# Abstract

I knew the Duke of York, he had 1,000 men

He marched them all up the hill, and marched them down again

When you’re up, you’re up

And when you’re down, you’re down

But when you’re only halfway up, you’re neither up nor down.

# Introduction

The Kuramoto-Shivashinsky (KS) equation was developed concurrently by two scientists focusing on development in laminar flames in [1] and [2]. The premise is that for a laminar flame, the combustion of various species is a chaotic behavior that results in certain instabilities within the flame front. To save the derivation, the KS equation is shown in Equation 1.

Equation . Kuramoto-Shivashinsky Equation with Coefficients. Adapted from [3].

There are five terms in this form of the KS equation. The first is the time-derivative, which has an obvious effect. It is accompanied by the second term, the convection term, which provides non-linearity to the equation. This takes the form like the Navier-Stokes equation. The third term is the diffusion term, which is the anti-diffusion, or coalescence, term in this form due to the lack of a negative sign. The fourth term is what is called a biharmonic operator [1], the most obvious example being an elastic deflection on a beam. On the right-hand side (RHS) of the equation, there is a forcing function that is not present during the rest of the study.

This study looks at solving the KS-equation to better understand the stability and underlying behavior.

# Methodology

This project uses the methods outlined in Table 1. A minimum of 4th order is required for adequate assessment of the 4th derivative, and this study will go two steps further. The time method uses the Runge-Kutta 4th order time integration, which seems to be more common in the research world. During some experimentation, the 5th order helped to avoid generation of random wave-spikes and instabilities along the boundaries.

Table . Solver Methods.

|  |  |
| --- | --- |
| Solver | Methodology |
| Spatial | 6th Order |
| Time | RK-4 |
| BC Spatial | 5th Order |

## Spatial Gradients

For the fourth order gradient, the stencil requires four points. The stencils for the interior are shown in Figure 1, Figure 2, and Figure 3.

A black circle with a black line

Description automatically generated

Figure . Interior Stencil.

A black circle with a black dot

Description automatically generated

Figure . LHS Boundary Stencil.

A black dot with a black line

Description automatically generated

Figure . RHS Boundary Stencil.

These stencils result in the numerical gradients in the following sections. These numerical gradients are in the matrices that solve the KS equation.

### 1st Derivative

Equation . Interior 1st Derivative, 6th Order.

Equation . LHS Boundary 1st Derivative, 6th Order.

Equation . RHS Boundary 1st Derivative, 6th Order.

### 2nd Derivative

Equation . Interior 2nd Derivative, 6th Order.

Equation . LHS Boundary 2nd Derivative, 6th Order.

Equation . RHS Boundary 2nd Derivative, 6th Order.

### 3rd Derivative

Equation . Interior 3rd Derivative, 6th Order.

Equation . LHS Boundary 3rd Derivative, 6th Order.

Equation . RHS Boundary 3rd Derivative, 6th Order.

### 4th Derivative

Equation . Interior 4th Derivative, 6th Order.

Equation . LHS Boundary 4th Derivative, 6th Order.

Equation . RHS Boundary 4th Derivative, 6th Order.

## Time Integration Method

As mentioned before, the RK-4 time integration method is used for explicit time stepping. The central equation for integration is Equation 2. There are four functions for one real and three virtual time steps, denoted by the values.

Equation . RK-4 Time Stepping Integration.

Equation . First Virtual Time Step.

Equation . Second Virtual Step.

Equation . Third Virtual Step.

### Alternative Time Integration Schemes

## Algorithm

### Initialization

The first step in the solve is to initialize the object that contains the data. i.e.:

\_\_init\_\_():

Initialization

### Linear System of Equations Setup

The next step is to set up a linear system of equations in the form Equation 6. Lower case variables represent arrays iterating, and . The upper case variables represent matrices to solve the system. In this cases, is the sum of the various matrices representing the higher derivatives, thus .

Equation . The Linear Equation for KS Solve.

i.e:

A=numericalGradient(1,<stencil>)(sparse)

B\_diffusion=numericalGradient(2,<stencil>)(sparse)

B\_third=numericalGradient(3,<stencil>)(sparse)

B\_fourth=numericalGradient(4,<stencil>)(sparse)

B=B\_diffusion+B\_third+B\_fourth

D=identity(Nx x Nx)(sparse)

e=zeros(Nx)

### Boundary Condition Setup

In this study, there is a fourth derivative, thus four boundary conditions are required. The code then becomes:

Pull 0th Derivative BCs

Condition A, B, D, and e for these BCs

Pull 1st Derivative BCs

Condition A, B, D, and e for these BCs

Pull 2nd Derivative BCs

Condition A, B, D, and e for these BCs

Pull 3rd Derivative BCs

Condition A, B, D, and e for these BCs

Pull 4th Derivative BCs

Condition A, B, D, and e for these BCs

Note that in this study, the boundary conditions are zero for the value of at the edges, and zero gradient at the edges, thus the 2nd, 3rd, and 4th orders

### Time Stepping

Finally, to complete the solve, the solver will go through the time stepping process, thus the algorithm is:

for time in time domain:

v = ½\*u^2

f = A\*v + B\*u + e

if time step index < time integration order or time integration order==1:

phi = dt \* f + u

next u = D\phi

elif time integration order==4:

Rn = f

# First virtual time step

phi1 = (dt/2)\*Rn + u

u1 = D\phi1

v1 = ½\*u1^2

R1 = A\*v1 + B\*u1 + e

# Second virtual time step

phi2 = (dt/2)\*R1 + u

u2 = D\phi2

v2 = ½\*u2^2

R2 = A\*v2 + B\*u2 + e

# Third virtual time step

phi3 = dt \* R2 + u

u3 = D\phi3

v3 = ½\*u3^2

R3 = A\*v3 + B\*u3 + e

# RK4 time step

phi = u + (dt/6)\*(Rn + 2R1 + 2R2 + R3)

next u = D\phi

# Analysis

## Algorithm FLOPs Intensity

### Numerical Gradients

Each of the numerical gradient calculations contain the same number of floating point operations in the numerator, which comes to be 11. Then each order of derivative adds on floating point operation to the calculation, thus:

Equation . FLOP Count for a Spatial Gradient.

These spatial gradients feed into the linear algebra equation in Equation 18. The flux formulation for advection as 2 FLOPs per spatial point. This means that the advection is 14 FLOPs per spatial point. Then the diffusion and stiffness gradients form 13 and 14 plus 1 FLOPs per spatial point. Then adding the vector adds an additional FLOP per spatial point. Combining these to formulate the FLOP count for each time step:

Equation . FLOP Count for Each Time Spatial Point.

### Time Stepping

Each time step for the RK-4 method has two steps, calculate the RHS of the time integration, which is 3 FLOPs per spatial point, and then solve the LHS of the time integration, which we will assume is some LU decomposition method. It is an algorithm called COLAND [4], which is some modification of the LU method, actually a QR decomposition [5], for improved efficiency. For this analysis, we will assume the efficiency of a LU decomposition. One can note that Gaussian is technically twice as efficient, but certainly does not fit ALU parallelization well. Thus, each time step solve requires FLOPs to solve, totaling:

Equation . FLOP Count for Each Time Integration.

For the RK-4 method, there are four time steps, which get combined by an equation that uses 6+2+1 FLOPs per spatial point. Thus:

Equation . FLOP Count Total at Each Time Step.

Bringing all these values together, the total FLOP count becomes:

Equation . FLOP Count Total.

## Convergence Analysis & Order of Accuracy

# Results

## Regime Definition

.

# Works Cited

1. Sivashinsky, G. I. (1977*). Nonlinear analysis of hydrodynamic instability of laminar flames – I. Derivation of basic equations*. Acta Astronautica, Vol 4, pg 1177-1206.
2. Kuramoto, Y. (1978). *Diffusion-Induced Chaos in Reaction Systems.* Supplement of the Progress of Theoretical Physics, No 64.
3. Kalogirou, A., Keaveny, E. E., and Papageorgiou, D. T. (2015). *An in-depth numerical study of the two-dimensional Kuramoto-Shivashinsky equation.* The Royal Society Publishing.
4. SciPY:Sparse. (2024). API Reference: scipy.sparse.linalg.spsolve. Available at: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.spsolve.html#r1c4f91118c9e-1>
5. Davis, T. A., Gilbert, J. R., Larimore, S. I., and Ng, E. G. (2000). *Algorithm 8xx: COLAMD, a column approximate minimum degree ordering algorithm.* University of Florida: Technical Report TR-00-006.