Analysis of a Parallel Solver for the Gray-Scott Reaction-Diffusion System

Parallel Programming for Large-Scale Problems SF2568
Teacher: Michael Hanke

Jonathan Ridenour, 780514-7779
Mateusz Herczka, 700624-9234

May 9, 2015

Problem description

The subject of this report is the parallelisation of a numerical method for a two-component reaction-diffusion system known as the Gray-Scott model. Originally introduced by Gray and Scott [1], the model is a system of two partial differential equations for the concentration of two chemical species, which are reacting with each other as they diffuse through a medium.

Pattern formation resulting from the Gray-Scott model is a rich area of research, with applications to a variety of physical, chemical, and biological phenomena, as well as cellular automata and the pattern formation of other partial differential equation systems [2]. We investigate a parallelisation strategy for the two-dimensional case. We make a series of trials, running the solver in parallel on 4, 16, 64, and 100 processes, and perform a speedup analysis using the results of these trials. Finally, we show a few of the various patterns that emerge from the Gray-Scott model with various input parameters.

Mathematical Formulation

The Gray-Scott model involves the reaction and diffusion of two generic chemical species, U and V, whose concentrations are described by the functions u and v, reacting according to the chemical equations

$$\begin{aligned} U + 2V &\to 3V, \\ V &\to P, \end{aligned}$$

where P is an inert byproduct. This system is governed by the following system of partial differential equations, known as the Gray-Scott equations:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1-u), \qquad u, v : \Omega \mapsto \mathbb{R},\ t \geq 0, \tag{1}$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F-K)v, \qquad u, v : \Omega \mapsto \mathbb{R}, t\ \geq 0. \tag{2}$$

Here, D_u and D_v are the diffusion constants for u and v respectively, and F and K are constants which govern the replenishment of the chemical species. The term uv^2 gives the reaction rate for the system. We are concerned with solving the Gray-Scott equations on a two-dimensional domain $\Omega = [0,1]^2$. Thus, $u = u(x,y,t)$ and $v = v(x,y,t)$, giving the laplacian as

$$\begin{aligned} \nabla^2 u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \\ \nabla^2 u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \end{aligned}$$

As in [2], we use periodic boundary conditions for (1) and (2); thus, our simulation on Ω models a small surface surrounded by similar elements reacting identically. We choose the diffusion constants $D_u = 2 \cdot 10^{-6}$, $D_v = 1 \cdot 10^{-6}$, and let F and K take values according to $F \in [0.02,\ 0.08]$, $K \in [0.05,\ 0.07]$ in order to investigate the different types of patterns that arise from varying combinations of the two.

The system is in a trivial state when $u = 1$ and $v = 0$. To initialize a reaction, we let the system assume this trivial state at time zero in all areas except a square in the middle of dimension one-sixth, where we set $u = 0.5$ and $v = 0.25$:

$$u(x, y, 0) = \begin{cases} 1/2, & 5/12 \geq x \geq 7/12,\ 5/12 \geq y \geq 7/12, \\ 1, & \text{otherwise,} \end{cases} \tag{3}$$

$$v(x, y, 0) = \begin{cases} 1/4, & 5/12 \geq x \geq 7/12,\ 5/12 \geq y \geq 7/12, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The reaction then diffuses outward from the middle, leaving behind a characteristic pattern, until all Ω is filled.

Numerical Method

We discretize Ω with the same stepsize in the x- and y-directions, h, corresponding to N^2 inner gridpoints equidistantly spread across the surface. We let $h = 1/(N+1)$ such that $x_i = ih$, $i = 1, 2, \ldots, N$ and $y_j = jh$, $j = 1, 2, \ldots, N$. Thus the function value $u(x, y, t)$ is approximated by $u(ih, jh, t) = u_{i,j}(t)$, $t \geq 0$. Using the traditional central difference formula for the discrete five-point laplacian with step length h, Δ_5^h, the system (1) + (2) can be written in semi-discrete form as follows:

$$\frac{\partial u_{i,j}}{\partial t} = \frac{D_u}{h^2} \Delta_5^h u_{i,j} - u_{i,j}(v_{i,j})^2 + F(1 - u_{i,j}), \tag{5}$$

$$\frac{\partial v_{i,j}}{\partial t} = \frac{D_v}{h^2} \Delta_5^h v_{i,j} + u_{i,j}(v_{i,j})^2 - (F - K)v_{i,j}. \tag{6}$$

We then discretize in time, with timestep dt, and approximate the time-derivative with the explicit Euler method. Letting $t = k \cdot dt$ for $k = 1, 2, 3, \ldots.$, and $u_{i,j}(k \cdot dt) = u_{i,j}^k$, $v_{i,j}(k \cdot dt) = v_{i,j}^k$, the system (5) + (6) becomes

$$\frac{u_{i,j}^{k+1} - u_{i,j}^k}{dt} = \frac{D_u}{h^2} \Delta_5^h u_{i,j}^k - u_{i,j}^k (v_{i,j}^k)^2 + F(1 - u_{i,j}^k), \tag{7}$$

$$\frac{v_{i,j}^{k+1} - v_{i,j}^k}{dt} = \frac{D_v}{h^2} \Delta_5^h v_{i,j}^k + u_{i,j}^k (v_{i,j}^k)^2 - (F - K)v_{i,j}^k. \tag{8}$$

The difference equations (7) and (8) give a simple formula for the update of the function values at time $(k+1) \cdot dt$ given that values at time $k \cdot dt$. We represent the discrete functions $u_{i,j}^k$ and $v_{i,j}^k$ as long vectors $\mathbf{u}^k$ and $\mathbf{v}^k$ with the following enumeration:

$$\mathbf{u}^k = (u_{1,1}^k,\ u_{2,1}^k,\ \ldots,\ u_{N,1}^k,\ u_{1,2}^k,\ u_{2,2}^k,\ \ldots,\ u_{N,N}^k),$$
$$\mathbf{v}^k = (v_{1,1}^k,\ v_{2,1}^k,\ \ldots,\ v_{N,1}^k,\ v_{1,2}^k,\ v_{2,2}^k,\ \ldots,\ v_{N,N}^k).$$

Using the long vectors, we can write the numerical operator Δ_5^h as a sparse matrix multiplication as follows:

$$\frac{D_u}{h^2}\Delta_5^h u_{i,j} = \mathbf{A}_u \mathbf{u}^n,$$

$$\frac{D_v}{h^2}\Delta_5^h v_{i,j} = \mathbf{A}_v \mathbf{v}^n,$$

where the matrices $\mathbf{A}_u$ and $\mathbf{A}_v$ are of dimension $[N^2 \times N^2]$ with block tridiagonal form:

$$\mathbf{A}_u = \text{tridiag}_N(\sigma_u \mathbf{I},\ \mathbf{T}_u,\ \sigma_u \mathbf{I}),$$
$$\mathbf{A}_v = \text{tridiag}_N(\sigma_v \mathbf{I},\ \mathbf{T}_v,\ \sigma_v \mathbf{I}),$$

where $\mathbf{I}$ is an identity matrix, multiplied by the constants σ_u or σ_v, and the matrices $\mathbf{T}_u$ and $\mathbf{T}_v$ are tridiagonal of the form

$$\mathbf{T}_u = \text{tridiag}_N(\sigma_u,\ -4\sigma_u,\ \sigma_u),$$
$$\mathbf{T}_v = \text{tridiag}_N(\sigma_v,\ -4\sigma_v,\ \sigma_v),$$

where

$$\sigma_u = \frac{D_u}{h^2},$$

$$\sigma_v = \frac{D_v}{h^2}.$$

If we let $f_u(\mathbf{u}^k, \mathbf{v}^k) = -\mathbf{u}^k(\mathbf{v}^k)^2 + F(1 - \mathbf{u}^k)$ and $f_v(\mathbf{u}^k, \mathbf{v}^k) = \mathbf{u}^k(\mathbf{v}^k)^2 - (F - k)\mathbf{v}^k$, we can compactly write the fully discretized equations as

$$\mathbf{u}^{k+1} = \mathbf{u}^k + dt[\mathbf{A}_u \mathbf{u}^k + f_u(\mathbf{u}^k, \mathbf{v}^k)], \tag{9}$$
$$\mathbf{v}^{k+1} = \mathbf{v}^k + dt[\mathbf{A}_v \mathbf{v}^k + f_v(\mathbf{u}^k, \mathbf{v}^k)], \tag{10}$$

and we have an exact update formula for each time step.

Initial and Boundary Conditions

The initial and boundary conditions are straightforward to implement numerically. The initial state is completely known, so $\mathbf{u}^0$ and $\mathbf{v}^0$ are given. To implement periodic boundary conditions we simply set

$$u^k_{-1,j} = u^k_{N-1,j},$$
$$u^k_{i,-1} = u^k_{i,N-1},$$
$$u^k_{N,j} = u^k_{0,j},$$
$$u^k_{i,N} = u^k_{i,0},$$

when updating the boundary points. The formula is likewise for v.

Stability Conditions

In order to ensure numerical stability for the differential operators $\mathbf{A}_u$ and $\mathbf{A}_v$ in (9) and (10), we must conform to the restriction

$$dt \cdot \lambda_{l,u} \in \mathcal{S},\ \forall\, l,\ l = 1, 2, \ldots L_u,$$
$$dt \cdot \lambda_{l,v} \in \mathcal{S},\ \forall\, l,\ l = 1, 2, \ldots L_v,$$

where $\lambda_{l,u}$ and $\lambda_{l,v}$ are the l-th eigenvalues of $\mathbf{A}_u$ and $\mathbf{A}_v$ respectively, and $\mathcal{S}$ is the stability region of the explicit Euler method: a circle in the complex plane centered at -1 with unit radius [3]. The number of unique eigenvalues possessed by each of the matrices is denoted by L_u and L_v.

For the parallel speedup trials, we discretize Ω using $N = 400$. Together with the chosen diffusion constants, the maximum eigenvalue, λ_{max}, of $\mathbf{A}_u$ and $\mathbf{A}_v$ has a strictly negative real value, and thus we must choose a timestep such that $dt \cdot \lambda_{max}$ is at least -2. The maximum eigenvalue is computed using Matlab as:

$$\lambda_{max} = -1.2735,$$

which provides the limitation on the timestep:

$$dt \leq \frac{-2}{-1.2735} = 1.5704.$$

Since the timestep also has a damping effect on the replenishment term and reaction rate, we must consider these as well when choosing the timestep. As in [2], we settle on a value which is comfortably beneath the max:

$$dt = 0.75.$$

This is well within the stability bounds imposed by $\mathbf{A}_u$ and $\mathbf{A}_v$; it also provides suitable damping on the replenishment and reaction rate terms for the observation of the characteristic Gray-Scott pattern formation.

Algorithm Description

Implementation details

To implement the update formulae (9) and (10) in parallel, the computational domain is divided into subdomains, one for each process. Each process is responsible for the update of it's local values. Along the subdomain boundaries, neighbouring values are needed for the update; these are obtained by coordinated message-passing. The boundary data are communicated between processes by means of a red-black communication schedule. When all necessary information is obtained, the local values are updated according to (9) and (10), see Algorithm 1.

For a domain size of 400×400, the global update boils down to two matrix multiplications, one using $\mathbf{A}_u$, one using $\mathbf{A}_v$, each of which contains 400^4 elements, most of which are zero. This can be accomplished by means of a sparse matrix multiplication method, or .... fill in your part here!

```
**Data**: $dt$, $\mathbf{u}^k$, $\mathbf{v}^k$, $\mathbf{A}_u$, $\mathbf{A}_v$, red
**Result**: $\mathbf{u}^{k+1}$ and $\mathbf{v}^{k+1}$
collect boundary data $\mathbf{b}_{out}$;
**if** *red* **then**
    send $\mathbf{b}_{out}$: north, south, east, west;
    receive $\mathbf{b}_{in}$: south, north, west, east;
**else**
    receive $\mathbf{b}_{in}$: south, north, west, east;
    send $\mathbf{b}_{out}$: north, south, east, west;
**end**
$\mathbf{u}^{k+1} = \mathbf{u}^k + dt[\mathbf{A}_u\mathbf{u}^k + f_u(\mathbf{u}^k, \mathbf{v}^k)] + \mathbf{b}_{in}$;
$\mathbf{v}^{k+1} = \mathbf{v}^k + dt[\mathbf{A}_v\mathbf{v}^k + f_v(\mathbf{u}^k, \mathbf{v}^k)] + \mathbf{b}_{in}$;
```

Algorithm 1: Communication and local update.

Performance Evaluation

Extrapolated Speedup

In an attempt to predict the parallel speedup we can expect to achieve as we increase the number of processes, we perform an initial run on 1 and 4 processes, compute the speedup S_4, and extrapolate to S_P using Amdahl's Law. For the test case employing sparse matrix multiplication, we get an run time of 793.15 s on one process, and 304.39 s on 4 processes. This gives the initial speedup:

$$S_4 = \frac{T_1}{T_4} = 2.606.$$

Using Amdahl's Law we can solve for f, the fraction of the computation which cannot be divided into parallel tasks:

$$S_4 = \frac{4}{1+3f} \Rightarrow f = \frac{4 - S_4}{3S_4} = 0.1783.$$

Applying this same analysis to the kernel method, we get the following calculation:

$$S_4 = \frac{T_1}{T_4} = \frac{113.0}{94.26} = 1.199 \Rightarrow f = \frac{4 - S_4}{3S_4} = 0.7789.$$

Table 1 shows the extrapolated speedup that these values for f indicate for 16, 64, and 100 processes. This method is obviously a poor predictor for both the parallelisation strategies we employ. Indeed, our experimental speedup trials indicate that the problem scales very well on increased numbers of processes, see figures 1 and 2.

Table 1: Extrapolated speedup given by Amdahl's Law.

	P	4	16	64	100
Sparse matrix method	S_P	2.6	4.3	5.2	5.4
Kernal method	S_P	1.2	1.3	1.3	1.3

Experimental Speedup

We run a series of trials on Ferlin and record the initialisation time, t_{init}, the communication time, t_{comm}, the calculation time, t_{calc}, and the total run time t_{tot}. Tables 2 and 3 show the maximum of these values for each of the trials at 1, 4, 16, 64, and 100 processes. The values of t_{tot} are used to compute the experimental speedup, shown in figures 1 and 2.

Table 2: Max time (seconds) measurements, sparse matrix method.

P	t_{init}	t_{comm}	t_{calc}	t_{tot}
1	0.1437	0.000	793.0	703.14
4	0.05663	36.48	301.8	304.4
16	0.008907	8.786	38.42	40.87
64	0.009324	2.8460	7.391	9.603
100	0.001253	3.206	4.287	7.419

Table 3: Max time (seconds) measurements, kernel method.

P	t_{init}	t_{comm}	t_{calc}	t_{tot}
1	0.0046	0.0000	113.0	113.0
4	0.0051	3.157	91.37	94.26
16	0.1540	3.115	6.559	9.778
64	0.1845	2.410	1.683	4.248
100	1.197	3.293	1.081	5.556

Speedup Analysis

There are a few reasons why the extrapolation of Amdahl's Law does not provide an accurate predictor for the parallel speedup. The most obvious reason is that f, the fraction of the computation which cannot be divided into parallel tasks, does not in itself

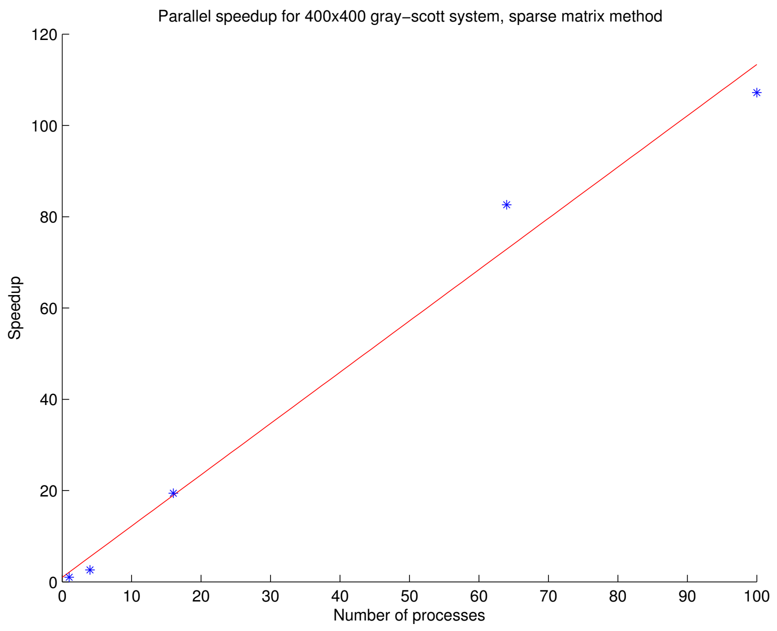

Figure 1: Experimental speedup, sparse matrix multiplication method.

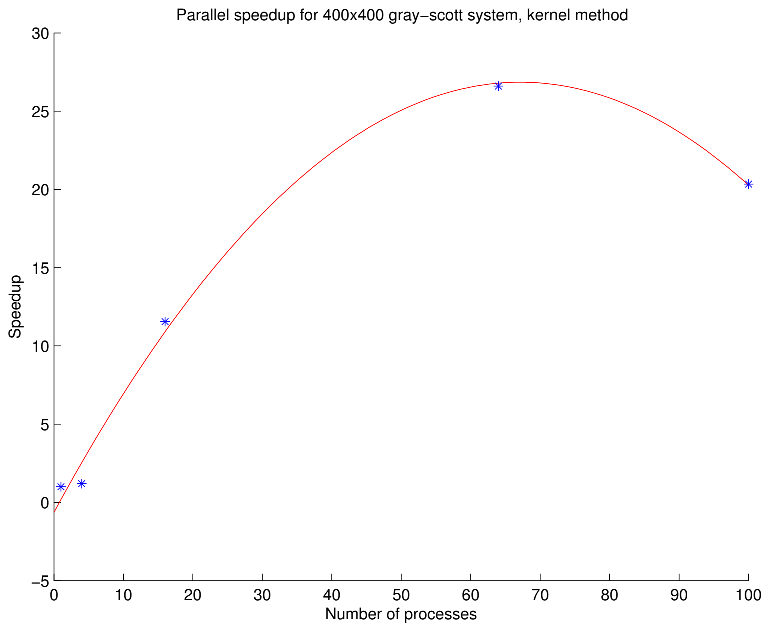

Figure 2: Experimental speedup, kernel method.

account for an upper bound on the speedup. In fact, our best measure of f is t_{init} and, as can be seen in tables 2 and 3, this accounts for only a minuscule portion of the total runtime. It is more likely that the speedup can be explained by the memory hierarchies employed by Ferlin. When the number of processes is low, so is the available cache. As the number of processes increases, more of the working set, and at some point the entire working set, can fit into the available caches, allowing for a quicker memory access time.

Patterns

Conclusions

References

[1] Gray P, Scott SK. *Sustained oscillations and other exotic patterns of behavior in isothermal reactions.* J Phys Chem 1985;89:22–32.

[2] Weiming Wang, Yezhi Lin, Feng Yang, Lei Zhang, Yongji Tan, *Numerical study of pattern formation in an extended Gray–Scott model*, Communications in Nonlinear Science and Numerical Simulation, 16 (2011).

[3] Lennart Edsberg *Introduction to Computation and Modelling for Differential Equations*, 2008, John Wiley and Sons, pp 50.