 - 2][i2][i3 - 4] + temp116*u[t1][i1 + 2][i2][i3 - 4] + temp118*u
[t1][i1 - 1][i2][i3 - 4] - temp118*u[t1][i1 + 1][i2][i3 - 4] + temp21*u[t1][i1 -
```

```
 3][i2][i3 - 4] - temp21*u[t1][i1 + 3][i2][i3 - 4]) + temp483*temp488*(temp129*u
[t1][i1][i2 - 1][i3 - 4] - temp132*u[t1][i1][i2 + 1][i3 - 4] + temp134*u[t1][i1]
[i2 + 2][i3 - 4] - temp136*u[t1][i1][i2 + 3][i3 - 4] + temp21*u[t1][i1][i2 + 4][
i3 - 4] - temp67*u[t1][i1][i2 - 2][i3 - 4] + temp858) - temp485*(temp600 - 4.0F/
3.0F*temp703 + (1.0F/2.0F)*temp704 - 2.0F/15.0F*temp705 - 1.0F/30.0F*temp757 + (
2.0F/5.0F)*temp758 + temp858)) - temp168*temp499*temp501*(-temp497*(temp129*u[t1
][i1 + 3][i2][i3 - 1] - temp132*u[t1][i1 + 3][i2][i3 + 1] + temp134*u[t1][i1 + 3
][i2][i3 + 2] + temp21*u[t1][i1 + 3][i2][i3 + 4] + temp640 - temp67*u[t1][i1 + 3
][i2][i3 - 2] + temp788) + temp499*temp501*temp639 + temp501*temp503*temp643) -
temp168*temp503*(-temp499*temp643 + temp503*temp639) - temp168*temp505*temp510*(
temp505*temp510*temp668 - temp507*(temp129*u[t1][i1][i2 - 4][i3 - 1] - temp132*u
[t1][i1][i2 - 4][i3 + 1] + temp134*u[t1][i1][i2 - 4][i3 + 2] - temp136*u[t1][i1]
[i2 - 4][i3 + 3] + temp21*u[t1][i1][i2 - 4][i3 + 4] + temp667 - temp67*u[t1][i1]
[i2 - 4][i3 - 2]) + temp508*temp510*temp666) + temp168*temp508*(temp505*temp666
- temp508*temp668) - temp182*temp283*(-temp182*(temp671 - 4.0F/3.0F*temp704 + (1
.0F/2.0F)*temp705 + (2.0F/5.0F)*temp717 - 1.0F/30.0F*temp758 + temp769 + temp789
) + temp760*(temp129*u[t1][i1][i2 - 1][i3 - 3] - temp132*u[t1][i1][i2 + 1][i3 -
3] + temp134*u[t1][i1][i2 + 2][i3 - 3] - temp136*u[t1][i1][i2 + 3][i3 - 3] + tem
p21*u[t1][i1][i2 + 4][i3 - 3] - temp67*u[t1][i1][i2 - 2][i3 - 3] + temp769) + te
mp768*(-temp116*u[t1][i1 - 2][i2][i3 - 3] + temp116*u[t1][i1 + 2][i2][i3 - 3] +
temp118*u[t1][i1 - 1][i2][i3 - 3] - temp118*u[t1][i1 + 1][i2][i3 - 3] - temp21*u
[t1][i1 + 3][i2][i3 - 3] + temp824)) + temp226*temp307*(-temp226*(temp678 + (2.0
F/5.0F)*temp703 - 4.0F/3.0F*temp705 - 2.0F/15.0F*temp707 - 1.0F/30.0F*temp717 +
temp720 + temp800) + temp790*(temp129*u[t1][i1][i2 - 1][i3 - 2] - temp132*u[t1][
i1][i2 + 1][i3 - 2] + temp134*u[t1][i1][i2 + 2][i3 - 2] - temp136*u[t1][i1][i2 +
 3][i3 - 2] + temp21*u[t1][i1][i2 + 4][i3 - 2] + temp800 + temp859) + temp799*(t
emp118*u[t1][i1 - 1][i2][i3 - 2] - temp118*u[t1][i1 + 1][i2][i3 - 2] + temp21*u[
t1][i1 - 3][i2][i3 - 2] - temp21*u[t1][i1 + 3][i2][i3 - 2] + temp832 + temp844))
 + temp249*temp350*(-temp249*(temp688 - 1.0F/30.0F*temp705 - 4.0F/3.0F*temp709 +
 (1.0F/2.0F)*temp710 - 2.0F/15.0F*temp714 + temp739 + temp817) + temp807*(temp12
9*u[t1][i1][i2 - 1][i3 + 1] + temp134*u[t1][i1][i2 + 2][i3 + 1] - temp136*u[t1][
```

```
i1][i2 + 3][i3 + 1] + temp21*u[t1][i1][i2 + 4][i3 + 1] - temp67*u[t1][i1][i2 - 2
][i3 + 1] + temp817 + temp861) + temp816*(-temp116*u[t1][i1 - 2][i2][i3 + 1] + t
emp116*u[t1][i1 + 2][i2][i3 + 1] + temp21*u[t1][i1 - 3][i2][i3 + 1] - temp21*u[t
1][i1 + 3][i2][i3 + 1] + temp837 + temp841)) + temp283*temp287*temp288*(-temp285
*(-temp116*u[t1][i1 - 3][i2][i3 - 2] + temp116*u[t1][i1 - 3][i2][i3 + 2] + temp1
18*u[t1][i1 - 3][i2][i3 - 1] - temp118*u[t1][i1 - 3][i2][i3 + 1] - temp21*u[t1][
i1 - 3][i2][i3 + 3] + temp824) + temp670*temp830 + temp672*temp831) + temp283*te
mp295*(-temp287*temp670 + temp295*temp672) + temp283*temp432*temp443*(-temp434*(
temp129*u[t1][i1][i2 - 3][i3 - 1] - temp132*u[t1][i1][i2 - 3][i3 + 1] + temp134*
u[t1][i1][i2 - 3][i3 + 2] - temp136*u[t1][i1][i2 - 3][i3 + 3] + temp21*u[t1][i1]
[i2 - 3][i3 + 4] + temp608 - temp67*u[t1][i1][i2 - 3][i3 - 2]) + temp673*temp853
 + temp674*temp852) - temp283*temp444*(temp432*temp673 - temp444*temp674) - temp
307*temp311*temp312*(-temp309*(temp118*u[t1][i1 - 2][i2][i3 - 1] - temp118*u[t1]
[i1 - 2][i2][i3 + 1] + temp21*u[t1][i1 - 2][i2][i3 - 3] - temp21*u[t1][i1 - 2][i
2][i3 + 3] + temp832 + temp833) + temp695*temp834 + temp696*temp835) - temp307*t
emp315*(-temp311*temp695 + temp315*temp696) - temp307*temp460*temp465*(-temp462*
(temp129*u[t1][i1][i2 - 2][i3 - 1] - temp132*u[t1][i1][i2 - 2][i3 + 1] + temp134
*u[t1][i1][i2 - 2][i3 + 2] - temp136*u[t1][i1][i2 - 2][i3 + 3] + temp21*u[t1][i1
][i2 - 2][i3 + 4] + temp646 + temp859) + temp677*temp855 + temp679*temp854) + te
mp307*temp466*(temp460*temp677 - temp466*temp679) + temp329*temp333*temp334*(-te
mp331*(-temp116*u[t1][i1 - 1][i2][i3 - 2] + temp116*u[t1][i1 - 1][i2][i3 + 2] +
temp21*u[t1][i1 - 1][i2][i3 - 3] - temp21*u[t1][i1 - 1][i2][i3 + 3] + temp836 -
temp837) + temp697*temp838 + temp698*temp839) + temp329*temp337*(-temp333*temp69
7 + temp337*temp698) + temp329*temp400*temp410*(-temp402*(-temp132*u[t1][i1][i2
- 1][i3 + 1] + temp134*u[t1][i1][i2 - 1][i3 + 2] - temp136*u[t1][i1][i2 - 1][i3
+ 3] + temp21*u[t1][i1][i2 - 1][i3 + 4] + temp558 - temp67*u[t1][i1][i2 - 1][i3
- 2] + temp860) + temp682*temp849 + temp684*temp848) - temp329*temp411*(temp400*
temp682 - temp411*temp684) - temp350*temp354*temp355*(-temp352*(-temp116*u[t1][i
1 + 1][i2][i3 - 2] + temp116*u[t1][i1 + 1][i2][i3 + 2] + temp21*u[t1][i1 + 1][i2
][i3 - 3] - temp21*u[t1][i1 + 1][i2][i3 + 3] + temp840 + temp841) + temp699*temp
842 + temp700*temp843) - temp350*temp358*(-temp354*temp699 + temp358*temp700) -
```

```
temp350*temp472*temp476*(-temp474*(temp129*u[t1][i1][i2 + 1][i3 - 1] + temp134*u
[t1][i1][i2 + 1][i3 + 2] - temp136*u[t1][i1][i2 + 1][i3 + 3] + temp21*u[t1][i1][
i2 + 1][i3 + 4] + temp657 - temp67*u[t1][i1][i2 + 1][i3 - 2] + temp861) + temp68
7*temp857 + temp689*temp856) + temp350*temp477*(temp472*temp687 - temp477*temp68
9) + temp375*temp379*temp380*(-temp377*(temp118*u[t1][i1 + 2][i2][i3 - 1] - temp
118*u[t1][i1 + 2][i2][i3 + 1] + temp21*u[t1][i1 + 2][i2][i3 - 3] - temp21*u[t1][
i1 + 2][i2][i3 + 3] - temp844 + temp845) + temp701*temp846 + temp702*temp847) +
temp375*temp383*(-temp379*temp701 + temp383*temp702) + temp375*temp417*temp425*(
-temp419*(temp129*u[t1][i1][i2 + 2][i3 - 1] - temp132*u[t1][i1][i2 + 2][i3 + 1]
- temp136*u[t1][i1][i2 + 2][i3 + 3] + temp21*u[t1][i1][i2 + 2][i3 + 4] + temp584
 - temp67*u[t1][i1][i2 + 2][i3 - 2] + temp862) + temp692*temp851 + temp694*temp8
50) - temp375*temp426*(temp417*temp692 - temp426*temp694) - temp712*temp716;
                  u[t2][i1][i2][i3] = temp4*(temp10*(temp511*delta[i1][i2][i3] +
 temp863*epsilon[i1][i2][i3]) + temp5*u[t1][i1][i2][i3] + temp9*u[t0][i1][i2][i3
]);
                  v[t2][i1][i2][i3] = temp4*(temp10*(temp511 + temp863*delta[i1]
[i2][i3]) + temp5*v[t1][i1][i2][i3] + temp9*v[t0][i1][i2][i3]);
                }
              }
        #pragma omp for schedule(static)
        for (int i1 = 64 - (58 % i1block); i1<64; i1++)
          for (int i2 = 6; i2<64 - (58 % i2block); i2++)
          {
            #pragma omp simd aligned(m, theta, delta, phi, u, damp, v, epsilon:6
4)
            for (int i3 = 6; i3<64; i3++)
            {
              float temp1 = 8.85879567828298e-1F*damp[i1][i2][i3];
              float temp3 = 2.0F*m[i1][i2][i3];
              float temp4 = 1.0F/(temp1 + temp3);
              float temp5 = 4.0F*m[i1][i2][i3];
```

```
                float temp9 = temp1 - temp3;
                float temp10 = 1.56956521739130F;
                float temp12 = 5.00000000000000e-2F;
                float temp13 = 2.91666666666667e-1F*temp12;
                float temp16 = sin(theta[i1][i2][i3]);
                float temp19 = cos(phi[i1][i2][i3]);
                float temp20 = cos(theta[i1][i2][i3]);
                float temp21 = (1.0F/60.0F)*temp12;
                float temp27 = temp12*v[t1][i1][i2][i3 - 2];
                float temp30 = temp12*v[t1][i1][i2][i3 - 1];
                float temp31 = (3.0F/4.0F)*temp30;
                float temp34 = temp12*v[t1][i1][i2][i3 + 1];
                float temp35 = (3.0F/4.0F)*temp34;
                float temp38 = temp12*v[t1][i1][i2][i3 + 2];
                float temp41 = temp21*v[t1][i1][i2][i3 + 3];
                float temp42 = sin(phi[i1][i2][i3]);
                float temp43 = temp16*temp42;
                float temp48 = temp12*v[t1][i1][i2 - 2][i3];
                float temp51 = temp12*v[t1][i1][i2 - 1][i3];
                float temp52 = (3.0F/4.0F)*temp51;
                float temp55 = temp12*v[t1][i1][i2 + 1][i3];
                float temp56 = (3.0F/4.0F)*temp55;
                float temp59 = temp12*v[t1][i1][i2 + 2][i3];
                float temp62 = temp21*v[t1][i1][i2 + 3][i3];
                float temp63 = temp16*temp19;
                float temp65 = temp12*v[t1][i1][i2][i3];
                float temp66 = (7.0F/12.0F)*temp65;
                float temp67 = (1.0F/30.0F)*temp12;
                float temp72 = temp12*v[t1][i1 - 1][i2][i3];
                float temp75 = temp12*v[t1][i1 + 1][i2][i3];
                float temp78 = temp12*v[t1][i1 + 2][i2][i3];
```

```
                float temp81 = temp12*v[t1][i1 + 3][i2][i3];
                float temp84 = temp21*v[t1][i1 + 4][i2][i3];
                float temp87 = temp12*v[t1][i1 - 2][i2][i3];
                float temp88 = (3.0F/4.0F)*temp72;
                float temp89 = (3.0F/4.0F)*temp75;
                float temp90 = temp21*v[t1][i1 + 3][i2][i3];
                float temp91 = temp12*v[t1][i1][i2][i3 + 3];
                float temp94 = temp21*v[t1][i1][i2][i3 + 4];
                float temp95 = temp12*v[t1][i1][i2 + 3][i3];
                float temp98 = temp21*v[t1][i1][i2 + 4][i3];
                float temp99 = 2.91666666666667e-1F*temp12*(temp20*((2.0F/5.0F)*te
mp30 - 4.0F/3.0F*temp34 + (1.0F/2.0F)*temp38 + temp66 - temp67*v[t1][i1][i2][i3
- 2] - 2.0F/15.0F*temp91 + temp94) + temp43*((2.0F/5.0F)*temp51 - 4.0F/3.0F*temp
55 + (1.0F/2.0F)*temp59 + temp66 - temp67*v[t1][i1][i2 - 2][i3] - 2.0F/15.0F*tem
p95 + temp98) + temp63*(temp21*v[t1][i1 - 3][i2][i3] + (3.0F/20.0F)*temp78 - 3.0
F/20.0F*temp87 + temp88 - temp89 - temp90));
                float temp100 = 3.75e-1F*temp12;
                float temp102 = cos(theta[i1][i2][i3 - 1]);
                float temp103 = (3.0F/4.0F)*temp65;
                float temp104 = -temp103;
                float temp107 = temp12*v[t1][i1][i2][i3 - 3];
                float temp108 = (3.0F/4.0F)*temp27;
                float temp109 = (3.0F/20.0F)*temp34;
                float temp110 = temp21*v[t1][i1][i2][i3 + 2];
                float temp112 = sin(phi[i1][i2][i3 - 1]);
                float temp113 = sin(theta[i1][i2][i3 - 1]);
                float temp114 = temp112*temp113;
                float temp116 = (3.0F/20.0F)*temp12;
                float temp118 = (3.0F/4.0F)*temp12;
                float temp120 = temp118*v[t1][i1][i2 - 1][i3 - 1];
                float temp122 = temp118*v[t1][i1][i2 + 1][i3 - 1];
```

```
                float temp125 = cos(phi[i1][i2][i3 - 1]);
                float temp126 = temp113*temp125;
                float temp127 = (7.0F/12.0F)*temp30;
                float temp129 = (2.0F/5.0F)*temp12;
                float temp131 = temp129*v[t1][i1 - 1][i2][i3 - 1];
                float temp132 = (4.0F/3.0F)*temp12;
                float temp134 = (1.0F/2.0F)*temp12;
                float temp136 = (2.0F/15.0F)*temp12;
                float temp138 = 7.5e-2F*temp12;
                float temp140 = cos(theta[i1][i2][i3 + 2]);
                float temp141 = (3.0F/20.0F)*temp65;
                float temp142 = -temp141;
                float temp143 = temp12*v[t1][i1][i2][i3 + 4];
                float temp146 = temp21*v[t1][i1][i2][i3 + 5];
                float temp148 = sin(phi[i1][i2][i3 + 2]);
                float temp149 = sin(theta[i1][i2][i3 + 2]);
                float temp150 = temp148*temp149;
                float temp153 = temp116*v[t1][i1][i2 - 2][i3 + 2];
                float temp157 = temp116*v[t1][i1][i2 + 2][i3 + 2];
                float temp159 = cos(phi[i1][i2][i3 + 2]);
                float temp160 = temp149*temp159;
                float temp161 = (7.0F/12.0F)*temp38;
                float temp166 = temp134*v[t1][i1 + 2][i2][i3 + 2];
                float temp168 = 8.33333333333333e-3F*temp12;
                float temp170 = cos(theta[i1][i2][i3 - 3]);
                float temp171 = (1.0F/60.0F)*temp65;
                float temp172 = -temp171;
                float temp177 = temp12*v[t1][i1][i2][i3 - 5];
                float temp178 = temp12*v[t1][i1][i2][i3 - 4];
                float temp179 = (3.0F/20.0F)*temp30;
                float temp181 = sin(phi[i1][i2][i3 - 3]);
```

```
                float temp182 = sin(theta[i1][i2][i3 - 3]);
                float temp183 = temp181*temp182;
                float temp184 = temp21*v[t1][i1][i2 - 3][i3 - 3];
                float temp190 = temp21*v[t1][i1][i2 + 3][i3 - 3];
                float temp191 = cos(phi[i1][i2][i3 - 3]);
                float temp192 = temp182*temp191;
                float temp193 = (7.0F/12.0F)*temp107;
                float temp200 = cos(theta[i1][i2][i3 + 3]);
                float temp201 = (3.0F/4.0F)*temp38;
                float temp202 = temp12*v[t1][i1][i2][i3 + 5];
                float temp204 = temp21*v[t1][i1][i2][i3 + 6];
                float temp206 = sin(phi[i1][i2][i3 + 3]);
                float temp207 = sin(theta[i1][i2][i3 + 3]);
                float temp209 = temp21*v[t1][i1][i2 - 3][i3 + 3];
                float temp214 = -temp21*v[t1][i1][i2 + 3][i3 + 3];
                float temp215 = cos(phi[i1][i2][i3 + 3]);
                float temp220 = -temp136*v[t1][i1 + 3][i2][i3 + 3];
                float temp222 = cos(theta[i1][i2][i3 - 2]);
                float temp223 = temp21*v[t1][i1][i2][i3 + 1];
                float temp225 = sin(phi[i1][i2][i3 - 2]);
                float temp226 = sin(theta[i1][i2][i3 - 2]);
                float temp227 = temp225*temp226;
                float temp230 = -temp116*v[t1][i1][i2 - 2][i3 - 2];
                float temp234 = temp116*v[t1][i1][i2 + 2][i3 - 2];
                float temp236 = cos(phi[i1][i2][i3 - 2]);
                float temp237 = temp226*temp236;
                float temp238 = (7.0F/12.0F)*temp27;
                float temp240 = -temp67*v[t1][i1 - 2][i2][i3 - 2];
                float temp246 = cos(theta[i1][i2][i3 + 1]);
                float temp248 = sin(phi[i1][i2][i3 + 1]);
                float temp249 = sin(theta[i1][i2][i3 + 1]);
```

```
                float temp250 = temp248*temp249;
                float temp254 = temp118*v[t1][i1][i2 - 1][i3 + 1];
                float temp256 = -temp118*v[t1][i1][i2 + 1][i3 + 1];
                float temp259 = cos(phi[i1][i2][i3 + 1]);
                float temp260 = temp249*temp259;
                float temp261 = (7.0F/12.0F)*temp34;
                float temp265 = -temp132*v[t1][i1 + 1][i2][i3 + 1];
                float temp270 = sin(theta[i1 - 4][i2][i3]);
                float temp272 = cos(phi[i1 - 4][i2][i3]);
                float temp273 = cos(theta[i1 - 4][i2][i3]);
                float temp274 = sin(phi[i1 - 4][i2][i3]);
                float temp279 = temp12*v[t1][i1 - 5][i2][i3];
                float temp281 = temp12*v[t1][i1 - 4][i2][i3];
                float temp282 = temp12*v[t1][i1 - 3][i2][i3];
                float temp283 = 6.66666666666667e-2F*temp12;
                float temp285 = sin(theta[i1 - 3][i2][i3]);
                float temp287 = cos(phi[i1 - 3][i2][i3]);
                float temp288 = cos(theta[i1 - 3][i2][i3]);
                float temp289 = temp21*v[t1][i1 - 3][i2][i3 - 3];
                float temp295 = sin(phi[i1 - 3][i2][i3]);
                float temp296 = temp285*temp295;
                float temp297 = temp21*v[t1][i1 - 3][i2 - 3][i3];
                float temp303 = temp285*temp287;
                float temp304 = -2.0F/15.0F*temp65;
                float temp305 = (7.0F/12.0F)*temp282;
                float temp306 = temp21*v[t1][i1 + 1][i2][i3];
                float temp307 = 2.5e-1F*temp12;
                float temp309 = sin(theta[i1 - 2][i2][i3]);
                float temp311 = cos(phi[i1 - 2][i2][i3]);
                float temp312 = cos(theta[i1 - 2][i2][i3]);
                float temp313 = -temp116*v[t1][i1 - 2][i2][i3 - 2];
```

```
                float temp314 = temp116*v[t1][i1 - 2][i2][i3 + 2];
                float temp315 = sin(phi[i1 - 2][i2][i3]);
                float temp316 = temp309*temp315;
                float temp319 = -temp116*v[t1][i1 - 2][i2 - 2][i3];
                float temp323 = temp116*v[t1][i1 - 2][i2 + 2][i3];
                float temp325 = temp309*temp311;
                float temp326 = (1.0F/2.0F)*temp65;
                float temp327 = (7.0F/12.0F)*temp87;
                float temp328 = temp21*v[t1][i1 + 2][i2][i3];
                float temp329 = 6.66666666666667e-1F*temp12;
                float temp331 = sin(theta[i1 - 1][i2][i3]);
                float temp333 = cos(phi[i1 - 1][i2][i3]);
                float temp334 = cos(theta[i1 - 1][i2][i3]);
                float temp335 = temp118*v[t1][i1 - 1][i2][i3 - 1];
                float temp336 = temp118*v[t1][i1 - 1][i2][i3 + 1];
                float temp337 = sin(phi[i1 - 1][i2][i3]);
                float temp338 = temp331*temp337;
                float temp342 = temp118*v[t1][i1 - 1][i2 - 1][i3];
                float temp344 = temp118*v[t1][i1 - 1][i2 + 1][i3];
                float temp347 = temp331*temp333;
                float temp348 = -4.0F/3.0F*temp65;
                float temp349 = (7.0F/12.0F)*temp72;
                float temp350 = 2.0e-1F*temp12;
                float temp352 = sin(theta[i1 + 1][i2][i3]);
                float temp354 = cos(phi[i1 + 1][i2][i3]);
                float temp355 = cos(theta[i1 + 1][i2][i3]);
                float temp356 = temp118*v[t1][i1 + 1][i2][i3 - 1];
                float temp357 = -temp118*v[t1][i1 + 1][i2][i3 + 1];
                float temp358 = sin(phi[i1 + 1][i2][i3]);
                float temp359 = temp352*temp358;
                float temp363 = temp118*v[t1][i1 + 1][i2 - 1][i3];
```

```
                float temp365 = -temp118*v[t1][i1 + 1][i2 + 1][i3];
                float temp368 = temp352*temp354;
                float temp369 = (2.0F/5.0F)*temp65;
                float temp370 = (7.0F/12.0F)*temp75;
                float temp371 = temp12*v[t1][i1 + 4][i2][i3];
                float temp374 = temp21*v[t1][i1 + 5][i2][i3];
                float temp375 = 1.66666666666667e-2F*temp12;
                float temp377 = sin(theta[i1 + 2][i2][i3]);
                float temp379 = cos(phi[i1 + 2][i2][i3]);
                float temp380 = cos(theta[i1 + 2][i2][i3]);
                float temp381 = temp116*v[t1][i1 + 2][i2][i3 - 2];
                float temp382 = temp116*v[t1][i1 + 2][i2][i3 + 2];
                float temp383 = sin(phi[i1 + 2][i2][i3]);
                float temp384 = temp377*temp383;
                float temp387 = temp116*v[t1][i1 + 2][i2 - 2][i3];
                float temp391 = temp116*v[t1][i1 + 2][i2 + 2][i3];
                float temp393 = temp377*temp379;
                float temp394 = -1.0F/30.0F*temp65;
                float temp395 = (7.0F/12.0F)*temp78;
                float temp396 = temp12*v[t1][i1 + 5][i2][i3];
                float temp398 = temp21*v[t1][i1 + 6][i2][i3];
                float temp400 = sin(phi[i1][i2 - 1][i3]);
                float temp402 = sin(theta[i1][i2 - 1][i3]);
                float temp403 = temp400*temp402;
                float temp406 = temp12*v[t1][i1][i2 - 3][i3];
                float temp407 = (3.0F/4.0F)*temp48;
                float temp408 = (3.0F/20.0F)*temp55;
                float temp409 = temp21*v[t1][i1][i2 + 2][i3];
                float temp410 = cos(theta[i1][i2 - 1][i3]);
                float temp411 = cos(phi[i1][i2 - 1][i3]);
                float temp412 = temp402*temp411;
```

```
                float temp413 = (7.0F/12.0F)*temp51;
                float temp414 = temp129*v[t1][i1 - 1][i2 - 1][i3];
                float temp417 = sin(phi[i1][i2 + 2][i3]);
                float temp419 = sin(theta[i1][i2 + 2][i3]);
                float temp420 = temp417*temp419;
                float temp421 = temp12*v[t1][i1][i2 + 4][i3];
                float temp424 = temp21*v[t1][i1][i2 + 5][i3];
                float temp425 = cos(theta[i1][i2 + 2][i3]);
                float temp426 = cos(phi[i1][i2 + 2][i3]);
                float temp427 = temp419*temp426;
                float temp428 = (7.0F/12.0F)*temp59;
                float temp429 = temp134*v[t1][i1 + 2][i2 + 2][i3];
                float temp432 = sin(phi[i1][i2 - 3][i3]);
                float temp434 = sin(theta[i1][i2 - 3][i3]);
                float temp435 = temp432*temp434;
                float temp440 = temp12*v[t1][i1][i2 - 5][i3];
                float temp441 = temp12*v[t1][i1][i2 - 4][i3];
                float temp442 = (3.0F/20.0F)*temp51;
                float temp443 = cos(theta[i1][i2 - 3][i3]);
                float temp444 = cos(phi[i1][i2 - 3][i3]);
                float temp445 = temp434*temp444;
                float temp446 = (7.0F/12.0F)*temp406;
                float temp449 = sin(phi[i1][i2 + 3][i3]);
                float temp451 = sin(theta[i1][i2 + 3][i3]);
                float temp452 = (3.0F/4.0F)*temp59;
                float temp453 = temp12*v[t1][i1][i2 + 5][i3];
                float temp455 = temp21*v[t1][i1][i2 + 6][i3];
                float temp456 = cos(theta[i1][i2 + 3][i3]);
                float temp457 = cos(phi[i1][i2 + 3][i3]);
                float temp458 = -temp136*v[t1][i1 + 3][i2 + 3][i3];
                float temp460 = sin(phi[i1][i2 - 2][i3]);
```

```
                float temp462 = sin(theta[i1][i2 - 2][i3]);
                float temp463 = temp460*temp462;
                float temp464 = temp21*v[t1][i1][i2 + 1][i3];
                float temp465 = cos(theta[i1][i2 - 2][i3]);
                float temp466 = cos(phi[i1][i2 - 2][i3]);
                float temp467 = temp462*temp466;
                float temp468 = (7.0F/12.0F)*temp48;
                float temp469 = -temp67*v[t1][i1 - 2][i2 - 2][i3];
                float temp472 = sin(phi[i1][i2 + 1][i3]);
                float temp474 = sin(theta[i1][i2 + 1][i3]);
                float temp475 = temp472*temp474;
                float temp476 = cos(theta[i1][i2 + 1][i3]);
                float temp477 = cos(phi[i1][i2 + 1][i3]);
                float temp478 = temp474*temp477;
                float temp479 = (7.0F/12.0F)*temp55;
                float temp480 = -temp132*v[t1][i1 + 1][i2 + 1][i3];
                float temp483 = cos(theta[i1][i2][i3 - 4]);
                float temp484 = (7.0F/12.0F)*temp178;
                float temp485 = sin(theta[i1][i2][i3 - 4]);
                float temp487 = cos(phi[i1][i2][i3 - 4]);
                float temp488 = sin(phi[i1][i2][i3 - 4]);
                float temp489 = -temp67*v[t1][i1][i2 - 2][i3 - 2];
                float temp490 = temp129*v[t1][i1][i2 - 1][i3 - 1];
                float temp491 = -temp132*v[t1][i1][i2 + 1][i3 + 1];
                float temp492 = temp134*v[t1][i1][i2 + 2][i3 + 2];
                float temp493 = (3.0F/4.0F)*temp87;
                float temp494 = (3.0F/20.0F)*temp75;
                float temp495 = (3.0F/20.0F)*temp72;
                float temp497 = sin(theta[i1 + 3][i2][i3]);
                float temp499 = cos(phi[i1 + 3][i2][i3]);
                float temp500 = (3.0F/4.0F)*temp78;
                float temp501 = cos(theta[i1 + 3][i2][i3]);
                float temp502 = (7.0F/12.0F)*temp81;
                float temp503 = sin(phi[i1 + 3][i2][i3]);
                float temp505 = sin(phi[i1][i2 - 4][i3]);
                float temp507 = sin(theta[i1][i2 - 4][i3]);
                float temp508 = cos(phi[i1][i2 - 4][i3]);
                float temp509 = (7.0F/12.0F)*temp441;
                float temp510 = cos(theta[i1][i2 - 4][i3]);
                float temp511 = temp100*temp102*(temp102*(temp104 - 3.0F/20.0F*tem
p107 + temp108 + temp109 - temp110 + temp21*v[t1][i1][i2][i3 - 4]) + temp114*(-t
emp116*v[t1][i1][i2 - 2][i3 - 1] + temp116*v[t1][i1][i2 + 2][i3 - 1] + temp120 -
 temp122 + temp21*v[t1][i1][i2 - 3][i3 - 1] - temp21*v[t1][i1][i2 + 3][i3 - 1])
+ temp126*(temp127 + temp131 - temp132*v[t1][i1 + 1][i2][i3 - 1] + temp134*v[t1]
[i1 + 2][i2][i3 - 1] - temp136*v[t1][i1 + 3][i2][i3 - 1] + temp21*v[t1][i1 + 4][
i2][i3 - 1] - temp67*v[t1][i1 - 2][i2][i3 - 1])) - temp100*temp246*(temp246*(tem
p103 - temp179 - temp201 + temp21*v[t1][i1][i2][i3 - 2] + (3.0F/20.0F)*temp91 -
temp94) + temp250*(-temp116*v[t1][i1][i2 - 2][i3 + 1] + temp116*v[t1][i1][i2 + 2
][i3 + 1] + temp21*v[t1][i1][i2 - 3][i3 + 1] - temp21*v[t1][i1][i2 + 3][i3 + 1]
+ temp254 + temp256) + temp260*(temp129*v[t1][i1 - 1][i2][i3 + 1] + temp134*v[t1
][i1 + 2][i2][i3 + 1] - temp136*v[t1][i1 + 3][i2][i3 + 1] + temp21*v[t1][i1 + 4]
[i2][i3 + 1] + temp261 + temp265 - temp67*v[t1][i1 - 2][i2][i3 + 1])) + temp100*
temp331*temp333*(temp334*(temp131 - temp132*v[t1][i1 - 1][i2][i3 + 1] + temp134*
v[t1][i1 - 1][i2][i3 + 2] - temp136*v[t1][i1 - 1][i2][i3 + 3] + temp21*v[t1][i1
- 1][i2][i3 + 4] + temp349 - temp67*v[t1][i1 - 1][i2][i3 - 2]) + temp338*(-temp1
32*v[t1][i1 - 1][i2 + 1][i3] + temp134*v[t1][i1 - 1][i2 + 2][i3] - temp136*v[t1]
[i1 - 1][i2 + 3][i3] + temp21*v[t1][i1 - 1][i2 + 4][i3] + temp349 + temp414 - te
mp67*v[t1][i1 - 1][i2 - 2][i3]) + temp347*(temp104 + temp21*v[t1][i1 - 4][i2][i3
] - 3.0F/20.0F*temp282 - temp328 + temp493 + temp494)) - temp100*temp352*temp354
*(temp355*(temp129*v[t1][i1 + 1][i2][i3 - 1] + temp134*v[t1][i1 + 1][i2][i3 + 2]
 - temp136*v[t1][i1 + 1][i2][i3 + 3] + temp21*v[t1][i1 + 1][i2][i3 + 4] + temp26
5 + temp370 - temp67*v[t1][i1 + 1][i2][i3 - 2]) + temp359*(temp129*v[t1][i1 + 1]
```

```
[i2 - 1][i3] + temp134*v[t1][i1 + 1][i2 + 2][i3] - temp136*v[t1][i1 + 1][i2 + 3]
[i3] + temp21*v[t1][i1 + 1][i2 + 4][i3] + temp370 + temp480 - temp67*v[t1][i1 +
1][i2 - 2][i3]) + temp368*(temp103 + temp21*v[t1][i1 - 2][i2][i3] - temp495 - te
mp500 + (3.0F/20.0F)*temp81 - temp84)) + temp100*temp400*temp402*(temp403*(temp1
04 + temp21*v[t1][i1][i2 - 4][i3] - 3.0F/20.0F*temp406 + temp407 + temp408 - tem
p409) + temp410*(-temp116*v[t1][i1][i2 - 1][i3 - 2] + temp116*v[t1][i1][i2 - 1][
i3 + 2] + temp120 + temp21*v[t1][i1][i2 - 1][i3 - 3] - temp21*v[t1][i1][i2 - 1][
i3 + 3] - temp254) + temp412*(-temp132*v[t1][i1 + 1][i2 - 1][i3] + temp134*v[t1]
[i1 + 2][i2 - 1][i3] - temp136*v[t1][i1 + 3][i2 - 1][i3] + temp21*v[t1][i1 + 4][
i2 - 1][i3] + temp413 + temp414 - temp67*v[t1][i1 - 2][i2 - 1][i3])) - temp100*t
emp472*temp474*(temp475*(temp103 + temp21*v[t1][i1][i2 - 2][i3] - temp442 - temp
452 + (3.0F/20.0F)*temp95 - temp98) + temp476*(-temp116*v[t1][i1][i2 + 1][i3 - 2
] + temp116*v[t1][i1][i2 + 1][i3 + 2] + temp122 + temp21*v[t1][i1][i2 + 1][i3 -
3] - temp21*v[t1][i1][i2 + 1][i3 + 3] + temp256) + temp478*(temp129*v[t1][i1 - 1
][i2 + 1][i3] + temp134*v[t1][i1 + 2][i2 + 1][i3] - temp136*v[t1][i1 + 3][i2 + 1
][i3] + temp21*v[t1][i1 + 4][i2 + 1][i3] + temp479 + temp480 - temp67*v[t1][i1 -
 2][i2 + 1][i3])) + temp102*temp329*(temp102*(temp127 + (2.0F/5.0F)*temp27 + (1.
0F/2.0F)*temp34 + temp348 - 2.0F/15.0F*temp38 + temp41 - temp67*v[t1][i1][i2][i3
 - 3]) + temp114*(temp127 - temp132*v[t1][i1][i2 + 1][i3 - 1] + temp134*v[t1][i1
][i2 + 2][i3 - 1] - temp136*v[t1][i1][i2 + 3][i3 - 1] + temp21*v[t1][i1][i2 + 4]
[i3 - 1] + temp490 - temp67*v[t1][i1][i2 - 2][i3 - 1]) + temp126*(-temp116*v[t1]
[i1 - 2][i2][i3 - 1] + temp116*v[t1][i1 + 2][i2][i3 - 1] + temp21*v[t1][i1 - 3][
i2][i3 - 1] - temp21*v[t1][i1 + 3][i2][i3 - 1] + temp335 - temp356)) - temp13*te
mp16*temp19*(temp20*(temp21*v[t1][i1][i2][i3 - 3] - 3.0F/20.0F*temp27 + temp31 -
 temp35 + (3.0F/20.0F)*temp38 - temp41) + temp43*(temp21*v[t1][i1][i2 - 3][i3] -
 3.0F/20.0F*temp48 + temp52 - temp56 + (3.0F/20.0F)*temp59 - temp62) + temp63*(t
emp66 - temp67*v[t1][i1 - 2][i2][i3] + (2.0F/5.0F)*temp72 - 4.0F/3.0F*temp75 + (
1.0F/2.0F)*temp78 - 2.0F/15.0F*temp81 + temp84)) + temp138*temp140*(temp140*(tem
p142 + (3.0F/20.0F)*temp143 - temp146 + temp21*v[t1][i1][i2][i3 - 1] + temp35 -
3.0F/4.0F*temp91) + temp150*(temp118*v[t1][i1][i2 - 1][i3 + 2] - temp118*v[t1][i
1][i2 + 1][i3 + 2] - temp153 + temp157 + temp21*v[t1][i1][i2 - 3][i3 + 2] - temp
```

```
21*v[t1][i1][i2 + 3][i3 + 2]) + temp160*(temp129*v[t1][i1 - 1][i2][i3 + 2] - tem
p132*v[t1][i1 + 1][i2][i3 + 2] - temp136*v[t1][i1 + 3][i2][i3 + 2] + temp161 + t
emp166 + temp21*v[t1][i1 + 4][i2][i3 + 2] - temp67*v[t1][i1 - 2][i2][i3 + 2])) -
 temp138*temp222*(temp222*((3.0F/4.0F)*temp107 + temp141 - 3.0F/20.0F*temp178 +
temp21*v[t1][i1][i2][i3 - 5] - temp223 - temp31) + temp227*(temp118*v[t1][i1][i2
 - 1][i3 - 2] - temp118*v[t1][i1][i2 + 1][i3 - 2] + temp21*v[t1][i1][i2 - 3][i3
- 2] - temp21*v[t1][i1][i2 + 3][i3 - 2] + temp230 + temp234) + temp237*(temp129*
v[t1][i1 - 1][i2][i3 - 2] - temp132*v[t1][i1 + 1][i2][i3 - 2] + temp134*v[t1][i1
 + 2][i2][i3 - 2] - temp136*v[t1][i1 + 3][i2][i3 - 2] + temp21*v[t1][i1 + 4][i2]
[i3 - 2] + temp238 + temp240)) - temp138*temp309*temp311*(temp312*(temp129*v[t1]
[i1 - 2][i2][i3 - 1] - temp132*v[t1][i1 - 2][i2][i3 + 1] + temp134*v[t1][i1 - 2]
[i2][i3 + 2] - temp136*v[t1][i1 - 2][i2][i3 + 3] + temp21*v[t1][i1 - 2][i2][i3 +
 4] + temp240 + temp327) + temp316*(temp129*v[t1][i1 - 2][i2 - 1][i3] - temp132*
v[t1][i1 - 2][i2 + 1][i3] + temp134*v[t1][i1 - 2][i2 + 2][i3] - temp136*v[t1][i1
 - 2][i2 + 3][i3] + temp21*v[t1][i1 - 2][i2 + 4][i3] + temp327 + temp469) + temp
325*(temp141 + temp21*v[t1][i1 - 5][i2][i3] - 3.0F/20.0F*temp281 + (3.0F/4.0F)*t
emp282 - temp306 - temp88)) + temp138*temp377*temp379*(temp380*(temp129*v[t1][i1
 + 2][i2][i3 - 1] - temp132*v[t1][i1 + 2][i2][i3 + 1] - temp136*v[t1][i1 + 2][i2
][i3 + 3] + temp166 + temp21*v[t1][i1 + 2][i2][i3 + 4] + temp395 - temp67*v[t1][
i1 + 2][i2][i3 - 2]) + temp384*(temp129*v[t1][i1 + 2][i2 - 1][i3] - temp132*v[t1
][i1 + 2][i2 + 1][i3] - temp136*v[t1][i1 + 2][i2 + 3][i3] + temp21*v[t1][i1 + 2]
[i2 + 4][i3] + temp395 + temp429 - temp67*v[t1][i1 + 2][i2 - 2][i3]) + temp393*(
temp142 + temp21*v[t1][i1 - 1][i2][i3] + (3.0F/20.0F)*temp371 - temp374 - 3.0F/4
.0F*temp81 + temp89)) + temp138*temp417*temp419*(temp420*(temp142 + temp21*v[t1]
[i1][i2 - 1][i3] + (3.0F/20.0F)*temp421 - temp424 + temp56 - 3.0F/4.0F*temp95) +
 temp425*(temp118*v[t1][i1][i2 + 2][i3 - 1] - temp118*v[t1][i1][i2 + 2][i3 + 1]
+ temp157 + temp21*v[t1][i1][i2 + 2][i3 - 3] - temp21*v[t1][i1][i2 + 2][i3 + 3]
- temp234) + temp427*(temp129*v[t1][i1 - 1][i2 + 2][i3] - temp132*v[t1][i1 + 1][
i2 + 2][i3] - temp136*v[t1][i1 + 3][i2 + 2][i3] + temp21*v[t1][i1 + 4][i2 + 2][i
3] + temp428 + temp429 - temp67*v[t1][i1 - 2][i2 + 2][i3])) - temp138*temp460*te
mp462*(temp463*(temp141 + temp21*v[t1][i1][i2 - 5][i3] + (3.0F/4.0F)*temp406 - 3
```

```
.0F/20.0F*temp441 - temp464 - temp52) + temp465*(temp118*v[t1][i1][i2 - 2][i3 -
1] - temp118*v[t1][i1][i2 - 2][i3 + 1] + temp153 + temp21*v[t1][i1][i2 - 2][i3 -
 3] - temp21*v[t1][i1][i2 - 2][i3 + 3] + temp230) + temp467*(temp129*v[t1][i1 -
1][i2 - 2][i3] - temp132*v[t1][i1 + 1][i2 - 2][i3] + temp134*v[t1][i1 + 2][i2 -
2][i3] - temp136*v[t1][i1 + 3][i2 - 2][i3] + temp21*v[t1][i1 + 4][i2 - 2][i3] +
temp468 + temp469)) + temp140*temp375*(temp140*((1.0F/2.0F)*temp143 + temp161 -
2.0F/15.0F*temp202 + temp204 + (2.0F/5.0F)*temp34 + temp394 - 4.0F/3.0F*temp91)
+ temp150*(temp129*v[t1][i1][i2 - 1][i3 + 2] - temp132*v[t1][i1][i2 + 1][i3 + 2]
 - temp136*v[t1][i1][i2 + 3][i3 + 2] + temp161 + temp21*v[t1][i1][i2 + 4][i3 + 2
] + temp492 - temp67*v[t1][i1][i2 - 2][i3 + 2]) + temp160*(temp118*v[t1][i1 - 1]
[i2][i3 + 2] - temp118*v[t1][i1 + 1][i2][i3 + 2] + temp21*v[t1][i1 - 3][i2][i3 +
 2] - temp21*v[t1][i1 + 3][i2][i3 + 2] - temp314 + temp382)) + temp168*temp170*(
temp170*(-temp108 + temp172 - 3.0F/20.0F*temp177 + (3.0F/4.0F)*temp178 + temp179
 + temp21*v[t1][i1][i2][i3 - 6]) + temp183*(-temp116*v[t1][i1][i2 - 2][i3 - 3] +
 temp116*v[t1][i1][i2 + 2][i3 - 3] + temp118*v[t1][i1][i2 - 1][i3 - 3] - temp118
*v[t1][i1][i2 + 1][i3 - 3] + temp184 - temp190) + temp192*(temp129*v[t1][i1 - 1]
[i2][i3 - 3] - temp132*v[t1][i1 + 1][i2][i3 - 3] + temp134*v[t1][i1 + 2][i2][i3
- 3] - temp136*v[t1][i1 + 3][i2][i3 - 3] + temp193 + temp21*v[t1][i1 + 4][i2][i3
 - 3] - temp67*v[t1][i1 - 2][i2][i3 - 3])) - temp168*temp200*(temp200*(-temp109
- 3.0F/4.0F*temp143 + temp171 + temp201 + (3.0F/20.0F)*temp202 - temp204) + temp
206*temp207*(-temp116*v[t1][i1][i2 - 2][i3 + 3] + temp116*v[t1][i1][i2 + 2][i3 +
 3] + temp118*v[t1][i1][i2 - 1][i3 + 3] - temp118*v[t1][i1][i2 + 1][i3 + 3] + te
mp209 + temp214) + temp207*temp215*(temp129*v[t1][i1 - 1][i2][i3 + 3] - temp132*
v[t1][i1 + 1][i2][i3 + 3] + temp134*v[t1][i1 + 2][i2][i3 + 3] + temp21*v[t1][i1
+ 4][i2][i3 + 3] + temp220 - temp67*v[t1][i1 - 2][i2][i3 + 3] + (7.0F/12.0F)*tem
p91)) - temp168*temp270*temp272*(temp270*temp272*(temp171 + (2.0F/5.0F)*temp279
+ (7.0F/12.0F)*temp281 - 4.0F/3.0F*temp282 - temp67*v[t1][i1 - 6][i2][i3] - 2.0F
/15.0F*temp72 + (1.0F/2.0F)*temp87) + temp270*temp274*(-temp116*v[t1][i1 - 4][i2
 - 2][i3] + temp116*v[t1][i1 - 4][i2 + 2][i3] + temp118*v[t1][i1 - 4][i2 - 1][i3
] - temp118*v[t1][i1 - 4][i2 + 1][i3] + temp21*v[t1][i1 - 4][i2 - 3][i3] - temp2
1*v[t1][i1 - 4][i2 + 3][i3]) + temp273*(-temp116*v[t1][i1 - 4][i2][i3 - 2] + tem
```

```
p116*v[t1][i1 - 4][i2][i3 + 2] + temp118*v[t1][i1 - 4][i2][i3 - 1] - temp118*v[t
1][i1 - 4][i2][i3 + 1] + temp21*v[t1][i1 - 4][i2][i3 - 3] - temp21*v[t1][i1 - 4]
[i2][i3 + 3])) + temp168*temp285*temp287*(temp288*(temp129*v[t1][i1 - 3][i2][i3
- 1] - temp132*v[t1][i1 - 3][i2][i3 + 1] + temp134*v[t1][i1 - 3][i2][i3 + 2] - t
emp136*v[t1][i1 - 3][i2][i3 + 3] + temp21*v[t1][i1 - 3][i2][i3 + 4] + temp305 -
temp67*v[t1][i1 - 3][i2][i3 - 2]) + temp296*(temp129*v[t1][i1 - 3][i2 - 1][i3] -
 temp132*v[t1][i1 - 3][i2 + 1][i3] + temp134*v[t1][i1 - 3][i2 + 2][i3] - temp136
*v[t1][i1 - 3][i2 + 3][i3] + temp21*v[t1][i1 - 3][i2 + 4][i3] + temp305 - temp67
*v[t1][i1 - 3][i2 - 2][i3]) + temp303*(temp172 + temp21*v[t1][i1 - 6][i2][i3] -
3.0F/20.0F*temp279 + (3.0F/4.0F)*temp281 - temp493 + temp495)) + temp168*temp432
*temp434*(temp435*(temp172 + temp21*v[t1][i1][i2 - 6][i3] - temp407 - 3.0F/20.0F
*temp440 + (3.0F/4.0F)*temp441 + temp442) + temp443*(-temp116*v[t1][i1][i2 - 3][
i3 - 2] + temp116*v[t1][i1][i2 - 3][i3 + 2] + temp118*v[t1][i1][i2 - 3][i3 - 1]
- temp118*v[t1][i1][i2 - 3][i3 + 1] + temp184 - temp209) + temp445*(temp129*v[t1
][i1 - 1][i2 - 3][i3] - temp132*v[t1][i1 + 1][i2 - 3][i3] + temp134*v[t1][i1 + 2
][i2 - 3][i3] - temp136*v[t1][i1 + 3][i2 - 3][i3] + temp21*v[t1][i1 + 4][i2 - 3]
[i3] + temp446 - temp67*v[t1][i1 - 2][i2 - 3][i3])) - temp168*temp449*temp451*(t
emp449*temp451*(temp171 - temp408 - 3.0F/4.0F*temp421 + temp452 + (3.0F/20.0F)*t
emp453 - temp455) + temp451*temp457*(temp129*v[t1][i1 - 1][i2 + 3][i3] - temp132
*v[t1][i1 + 1][i2 + 3][i3] + temp134*v[t1][i1 + 2][i2 + 3][i3] + temp21*v[t1][i1
 + 4][i2 + 3][i3] + temp458 - temp67*v[t1][i1 - 2][i2 + 3][i3] + (7.0F/12.0F)*te
mp95) + temp456*(-temp116*v[t1][i1][i2 + 3][i3 - 2] + temp116*v[t1][i1][i2 + 3][
i3 + 2] + temp118*v[t1][i1][i2 + 3][i3 - 1] - temp118*v[t1][i1][i2 + 3][i3 + 1]
+ temp190 + temp214)) - temp168*temp483*(temp483*(-4.0F/3.0F*temp107 + temp171 +
 (2.0F/5.0F)*temp177 + (1.0F/2.0F)*temp27 - 2.0F/15.0F*temp30 + temp484 - temp67
*v[t1][i1][i2][i3 - 6]) + temp485*temp487*(-temp116*v[t1][i1 - 2][i2][i3 - 4] +
temp116*v[t1][i1 + 2][i2][i3 - 4] + temp118*v[t1][i1 - 1][i2][i3 - 4] - temp118*
v[t1][i1 + 1][i2][i3 - 4] + temp21*v[t1][i1 - 3][i2][i3 - 4] - temp21*v[t1][i1 +
 3][i2][i3 - 4]) + temp485*temp488*(temp129*v[t1][i1][i2 - 1][i3 - 4] - temp132*
v[t1][i1][i2 + 1][i3 - 4] + temp134*v[t1][i1][i2 + 2][i3 - 4] - temp136*v[t1][i1
][i2 + 3][i3 - 4] + temp21*v[t1][i1][i2 + 4][i3 - 4] + temp484 - temp67*v[t1][i1
```

```
][i2 - 2][i3 - 4])) - temp168*temp497*temp499*(temp497*temp499*(temp171 - 3.0F/4
.0F*temp371 + (3.0F/20.0F)*temp396 - temp398 - temp494 + temp500) + temp497*temp
503*(temp129*v[t1][i1 + 3][i2 - 1][i3] - temp132*v[t1][i1 + 3][i2 + 1][i3] + tem
p134*v[t1][i1 + 3][i2 + 2][i3] + temp21*v[t1][i1 + 3][i2 + 4][i3] + temp458 + te
mp502 - temp67*v[t1][i1 + 3][i2 - 2][i3]) + temp501*(temp129*v[t1][i1 + 3][i2][i
3 - 1] - temp132*v[t1][i1 + 3][i2][i3 + 1] + temp134*v[t1][i1 + 3][i2][i3 + 2] +
 temp21*v[t1][i1 + 3][i2][i3 + 4] + temp220 + temp502 - temp67*v[t1][i1 + 3][i2]
[i3 - 2])) - temp168*temp505*temp507*(temp505*temp507*(temp171 - 4.0F/3.0F*temp4
06 + (2.0F/5.0F)*temp440 + (1.0F/2.0F)*temp48 + temp509 - 2.0F/15.0F*temp51 - te
mp67*v[t1][i1][i2 - 6][i3]) + temp507*temp508*(-temp116*v[t1][i1 - 2][i2 - 4][i3
] + temp116*v[t1][i1 + 2][i2 - 4][i3] + temp118*v[t1][i1 - 1][i2 - 4][i3] - temp
118*v[t1][i1 + 1][i2 - 4][i3] + temp21*v[t1][i1 - 3][i2 - 4][i3] - temp21*v[t1][
i1 + 3][i2 - 4][i3]) + temp510*(temp129*v[t1][i1][i2 - 4][i3 - 1] - temp132*v[t1
][i1][i2 - 4][i3 + 1] + temp134*v[t1][i1][i2 - 4][i3 + 2] - temp136*v[t1][i1][i2
 - 4][i3 + 3] + temp21*v[t1][i1][i2 - 4][i3 + 4] + temp509 - temp67*v[t1][i1][i2
 - 4][i3 - 2])) + temp170*temp283*(temp170*((2.0F/5.0F)*temp178 + temp193 + temp
223 - 4.0F/3.0F*temp27 + (1.0F/2.0F)*temp30 + temp304 - temp67*v[t1][i1][i2][i3
- 5]) + temp183*(temp129*v[t1][i1][i2 - 1][i3 - 3] - temp132*v[t1][i1][i2 + 1][i
3 - 3] + temp134*v[t1][i1][i2 + 2][i3 - 3] - temp136*v[t1][i1][i2 + 3][i3 - 3] +
 temp193 + temp21*v[t1][i1][i2 + 4][i3 - 3] - temp67*v[t1][i1][i2 - 2][i3 - 3])
+ temp192*(-temp116*v[t1][i1 - 2][i2][i3 - 3] + temp116*v[t1][i1 + 2][i2][i3 - 3
] + temp118*v[t1][i1 - 1][i2][i3 - 3] - temp118*v[t1][i1 + 1][i2][i3 - 3] - temp
21*v[t1][i1 + 3][i2][i3 - 3] + temp289)) - temp20*temp99 - temp222*temp307*(temp
222*((2.0F/5.0F)*temp107 + temp110 + temp238 - 4.0F/3.0F*temp30 + temp326 - 2.0F
/15.0F*temp34 - temp67*v[t1][i1][i2][i3 - 4]) + temp227*(temp129*v[t1][i1][i2 -
1][i3 - 2] - temp132*v[t1][i1][i2 + 1][i3 - 2] + temp134*v[t1][i1][i2 + 2][i3 -
2] - temp136*v[t1][i1][i2 + 3][i3 - 2] + temp21*v[t1][i1][i2 + 4][i3 - 2] + temp
238 + temp489) + temp237*(temp118*v[t1][i1 - 1][i2][i3 - 2] - temp118*v[t1][i1 +
 1][i2][i3 - 2] + temp21*v[t1][i1 - 3][i2][i3 - 2] - temp21*v[t1][i1 + 3][i2][i3
 - 2] + temp313 + temp381)) - temp246*temp350*(temp246*(-2.0F/15.0F*temp143 + te
mp146 + temp261 + temp369 - 4.0F/3.0F*temp38 - temp67*v[t1][i1][i2][i3 - 1] + (1
```

```
.0F/2.0F)*temp91) + temp250*(temp129*v[t1][i1][i2 - 1][i3 + 1] + temp134*v[t1][i
1][i2 + 2][i3 + 1] - temp136*v[t1][i1][i2 + 3][i3 + 1] + temp21*v[t1][i1][i2 + 4
][i3 + 1] + temp261 + temp491 - temp67*v[t1][i1][i2 - 2][i3 + 1]) + temp260*(-te
mp116*v[t1][i1 - 2][i2][i3 + 1] + temp116*v[t1][i1 + 2][i2][i3 + 1] + temp21*v[t
1][i1 - 3][i2][i3 + 1] - temp21*v[t1][i1 + 3][i2][i3 + 1] + temp336 + temp357))
+ temp283*temp285*temp287*(temp288*(-temp116*v[t1][i1 - 3][i2][i3 - 2] + temp116
*v[t1][i1 - 3][i2][i3 + 2] + temp118*v[t1][i1 - 3][i2][i3 - 1] - temp118*v[t1][i
1 - 3][i2][i3 + 1] - temp21*v[t1][i1 - 3][i2][i3 + 3] + temp289) + temp296*(-tem
p116*v[t1][i1 - 3][i2 - 2][i3] + temp116*v[t1][i1 - 3][i2 + 2][i3] + temp118*v[t
1][i1 - 3][i2 - 1][i3] - temp118*v[t1][i1 - 3][i2 + 1][i3] - temp21*v[t1][i1 - 3
][i2 + 3][i3] + temp297) + temp303*((2.0F/5.0F)*temp281 + temp304 + temp305 + te
mp306 - temp67*v[t1][i1 - 5][i2][i3] + (1.0F/2.0F)*temp72 - 4.0F/3.0F*temp87)) +
 temp283*temp432*temp434*(temp435*(temp304 + (2.0F/5.0F)*temp441 + temp446 + tem
p464 - 4.0F/3.0F*temp48 + (1.0F/2.0F)*temp51 - temp67*v[t1][i1][i2 - 5][i3]) + t
emp443*(temp129*v[t1][i1][i2 - 3][i3 - 1] - temp132*v[t1][i1][i2 - 3][i3 + 1] +
temp134*v[t1][i1][i2 - 3][i3 + 2] - temp136*v[t1][i1][i2 - 3][i3 + 3] + temp21*v
[t1][i1][i2 - 3][i3 + 4] + temp446 - temp67*v[t1][i1][i2 - 3][i3 - 2]) + temp445
*(-temp116*v[t1][i1 - 2][i2 - 3][i3] + temp116*v[t1][i1 + 2][i2 - 3][i3] + temp1
18*v[t1][i1 - 1][i2 - 3][i3] - temp118*v[t1][i1 + 1][i2 - 3][i3] - temp21*v[t1][
i1 + 3][i2 - 3][i3] + temp297)) - temp307*temp309*temp311*(temp312*(temp118*v[t1
][i1 - 2][i2][i3 - 1] - temp118*v[t1][i1 - 2][i2][i3 + 1] + temp21*v[t1][i1 - 2]
[i2][i3 - 3] - temp21*v[t1][i1 - 2][i2][i3 + 3] + temp313 + temp314) + temp316*(
temp118*v[t1][i1 - 2][i2 - 1][i3] - temp118*v[t1][i1 - 2][i2 + 1][i3] + temp21*v
[t1][i1 - 2][i2 - 3][i3] - temp21*v[t1][i1 - 2][i2 + 3][i3] + temp319 + temp323)
 + temp325*((2.0F/5.0F)*temp282 + temp326 + temp327 + temp328 - temp67*v[t1][i1
- 4][i2][i3] - 4.0F/3.0F*temp72 - 2.0F/15.0F*temp75)) - temp307*temp460*temp462*
(temp463*(temp326 + (2.0F/5.0F)*temp406 + temp409 + temp468 - 4.0F/3.0F*temp51 -
 2.0F/15.0F*temp55 - temp67*v[t1][i1][i2 - 4][i3]) + temp465*(temp129*v[t1][i1][
i2 - 2][i3 - 1] - temp132*v[t1][i1][i2 - 2][i3 + 1] + temp134*v[t1][i1][i2 - 2][
i3 + 2] - temp136*v[t1][i1][i2 - 2][i3 + 3] + temp21*v[t1][i1][i2 - 2][i3 + 4] +
 temp468 + temp489) + temp467*(temp118*v[t1][i1 - 1][i2 - 2][i3] - temp118*v[t1]
```

```
[i1 + 1][i2 - 2][i3] + temp21*v[t1][i1 - 3][i2 - 2][i3] - temp21*v[t1][i1 + 3][i
2 - 2][i3] + temp319 + temp387)) + temp329*temp331*temp333*(temp334*(-temp116*v[
t1][i1 - 1][i2][i3 - 2] + temp116*v[t1][i1 - 1][i2][i3 + 2] + temp21*v[t1][i1 -
1][i2][i3 - 3] - temp21*v[t1][i1 - 1][i2][i3 + 3] + temp335 - temp336) + temp338
*(-temp116*v[t1][i1 - 1][i2 - 2][i3] + temp116*v[t1][i1 - 1][i2 + 2][i3] + temp2
1*v[t1][i1 - 1][i2 - 3][i3] - temp21*v[t1][i1 - 1][i2 + 3][i3] + temp342 - temp3
44) + temp347*(temp348 + temp349 - temp67*v[t1][i1 - 3][i2][i3] + (1.0F/2.0F)*te
mp75 - 2.0F/15.0F*temp78 + (2.0F/5.0F)*temp87 + temp90)) + temp329*temp400*temp4
02*(temp403*(temp348 + temp413 + (2.0F/5.0F)*temp48 + (1.0F/2.0F)*temp55 - 2.0F/
15.0F*temp59 + temp62 - temp67*v[t1][i1][i2 - 3][i3]) + temp410*(-temp132*v[t1][
i1][i2 - 1][i3 + 1] + temp134*v[t1][i1][i2 - 1][i3 + 2] - temp136*v[t1][i1][i2 -
 1][i3 + 3] + temp21*v[t1][i1][i2 - 1][i3 + 4] + temp413 + temp490 - temp67*v[t1
][i1][i2 - 1][i3 - 2]) + temp412*(-temp116*v[t1][i1 - 2][i2 - 1][i3] + temp116*v
[t1][i1 + 2][i2 - 1][i3] + temp21*v[t1][i1 - 3][i2 - 1][i3] - temp21*v[t1][i1 +
3][i2 - 1][i3] + temp342 - temp363)) - temp350*temp352*temp354*(temp355*(-temp11
6*v[t1][i1 + 1][i2][i3 - 2] + temp116*v[t1][i1 + 1][i2][i3 + 2] + temp21*v[t1][i
1 + 1][i2][i3 - 3] - temp21*v[t1][i1 + 1][i2][i3 + 3] + temp356 + temp357) + tem
p359*(-temp116*v[t1][i1 + 1][i2 - 2][i3] + temp116*v[t1][i1 + 1][i2 + 2][i3] + t
emp21*v[t1][i1 + 1][i2 - 3][i3] - temp21*v[t1][i1 + 1][i2 + 3][i3] + temp363 + t
emp365) + temp368*(temp369 + temp370 - 2.0F/15.0F*temp371 + temp374 - temp67*v[t
1][i1 - 1][i2][i3] - 4.0F/3.0F*temp78 + (1.0F/2.0F)*temp81)) - temp350*temp472*t
emp474*(temp475*(temp369 - 2.0F/15.0F*temp421 + temp424 + temp479 - 4.0F/3.0F*te
mp59 - temp67*v[t1][i1][i2 - 1][i3] + (1.0F/2.0F)*temp95) + temp476*(temp129*v[t
1][i1][i2 + 1][i3 - 1] + temp134*v[t1][i1][i2 + 1][i3 + 2] - temp136*v[t1][i1][i
2 + 1][i3 + 3] + temp21*v[t1][i1][i2 + 1][i3 + 4] + temp479 + temp491 - temp67*v
[t1][i1][i2 + 1][i3 - 2]) + temp478*(-temp116*v[t1][i1 - 2][i2 + 1][i3] + temp11
6*v[t1][i1 + 2][i2 + 1][i3] + temp21*v[t1][i1 - 3][i2 + 1][i3] - temp21*v[t1][i1
 + 3][i2 + 1][i3] + temp344 + temp365)) + temp375*temp377*temp379*(temp380*(temp
118*v[t1][i1 + 2][i2][i3 - 1] - temp118*v[t1][i1 + 2][i2][i3 + 1] + temp21*v[t1]
[i1 + 2][i2][i3 - 3] - temp21*v[t1][i1 + 2][i2][i3 + 3] - temp381 + temp382) + t
emp384*(temp118*v[t1][i1 + 2][i2 - 1][i3] - temp118*v[t1][i1 + 2][i2 + 1][i3] +
```

```
temp21*v[t1][i1 + 2][i2 - 3][i3] - temp21*v[t1][i1 + 2][i2 + 3][i3] - temp387 +
temp391) + temp393*((1.0F/2.0F)*temp371 + temp394 + temp395 - 2.0F/15.0F*temp396
 + temp398 + (2.0F/5.0F)*temp75 - 4.0F/3.0F*temp81)) + temp375*temp417*temp419*(
temp420*(temp394 + (1.0F/2.0F)*temp421 + temp428 - 2.0F/15.0F*temp453 + temp455
+ (2.0F/5.0F)*temp55 - 4.0F/3.0F*temp95) + temp425*(temp129*v[t1][i1][i2 + 2][i3
 - 1] - temp132*v[t1][i1][i2 + 2][i3 + 1] - temp136*v[t1][i1][i2 + 2][i3 + 3] +
temp21*v[t1][i1][i2 + 2][i3 + 4] + temp428 + temp492 - temp67*v[t1][i1][i2 + 2][
i3 - 2]) + temp427*(temp118*v[t1][i1 - 1][i2 + 2][i3] - temp118*v[t1][i1 + 1][i2
 + 2][i3] + temp21*v[t1][i1 - 3][i2 + 2][i3] - temp21*v[t1][i1 + 3][i2 + 2][i3]
- temp323 + temp391)) - temp43*temp99;
                float temp514 = temp12*u[t1][i1][i2 - 2][i3];
                float temp516 = temp12*u[t1][i1][i2 - 1][i3];
                float temp517 = (3.0F/4.0F)*temp516;
                float temp519 = temp12*u[t1][i1][i2 + 1][i3];
                float temp520 = (3.0F/4.0F)*temp519;
                float temp522 = temp12*u[t1][i1][i2 + 2][i3];
                float temp524 = temp21*u[t1][i1][i2 + 3][i3];
                float temp525 = temp21*u[t1][i1][i2 - 3][i3] - 3.0F/20.0F*temp514
+ temp517 - temp520 + (3.0F/20.0F)*temp522 - temp524;
                float temp526 = temp12*u[t1][i1][i2][i3];
                float temp527 = (7.0F/12.0F)*temp526;
                float temp530 = temp12*u[t1][i1 - 1][i2][i3];
                float temp532 = temp12*u[t1][i1 + 1][i2][i3];
                float temp534 = temp12*u[t1][i1 + 2][i2][i3];
                float temp536 = temp12*u[t1][i1 + 3][i2][i3];
                float temp538 = temp21*u[t1][i1 + 4][i2][i3];
                float temp539 = temp527 + (2.0F/5.0F)*temp530 - 4.0F/3.0F*temp532
+ (1.0F/2.0F)*temp534 - 2.0F/15.0F*temp536 + temp538 - temp67*u[t1][i1 - 2][i2][
i3];
                float temp541 = temp12*u[t1][i1 - 2][i2][i3];
                float temp542 = (3.0F/4.0F)*temp530;
                float temp543 = (3.0F/4.0F)*temp532;
                float temp544 = temp21*u[t1][i1 + 3][i2][i3];
                float temp545 = temp21*u[t1][i1 - 3][i2][i3] + (3.0F/20.0F)*temp53
4 - 3.0F/20.0F*temp541 + temp542 - temp543 - temp544;
                float temp546 = temp12*u[t1][i1][i2 + 3][i3];
                float temp548 = temp21*u[t1][i1][i2 + 4][i3];
                float temp549 = (2.0F/5.0F)*temp516 - 4.0F/3.0F*temp519 + (1.0F/2.
0F)*temp522 + temp527 - 2.0F/15.0F*temp546 + temp548 - temp67*u[t1][i1][i2 - 2][
i3];
                float temp550 = (3.0F/4.0F)*temp526;
                float temp551 = -temp550;
                float temp553 = temp12*u[t1][i1][i2 - 3][i3];
                float temp554 = (3.0F/4.0F)*temp514;
                float temp555 = (3.0F/20.0F)*temp519;
                float temp556 = temp21*u[t1][i1][i2 + 2][i3];
                float temp557 = temp21*u[t1][i1][i2 - 4][i3] + temp551 - 3.0F/20.0
F*temp553 + temp554 + temp555 - temp556;
                float temp558 = (7.0F/12.0F)*temp516;
                float temp561 = temp129*u[t1][i1 - 1][i2 - 1][i3];
                float temp565 = -temp132*u[t1][i1 + 1][i2 - 1][i3] + temp134*u[t1]
[i1 + 2][i2 - 1][i3] - temp136*u[t1][i1 + 3][i2 - 1][i3] + temp21*u[t1][i1 + 4][
i2 - 1][i3] + temp558 + temp561 - temp67*u[t1][i1 - 2][i2 - 1][i3];
                float temp567 = temp12*u[t1][i1 - 3][i2][i3];
                float temp568 = (3.0F/4.0F)*temp541;
                float temp569 = (3.0F/20.0F)*temp532;
                float temp570 = temp21*u[t1][i1 + 2][i2][i3];
                float temp571 = temp21*u[t1][i1 - 4][i2][i3] + temp551 - 3.0F/20.0
F*temp567 + temp568 + temp569 - temp570;
                float temp572 = (7.0F/12.0F)*temp530;
                float temp577 = -temp132*u[t1][i1 - 1][i2 + 1][i3] + temp134*u[t1]
[i1 - 1][i2 + 2][i3] - temp136*u[t1][i1 - 1][i2 + 3][i3] + temp21*u[t1][i1 - 1][
```

```
i2 + 4][i3] + temp561 + temp572 - temp67*u[t1][i1 - 1][i2 - 2][i3];
                float temp578 = (3.0F/20.0F)*temp526;
                float temp579 = -temp578;
                float temp580 = temp12*u[t1][i1][i2 + 4][i3];
                float temp582 = temp21*u[t1][i1][i2 + 5][i3];
                float temp583 = temp21*u[t1][i1][i2 - 1][i3] + temp520 - 3.0F/4.0F
*temp546 + temp579 + (3.0F/20.0F)*temp580 - temp582;
                float temp584 = (7.0F/12.0F)*temp522;
                float temp588 = temp134*u[t1][i1 + 2][i2 + 2][i3];
                float temp590 = temp129*u[t1][i1 - 1][i2 + 2][i3] - temp132*u[t1][
i1 + 1][i2 + 2][i3] - temp136*u[t1][i1 + 3][i2 + 2][i3] + temp21*u[t1][i1 + 4][i
2 + 2][i3] + temp584 + temp588 - temp67*u[t1][i1 - 2][i2 + 2][i3];
                float temp591 = temp12*u[t1][i1 + 4][i2][i3];
                float temp593 = temp21*u[t1][i1 + 5][i2][i3];
                float temp594 = temp21*u[t1][i1 - 1][i2][i3] - 3.0F/4.0F*temp536 +
 temp543 + temp579 + (3.0F/20.0F)*temp591 - temp593;
                float temp595 = (7.0F/12.0F)*temp534;
                float temp599 = temp129*u[t1][i1 + 2][i2 - 1][i3] - temp132*u[t1][
i1 + 2][i2 + 1][i3] - temp136*u[t1][i1 + 2][i2 + 3][i3] + temp21*u[t1][i1 + 2][i
2 + 4][i3] + temp588 + temp595 - temp67*u[t1][i1 + 2][i2 - 2][i3];
                float temp600 = (1.0F/60.0F)*temp526;
                float temp601 = -temp600;
                float temp604 = temp12*u[t1][i1][i2 - 5][i3];
                float temp605 = temp12*u[t1][i1][i2 - 4][i3];
                float temp606 = (3.0F/20.0F)*temp516;
                float temp607 = temp21*u[t1][i1][i2 - 6][i3] - temp554 + temp601 -
 3.0F/20.0F*temp604 + (3.0F/4.0F)*temp605 + temp606;
                float temp608 = (7.0F/12.0F)*temp553;
                float temp614 = temp129*u[t1][i1 - 1][i2 - 3][i3] - temp132*u[t1][
i1 + 1][i2 - 3][i3] + temp134*u[t1][i1 + 2][i2 - 3][i3] - temp136*u[t1][i1 + 3][
i2 - 3][i3] + temp21*u[t1][i1 + 4][i2 - 3][i3] + temp608 - temp67*u[t1][i1 - 2][
```

```
i2 - 3][i3];
                float temp617 = temp12*u[t1][i1 - 5][i2][i3];
                float temp618 = temp12*u[t1][i1 - 4][i2][i3];
                float temp619 = (3.0F/20.0F)*temp530;
                float temp620 = temp21*u[t1][i1 - 6][i2][i3] - temp568 + temp601 -
 3.0F/20.0F*temp617 + (3.0F/4.0F)*temp618 + temp619;
                float temp621 = (7.0F/12.0F)*temp567;
                float temp627 = temp129*u[t1][i1 - 3][i2 - 1][i3] - temp132*u[t1][
i1 - 3][i2 + 1][i3] + temp134*u[t1][i1 - 3][i2 + 2][i3] - temp136*u[t1][i1 - 3][
i2 + 3][i3] + temp21*u[t1][i1 - 3][i2 + 4][i3] + temp621 - temp67*u[t1][i1 - 3][
i2 - 2][i3];
                float temp628 = (3.0F/4.0F)*temp522;
                float temp629 = temp12*u[t1][i1][i2 + 5][i3];
                float temp630 = temp21*u[t1][i1][i2 + 6][i3];
                float temp631 = -temp555 - 3.0F/4.0F*temp580 + temp600 + temp628 +
 (3.0F/20.0F)*temp629 - temp630;
                float temp634 = -temp136*u[t1][i1 + 3][i2 + 3][i3];
                float temp635 = temp129*u[t1][i1 - 1][i2 + 3][i3] - temp132*u[t1][
i1 + 1][i2 + 3][i3] + temp134*u[t1][i1 + 2][i2 + 3][i3] + temp21*u[t1][i1 + 4][i
2 + 3][i3] + (7.0F/12.0F)*temp546 + temp634 - temp67*u[t1][i1 - 2][i2 + 3][i3];
                float temp636 = (3.0F/4.0F)*temp534;
                float temp637 = temp12*u[t1][i1 + 5][i2][i3];
                float temp638 = temp21*u[t1][i1 + 6][i2][i3];
                float temp639 = -temp569 - 3.0F/4.0F*temp591 + temp600 + temp636 +
 (3.0F/20.0F)*temp637 - temp638;
                float temp640 = (7.0F/12.0F)*temp536;
                float temp643 = temp129*u[t1][i1 + 3][i2 - 1][i3] - temp132*u[t1][
i1 + 3][i2 + 1][i3] + temp134*u[t1][i1 + 3][i2 + 2][i3] + temp21*u[t1][i1 + 3][i
2 + 4][i3] + temp634 + temp640 - temp67*u[t1][i1 + 3][i2 - 2][i3];
                float temp644 = temp21*u[t1][i1][i2 + 1][i3];
                float temp645 = temp21*u[t1][i1][i2 - 5][i3] - temp517 + (3.0F/4.0
```

```
F)*temp553 + temp578 - 3.0F/20.0F*temp605 - temp644;
                float temp646 = (7.0F/12.0F)*temp514;
                float temp648 = -temp67*u[t1][i1 - 2][i2 - 2][i3];
                float temp650 = temp129*u[t1][i1 - 1][i2 - 2][i3] - temp132*u[t1][
i1 + 1][i2 - 2][i3] + temp134*u[t1][i1 + 2][i2 - 2][i3] - temp136*u[t1][i1 + 3][
i2 - 2][i3] + temp21*u[t1][i1 + 4][i2 - 2][i3] + temp646 + temp648;
                float temp651 = temp21*u[t1][i1 + 1][i2][i3];
                float temp652 = temp21*u[t1][i1 - 5][i2][i3] - temp542 + (3.0F/4.0
F)*temp567 + temp578 - 3.0F/20.0F*temp618 - temp651;
                float temp653 = (7.0F/12.0F)*temp541;
                float temp655 = temp129*u[t1][i1 - 2][i2 - 1][i3] - temp132*u[t1][
i1 - 2][i2 + 1][i3] + temp134*u[t1][i1 - 2][i2 + 2][i3] - temp136*u[t1][i1 - 2][
i2 + 3][i3] + temp21*u[t1][i1 - 2][i2 + 4][i3] + temp648 + temp653;
                float temp656 = temp21*u[t1][i1][i2 - 2][i3] + (3.0F/20.0F)*temp54
6 - temp548 + temp550 - temp606 - temp628;
                float temp657 = (7.0F/12.0F)*temp519;
                float temp659 = -temp132*u[t1][i1 + 1][i2 + 1][i3];
                float temp660 = temp129*u[t1][i1 - 1][i2 + 1][i3] + temp134*u[t1][
i1 + 2][i2 + 1][i3] - temp136*u[t1][i1 + 3][i2 + 1][i3] + temp21*u[t1][i1 + 4][i
2 + 1][i3] + temp657 + temp659 - temp67*u[t1][i1 - 2][i2 + 1][i3];
                float temp661 = temp21*u[t1][i1 - 2][i2][i3] + (3.0F/20.0F)*temp53
6 - temp538 + temp550 - temp619 - temp636;
                float temp662 = (7.0F/12.0F)*temp532;
                float temp663 = temp129*u[t1][i1 + 1][i2 - 1][i3] + temp134*u[t1][
i1 + 1][i2 + 2][i3] - temp136*u[t1][i1 + 1][i2 + 3][i3] + temp21*u[t1][i1 + 1][i
2 + 4][i3] + temp659 + temp662 - temp67*u[t1][i1 + 1][i2 - 2][i3];
                float temp664 = -temp116*u[t1][i1 - 4][i2 - 2][i3] + temp116*u[t1]
[i1 - 4][i2 + 2][i3] + temp118*u[t1][i1 - 4][i2 - 1][i3] - temp118*u[t1][i1 - 4]
[i2 + 1][i3] + temp21*u[t1][i1 - 4][i2 - 3][i3] - temp21*u[t1][i1 - 4][i2 + 3][i
3];
                float temp665 = -2.0F/15.0F*temp530 + (1.0F/2.0F)*temp541 - 4.0F/3
```

```
.0F*temp567 + temp600 + (2.0F/5.0F)*temp617 + (7.0F/12.0F)*temp618 - temp67*u[t1
][i1 - 6][i2][i3];
                float temp666 = -temp116*u[t1][i1 - 2][i2 - 4][i3] + temp116*u[t1]
[i1 + 2][i2 - 4][i3] + temp118*u[t1][i1 - 1][i2 - 4][i3] - temp118*u[t1][i1 + 1]
[i2 - 4][i3] + temp21*u[t1][i1 - 3][i2 - 4][i3] - temp21*u[t1][i1 + 3][i2 - 4][i
3];
                float temp667 = (7.0F/12.0F)*temp605;
                float temp668 = (1.0F/2.0F)*temp514 - 2.0F/15.0F*temp516 - 4.0F/3.
0F*temp553 + temp600 + (2.0F/5.0F)*temp604 + temp667 - temp67*u[t1][i1][i2 - 6][
i3];
                float temp669 = temp21*u[t1][i1 - 3][i2 - 3][i3];
                float temp670 = -temp116*u[t1][i1 - 3][i2 - 2][i3] + temp116*u[t1]
[i1 - 3][i2 + 2][i3] + temp118*u[t1][i1 - 3][i2 - 1][i3] - temp118*u[t1][i1 - 3]
[i2 + 1][i3] - temp21*u[t1][i1 - 3][i2 + 3][i3] + temp669;
                float temp671 = -2.0F/15.0F*temp526;
                float temp672 = (1.0F/2.0F)*temp530 - 4.0F/3.0F*temp541 + (2.0F/5.
0F)*temp618 + temp621 + temp651 - temp67*u[t1][i1 - 5][i2][i3] + temp671;
                float temp673 = -temp116*u[t1][i1 - 2][i2 - 3][i3] + temp116*u[t1]
[i1 + 2][i2 - 3][i3] + temp118*u[t1][i1 - 1][i2 - 3][i3] - temp118*u[t1][i1 + 1]
[i2 - 3][i3] - temp21*u[t1][i1 + 3][i2 - 3][i3] + temp669;
                float temp674 = -4.0F/3.0F*temp514 + (1.0F/2.0F)*temp516 + (2.0F/5
.0F)*temp605 + temp608 + temp644 - temp67*u[t1][i1][i2 - 5][i3] + temp671;
                float temp675 = -temp116*u[t1][i1 - 2][i2 - 2][i3];
                float temp676 = temp116*u[t1][i1 + 2][i2 - 2][i3];
                float temp677 = temp118*u[t1][i1 - 1][i2 - 2][i3] - temp118*u[t1][
i1 + 1][i2 - 2][i3] + temp21*u[t1][i1 - 3][i2 - 2][i3] - temp21*u[t1][i1 + 3][i2
 - 2][i3] + temp675 + temp676;
                float temp678 = (1.0F/2.0F)*temp526;
                float temp679 = -4.0F/3.0F*temp516 - 2.0F/15.0F*temp519 + (2.0F/5.
0F)*temp553 + temp556 + temp646 - temp67*u[t1][i1][i2 - 4][i3] + temp678;
                float temp680 = temp118*u[t1][i1 - 1][i2 - 1][i3];
```

```
                float temp681 = temp118*u[t1][i1 + 1][i2 - 1][i3];
                float temp682 = -temp116*u[t1][i1 - 2][i2 - 1][i3] + temp116*u[t1]
[i1 + 2][i2 - 1][i3] + temp21*u[t1][i1 - 3][i2 - 1][i3] - temp21*u[t1][i1 + 3][i
2 - 1][i3] + temp680 - temp681;
                float temp683 = -4.0F/3.0F*temp526;
                float temp684 = (2.0F/5.0F)*temp514 + (1.0F/2.0F)*temp519 - 2.0F/1
5.0F*temp522 + temp524 + temp558 - temp67*u[t1][i1][i2 - 3][i3] + temp683;
                float temp685 = temp118*u[t1][i1 - 1][i2 + 1][i3];
                float temp686 = -temp118*u[t1][i1 + 1][i2 + 1][i3];
                float temp687 = -temp116*u[t1][i1 - 2][i2 + 1][i3] + temp116*u[t1]
[i1 + 2][i2 + 1][i3] + temp21*u[t1][i1 - 3][i2 + 1][i3] - temp21*u[t1][i1 + 3][i
2 + 1][i3] + temp685 + temp686;
                float temp688 = (2.0F/5.0F)*temp526;
                float temp689 = -4.0F/3.0F*temp522 + (1.0F/2.0F)*temp546 - 2.0F/15
.0F*temp580 + temp582 + temp657 - temp67*u[t1][i1][i2 - 1][i3] + temp688;
                float temp690 = temp116*u[t1][i1 - 2][i2 + 2][i3];
                float temp691 = temp116*u[t1][i1 + 2][i2 + 2][i3];
                float temp692 = temp118*u[t1][i1 - 1][i2 + 2][i3] - temp118*u[t1][
i1 + 1][i2 + 2][i3] + temp21*u[t1][i1 - 3][i2 + 2][i3] - temp21*u[t1][i1 + 3][i2
 + 2][i3] - temp690 + temp691;
                float temp693 = -1.0F/30.0F*temp526;
                float temp694 = (2.0F/5.0F)*temp519 - 4.0F/3.0F*temp546 + (1.0F/2.
0F)*temp580 + temp584 - 2.0F/15.0F*temp629 + temp630 + temp693;
                float temp695 = temp118*u[t1][i1 - 2][i2 - 1][i3] - temp118*u[t1][
i1 - 2][i2 + 1][i3] + temp21*u[t1][i1 - 2][i2 - 3][i3] - temp21*u[t1][i1 - 2][i2
 + 3][i3] + temp675 + temp690;
                float temp696 = -4.0F/3.0F*temp530 - 2.0F/15.0F*temp532 + (2.0F/5.
0F)*temp567 + temp570 + temp653 - temp67*u[t1][i1 - 4][i2][i3] + temp678;
                float temp697 = -temp116*u[t1][i1 - 1][i2 - 2][i3] + temp116*u[t1]
[i1 - 1][i2 + 2][i3] + temp21*u[t1][i1 - 1][i2 - 3][i3] - temp21*u[t1][i1 - 1][i
2 + 3][i3] + temp680 - temp685;
```

```
                float temp698 = (1.0F/2.0F)*temp532 - 2.0F/15.0F*temp534 + (2.0F/5
.0F)*temp541 + temp544 + temp572 - temp67*u[t1][i1 - 3][i2][i3] + temp683;
                float temp699 = -temp116*u[t1][i1 + 1][i2 - 2][i3] + temp116*u[t1]
[i1 + 1][i2 + 2][i3] + temp21*u[t1][i1 + 1][i2 - 3][i3] - temp21*u[t1][i1 + 1][i
2 + 3][i3] + temp681 + temp686;
                float temp700 = -4.0F/3.0F*temp534 + (1.0F/2.0F)*temp536 - 2.0F/15
.0F*temp591 + temp593 + temp662 - temp67*u[t1][i1 - 1][i2][i3] + temp688;
                float temp701 = temp118*u[t1][i1 + 2][i2 - 1][i3] - temp118*u[t1][
i1 + 2][i2 + 1][i3] + temp21*u[t1][i1 + 2][i2 - 3][i3] - temp21*u[t1][i1 + 2][i2
 + 3][i3] - temp676 + temp691;
                float temp702 = (2.0F/5.0F)*temp532 - 4.0F/3.0F*temp536 + (1.0F/2.
0F)*temp591 + temp595 - 2.0F/15.0F*temp637 + temp638 + temp693;
                float temp703 = temp12*u[t1][i1][i2][i3 - 3];
                float temp704 = temp12*u[t1][i1][i2][i3 - 2];
                float temp705 = temp12*u[t1][i1][i2][i3 - 1];
                float temp706 = (3.0F/4.0F)*temp705;
                float temp707 = temp12*u[t1][i1][i2][i3 + 1];
                float temp708 = (3.0F/4.0F)*temp707;
                float temp709 = temp12*u[t1][i1][i2][i3 + 2];
                float temp710 = temp12*u[t1][i1][i2][i3 + 3];
                float temp711 = (1.0F/60.0F)*temp710;
                float temp712 = temp20*temp42;
                float temp713 = temp19*temp20;
                float temp714 = temp12*u[t1][i1][i2][i3 + 4];
                float temp715 = (1.0F/60.0F)*temp714;
                float temp716 = 2.91666666666667e-1F*temp12*(-temp16*(temp527 - 1.
0F/30.0F*temp704 + (2.0F/5.0F)*temp705 - 4.0F/3.0F*temp707 + (1.0F/2.0F)*temp709
 - 2.0F/15.0F*temp710 + temp715) + temp545*temp713 + temp549*temp712);
                float temp717 = temp12*u[t1][i1][i2][i3 - 4];
                float temp718 = (3.0F/4.0F)*temp704;
                float temp719 = (3.0F/20.0F)*temp707;
```

```
                float temp720 = (1.0F/60.0F)*temp709;
                float temp721 = temp102*temp112;
                float temp725 = temp118*u[t1][i1][i2 - 1][i3 - 1];
                float temp727 = temp118*u[t1][i1][i2 + 1][i3 - 1];
                float temp730 = temp102*temp125;
                float temp731 = (7.0F/12.0F)*temp705;
                float temp734 = temp129*u[t1][i1 - 1][i2][i3 - 1];
                float temp738 = temp12*u[t1][i1][i2][i3 + 5];
                float temp739 = (1.0F/60.0F)*temp738;
                float temp740 = temp140*temp148;
                float temp743 = temp116*u[t1][i1][i2 - 2][i3 + 2];
                float temp747 = temp116*u[t1][i1][i2 + 2][i3 + 2];
                float temp749 = temp140*temp159;
                float temp750 = (7.0F/12.0F)*temp709;
                float temp755 = temp134*u[t1][i1 + 2][i2][i3 + 2];
                float temp757 = temp12*u[t1][i1][i2][i3 - 6];
                float temp758 = temp12*u[t1][i1][i2][i3 - 5];
                float temp759 = (3.0F/20.0F)*temp705;
                float temp760 = temp170*temp181;
                float temp761 = temp21*u[t1][i1][i2 - 3][i3 - 3];
                float temp767 = temp21*u[t1][i1][i2 + 3][i3 - 3];
                float temp768 = temp170*temp191;
                float temp769 = (7.0F/12.0F)*temp703;
                float temp775 = (3.0F/4.0F)*temp709;
                float temp776 = temp21*u[t1][i1][i2][i3 + 6];
                float temp778 = temp21*u[t1][i1][i2 - 3][i3 + 3];
                float temp783 = -temp21*u[t1][i1][i2 + 3][i3 + 3];
                float temp788 = -temp136*u[t1][i1 + 3][i2][i3 + 3];
                float temp789 = (1.0F/60.0F)*temp707;
                float temp790 = temp222*temp225;
                float temp793 = -temp116*u[t1][i1][i2 - 2][i3 - 2];
```

```
                float temp797 = temp116*u[t1][i1][i2 + 2][i3 - 2];
                float temp799 = temp222*temp236;
                float temp800 = (7.0F/12.0F)*temp704;
                float temp802 = -temp67*u[t1][i1 - 2][i2][i3 - 2];
                float temp807 = temp246*temp248;
                float temp811 = temp118*u[t1][i1][i2 - 1][i3 + 1];
                float temp813 = -temp118*u[t1][i1][i2 + 1][i3 + 1];
                float temp816 = temp246*temp259;
                float temp817 = (7.0F/12.0F)*temp707;
                float temp821 = -temp132*u[t1][i1 + 1][i2][i3 + 1];
                float temp824 = temp21*u[t1][i1 - 3][i2][i3 - 3];
                float temp830 = temp288*temp295;
                float temp831 = temp287*temp288;
                float temp832 = -temp116*u[t1][i1 - 2][i2][i3 - 2];
                float temp833 = temp116*u[t1][i1 - 2][i2][i3 + 2];
                float temp834 = temp312*temp315;
                float temp835 = temp311*temp312;
                float temp836 = temp118*u[t1][i1 - 1][i2][i3 - 1];
                float temp837 = temp118*u[t1][i1 - 1][i2][i3 + 1];
                float temp838 = temp334*temp337;
                float temp839 = temp333*temp334;
                float temp840 = temp118*u[t1][i1 + 1][i2][i3 - 1];
                float temp841 = -temp118*u[t1][i1 + 1][i2][i3 + 1];
                float temp842 = temp355*temp358;
                float temp843 = temp354*temp355;
                float temp844 = temp116*u[t1][i1 + 2][i2][i3 - 2];
                float temp845 = temp116*u[t1][i1 + 2][i2][i3 + 2];
                float temp846 = temp380*temp383;
                float temp847 = temp379*temp380;
                float temp848 = temp400*temp410;
                float temp849 = temp410*temp411;
```

```
              float temp850 = temp417*temp425;
              float temp851 = temp425*temp426;
              float temp852 = temp432*temp443;
              float temp853 = temp443*temp444;
              float temp854 = temp460*temp465;
              float temp855 = temp465*temp466;
              float temp856 = temp472*temp476;
              float temp857 = temp476*temp477;
              float temp858 = (7.0F/12.0F)*temp717;
              float temp859 = -temp67*u[t1][i1][i2 - 2][i3 - 2];
              float temp860 = temp129*u[t1][i1][i2 - 1][i3 - 1];
              float temp861 = -temp132*u[t1][i1][i2 + 1][i3 + 1];
              float temp862 = temp134*u[t1][i1][i2 + 2][i3 + 2];
              float temp863 = -temp100*temp113*(-temp113*(temp551 - 3.0F/20.0F*t
emp703 + (1.0F/60.0F)*temp717 + temp718 + temp719 - temp720) + temp721*(-temp116
*u[t1][i1][i2 - 2][i3 - 1] + temp116*u[t1][i1][i2 + 2][i3 - 1] + temp21*u[t1][i1
][i2 - 3][i3 - 1] - temp21*u[t1][i1][i2 + 3][i3 - 1] + temp725 - temp727) + temp
730*(-temp132*u[t1][i1 + 1][i2][i3 - 1] + temp134*u[t1][i1 + 2][i2][i3 - 1] - te
mp136*u[t1][i1 + 3][i2][i3 - 1] + temp21*u[t1][i1 + 4][i2][i3 - 1] - temp67*u[t1
][i1 - 2][i2][i3 - 1] + temp731 + temp734)) + temp100*temp249*(-temp249*(temp550
 + (1.0F/60.0F)*temp704 + (3.0F/20.0F)*temp710 - temp715 - temp759 - temp775) +
temp807*(-temp116*u[t1][i1][i2 - 2][i3 + 1] + temp116*u[t1][i1][i2 + 2][i3 + 1]
+ temp21*u[t1][i1][i2 - 3][i3 + 1] - temp21*u[t1][i1][i2 + 3][i3 + 1] + temp811
+ temp813) + temp816*(temp129*u[t1][i1 - 1][i2][i3 + 1] + temp134*u[t1][i1 + 2][
i2][i3 + 1] - temp136*u[t1][i1 + 3][i2][i3 + 1] + temp21*u[t1][i1 + 4][i2][i3 +
1] - temp67*u[t1][i1 - 2][i2][i3 + 1] + temp817 + temp821)) + temp100*temp333*te
mp334*(-temp331*(-temp132*u[t1][i1 - 1][i2][i3 + 1] + temp134*u[t1][i1 - 1][i2][
i3 + 2] - temp136*u[t1][i1 - 1][i2][i3 + 3] + temp21*u[t1][i1 - 1][i2][i3 + 4] +
 temp572 - temp67*u[t1][i1 - 1][i2][i3 - 2] + temp734) + temp571*temp839 + temp5
77*temp838) + temp100*temp337*(-temp333*temp577 + temp337*temp571) - temp100*tem
p354*temp355*(-temp352*(temp129*u[t1][i1 + 1][i2][i3 - 1] + temp134*u[t1][i1 + 1
```

```
][i2][i3 + 2] - temp136*u[t1][i1 + 1][i2][i3 + 3] + temp21*u[t1][i1 + 1][i2][i3
+ 4] + temp662 - temp67*u[t1][i1 + 1][i2][i3 - 2] + temp821) + temp661*temp843 +
 temp663*temp842) - temp100*temp358*(-temp354*temp663 + temp358*temp661) + temp1
00*temp400*temp410*(-temp402*(-temp116*u[t1][i1][i2 - 1][i3 - 2] + temp116*u[t1]
[i1][i2 - 1][i3 + 2] + temp21*u[t1][i1][i2 - 1][i3 - 3] - temp21*u[t1][i1][i2 -
1][i3 + 3] + temp725 - temp811) + temp557*temp848 + temp565*temp849) - temp100*t
emp411*(temp400*temp565 - temp411*temp557) - temp100*temp472*temp476*(-temp474*(
-temp116*u[t1][i1][i2 + 1][i3 - 2] + temp116*u[t1][i1][i2 + 1][i3 + 2] + temp21*
u[t1][i1][i2 + 1][i3 - 3] - temp21*u[t1][i1][i2 + 1][i3 + 3] + temp727 + temp813
) + temp656*temp856 + temp660*temp857) + temp100*temp477*(temp472*temp660 - temp
477*temp656) - temp113*temp329*(-temp113*(temp683 - 1.0F/30.0F*temp703 + (2.0F/5
.0F)*temp704 + (1.0F/2.0F)*temp707 - 2.0F/15.0F*temp709 + temp711 + temp731) + t
emp721*(-temp132*u[t1][i1][i2 + 1][i3 - 1] + temp134*u[t1][i1][i2 + 2][i3 - 1] -
 temp136*u[t1][i1][i2 + 3][i3 - 1] + temp21*u[t1][i1][i2 + 4][i3 - 1] - temp67*u
[t1][i1][i2 - 2][i3 - 1] + temp731 + temp860) + temp730*(-temp116*u[t1][i1 - 2][
i2][i3 - 1] + temp116*u[t1][i1 + 2][i2][i3 - 1] + temp21*u[t1][i1 - 3][i2][i3 -
1] - temp21*u[t1][i1 + 3][i2][i3 - 1] + temp836 - temp840)) - temp13*temp19*temp
20*(-temp16*((1.0F/60.0F)*temp703 - 3.0F/20.0F*temp704 + temp706 - temp708 + (3.
0F/20.0F)*temp709 - temp711) + temp525*temp712 + temp539*temp713) + temp13*temp1
9*(-temp19*temp549 + temp42*temp545) - temp13*temp42*(-temp19*temp525 + temp42*t
emp539) - temp138*temp149*(-temp149*(temp579 + (1.0F/60.0F)*temp705 + temp708 -
3.0F/4.0F*temp710 + (3.0F/20.0F)*temp714 - temp739) + temp740*(temp118*u[t1][i1]
[i2 - 1][i3 + 2] - temp118*u[t1][i1][i2 + 1][i3 + 2] + temp21*u[t1][i1][i2 - 3][
i3 + 2] - temp21*u[t1][i1][i2 + 3][i3 + 2] - temp743 + temp747) + temp749*(temp1
29*u[t1][i1 - 1][i2][i3 + 2] - temp132*u[t1][i1 + 1][i2][i3 + 2] - temp136*u[t1]
[i1 + 3][i2][i3 + 2] + temp21*u[t1][i1 + 4][i2][i3 + 2] - temp67*u[t1][i1 - 2][i
2][i3 + 2] + temp750 + temp755)) + temp138*temp226*(-temp226*(temp578 + (3.0F/4.
0F)*temp703 - temp706 - 3.0F/20.0F*temp717 + (1.0F/60.0F)*temp758 - temp789) + t
emp790*(temp118*u[t1][i1][i2 - 1][i3 - 2] - temp118*u[t1][i1][i2 + 1][i3 - 2] +
temp21*u[t1][i1][i2 - 3][i3 - 2] - temp21*u[t1][i1][i2 + 3][i3 - 2] + temp793 +
temp797) + temp799*(temp129*u[t1][i1 - 1][i2][i3 - 2] - temp132*u[t1][i1 + 1][i2
```

```
][i3 - 2] + temp134*u[t1][i1 + 2][i2][i3 - 2] - temp136*u[t1][i1 + 3][i2][i3 - 2
] + temp21*u[t1][i1 + 4][i2][i3 - 2] + temp800 + temp802)) - temp138*temp311*tem
p312*(-temp309*(temp129*u[t1][i1 - 2][i2][i3 - 1] - temp132*u[t1][i1 - 2][i2][i3
 + 1] + temp134*u[t1][i1 - 2][i2][i3 + 2] - temp136*u[t1][i1 - 2][i2][i3 + 3] +
temp21*u[t1][i1 - 2][i2][i3 + 4] + temp653 + temp802) + temp652*temp835 + temp65
5*temp834) - temp138*temp315*(-temp311*temp655 + temp315*temp652) + temp138*temp
379*temp380*(-temp377*(temp129*u[t1][i1 + 2][i2][i3 - 1] - temp132*u[t1][i1 + 2]
[i2][i3 + 1] - temp136*u[t1][i1 + 2][i2][i3 + 3] + temp21*u[t1][i1 + 2][i2][i3 +
 4] + temp595 - temp67*u[t1][i1 + 2][i2][i3 - 2] + temp755) + temp594*temp847 +
temp599*temp846) + temp138*temp383*(-temp379*temp599 + temp383*temp594) + temp13
8*temp417*temp425*(-temp419*(temp118*u[t1][i1][i2 + 2][i3 - 1] - temp118*u[t1][i
1][i2 + 2][i3 + 1] + temp21*u[t1][i1][i2 + 2][i3 - 3] - temp21*u[t1][i1][i2 + 2]
[i3 + 3] + temp747 - temp797) + temp583*temp850 + temp590*temp851) - temp138*tem
p426*(temp417*temp590 - temp426*temp583) - temp138*temp460*temp465*(-temp462*(te
mp118*u[t1][i1][i2 - 2][i3 - 1] - temp118*u[t1][i1][i2 - 2][i3 + 1] + temp21*u[t
1][i1][i2 - 2][i3 - 3] - temp21*u[t1][i1][i2 - 2][i3 + 3] + temp743 + temp793) +
 temp645*temp854 + temp650*temp855) + temp138*temp466*(temp460*temp650 - temp466
*temp645) - temp149*temp375*(-temp149*(temp693 + (2.0F/5.0F)*temp707 - 4.0F/3.0F
*temp710 + (1.0F/2.0F)*temp714 - 2.0F/15.0F*temp738 + temp750 + temp776) + temp7
40*(temp129*u[t1][i1][i2 - 1][i3 + 2] - temp132*u[t1][i1][i2 + 1][i3 + 2] - temp
136*u[t1][i1][i2 + 3][i3 + 2] + temp21*u[t1][i1][i2 + 4][i3 + 2] - temp67*u[t1][
i1][i2 - 2][i3 + 2] + temp750 + temp862) + temp749*(temp118*u[t1][i1 - 1][i2][i3
 + 2] - temp118*u[t1][i1 + 1][i2][i3 + 2] + temp21*u[t1][i1 - 3][i2][i3 + 2] - t
emp21*u[t1][i1 + 3][i2][i3 + 2] - temp833 + temp845)) + temp16*temp716 - temp168
*temp182*(-temp182*(temp601 + (3.0F/4.0F)*temp717 - temp718 + (1.0F/60.0F)*temp7
57 - 3.0F/20.0F*temp758 + temp759) + temp760*(-temp116*u[t1][i1][i2 - 2][i3 - 3]
 + temp116*u[t1][i1][i2 + 2][i3 - 3] + temp118*u[t1][i1][i2 - 1][i3 - 3] - temp1
18*u[t1][i1][i2 + 1][i3 - 3] + temp761 - temp767) + temp768*(temp129*u[t1][i1 -
1][i2][i3 - 3] - temp132*u[t1][i1 + 1][i2][i3 - 3] + temp134*u[t1][i1 + 2][i2][i
3 - 3] - temp136*u[t1][i1 + 3][i2][i3 - 3] + temp21*u[t1][i1 + 4][i2][i3 - 3] -
temp67*u[t1][i1 - 2][i2][i3 - 3] + temp769)) + temp168*temp207*(temp200*temp206*
```

```
(-temp116*u[t1][i1][i2 - 2][i3 + 3] + temp116*u[t1][i1][i2 + 2][i3 + 3] + temp11
8*u[t1][i1][i2 - 1][i3 + 3] - temp118*u[t1][i1][i2 + 1][i3 + 3] + temp778 + temp
783) + temp200*temp215*(temp129*u[t1][i1 - 1][i2][i3 + 3] - temp132*u[t1][i1 + 1
][i2][i3 + 3] + temp134*u[t1][i1 + 2][i2][i3 + 3] + temp21*u[t1][i1 + 4][i2][i3
+ 3] - temp67*u[t1][i1 - 2][i2][i3 + 3] + (7.0F/12.0F)*temp710 + temp788) - temp
207*(temp600 - 3.0F/4.0F*temp714 - temp719 + (3.0F/20.0F)*temp738 + temp775 - te
mp776)) - temp168*temp272*temp273*(-temp270*(-temp116*u[t1][i1 - 4][i2][i3 - 2]
+ temp116*u[t1][i1 - 4][i2][i3 + 2] + temp118*u[t1][i1 - 4][i2][i3 - 1] - temp11
8*u[t1][i1 - 4][i2][i3 + 1] + temp21*u[t1][i1 - 4][i2][i3 - 3] - temp21*u[t1][i1
 - 4][i2][i3 + 3]) + temp272*temp273*temp665 + temp273*temp274*temp664) - temp16
8*temp274*(-temp272*temp664 + temp274*temp665) + temp168*temp287*temp288*(-temp2
85*(temp129*u[t1][i1 - 3][i2][i3 - 1] - temp132*u[t1][i1 - 3][i2][i3 + 1] + temp
134*u[t1][i1 - 3][i2][i3 + 2] - temp136*u[t1][i1 - 3][i2][i3 + 3] + temp21*u[t1]
[i1 - 3][i2][i3 + 4] + temp621 - temp67*u[t1][i1 - 3][i2][i3 - 2]) + temp620*tem
p831 + temp627*temp830) + temp168*temp295*(-temp287*temp627 + temp295*temp620) +
 temp168*temp432*temp443*(-temp434*(-temp116*u[t1][i1][i2 - 3][i3 - 2] + temp116
*u[t1][i1][i2 - 3][i3 + 2] + temp118*u[t1][i1][i2 - 3][i3 - 1] - temp118*u[t1][i
1][i2 - 3][i3 + 1] + temp761 - temp778) + temp607*temp852 + temp614*temp853) - t
emp168*temp444*(temp432*temp614 - temp444*temp607) - temp168*temp449*temp456*(te
mp449*temp456*temp631 - temp451*(-temp116*u[t1][i1][i2 + 3][i3 - 2] + temp116*u[
t1][i1][i2 + 3][i3 + 2] + temp118*u[t1][i1][i2 + 3][i3 - 1] - temp118*u[t1][i1][
i2 + 3][i3 + 1] + temp767 + temp783) + temp456*temp457*temp635) + temp168*temp45
7*(temp449*temp635 - temp457*temp631) + temp168*temp485*(temp483*temp487*(-temp1
16*u[t1][i1 - 2][i2][i3 - 4] + temp116*u[t1][i1 + 2][i2][i3 - 4] + temp118*u[t1]
[i1 - 1][i2][i3 - 4] - temp118*u[t1][i1 + 1][i2][i3 - 4] + temp21*u[t1][i1 - 3][
i2][i3 - 4] - temp21*u[t1][i1 + 3][i2][i3 - 4]) + temp483*temp488*(temp129*u[t1]
[i1][i2 - 1][i3 - 4] - temp132*u[t1][i1][i2 + 1][i3 - 4] + temp134*u[t1][i1][i2
+ 2][i3 - 4] - temp136*u[t1][i1][i2 + 3][i3 - 4] + temp21*u[t1][i1][i2 + 4][i3 -
 4] - temp67*u[t1][i1][i2 - 2][i3 - 4] + temp858) - temp485*(temp600 - 4.0F/3.0F
*temp703 + (1.0F/2.0F)*temp704 - 2.0F/15.0F*temp705 - 1.0F/30.0F*temp757 + (2.0F
/5.0F)*temp758 + temp858)) - temp168*temp499*temp501*(-temp497*(temp129*u[t1][i1
```

```
 + 3][i2][i3 - 1] - temp132*u[t1][i1 + 3][i2][i3 + 1] + temp134*u[t1][i1 + 3][i2
][i3 + 2] + temp21*u[t1][i1 + 3][i2][i3 + 4] + temp640 - temp67*u[t1][i1 + 3][i2
][i3 - 2] + temp788) + temp499*temp501*temp639 + temp501*temp503*temp643) - temp
168*temp503*(-temp499*temp643 + temp503*temp639) - temp168*temp505*temp510*(temp
505*temp510*temp668 - temp507*(temp129*u[t1][i1][i2 - 4][i3 - 1] - temp132*u[t1]
[i1][i2 - 4][i3 + 1] + temp134*u[t1][i1][i2 - 4][i3 + 2] - temp136*u[t1][i1][i2
- 4][i3 + 3] + temp21*u[t1][i1][i2 - 4][i3 + 4] + temp667 - temp67*u[t1][i1][i2
- 4][i3 - 2]) + temp508*temp510*temp666) + temp168*temp508*(temp505*temp666 - te
mp508*temp668) - temp182*temp283*(-temp182*(temp671 - 4.0F/3.0F*temp704 + (1.0F/
2.0F)*temp705 + (2.0F/5.0F)*temp717 - 1.0F/30.0F*temp758 + temp769 + temp789) +
temp760*(temp129*u[t1][i1][i2 - 1][i3 - 3] - temp132*u[t1][i1][i2 + 1][i3 - 3] +
 temp134*u[t1][i1][i2 + 2][i3 - 3] - temp136*u[t1][i1][i2 + 3][i3 - 3] + temp21*
u[t1][i1][i2 + 4][i3 - 3] - temp67*u[t1][i1][i2 - 2][i3 - 3] + temp769) + temp76
8*(-temp116*u[t1][i1 - 2][i2][i3 - 3] + temp116*u[t1][i1 + 2][i2][i3 - 3] + temp
118*u[t1][i1 - 1][i2][i3 - 3] - temp118*u[t1][i1 + 1][i2][i3 - 3] - temp21*u[t1]
[i1 + 3][i2][i3 - 3] + temp824)) + temp226*temp307*(-temp226*(temp678 + (2.0F/5.
0F)*temp703 - 4.0F/3.0F*temp705 - 2.0F/15.0F*temp707 - 1.0F/30.0F*temp717 + temp
720 + temp800) + temp790*(temp129*u[t1][i1][i2 - 1][i3 - 2] - temp132*u[t1][i1][
i2 + 1][i3 - 2] + temp134*u[t1][i1][i2 + 2][i3 - 2] - temp136*u[t1][i1][i2 + 3][
i3 - 2] + temp21*u[t1][i1][i2 + 4][i3 - 2] + temp800 + temp859) + temp799*(temp1
18*u[t1][i1 - 1][i2][i3 - 2] - temp118*u[t1][i1 + 1][i2][i3 - 2] + temp21*u[t1][
i1 - 3][i2][i3 - 2] - temp21*u[t1][i1 + 3][i2][i3 - 2] + temp832 + temp844)) + t
emp249*temp350*(-temp249*(temp688 - 1.0F/30.0F*temp705 - 4.0F/3.0F*temp709 + (1.
0F/2.0F)*temp710 - 2.0F/15.0F*temp714 + temp739 + temp817) + temp807*(temp129*u[
t1][i1][i2 - 1][i3 + 1] + temp134*u[t1][i1][i2 + 2][i3 + 1] - temp136*u[t1][i1][
i2 + 3][i3 + 1] + temp21*u[t1][i1][i2 + 4][i3 + 1] - temp67*u[t1][i1][i2 - 2][i3
 + 1] + temp817 + temp861) + temp816*(-temp116*u[t1][i1 - 2][i2][i3 + 1] + temp1
16*u[t1][i1 + 2][i2][i3 + 1] + temp21*u[t1][i1 - 3][i2][i3 + 1] - temp21*u[t1][i
1 + 3][i2][i3 + 1] + temp837 + temp841)) + temp283*temp287*temp288*(-temp285*(-t
emp116*u[t1][i1 - 3][i2][i3 - 2] + temp116*u[t1][i1 - 3][i2][i3 + 2] + temp118*u
[t1][i1 - 3][i2][i3 - 1] - temp118*u[t1][i1 - 3][i2][i3 + 1] - temp21*u[t1][i1 -
```

```
 3][i2][i3 + 3] + temp824) + temp670*temp830 + temp672*temp831) + temp283*temp29
5*(-temp287*temp670 + temp295*temp672) + temp283*temp432*temp443*(-temp434*(temp
129*u[t1][i1][i2 - 3][i3 - 1] - temp132*u[t1][i1][i2 - 3][i3 + 1] + temp134*u[t1
][i1][i2 - 3][i3 + 2] - temp136*u[t1][i1][i2 - 3][i3 + 3] + temp21*u[t1][i1][i2
- 3][i3 + 4] + temp608 - temp67*u[t1][i1][i2 - 3][i3 - 2]) + temp673*temp853 + t
emp674*temp852) - temp283*temp444*(temp432*temp673 - temp444*temp674) - temp307*
temp311*temp312*(-temp309*(temp118*u[t1][i1 - 2][i2][i3 - 1] - temp118*u[t1][i1
- 2][i2][i3 + 1] + temp21*u[t1][i1 - 2][i2][i3 - 3] - temp21*u[t1][i1 - 2][i2][i
3 + 3] + temp832 + temp833) + temp695*temp834 + temp696*temp835) - temp307*temp3
15*(-temp311*temp695 + temp315*temp696) - temp307*temp460*temp465*(-temp462*(tem
p129*u[t1][i1][i2 - 2][i3 - 1] - temp132*u[t1][i1][i2 - 2][i3 + 1] + temp134*u[t
1][i1][i2 - 2][i3 + 2] - temp136*u[t1][i1][i2 - 2][i3 + 3] + temp21*u[t1][i1][i2
 - 2][i3 + 4] + temp646 + temp859) + temp677*temp855 + temp679*temp854) + temp30
7*temp466*(temp460*temp677 - temp466*temp679) + temp329*temp333*temp334*(-temp33
1*(-temp116*u[t1][i1 - 1][i2][i3 - 2] + temp116*u[t1][i1 - 1][i2][i3 + 2] + temp
21*u[t1][i1 - 1][i2][i3 - 3] - temp21*u[t1][i1 - 1][i2][i3 + 3] + temp836 - temp
837) + temp697*temp838 + temp698*temp839) + temp329*temp337*(-temp333*temp697 +
temp337*temp698) + temp329*temp400*temp410*(-temp402*(-temp132*u[t1][i1][i2 - 1]
[i3 + 1] + temp134*u[t1][i1][i2 - 1][i3 + 2] - temp136*u[t1][i1][i2 - 1][i3 + 3]
 + temp21*u[t1][i1][i2 - 1][i3 + 4] + temp558 - temp67*u[t1][i1][i2 - 1][i3 - 2]
 + temp860) + temp682*temp849 + temp684*temp848) - temp329*temp411*(temp400*temp
682 - temp411*temp684) - temp350*temp354*temp355*(-temp352*(-temp116*u[t1][i1 +
1][i2][i3 - 2] + temp116*u[t1][i1 + 1][i2][i3 + 2] + temp21*u[t1][i1 + 1][i2][i3
 - 3] - temp21*u[t1][i1 + 1][i2][i3 + 3] + temp840 + temp841) + temp699*temp842
+ temp700*temp843) - temp350*temp358*(-temp354*temp699 + temp358*temp700) - temp
350*temp472*temp476*(-temp474*(temp129*u[t1][i1][i2 + 1][i3 - 1] + temp134*u[t1]
[i1][i2 + 1][i3 + 2] - temp136*u[t1][i1][i2 + 1][i3 + 3] + temp21*u[t1][i1][i2 +
 1][i3 + 4] + temp657 - temp67*u[t1][i1][i2 + 1][i3 - 2] + temp861) + temp687*te
mp857 + temp689*temp856) + temp350*temp477*(temp472*temp687 - temp477*temp689) +
 temp375*temp379*temp380*(-temp377*(temp118*u[t1][i1 + 2][i2][i3 - 1] - temp118*
u[t1][i1 + 2][i2][i3 + 1] + temp21*u[t1][i1 + 2][i2][i3 - 3] - temp21*u[t1][i1 +
```

```
 2][i2][i3 + 3] - temp844 + temp845) + temp701*temp846 + temp702*temp847) + temp
375*temp383*(-temp379*temp701 + temp383*temp702) + temp375*temp417*temp425*(-tem
p419*(temp129*u[t1][i1][i2 + 2][i3 - 1] - temp132*u[t1][i1][i2 + 2][i3 + 1] - te
mp136*u[t1][i1][i2 + 2][i3 + 3] + temp21*u[t1][i1][i2 + 2][i3 + 4] + temp584 - t
emp67*u[t1][i1][i2 + 2][i3 - 2] + temp862) + temp692*temp851 + temp694*temp850)
- temp375*temp426*(temp417*temp692 - temp426*temp694) - temp712*temp716;
              u[t2][i1][i2][i3] = temp4*(temp10*(temp511*delta[i1][i2][i3] + tem
p863*epsilon[i1][i2][i3]) + temp5*u[t1][i1][i2][i3] + temp9*u[t0][i1][i2][i3]);
              v[t2][i1][i2][i3] = temp4*(temp10*(temp511 + temp863*delta[i1][i2]
[i3]) + temp5*v[t1][i1][i2][i3] + temp9*v[t0][i1][i2][i3]);
            }
          }
        #pragma omp for schedule(static)
        for (int i1 = 6; i1<64; i1++)
          for (int i2 = 64 - (58 % i2block); i2<64; i2++)
          {
            #pragma omp simd aligned(m, theta, delta, phi, u, damp, v, epsilon:6
4)
            for (int i3 = 6; i3<64; i3++)
            {
              float temp1 = 8.85879567828298e-1F*damp[i1][i2][i3];
              float temp3 = 2.0F*m[i1][i2][i3];
              float temp4 = 1.0F/(temp1 + temp3);
              float temp5 = 4.0F*m[i1][i2][i3];
              float temp9 = temp1 - temp3;
              float temp10 = 1.56956521739130F;
              float temp12 = 5.00000000000000e-2F;
              float temp13 = 2.91666666666667e-1F*temp12;
              float temp16 = sin(theta[i1][i2][i3]);
              float temp19 = cos(phi[i1][i2][i3]);
              float temp20 = cos(theta[i1][i2][i3]);
```

```
              float temp21 = (1.0F/60.0F)*temp12;
              float temp27 = temp12*v[t1][i1][i2][i3 - 2];
              float temp30 = temp12*v[t1][i1][i2][i3 - 1];
              float temp31 = (3.0F/4.0F)*temp30;
              float temp34 = temp12*v[t1][i1][i2][i3 + 1];
              float temp35 = (3.0F/4.0F)*temp34;
              float temp38 = temp12*v[t1][i1][i2][i3 + 2];
              float temp41 = temp21*v[t1][i1][i2][i3 + 3];
              float temp42 = sin(phi[i1][i2][i3]);
              float temp43 = temp16*temp42;
              float temp48 = temp12*v[t1][i1][i2 - 2][i3];
              float temp51 = temp12*v[t1][i1][i2 - 1][i3];
              float temp52 = (3.0F/4.0F)*temp51;
              float temp55 = temp12*v[t1][i1][i2 + 1][i3];
              float temp56 = (3.0F/4.0F)*temp55;
              float temp59 = temp12*v[t1][i1][i2 + 2][i3];
              float temp62 = temp21*v[t1][i1][i2 + 3][i3];
              float temp63 = temp16*temp19;
              float temp65 = temp12*v[t1][i1][i2][i3];
              float temp66 = (7.0F/12.0F)*temp65;
              float temp67 = (1.0F/30.0F)*temp12;
              float temp72 = temp12*v[t1][i1 - 1][i2][i3];
              float temp75 = temp12*v[t1][i1 + 1][i2][i3];
              float temp78 = temp12*v[t1][i1 + 2][i2][i3];
              float temp81 = temp12*v[t1][i1 + 3][i2][i3];
              float temp84 = temp21*v[t1][i1 + 4][i2][i3];
              float temp87 = temp12*v[t1][i1 - 2][i2][i3];
              float temp88 = (3.0F/4.0F)*temp72;
              float temp89 = (3.0F/4.0F)*temp75;
              float temp90 = temp21*v[t1][i1 + 3][i2][i3];
              float temp91 = temp12*v[t1][i1][i2][i3 + 3];
```

```
              float temp94 = temp21*v[t1][i1][i2][i3 + 4];
              float temp95 = temp12*v[t1][i1][i2 + 3][i3];
              float temp98 = temp21*v[t1][i1][i2 + 4][i3];
              float temp99 = 2.91666666666667e-1F*temp12*(temp20*((2.0F/5.0F)*te
mp30 - 4.0F/3.0F*temp34 + (1.0F/2.0F)*temp38 + temp66 - temp67*v[t1][i1][i2][i3
- 2] - 2.0F/15.0F*temp91 + temp94) + temp43*((2.0F/5.0F)*temp51 - 4.0F/3.0F*temp
55 + (1.0F/2.0F)*temp59 + temp66 - temp67*v[t1][i1][i2 - 2][i3] - 2.0F/15.0F*tem
p95 + temp98) + temp63*(temp21*v[t1][i1 - 3][i2][i3] + (3.0F/20.0F)*temp78 - 3.0
F/20.0F*temp87 + temp88 - temp89 - temp90));
              float temp100 = 3.75e-1F*temp12;
              float temp102 = cos(theta[i1][i2][i3 - 1]);
              float temp103 = (3.0F/4.0F)*temp65;
              float temp104 = -temp103;
              float temp107 = temp12*v[t1][i1][i2][i3 - 3];
              float temp108 = (3.0F/4.0F)*temp27;
              float temp109 = (3.0F/20.0F)*temp34;
              float temp110 = temp21*v[t1][i1][i2][i3 + 2];
              float temp112 = sin(phi[i1][i2][i3 - 1]);
              float temp113 = sin(theta[i1][i2][i3 - 1]);
              float temp114 = temp112*temp113;
              float temp116 = (3.0F/20.0F)*temp12;
              float temp118 = (3.0F/4.0F)*temp12;
              float temp120 = temp118*v[t1][i1][i2 - 1][i3 - 1];
              float temp122 = temp118*v[t1][i1][i2 + 1][i3 - 1];
              float temp125 = cos(phi[i1][i2][i3 - 1]);
              float temp126 = temp113*temp125;
              float temp127 = (7.0F/12.0F)*temp30;
              float temp129 = (2.0F/5.0F)*temp12;
              float temp131 = temp129*v[t1][i1 - 1][i2][i3 - 1];
              float temp132 = (4.0F/3.0F)*temp12;
              float temp134 = (1.0F/2.0F)*temp12;
```

```
          float temp136 = (2.0F/15.0F)*temp12;
          float temp138 = 7.5e-2F*temp12;
          float temp140 = cos(theta[i1][i2][i3 + 2]);
          float temp141 = (3.0F/20.0F)*temp65;
          float temp142 = -temp141;
          float temp143 = temp12*v[t1][i1][i2][i3 + 4];
          float temp146 = temp21*v[t1][i1][i2][i3 + 5];
          float temp148 = sin(phi[i1][i2][i3 + 2]);
          float temp149 = sin(theta[i1][i2][i3 + 2]);
          float temp150 = temp148*temp149;
          float temp153 = temp116*v[t1][i1][i2 - 2][i3 + 2];
          float temp157 = temp116*v[t1][i1][i2 + 2][i3 + 2];
          float temp159 = cos(phi[i1][i2][i3 + 2]);
          float temp160 = temp149*temp159;
          float temp161 = (7.0F/12.0F)*temp38;
          float temp166 = temp134*v[t1][i1 + 2][i2][i3 + 2];
          float temp168 = 8.33333333333333e-3F*temp12;
          float temp170 = cos(theta[i1][i2][i3 - 3]);
          float temp171 = (1.0F/60.0F)*temp65;
          float temp172 = -temp171;
          float temp177 = temp12*v[t1][i1][i2][i3 - 5];
          float temp178 = temp12*v[t1][i1][i2][i3 - 4];
          float temp179 = (3.0F/20.0F)*temp30;
          float temp181 = sin(phi[i1][i2][i3 - 3]);
          float temp182 = sin(theta[i1][i2][i3 - 3]);
          float temp183 = temp181*temp182;
          float temp184 = temp21*v[t1][i1][i2 - 3][i3 - 3];
          float temp190 = temp21*v[t1][i1][i2 + 3][i3 - 3];
          float temp191 = cos(phi[i1][i2][i3 - 3]);
          float temp192 = temp182*temp191;
          float temp193 = (7.0F/12.0F)*temp107;
```

```
          float temp200 = cos(theta[i1][i2][i3 + 3]);
          float temp201 = (3.0F/4.0F)*temp38;
          float temp202 = temp12*v[t1][i1][i2][i3 + 5];
          float temp204 = temp21*v[t1][i1][i2][i3 + 6];
          float temp206 = sin(phi[i1][i2][i3 + 3]);
          float temp207 = sin(theta[i1][i2][i3 + 3]);
          float temp209 = temp21*v[t1][i1][i2 - 3][i3 + 3];
          float temp214 = -temp21*v[t1][i1][i2 + 3][i3 + 3];
          float temp215 = cos(phi[i1][i2][i3 + 3]);
          float temp220 = -temp136*v[t1][i1 + 3][i2][i3 + 3];
          float temp222 = cos(theta[i1][i2][i3 - 2]);
          float temp223 = temp21*v[t1][i1][i2][i3 + 1];
          float temp225 = sin(phi[i1][i2][i3 - 2]);
          float temp226 = sin(theta[i1][i2][i3 - 2]);
          float temp227 = temp225*temp226;
          float temp230 = -temp116*v[t1][i1][i2 - 2][i3 - 2];
          float temp234 = temp116*v[t1][i1][i2 + 2][i3 - 2];
          float temp236 = cos(phi[i1][i2][i3 - 2]);
          float temp237 = temp226*temp236;
          float temp238 = (7.0F/12.0F)*temp27;
          float temp240 = -temp67*v[t1][i1 - 2][i2][i3 - 2];
          float temp246 = cos(theta[i1][i2][i3 + 1]);
          float temp248 = sin(phi[i1][i2][i3 + 1]);
          float temp249 = sin(theta[i1][i2][i3 + 1]);
          float temp250 = temp248*temp249;
          float temp254 = temp118*v[t1][i1][i2 - 1][i3 + 1];
          float temp256 = -temp118*v[t1][i1][i2 + 1][i3 + 1];
          float temp259 = cos(phi[i1][i2][i3 + 1]);
          float temp260 = temp249*temp259;
          float temp261 = (7.0F/12.0F)*temp34;
          float temp265 = -temp132*v[t1][i1 + 1][i2][i3 + 1];
```

```
          float temp270 = sin(theta[i1 - 4][i2][i3]);
          float temp272 = cos(phi[i1 - 4][i2][i3]);
          float temp273 = cos(theta[i1 - 4][i2][i3]);
          float temp274 = sin(phi[i1 - 4][i2][i3]);
          float temp279 = temp12*v[t1][i1 - 5][i2][i3];
          float temp281 = temp12*v[t1][i1 - 4][i2][i3];
          float temp282 = temp12*v[t1][i1 - 3][i2][i3];
          float temp283 = 6.66666666666667e-2F*temp12;
          float temp285 = sin(theta[i1 - 3][i2][i3]);
          float temp287 = cos(phi[i1 - 3][i2][i3]);
          float temp288 = cos(theta[i1 - 3][i2][i3]);
          float temp289 = temp21*v[t1][i1 - 3][i2][i3 - 3];
          float temp295 = sin(phi[i1 - 3][i2][i3]);
          float temp296 = temp285*temp295;
          float temp297 = temp21*v[t1][i1 - 3][i2 - 3][i3];
          float temp303 = temp285*temp287;
          float temp304 = -2.0F/15.0F*temp65;
          float temp305 = (7.0F/12.0F)*temp282;
          float temp306 = temp21*v[t1][i1 + 1][i2][i3];
          float temp307 = 2.5e-1F*temp12;
          float temp309 = sin(theta[i1 - 2][i2][i3]);
          float temp311 = cos(phi[i1 - 2][i2][i3]);
          float temp312 = cos(theta[i1 - 2][i2][i3]);
          float temp313 = -temp116*v[t1][i1 - 2][i2][i3 - 2];
          float temp314 = temp116*v[t1][i1 - 2][i2][i3 + 2];
          float temp315 = sin(phi[i1 - 2][i2][i3]);
          float temp316 = temp309*temp315;
          float temp319 = -temp116*v[t1][i1 - 2][i2 - 2][i3];
          float temp323 = temp116*v[t1][i1 - 2][i2 + 2][i3];
          float temp325 = temp309*temp311;
          float temp326 = (1.0F/2.0F)*temp65;
```

```
          float temp327 = (7.0F/12.0F)*temp87;
          float temp328 = temp21*v[t1][i1 + 2][i2][i3];
          float temp329 = 6.66666666666667e-1F*temp12;
          float temp331 = sin(theta[i1 - 1][i2][i3]);
          float temp333 = cos(phi[i1 - 1][i2][i3]);
          float temp334 = cos(theta[i1 - 1][i2][i3]);
          float temp335 = temp118*v[t1][i1 - 1][i2][i3 - 1];
          float temp336 = temp118*v[t1][i1 - 1][i2][i3 + 1];
          float temp337 = sin(phi[i1 - 1][i2][i3]);
          float temp338 = temp331*temp337;
          float temp342 = temp118*v[t1][i1 - 1][i2 - 1][i3];
          float temp344 = temp118*v[t1][i1 - 1][i2 + 1][i3];
          float temp347 = temp331*temp333;
          float temp348 = -4.0F/3.0F*temp65;
          float temp349 = (7.0F/12.0F)*temp72;
          float temp350 = 2.0e-1F*temp12;
          float temp352 = sin(theta[i1 + 1][i2][i3]);
          float temp354 = cos(phi[i1 + 1][i2][i3]);
          float temp355 = cos(theta[i1 + 1][i2][i3]);
          float temp356 = temp118*v[t1][i1 + 1][i2][i3 - 1];
          float temp357 = -temp118*v[t1][i1 + 1][i2][i3 + 1];
          float temp358 = sin(phi[i1 + 1][i2][i3]);
          float temp359 = temp352*temp358;
          float temp363 = temp118*v[t1][i1 + 1][i2 - 1][i3];
          float temp365 = -temp118*v[t1][i1 + 1][i2 + 1][i3];
          float temp368 = temp352*temp354;
          float temp369 = (2.0F/5.0F)*temp65;
          float temp370 = (7.0F/12.0F)*temp75;
          float temp371 = temp12*v[t1][i1 + 4][i2][i3];
          float temp374 = temp21*v[t1][i1 + 5][i2][i3];
          float temp375 = 1.66666666666667e-2F*temp12;
```

```
          float temp377 = sin(theta[i1 + 2][i2][i3]);
          float temp379 = cos(phi[i1 + 2][i2][i3]);
          float temp380 = cos(theta[i1 + 2][i2][i3]);
          float temp381 = temp116*v[t1][i1 + 2][i2][i3 - 2];
          float temp382 = temp116*v[t1][i1 + 2][i2][i3 + 2];
          float temp383 = sin(phi[i1 + 2][i2][i3]);
          float temp384 = temp377*temp383;
          float temp387 = temp116*v[t1][i1 + 2][i2 - 2][i3];
          float temp391 = temp116*v[t1][i1 + 2][i2 + 2][i3];
          float temp393 = temp377*temp379;
          float temp394 = -1.0F/30.0F*temp65;
          float temp395 = (7.0F/12.0F)*temp78;
          float temp396 = temp12*v[t1][i1 + 5][i2][i3];
          float temp398 = temp21*v[t1][i1 + 6][i2][i3];
          float temp400 = sin(phi[i1][i2 - 1][i3]);
          float temp402 = sin(theta[i1][i2 - 1][i3]);
          float temp403 = temp400*temp402;
          float temp406 = temp12*v[t1][i1][i2 - 3][i3];
          float temp407 = (3.0F/4.0F)*temp48;
          float temp408 = (3.0F/20.0F)*temp55;
          float temp409 = temp21*v[t1][i1][i2 + 2][i3];
          float temp410 = cos(theta[i1][i2 - 1][i3]);
          float temp411 = cos(phi[i1][i2 - 1][i3]);
          float temp412 = temp402*temp411;
          float temp413 = (7.0F/12.0F)*temp51;
          float temp414 = temp129*v[t1][i1 - 1][i2 - 1][i3];
          float temp417 = sin(phi[i1][i2 + 2][i3]);
          float temp419 = sin(theta[i1][i2 + 2][i3]);
          float temp420 = temp417*temp419;
          float temp421 = temp12*v[t1][i1][i2 + 4][i3];
          float temp424 = temp21*v[t1][i1][i2 + 5][i3];
```

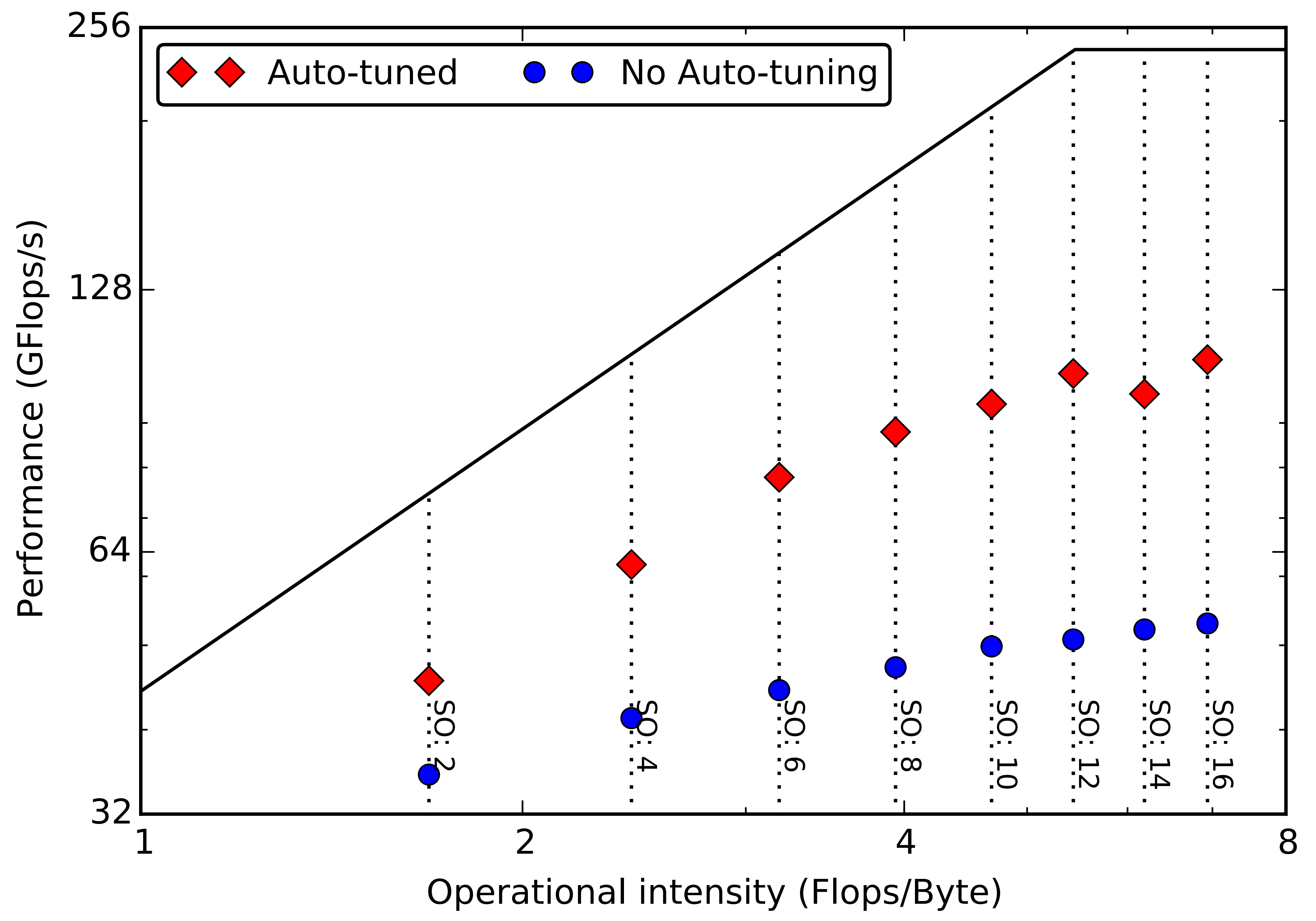
```
          float temp425 = cos(theta[i1][i2 + 2][i3]);
          float temp426 = cos(phi[i1][i2 + 2][i3]);
          float temp427 = temp419*temp426;
          float temp428 = (7.0F/12.0F)*temp59;
          float temp429 = temp134*v[t1][i1 + 2][i2 + 2][i3];
          float temp432 = sin(phi[i1][i2 - 3][i3]);
          float temp434 = sin(theta[i1][i2 - 3][i3]);
          float temp435 = temp432*temp434;
          float temp440 = temp12*v[t1][i1][i2 - 5][i3];
          float temp441 = temp12*v[t1][i1][i2 - 4][i3];
          float temp442 = (3.0F/20.0F)*temp51;
          float temp443 = cos(theta[i1][i2 - 3][i3]);
          float temp444 = cos(phi[i1][i2 - 3][i3]);
          float temp445 = temp434*temp444;
          float temp446 = (7.0F/12.0F)*temp406;
          float temp449 = sin(phi[i1][i2 + 3][i3]);
          float temp451 = sin(theta[i1][i2 + 3][i3]);
          float temp452 = (3.0F/4.0F)*temp59;
          float temp453 = temp12*v[t1][i1][i2 + 5][i3];
          float temp455 = temp21*v[t1][i1][i2 + 6][i3];
          float temp456 = cos(theta[i1][i2 + 3][i3]);
          float temp457 = cos(phi[i1][i2 + 3][i3]);
          float temp458 = -temp136*v[t1][i1 + 3][i2 + 3][i3];
          float temp460 = sin(phi[i1][i2 - 2][i3]);
          float temp462 = sin(theta[i1][i2 - 2][i3]);
          float temp463 = temp460*temp462;
          float temp464 = temp21*v[t1][i1][i2 + 1][i3];
          float temp465 = cos(theta[i1][i2 - 2][i3]);
          float temp466 = cos(phi[i1][i2 - 2][i3]);
          float temp467 = temp462*temp466;
          float temp468 = (7.0F/12.0F)*temp48;
```

```
          float temp469 = -temp67*v[t1][i1 - 2][i2 - 2][i3];
          float temp472 = sin(phi[i1][i2 + 1][i3]);
          float temp474 = sin(theta[i1][i2 + 1][i3]);
          float temp475 = temp472*temp474;
          float temp476 = cos(theta[i1][i2 + 1][i3]);
          float temp477 = cos(phi[i1][i2 + 1][i3]);
          float temp478 = temp474*temp477;
          float temp479 = (7.0F/12.0F)*temp55;
          float temp480 = -temp132*v[t1][i1 + 1][i2 + 1][i3];
          float temp483 = cos(theta[i1][i2][i3 - 4]);
          float temp484 = (7.0F/12.0F)*temp178;
          float temp485 = sin(theta[i1][i2][i3 - 4]);
          float temp487 = cos(phi[i1][i2][i3 - 4]);
          float temp488 = sin(phi[i1][i2][i3 - 4]);
          float temp489 = -temp67*v[t1][i1][i2 - 2][i3 - 2];
          float temp490 = temp129*v[t1][i1][i2 - 1][i3 - 1];
          float temp491 = -temp132*v[t1][i1][i2 + 1][i3 + 1];
          float temp492 = temp134*v[t1][i1][i2 + 2][i3 + 2];
          float temp493 = (3.0F/4.0F)*temp87;
          float temp494 = (3.0F/20.0F)*temp75;
          float temp495 = (3.0F/20.0F)*temp72;
          float temp497 = sin(theta[i1 + 3][i2][i3]);
          float temp499 = cos(phi[i1 + 3][i2][i3]);
          float temp500 = (3.0F/4.0F)*temp78;
          float temp501 = cos(theta[i1 + 3][i2][i3]);
          float temp502 = (7.0F/12.0F)*temp81;
          float temp503 = sin(phi[i1 + 3][i2][i3]);
          float temp505 = sin(phi[i1][i2 - 4][i3]);
          float temp507 = sin(theta[i1][i2 - 4][i3]);
          float temp508 = cos(phi[i1][i2 - 4][i3]);
          float temp509 = (7.0F/12.0F)*temp441;
```

```
          float temp510 = cos(theta[i1][i2 - 4][i3]);
          float temp511 = temp100*temp102*(temp102*(temp104 - 3.0F/20.0F*tem
p107 + temp108 + temp109 - temp110 + temp21*v[t1][i1][i2][i3 - 4]) + temp114*(-t
emp116*v[t1][i1][i2 - 2][i3 - 1] + temp116*v[t1][i1][i2 + 2][i3 - 1] + temp120 -
 temp122 + temp21*v[t1][i1][i2 - 3][i3 - 1] - temp21*v[t1][i1][i2 + 3][i3 - 1])
+ temp126*(temp127 + temp131 - temp132*v[t1][i1 + 1][i2][i3 - 1] + temp134*v[t1]
[i1 + 2][i2][i3 - 1] - temp136*v[t1][i1 + 3][i2][i3 - 1] + temp21*v[t1][i1 + 4][
i2][i3 - 1] - temp67*v[t1][i1 - 2][i2][i3 - 1])) - temp100*temp246*(temp246*(tem
p103 - temp179 - temp201 + temp21*v[t1][i1][i2][i3 - 2] + (3.0F/20.0F)*temp91 -
temp94) + temp250*(-temp116*v[t1][i1][i2 - 2][i3 + 1] + temp116*v[t1][i1][i2 + 2
][i3 + 1] + temp21*v[t1][i1][i2 - 3][i3 + 1] - temp21*v[t1][i1][i2 + 3][i3 + 1]
+ temp254 + temp256) + temp260*(temp129*v[t1][i1 - 1][i2][i3 + 1] + temp134*v[t1
][i1 + 2][i2][i3 + 1] - temp136*v[t1][i1 + 3][i2][i3 + 1] + temp21*v[t1][i1 + 4]
[i2][i3 + 1] + temp261 + temp265 - temp67*v[t1][i1 - 2][i2][i3 + 1])) + temp100*
temp331*temp333*(temp334*(temp131 - temp132*v[t1][i1 - 1][i2][i3 + 1] + temp134*
v[t1][i1 - 1][i2][i3 + 2] - temp136*v[t1][i1 - 1][i2][i3 + 3] + temp21*v[t1][i1
- 1][i2][i3 + 4] + temp349 - temp67*v[t1][i1 - 1][i2][i3 - 2]) + temp338*(-temp1
32*v[t1][i1 - 1][i2 + 1][i3] + temp134*v[t1][i1 - 1][i2 + 2][i3] - temp136*v[t1]
[i1 - 1][i2 + 3][i3] + temp21*v[t1][i1 - 1][i2 + 4][i3] + temp349 + temp414 - te
mp67*v[t1][i1 - 1][i2 - 2][i3]) + temp347*(temp104 + temp21*v[t1][i1 - 4][i2][i3
] - 3.0F/20.0F*temp282 - temp328 + temp493 + temp494)) - temp100*temp352*temp354
*(temp355*(temp129*v[t1][i1 + 1][i2][i3 - 1] + temp134*v[t1][i1 + 1][i2][i3 + 2]
 - temp136*v[t1][i1 + 1][i2][i3 + 3] + temp21*v[t1][i1 + 1][i2][i3 + 4] + temp26
5 + temp370 - temp67*v[t1][i1 + 1][i2][i3 - 2]) + temp359*(temp129*v[t1][i1 + 1]
[i2 - 1][i3] + temp134*v[t1][i1 + 1][i2 + 2][i3] - temp136*v[t1][i1 + 1][i2 + 3]
[i3] + temp21*v[t1][i1 + 1][i2 + 4][i3] + temp370 + temp480 - temp67*v[t1][i1 +
1][i2 - 2][i3]) + temp368*(temp103 + temp21*v[t1][i1 - 2][i2][i3] - temp495 - te
mp500 + (3.0F/20.0F)*temp81 - temp84)) + temp100*temp400*temp402*(temp403*(temp1
04 + temp21*v[t1][i1][i2 - 4][i3] - 3.0F/20.0F*temp406 + temp407 + temp408 - tem
p409) + temp410*(-temp116*v[t1][i1][i2 - 1][i3 - 2] + temp116*v[t1][i1][i2 - 1][
i3 + 2] + temp120 + temp21*v[t1][i1][i2 - 1][i3 - 3] - temp21*v[t1][i1][i2 - 1][
```

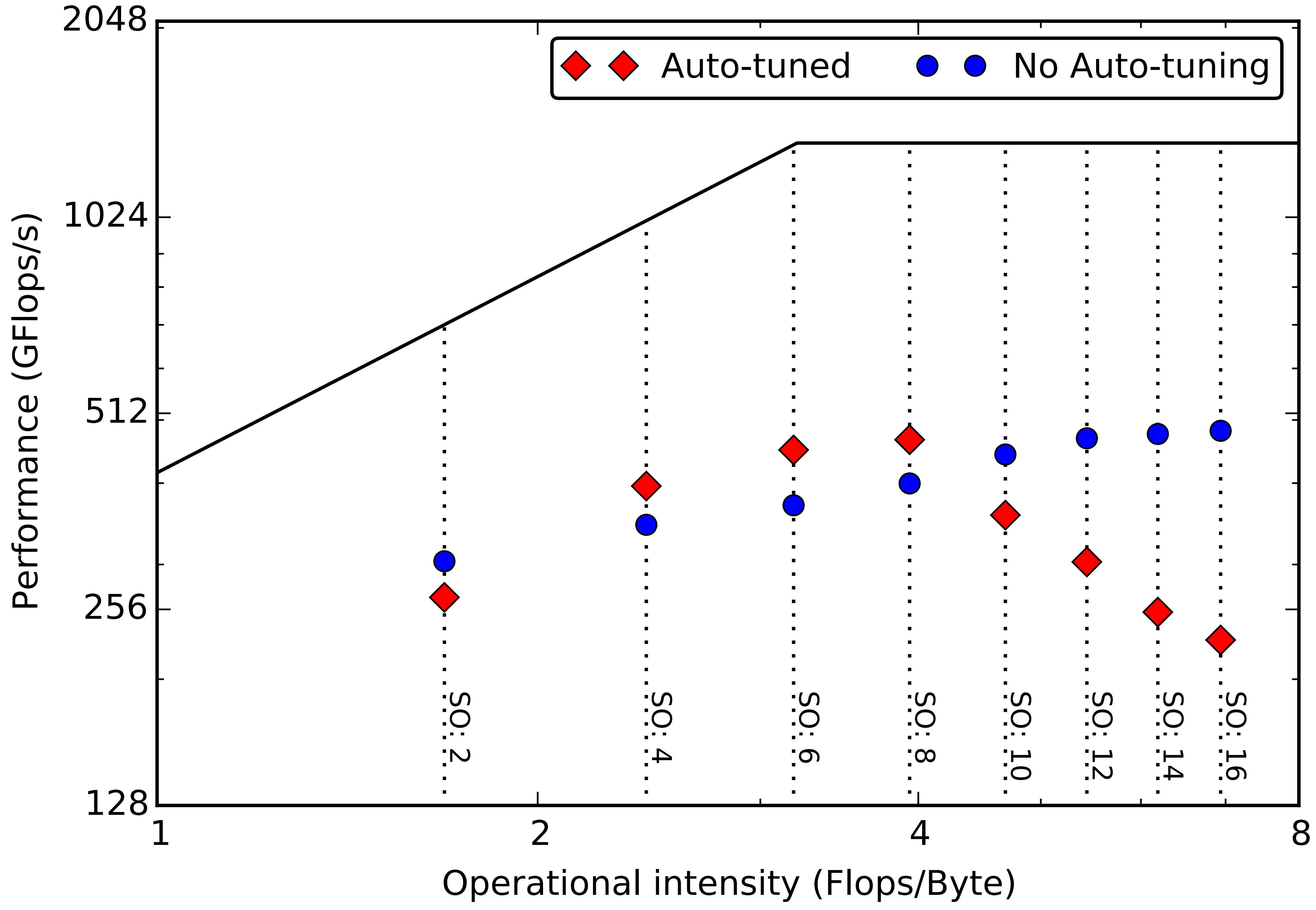
```
i3 + 3] - temp254) + temp412*(-temp132*v[t1][i1 + 1][i2 - 1][i3] + temp134*v[t1]
[i1 + 2][i2 - 1][i3] - temp136*v[t1][i1 + 3][i2 - 1][i3] + temp21*v[t1][i1 + 4][
i2 - 1][i3] + temp413 + temp414 - temp67*v[t1][i1 - 2][i2 - 1][i3])) - temp100*t
emp472*temp474*(temp475*(temp103 + temp21*v[t1][i1][i2 - 2][i3] - temp442 - temp
452 + (3.0F/20.0F)*temp95 - temp98) + temp476*(-temp116*v[t1][i1][i2 + 1][i3 - 2
] + temp116*v[t1][i1][i2 + 1][i3 + 2] + temp122 + temp21*v[t1][i1][i2 + 1][i3 -
3] - temp21*v[t1][i1][i2 + 1][i3 + 3] + temp256) + temp478*(temp129*v[t1][i1 - 1
][i2 + 1][i3] + temp134*v[t1][i1 + 2][i2 + 1][i3] - temp136*v[t1][i1 + 3][i2 + 1
][i3] + temp21*v[t1][i1 + 4][i2 + 1][i3] + temp479 + temp480 - temp67*v[t1][i1 -
 2][i2 + 1][i3])) + temp102*temp329*(temp102*(temp127 + (2.0F/5.0F)*temp27 + (1.
0F/2.0F)*temp34 + temp348 - 2.0F/15.0F*temp38 + temp41 - temp67*v[t1][i1][i2][i3
 - 3]) + temp114*(temp127 - temp132*v[t1][i1][i2 + 1][i3 - 1] + temp134*v[t1][i1
][i2 + 2][i3 - 1] - temp136*v[t1][i1][i2 + 3][i3 - 1] + temp21*v[t1][i1][i2 + 4]
[i3 - 1] + temp490 - temp67*v[t1][i1][i2 - 2][i3 - 1]) + temp126*(-temp116*v[t1]
[i1 - 2][i2][i3 - 1] + temp116*v[t1][i1 + 2][i2][i3 - 1] + temp21*v[t1][i1 - 3][
i2][i3 - 1] - temp21*v[t1][i1 + 3][i2][i3 - 1] + temp335 - temp356)) - temp13*te
mp16*temp19*(temp20*(temp21*v[t1][i1][i2][i3 - 3] - 3.0F/20.0F*temp27 + temp31 -
 temp35 + (3.0F/20.0F)*temp38 - temp41) + temp43*(temp21*v[t1][i1][i2 - 3][i3] -
 3.0F/20.0F*temp48 + temp52 - temp56 + (3.0F/20.0F)*temp59 - temp62) + temp63*(t
emp66 - temp67*v[t1][i1 - 2][i2][i3] + (2.0F/5.0F)*temp72 - 4.0F/3.0F*temp75 + (
1.0F/2.0F)*temp78 - 2.0F/15.0F*temp81 + temp84)) + temp138*temp140*(temp140*(tem
p142 + (3.0F/20.0F)*temp143 - temp146 + temp21*v[t1][i1][i2][i3 - 1] + temp35 -
3.0F/4.0F*temp91) + temp150*(temp118*v[t1][i1][i2 - 1][i3 + 2] - temp118*v[t1][i
1][i2 + 1][i3 + 2] - temp153 + temp157 + temp21*v[t1][i1][i2 - 3][i3 + 2] - temp
21*v[t1][i1][i2 + 3][i3 + 2]) + temp160*(temp129*v[t1][i1 - 1][i2][i3 + 2] - tem
p132*v[t1][i1 + 1][i2][i3 + 2] - temp136*v[t1][i1 + 3][i2][i3 + 2] + temp161 + t
emp166 + temp21*v[t1][i1 + 4][i2][i3 + 2] - temp67*v[t1][i1 - 2][i2][i3 + 2])) -
 temp138*temp222*(temp222*((3.0F/4.0F)*temp107 + temp141 - 3.0F/20.0F*temp178 +
temp21*v[t1][i1][i2][i3 - 5] - temp223 - temp31) + temp227*(temp118*v[t1][i1][i2
 - 1][i3 - 2] - temp118*v[t1][i1][i2 + 1][i3 - 2] + temp21*v[t1][i1][i2 - 3][i3
- 2] - temp21*v[t1][i1][i2 + 3][i3 - 2] + temp230 + temp234) + temp237*(temp129*
```

3D scalar acoustic — Intel's ivy bridge (Endeavour)



Imperial College London

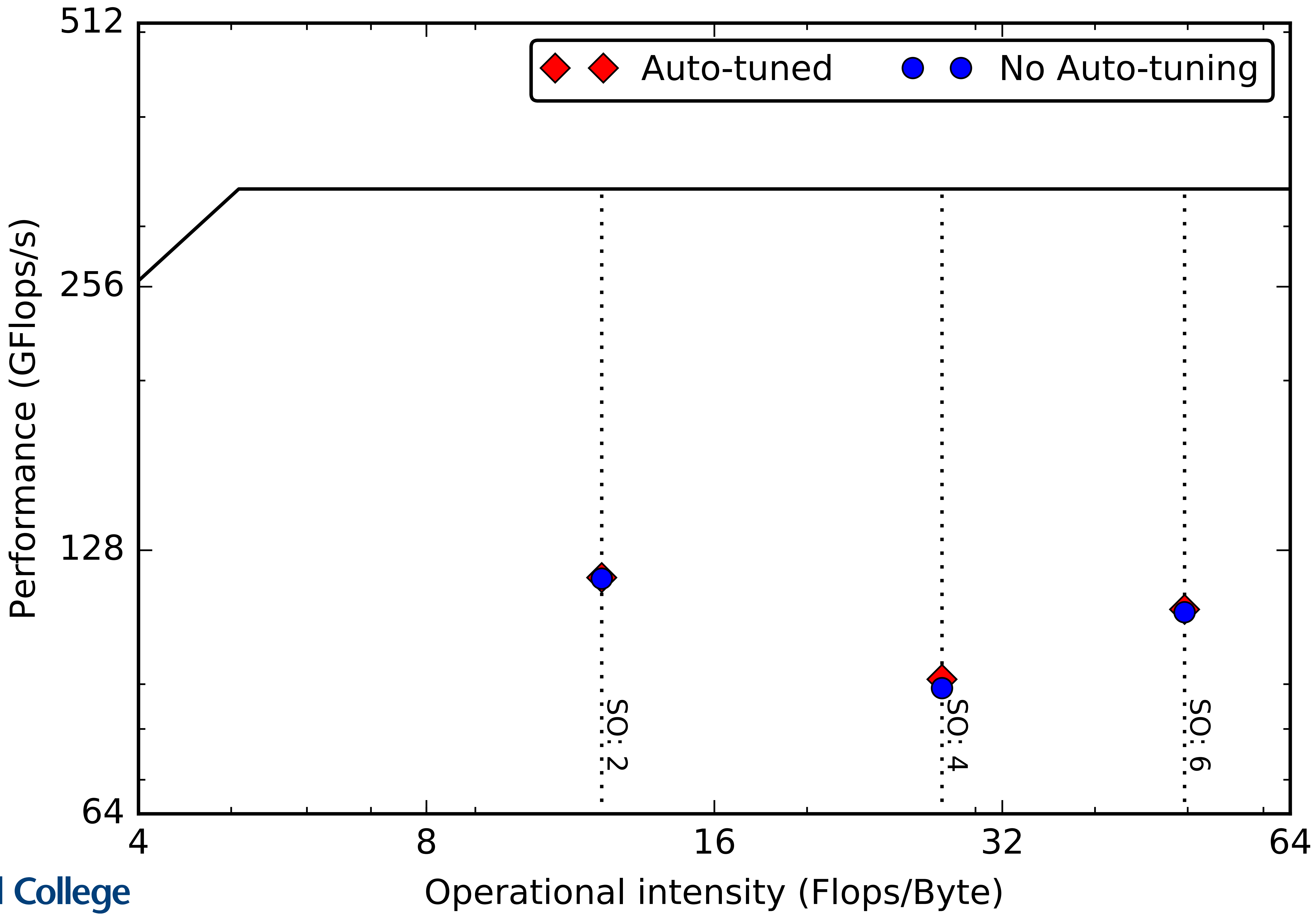
3D acoustic — Intel's KNL



Clearly optimisation strategy needs to adapt for KNL - OPESCI compiler team currently at Intel's meeting in Toulouse learning what needs to be done.

Imperial College
London

3D TTI — Intel's broadwell



Imperial College London

3D TTI - 91 GFLOPS and counting

- Complex equation, complex implementation, extremely challenging to optimise (527, 1213, 2211 FLOPs in the 2nd, 4th, and 6th stencils).
- Symbolic engine again plays critical role:
  - Common subexpression elimination (CSE) critical **before** source is generated - we extended SymPy for this purpose.
  - Greatly speeds up code generation (critical for JIT).
  - Allows further transformations (eg factorisation, expansion).
  - Without this high level CSE TTI cannot even be compiled!
  - Replaced trigometric functions with Chebyshev polynomials of the first kind - greatly increases FLOPs and complexity of expression but results in 1.3x speed-up at order 2.

3D TTI cont...

- Future work:
  - Systematically optimise these expressions (ie to reduce the flop count), exploiting mathematical properties.
  - Not feasible to run space order 8 with the current technology - but have a solution.
  - Vectorisation is crucial.
  - Many temporaries — register spilling!

Reflections

- SymPy as a DSL for finite difference is a key component. Migrate changes upstream to be supported by the wider community
- Stencil compilers are a very active area of computer science research
  - Need to maintain intermediate representation layer
  - Continue with our own stencil compiler - retain flexibility to add domain specific extensions.
  - Prototype integration with existing stencil DSL's - see where abstractions break down.
- Integration into FWI packages:
  - Unique selling points: Agility; performance; adjoint
  - Code generation allows us to generate many different API's. One target is to provide a "library" for existing C or Fortran codes. However, a high level implementation in a data flow framework such as TensorFlow is a clear topic for research.

Imperial College London

Currently designing 3rd generation of Devito

Maintaining current gneration as stable release for SINBAD. But already designing next generation of technology.

- MPI parallelism for larger models.
- Integrate with stencil and polyhedral compilers.
- Additional symbolic optimisation (factorisation, hoisting, etc.) - **critical for TTI**.
- Polyhedral compilation.
- Automated data layout optimisations.
- FLOP reduction and register pressure reduction.
- Extend feature range to facilitate more science.
- Integration of verification tools such as CodeThorn.

Imperial College
London

Thank You

OPESCI

intel
Intel PCC

Sistema FIEB SENAI
Federação das Indústrias do Estado da Bahia

EPSRC
Engineering and Physical Sciences
Research Council

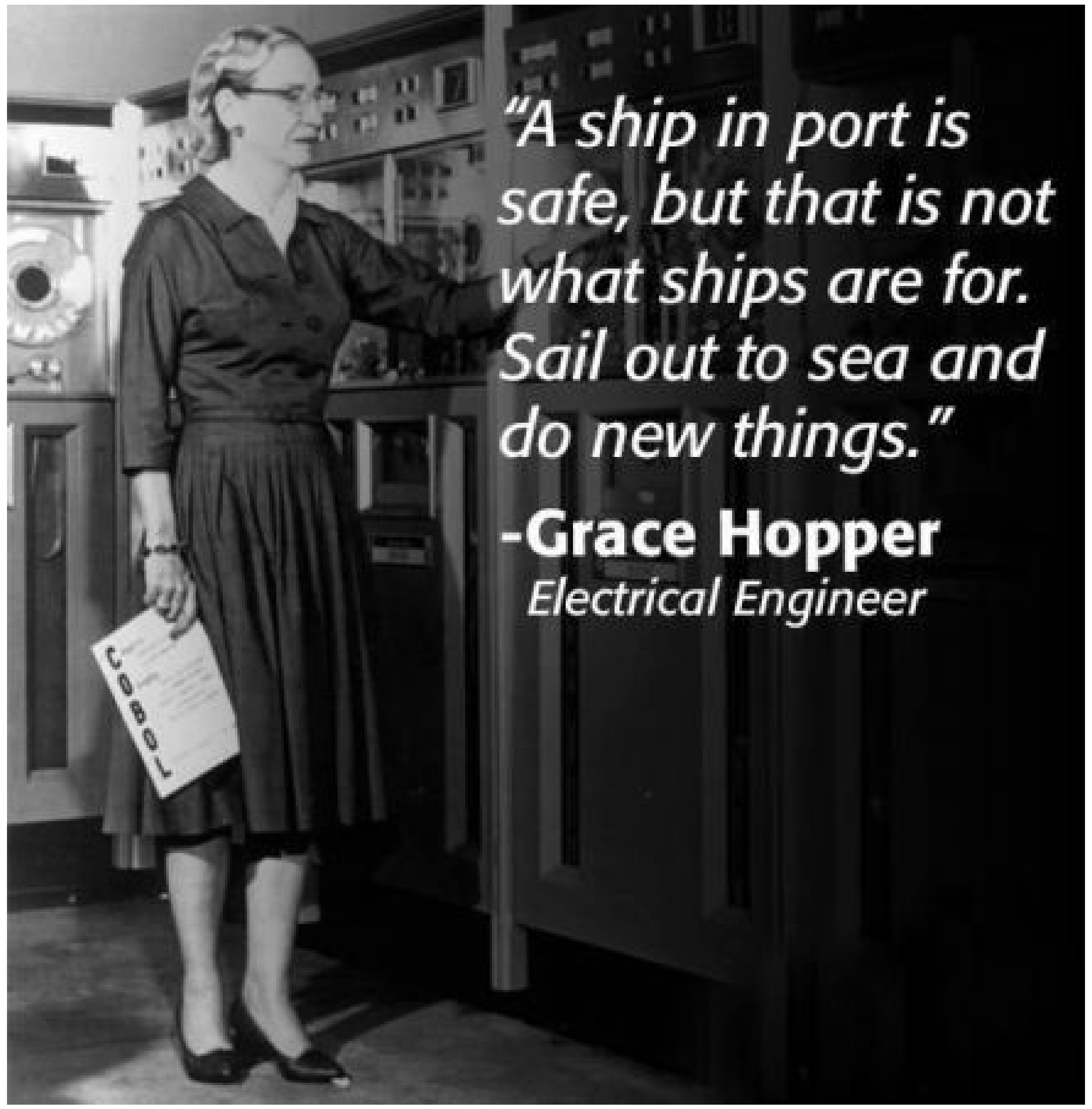
BG GROUP

SINBAD

Imperial College
London

Acknowledgments

This work was funded by the Intel Parallel Computing Centre program in partnership with BG Group, and ESPRC platform grant EP/L000407/1 - *Underpinning Technologies for Finite Element Simulation*.


"A ship in port is safe, but that is not what ships are for. Sail out to sea and do new things."
-Grace Hopper
Electrical Engineer
COBOL

Imperial College London