Solving Navier-Stokes Equations with Sequential Regularization Methods

Colin Macdonald

21/06/2001

Contents

Abstract

We introduce the notion of partial differential-algebraic equations (PDAEs) by discussing the Navier-Stokes equations for an incompressible viscous fluid. We note that there are two concerns for solving such problems: the involvment of PDEs rather than ODEs, and the presense of *algebraic constraints* on these PDEs.

To deal with the first concern, we introduce solving PDEs numerically with the *method of lines* and *finite differences* with the help of an example based on the one-dimensional heat equation.

After discussing some of the difficulties involved in the numerical solution to differential-algebraic equations (DAEs), we introduce the *sequential regularization method (SRM)* as a method for solving certain types of DAE problems. We present a SRM for Navier-Stokes equations as a possible solution to the second concern above. A possible spatial discretization of the SRM for Navier-Stokes equations is then demonstrated.

We introduce a *predicted sequential regularization method (PSRM)* which significantly improves computational time over the SRM.

We test the SRM and PSRM on two problems: a problem with an exact solution and a physically motivated cavity problem. Based on these results, a discussion of the various CFL restrictions of these methods is presented.

1 Navier-Stokes Equations

Navier-Stokes equations are used to model the flow of fluids based on Newton's second law of motion applied to localized points in the flow. They are used in many areas of engineering and form the basis for the study of fluid dynamics.

The velocity $u(x, t)$ and pressure $p(x, t)$ of a viscous fluid over a domain $\Omega \subseteq \mathbb{R}^n$ are related by the PDE

$$u_t + (u \cdot \nabla)u = \mu\nabla^2 u - \nabla p + f, \tag{1}$$

where $u : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, $p : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, $\mu \in \mathbb{R}$ is the *constant of viscosity*, and $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ represents the external forces applied to the fluid (such as, for example, gravity or a driving force due to wind).

1.1 Navier-Stokes Equations as PDAEs

In many fluid dynamics problems, the fluid is assumed to be *incompressible*; that is, the volume of fluid flowing into a system must equal the volume flowing out. This means that the system is *divergence free* and we expect that

$$\nabla \cdot u = 0. \tag{2}$$

However, simply ignoring (2) and solving (1) numerically will result in solutions which may or may not satisfy the incompressibility condition. For example, if the problem describes water flowing down a pipe which narrows at some point, then solving the PDE without the constraint may well result in a solution where there is simply less water after the narrowing. Clearly, this leads to unphysical solutions.

Instead we consider (2) as an *algebraic constraint* upon the PDE (1). We can thus rewrite the Navier-Stokes equation with the incompressibility constraint in what is known as a *partial differential-algebraic equation* (PDAE) of the form

$$u_t + (u \cdot \nabla)u = \mu\nabla^2 u - \nabla p + f, \tag{3a}$$

$$\nabla \cdot u = 0, \tag{3b}$$

subject to the boundary conditions

$$u|_{\partial\Omega} = b(t), \tag{3c}$$

and the initial conditions

$$u(x, 0) = a(x). \tag{3d}$$

This system now has the *divergence-free* condition (2) built in and, in fact, (3b) is known as the *constraint equation.*

For this project, we seek numerical solutions to problems in the form of (3). This presents two distinct concerns:

1. The problem involves PDEs rather then ODEs.

2. The presense of the constraint equation.

To deal with the first concern, we begin by examining a widely used discretization technique for replacing PDEs with ODEs known as the *method of lines.* Later on, we present a *sequential regularization method* for the numerical solution of differential equations with algebraic constraints.

2 The Method of Lines

The *method of lines (MOL)* provides a way of numerically replacing PDEs with ODEs. This is accomplished by discretizing the spatial domain and forming a coupled system of ODEs in time. This discretization can be accomplished by a variety of methods; however, if we restrict ourselves to a rectangular domain[1], then a simple discretization known as *finite differences* is effective.

2.1 Finite Differences

Consider the Taylor series for $f(x+h)$ and $f(x-h)$

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2!}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \ldots, \tag{4a}$$

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2!}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + \ldots. \tag{4b}$$

By rearranging (4a) and (4b), we obtain the following formulae for the first derivative in terms of infinite series:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)h}{2} + \ldots,$$

$$f'(x) = \frac{f(x) - f(x+h)}{h} + \frac{f''(x)h}{2} - \ldots.$$

Now if we truncate these series before the $\mathcal{O}(h)$ terms, then we obtain the following approximations for the first derivatives

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \tag{5a}$$

$$f'(x) \approx \frac{f(x) - f(x+h)}{h}, \tag{5b}$$

[1]That is, rectangular with respect to some coordinate system of the domain.

respectively called *forward difference* and *backward difference.* Both are known as $\mathcal{O}(h)$ *approximations* because the most significant term (with respect to h) that was truncated was $\mathcal{O}(h)$. That is, the error on these approximations is ch for some $c \in \mathbb{R}$.

By adding (4a) to (4b) and solving for $f''(x)$, we obtain

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{f^{(iv)}(x)}{12}h^2 + \ldots, \tag{6}$$

and similar to above, by dropping the $\mathcal{O}(h^2)$ and higher terms, we obtain a $\mathcal{O}(h^2)$ approximation to the second derivative

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{7}$$

As one might imagine, similar formulae exist for other derivatives though it may be necessary to make use of the Taylor series for other points near x such as $f(x+\frac{h}{2})$ or $f(x+2h)$.

The MOL *discretizes* all spatial partial derivatives with appropriate finite differences. For the purposes of this project, the spatial domain of the problem is first divided into to a uniform[2] *mesh* which provides the points necessary for the finite differences. This process results in a system of ODEs in time of $N+1$ variables where N is the number of points in the mesh. The following example based on the heat equation will demonstrate this technique.

[2]In general, the mesh does not need to be uniform; in fact there is a lot of research into using non-uniform meshes which move around to keep a locally finer mesh over "interesting areas" of the domain while those areas move with time. Such techniques are particularly useful to model problems such as the temperature of a surface as a flame front burns across it.

2.2 A Simple Spatial Discretization Example

Consider the one-dimensional heat equation

$$u_t = cu_{xx} \tag{8}$$

on the unit interval with boundary conditions $u(t,0) = \alpha$, $u(t,1) = \beta$ and some initial conditions. Suppose we choose a uniform mesh by partitioning $[0,1]$ in N equal subintervals, each of width h. We refer to each point in the mesh as x_j for $j = 0,\ldots,N$, and adopt the common notation that $u(t,x_j) = u_j(t)$. Then, using a finite difference formula (7) in place of u_{xx}, we approximate the PDE with

$$\frac{\mathrm{d}u_j}{\mathrm{d}t} = c\frac{u_{j-1} - 2u_j + u_{j+1}}{h^2}, \quad j = 1,\ldots,N-1, \tag{9a}$$

$$u_0 = \alpha, \tag{9b}$$

$$u_N = \beta. \tag{9c}$$

We can combine (9a) for all $n = 1,2,\ldots,(N-1)$ into a system of $N-1$ ODEs. Because (9b) and (9c) will be satisfied by the initial conditions, two additional trivial ODEs can be added to our system, namely: $(u_0)' = 0$ and $(u_N)' = 0$. Letting $\gamma = \frac{c}{h^2}$ and $\delta = -2\frac{c}{h^2}$, we obtain the coupled system of $N+1$ ODEs

$$\begin{pmatrix} u_0 \\ u_1 \\ \\ \vdots \\ \\ u_{N-1} \\ u_N \end{pmatrix}'(t) = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \gamma & \delta & \gamma & 0 & \cdots & 0 & 0 \\ 0 & \gamma & \delta & \gamma & 0 & \cdots & 0 \\ \vdots & 0 & \gamma & \delta & \gamma & \ddots & \vdots \\ 0 & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \gamma & \delta & \gamma \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \\ \vdots \\ \\ u_{N-1} \\ u_N \end{pmatrix}(t), \tag{10}$$

which can be solved using various numerical techniques for ODEs such *Forward Euler* or various *Runge-Kutta* methods. Note in particular that the tri-diagonal structure of the matrix can be exploited for efficiency.

To summarize, the method of lines discretizes PDEs in space to obtain ODEs in time from which numerical solutions can be obtained using time discretization techniques. However, for a PDAE problem such as the (3), the method of lines spatial discretization does not yield a strictly ODE form. For these problems, it becomes necessary to numerically solve a system of $N+1$ *differential-algebraic equations* (DAEs). Towards this end, the next section introduces a newly developed method for the numerical solution of certain types of DAEs.

3 The Sequential Regularization Method

3.1 Differential-Algebraic Equations

Differential-algebraic equations (DAEs) are systems of (ordinary) differential equations where the unknown functions satisfy additional algebraic equations. Consider the DAE

$$x' = f(t,x) - B(t,x)y, \tag{11a}$$

$$0 = g(t,x), \tag{11b}$$

where x is the called the *differential variable*, y is the *algebraic variable*, and (11b) is an *algebraic constraint* imposed by the term By (in this sense, y behaves as a vector of Lagrange multipliers).

It turns out that many DAEs may be expressed as ODEs by repeatedly differentiating their constraint equations (and making substitutions of (11a) where necessary). The number of such differentiations that must be carried out to obtain an ODE for

y is known as the *index* of the PDE and in some sense characterizes the "difficulty" of a PDAE .

3.2 Numerical Solution of DAEs

Although it may be possible, using multiple *index reductions*, to analytically convert a DAE to an ODE, such manipulations introduce mild numerical instabilities into the problem. When numerically solving the *index-reduced* problem, these instabilities manifest themselves over time in a phenomenon known as *drift* whereby the solutions satisfy the differential equation of the DAE but no longer satisfy the algebraic constraint equation. Various *stabilization methods* have been developed to overcome the problem of drift. Perhaps the most popular of these techniques is *Baumgarte's technique* (see, [6]) which involves introducing an ODE for the algebraic variable by replacing the constraint equation (11b) with

$$\alpha_1 \frac{\mathrm{d}}{\mathrm{d}t} g(t,x) + \alpha_2 g(t,x) = 0, \tag{12}$$

where $\alpha_1 \geq 0, \alpha_2 \geq 0$ are known as *Baumgarte's parameters.* On its own, Baumgarte's technique suffers from a shortcoming in that α_1 and α_2 are notoriously difficult to choose correctly. Despite this, Baumgarte's technique is used by many engineers (see, e.g., [2, 1]), particularly for modeling mechanical problems known as *multibody systems.*

Another common technique, known as *regularization*, introduces an ODE for the algebraic variable by replacing the constraint equation (11b) with

$$\varepsilon T^{-1}(x,t) y' = g(x,t),$$

or, as an ODE for y,

$$y' = \frac{1}{\varepsilon} T(x,t) g(x,t), \tag{13}$$

where ε is a small scalar known as the *regularization parameter* and $T(x,t)$ is known as the *regularization matrix*. The latter is chosen to stabilize the problem and typically we use $T = (GB)^{-1}$, $T = (GB)^T$, or $T = I$. Unfortunately, because ε is small, the regularized problem is more *stiff* then the original problem. That is, a simple explicit solver may require cripplingly small step-sizes; generally one must switch to an implicit solver which is generally undesirable due to a significantly higher computation expense (as compared to an explicit solver).

The sequential regularization method (SRM) is a new method discussed in [3, 4] that borrows an iterated technique from optimization theory and combines it with regularization and Baumgarte's stabilization technique.

3.3 The SRM for index-2 DAEs

Instead of solving one stiff equation we solve a sequence *of less-stiff equations for* $x^{(\sigma)}$ *and* $y^{(\sigma)}$*:*

$$\begin{aligned}\left(x^{(\sigma)}\right)' &= f(x^{(\sigma)}, t) \\ &\quad - B(x^{(\sigma)}, t)\left(y^{(\sigma-1)} + \frac{1}{\varepsilon} T(x^{(\sigma)}, t)\left(\alpha_1 \frac{\mathrm{d}}{\mathrm{d}t} g(x^{(\sigma)}, t) + \alpha_2 g(x^{(\sigma)}, t)\right)\right), \quad &(14a)\\ y^{(\sigma)} &= y^{(\sigma-1)} + \frac{1}{\varepsilon} T(x^{(\sigma)}, t)\left(\alpha_1 \frac{\mathrm{d}}{\mathrm{d}t} g(x^{(\sigma)}, t) + \alpha_2 g(x^{(\sigma)}, t)\right), &(14b)\end{aligned}$$

where α_1, α_2 are Baumgarte's parameters and ε and $T(x,t)$ are respectively the regularization parameter and the regularization matrix mentioned above. However, because we are solving a sequence of problems, ε *does not* need to be so small as to make the problem stiff (see, [3, 4]). Also if we let $\alpha_1 = 0$, then a linear system solve is a avoided and the SRM can be a fully explicit method (see [3, 4] for analysis that supports this choice). The SRM is thus rewritten as

$$\left(x^{(\sigma)}\right)' = f(x^{(\sigma)},t) - B(x^{(\sigma)},t)\left(y^{(\sigma-1)} + \frac{1}{\varepsilon}T(x^{(\sigma)},t)g(x^{(\sigma)},t)\right), \tag{15a}$$

$$y^{(\sigma)} = y^{(\sigma-1)} + \frac{1}{\varepsilon}T(x^{(\sigma)},t)g(x^{(\sigma)},t), \tag{15b}$$

The SRM Algorithm

Given an somewhat arbitrary $y^{(0)}(t)$ (see, [9]) and an initial value for $x^{(\sigma)}(0)$, we repeat the following steps m times

① solve the *ODE* (15a) to advance from $x^{(\sigma)}_{n-1}$ to $x^{(\sigma)}_n$ (in general this will require $y^{(\sigma-1)}(t)$.

② use the *update formula* (15b) to obtain $y^{(\sigma)}_n$ (in general this will require $y^{(\sigma-1)}(t)$ and $x^{(\sigma)}(t)$).

The algorithm can be visualized as per Figure 1 and is described in detail in [8]. Note that m is typically chosen to be the order of the underlying Runge-Kutta scheme (or other ODE solver).

3.4 A SRM for Navier-Stokes Equations

Instead of solving the PDAE (3), we solve a *sequence* of PDE problems given by

$$(u^{(\sigma)})_t + (u^{(\sigma)}\cdot\nabla)u^{(\sigma)} = \mu\nabla^2 u^{(\sigma)} - \nabla\left(p^{(\sigma-1)} - \frac{1}{\varepsilon}(\nabla\cdot u^{(\sigma)})\right) + f, \tag{16a}$$

$$p^{(\sigma)} = p^{(\sigma-1)} - \frac{1}{\varepsilon}(\nabla\cdot u^{(\sigma)}), \tag{16b}$$

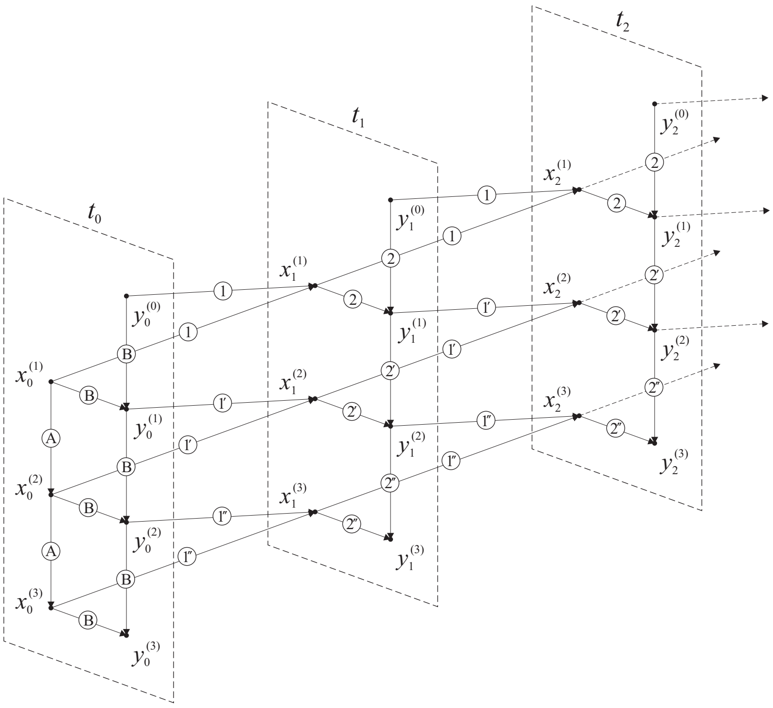

Figure 1: Graphical representation of the SRM algorithm.

where each PDE is subject to the original boundary and initial conditions, namely

$$u^{(\sigma)}|_{\partial\Omega} = b(t) \quad \text{and} \quad u^{(\sigma)}(0,x) = a(x). \tag{17}$$

We will concentrate on the 2-dimensional case in a rectangular domain and note that we can apply similar techniques in 3 dimensions. By expanding the operators in (16), we obtain the following formulation for the 2-dimensional incompressible Navier-Stokes SRM

$$\begin{pmatrix} u \\ v \end{pmatrix}' = \mu \begin{pmatrix} u_{xx} + u_{yy} \\ v_{xx} + v_{yy} \end{pmatrix} - \begin{pmatrix} p_x^{(\sigma-1)} \\ p_y^{(\sigma-1)} \end{pmatrix} + \frac{1}{\varepsilon} \begin{pmatrix} u_{xx} + v_{yx} \\ u_{xy} + v_{yy} \end{pmatrix} - \begin{pmatrix} uu_x + vu_y \\ uv_x + vv_y \end{pmatrix} + f, \tag{18a}$$

$$p^{(\sigma)} = p^{(\sigma-1)} - \frac{1}{\varepsilon} \left(u_x + v_y \right), \tag{18b}$$

where $u = u^{(\sigma)}$ and $v = v^{(\sigma)}$. We can discretize (18) by first dividing the domain into N_x horizontal and N_y vertical equally spaced mesh lines such that the distance between adjacent vertical mesh lines is h_x and likewise the distance between horizontal mesh lines is h_y. The PDE can then be descritized using the following difference formulas

$$w_{x_b} = \frac{w_{i,j} - w_{i-1,j}}{h_x}, \tag{19a}$$

$$w_{y_b} = \frac{w_{i,j} - w_{i,j-1}}{h_y}, \tag{19b}$$

$$w_{x_b x_f} = \frac{w_{i+1,j} - 2w_{i,j} + w_{i-1,j}}{h_x^2}, \tag{19c}$$

$$w_{y_b y_f} = \frac{w_{i,j+1} - 2w_{i,j} + w_{i,j-1}}{h_y^2}, \tag{19d}$$

$$w_{x_b y_f} = \frac{(w_{i,j+1} - w_{i,j}) - (w_{i-1,j+1} - w_{i-1,j})}{h_x h_y}, \tag{19e}$$

$$w_{y_b x_f} = \frac{(w_{i+1,j} - w_{i,j}) - (w_{i+1,j-1} - w_{i,j-1})}{h_x h_y}, \tag{19f}$$

where w can be either u or v and, for clarity, we use f and b to indicate the use of forward and backward differences, respectively. For the discretization of p, we use forward differences; that is

$$p_{x_f} = \frac{p_{i+1,j} - p_{i,j}}{h_x}, \tag{19g}$$

$$p_{y_f} = \frac{p_{i,j+1} - p_{i,j}}{h_y}. \tag{19h}$$

Finally note that some care must be taken around the boundaries, particularly for the update formula (18b) where it is necessary to update p on the "top" and "right" boundaries because of the choice of forward differences (19g), (19h) in (18a) for the discretization of p_x and p_y.

This discretization is not unique and is not optimal in the sense that the first derivative differences are $\mathcal{O}(h_x)$ or $\mathcal{O}(h_y)$; ideally, we would want $\mathcal{O}(h^2)$ differences since a discretization is generally only as accurate as its lowest-order error term. We expect that higher-order Runge-Kutta methods such as Trapezoidal Rule will produce negligible improvements in error because the error will be dominated by the spatial discretization error.

3.5 Implementation Details

The velocities and pressure data is presented by stacking, row-by-row, the values at the mesh vertices into a vector. Each of the difference schemes above can then be performed over the entire domain with a matrix multiplication by these vectors. Using MATLAB's sparse matrix functionality, this results in a (relatively) fast implementation of (18a) and (18b).

As per the SRM algorithm discussed above, Forward Euler or a *Runge-Kutta* method such as the 2-stage Trapezoidal Rule is used to solve the ODE (18a) and advance the velocity solution in time by a single step. The pressure update formula (18b) is then used to estimate the pressure at the next step. The process is then iterated m times to allow for convergence before advancing to the next time-step.

4 A Predicted Sequential Regularization Method

The SRM is not optimal in the sense that it must be iterated m times: the Predicted Sequential Regularization Method (PSRM) attempts to make a better guess at the initial pressure and then use $m = 1$ (a single iteration of the SRM). Recall that for the SRM, we start each time-step with an arbitrary pressure over the domain. On the assumption that the pressure will change smoothly, the *PSRM with Simple Prediction* uses the pressure distribution from the previous time-step as an initial guess at the current step. It is expected that this simple technique will work well for Navier-Stokes due to the continuous nature of fluids. Also, due to CFL restrictions when using the MOL, the time-steps will likely be relatively small.

5 Results and Discussion

We test the SRM and PSRM on two problems. The first is a constructed problem with an exact solution that appears in [9]. The second appears in [11] and is a well-known benchmark problem for testing numerical Navier-Stokes solvers; it does not have an analytic solution but intuitively we know what to expect (and a visual comparison to figures which appear in [11] can be made).

Problem 1 *This problem is defined on the unit square to have exact solution*

$$u = 50x^2(1-x)^2 y(1-y)(1-2y)\left(1+e^{-t}\right) \tag{20a}$$

$$v = -50y^2(1-y)^2 x(1-x)(1-2x)\left(1+e^{-t}\right) \tag{20b}$$

$$p = \left[-x\left(\frac{x}{2}+2\right) - y\left(\frac{y}{2}-2\right) + \frac{1}{3}\right]\left(1+e^{-t}\right)^2. \tag{20c}$$

The function f as it appears in the Navier-Stokes formulation was derived using MAPLE *from this exact solution. The boundary conditions can be derived from this solution and turn out to be homogeneous Dirichlet conditions.*

Problem 2 (Cavity Problem) *This problem consists of a unit square with homogeneous Dirichlet conditions on the top and sides and a constant unit velocity specified along the bottom. This problem corresponds to the physical situation of a (small) cavity in, say, the wall of a pipe containing a flowing fluid (see Figure 2).*

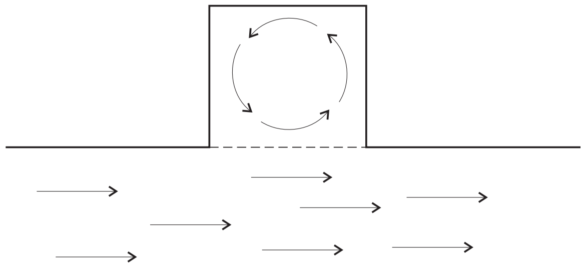

Figure 2: A physical manifestation of the cavity problem.

Using the SRM with first Forward Euler and then Trapezoidal Rule to solve Problem 1 with $\varepsilon = 0.5$ and $\mu = 0.1$ (as in [9]), $N_x = N_y = 16$, and $t_f = 1$ results in errors shown in Table 1. Notice that neither halving the step-size nor using the higher-order

		Forward Euler		Trapezoidal Rule	
k	m	error in $\mathbf{u}$	error in p	error in $\mathbf{u}$	error in p
0.0005	1	<unstable>		<unstable>	
0.0004	1	0.36770	0.74314	0.36770	0.74312
	2	0.07504	0.25840	0.07504	0.25839
	3	0.04522	0.17883	0.04521	0.17883
	4	0.04213	0.16482	0.04213	0.16482
	5	0.04173	0.16232	0.04173	0.16232
	6	0.04167	0.16184	0.04168	0.16184
	7	0.04167	0.16175	0.04167	0.16175
0.0002	1	0.36770	0.74313	0.36770	0.74312
	2	0.07503	0.25839	0.07504	0.25839
	3	0.04521	0.17883	0.04521	0.17883
	4	0.04213	0.16482	0.04213	0.16482
	5	0.04172	0.16232	0.04173	0.16232
	6	0.04167	0.16184	0.04168	0.16184
	7	0.04167	0.16175	0.04167	0.16175

Table 1: Maximum absolute errors in velocity and pressure using the SRM to solve Problem 1. Note these errors are calculated only on the interior because $\mathbf{u}$ will always be correct on the boundaries and p is only partially defined on the boundaries.

Trapezoidal Rule has any significant effect on the errors. A step-size of 0.0005 or larger resulted in an unstable system: the pressure field quickly became unbounded.

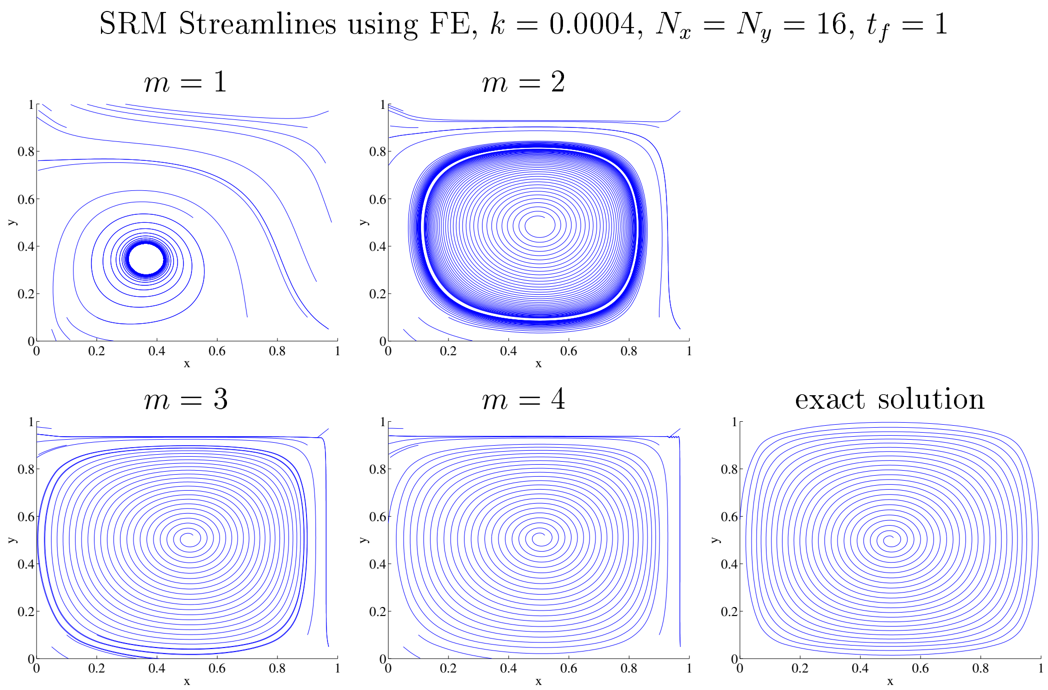

Figure 3: Streamline patterns for Problem 1 as output by the SRM with Forward Euler for various values of m and as calculated based on the exact solution.

The streamline pattern calculated versus the actual streamline pattern is shown in Figure 3. Note that by $m = 3$ and $m = 4$, the patterns seems quite similar (at least away from the boundaries).

Table 2 shows the result of using the PSRM to solve Problem 1 with both Forward Euler and Trapezoidal Rule. By comparing to Tables 1 and 2, we notice that we

method	k	m	error on u	error on p
fe	0.0005	1	<unstable>	
	0.0004	1	<unstable>	
	0.0002	1	0.041637	0.16133
trap	0.0005	1	<unstable>	
	0.0004	1	0.041602	0.16133
	0.0002	1	0.041636	0.16153

Table 2: Maximum absolute errors in velocity and pressure using the PSRM with Forward Euler and Trapezoidal Rule.

achieve a slightly more accurate solution using the PSRM then the SRM with $m = 4$

or even $m = 5$. Because the PSRM with Simple Prediction costs very little more then a single iteration of the SRM, this represents a four- or five-fold reduction in computation expense.

In Table 2, we note a discrepancy in CFL restrictions between the SRM and PSRM. In Table 1 both Forward Euler and Trapezoidal Rule seem to have the same CFL restriction, whereas in Table 2, Forward Euler seems to have a worsened CFL restriction then Trapezoidal Rule. We examine this idea in more depth in Tables 3 and 4. Note that we also add a three stage, second-order Runge-Kutta method known as SSP32 (see, [12]) which has the property of having an improved CFL restriction over Forward Euler and Trapezoidal rule. In fact, for the solution of pure PDEs,

	error on $\mathbf{u}$		
k	Forward Euler	Trapezoidal Rule	SSP32
0.0012	<unstable>	<unstable>	<unstable>
0.0011	<unstable>	<unstable>	4.14926
0.001	<unstable>	<unstable>	0.36770
⋮	⋮	⋮	⋮
0.00049	<unstable>	<unstable>	0.36769
0.00048	3.24156	3.24136	0.36766
0.00047	0.36768	0.36768	0.36768
0.00046	0.36770	0.36769	0.36769

Table 3: CFL restrictions for the SRM using various Runge-Kutta Schemes.

SSP32 has a CFL number of 2 (again see, [12]), or in others words, SSP32 should allow twice as large a time step as Forward Euler. For the SRM, as seen in Table 3, Trapezoidal Rule has the same CFL restriction as Forward Euler (according to [12], Trapezoidal Rule has CFL number 1 so this result is expected). Again as expected, SSP32 allows approximately twice as large a time step as Forward Euler.

However, for the PSRM, the results are less predictable (no pun intended), and we note a worsened CFL restriction for Forward Euler in Table 3 over the SRM case. The CFL restrictions for Trapezoidal Rule and SSP32 have also worsened, though

k	error on $\mathbf{u}$ FE PSRM	TRAP PSRM	SSP32 PSRM
0.00096	<unstable>	<unstable>	<unstable>
0.00095	<unstable>	<unstable>	2.71915
0.00094	<unstable>	<unstable>	0.04152
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0.00047	<unstable>	<unstable>	0.04160
0.00046	<unstable>	0.04159	0.04159
0.00045	<unstable>	0.04161	0.04161
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0.00033	<unstable>	0.04163	0.04163
0.00032	3.48482	0.04162	0.04162
0.00031	0.04162	0.04162	0.04162
0.0003	0.04163	0.04163	0.04163

Table 4: CFL restrictions for the PSRM using various Runge-Kutta Schemes.

not as significantly.

As in [11], the cavity problem was discretized with $N_x = N_y = 64$. Also as per [11] a Reynolds number of 100 (i.e., $\mu = 0.01$) was chosen. Figures 4 and 5 show the results of solving the problem to $t = 45$ using the PSRM with Forward Euler. Larger step-sizes than $k = .00002$ were unstable due to CFL restrictions. The results appear reasonable and match those generated by the code used in [11].

6 Conclusions

The method of lines is a powerful technique for numerically replaced PDEs with large systems of ODEs. The sequential regularization method is a new numerical methods for solving particular types of DAEs. Combining the two creates a useful and intuitive method for the numerical solution of incompressible Navier-Stokes equations.

Some care should be taken in choosing a spatial discretization as a poor choice will prevent the usage of the SRM/PSRM to their full potentials. Specifically, the

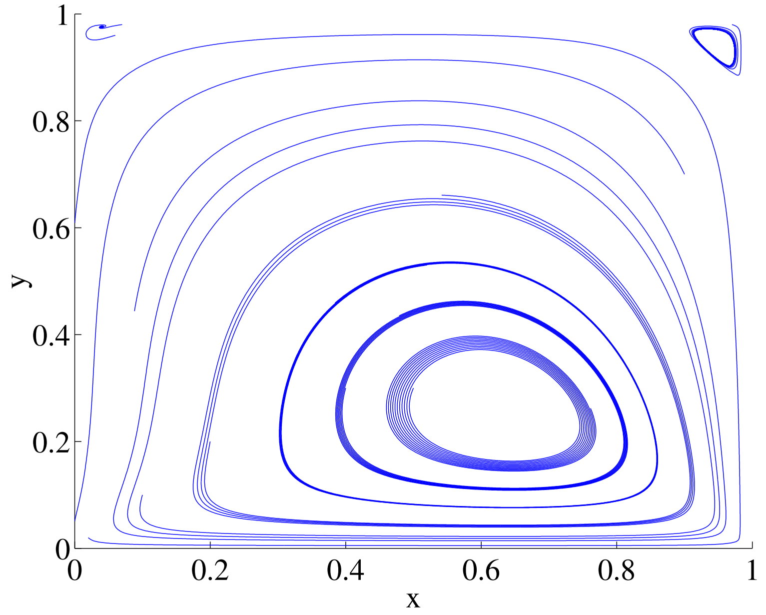

Figure 4: Streamline pattern of the velocity field over the cavity domain at $t = 45$ for a fluid with $\mu = 0.01$. Solved using the PSRM with Forward Euler, $N_x = N_y = 64$, $\varepsilon = 0.5$, and $k = 0.00002$.

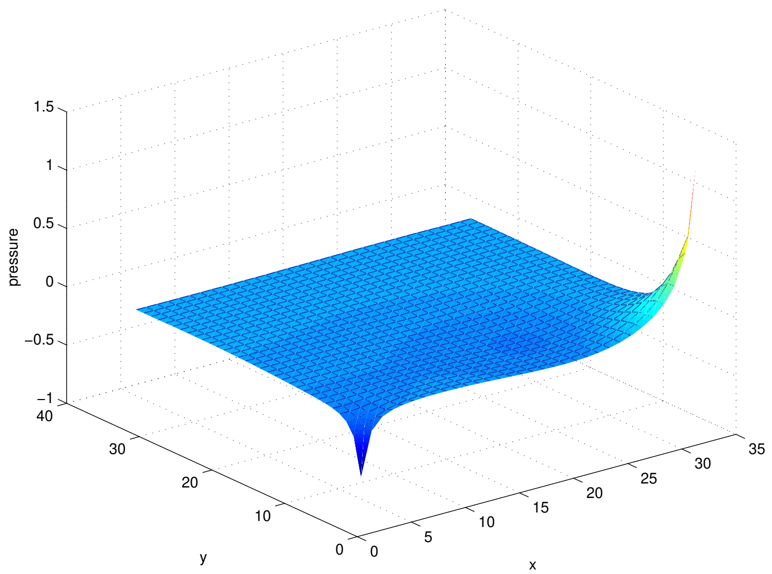

Figure 5: The pressure over the cavity domain at $t = 45$ for a fuild with $\mu = 0.01$. Solved using the PSRM with Forward Euler, $N_x = N_y = 64$, $\varepsilon = 0.5$, and $k = 0.00002$.

previously presented discretization limits the ODE solver to Forward Euler in the sense that using even a second-order Runge-Kutta offers no additional accuracy.

The use of a predicted sequential regularization method over the sequential regularization method results in a substantial reduction in computational expense. Our results seem to indicate that this reduction may be as much as five-fold for a given fixed time step size. However, for the problem presented, using the PSRM with Forward Euler required a 66% smaller time step so the effective savings is slightly more than a three-fold reduction in computation expense.

It is interesting to note that, with respect to the PSRM with Forward Euler, the SSP32 scheme has an effective CFL number of 3. Thus, with the PSRM, it costs no more to use SSP32 over Forward Euler and we achieve an second-order solution. Hence using a higher-order (for example, $\mathcal{O}(h^2)$) spatial discretization with the PSRM and SSP32 may produce a more accurate solution than Forward Euler for very little or no additional computation expense.

Bibliography

[1] U. Ascher, H. Chin, L. Petzold, and S. Reich, *Stabilization of constrained mechanical systems with DAEs and invariant manifolds*, J. Mech. Structures Mach., **23** (1995), 135–157.

[2] U. Ascher, H. Chin, and S. Reich, *Stabilization of DAEs and invariant manifolds*, Numer. Math., **67** (1994), 131–149.

[3] U. Ascher and P. Lin, *Sequential regularization methods for higher index DAEs with constraint singularities: Linear index-2 case*, SIAM J. Numer. Anal. **33** (1996), 1921–1940.

[4] U. Ascher and P. Lin, *Sequential regularization methods for nonlinear higher-index differential-algebraic equations*, SIAM J. Sci. Comput. **18** (1997), 160–181.

[5] B. Baird, M. Ball, S. DuBois, and R. Wellwood, *The uniform flow of a fluid about an infinitely long cyclinder*, Math 3743 Final Project, 2000.

[6] J. Baumgarte, *Stabilization of constraints and integrals of motion in dynamical systems*, Comp. Math. Appl. Mech. Eng. **1** (1972), 1–16.

[7] K. Brenan, S. Campbell, L. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*, Elsevier Science Publishers, New York, 1989.

[8] C. Macdonald, *The predicted sequential regularization method for differential-algebraic equations*, Undergraduate Honours Thesis, 2001.

[9] P. Lin, *A sequential regularization method for time-dependent incompressible Navier-Stokes equations*, SIAM J. Numer. Anal. **34** (1997), 1051–1071.

[10] P. Lin and R. Spiteri, *A predicted sequential regularization method for index-*2 *Hessenberg DAEs*, SIAM J. Numer. Anal. to appear

[11] P. Lin, Q. Guo, and X. Chen, *A sequential regularization formulation for incompressible flow computation*, under review.

[12] R. Spiteri and S. Ruuth, *A new class of optimal high-order strong-stability-preserving time discretization methods*, under review.