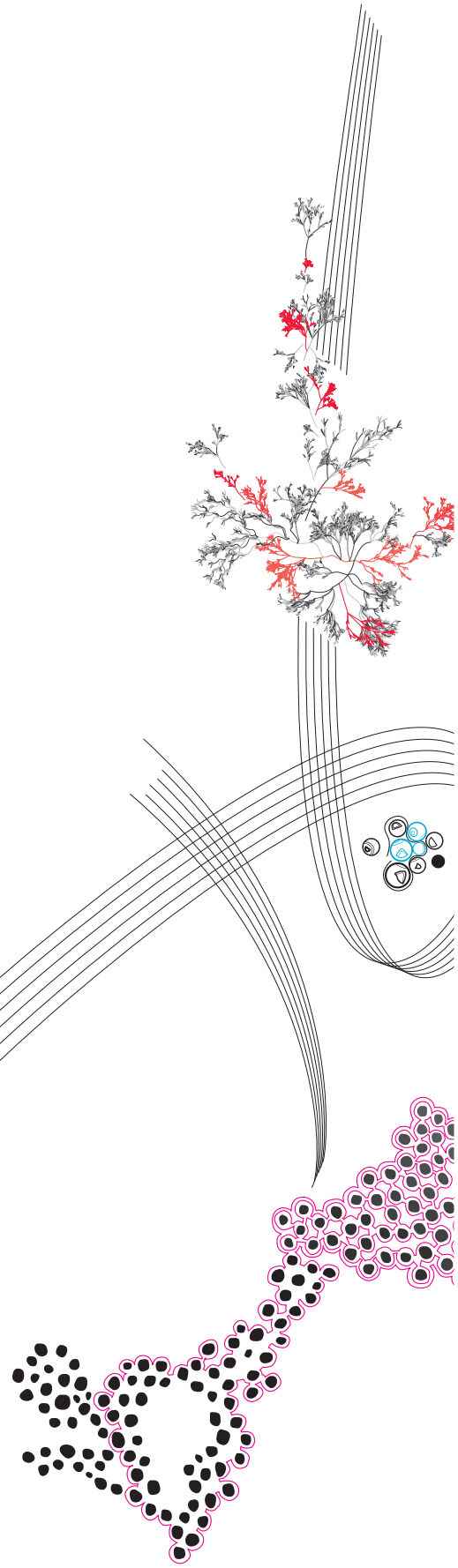

BSc Thesis Applied Mathematics

An analysis of Galerkin methods for solving transport problems

Maik Overmars

Supervisor: Matthias Schlottbom

June, 2019

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

I want to thank my supervisor for his help during the assignment by answering questions, providing insights and giving general advice.

An analysis of Galerkin methods for solving transport problems

Maik Overmars

June, 2019

Abstract

An analysis is performed of Galerkin methods for solving a one-dimensinal transport problem. In particular, a discontinuous Galerkin and continuous Petrov-Galerkin method are analyzed and compared. The methods both find a piecewise polynomial as a solution with polynomials of order k. For both methods it is shown that they are strongly A-stable. Furthermore, an error analysis is done in which it was obtained that the error in each method is of order $O(h^{\min(k+1,p)})$ with p as the order of a numerical quadrature. Finally, a numerical analysis is done to compare the results of the analysis to numerical results.

Keywords: Galerkin, DG, Petrov-Galerkin, CPG, Differential equation, ODE, Transport problem

1 Introduction

The radiative transfer equation is a differential equation which describes the propagation of radiation trough a domain. The radiation is affected by absorption, emission and scattering, of which the effects are included in the differential equation. Furthermore, the solution to the equation is a function of position and direction. In this article we look at a reduced problem where the direction is fixed and only the spatial discretization is taken into account. In addition, the terms responsible for absorption, emission and scattering will be combined into a function f. These terms are non-linear in general, so f can also depend on the solution. Thus, the transport problem for fixed direction s and function f that will be analysed is defined on a domain Ω and is given by

$$\begin{aligned} s \cdot \nabla u &= f \quad \text{on } \Omega \\ u &= g \quad \text{on } \partial_{in}\Omega, \end{aligned} \tag{1}$$

with g defined on $\partial_{in}\Omega$, the part of the boundary for which s points inwards. In this paper, we wish to develop Galerkin methods to find an approximate solution to this equation. This will be done for the one-dimensional case. Let's however first consider a two-dimensional domain. In this case, the problem is illustrated in figure 1. Equation (1) then holds for $s = (\mu, \nu)$ position (x, y).

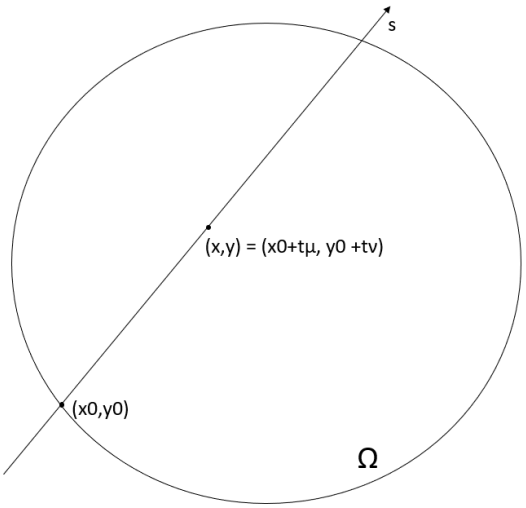

FIGURE 1: The transport problem along direction s

Now, if we were to fix the point (x, y), we can trace back along fixed direction $s = (\mu, \nu)$ to the point at (x_0, y_0) on the inflow boundary, as also depicted in figure 1. Along this line we essentially have a one-dimensional problem, since for any point along this line we have $(x, y) = (x_0 + t\mu, y_0 + t\nu)$ for $t \in [0, T]$. Let's now define the function ϕ as:

$$\phi(t) := u(x, y, \mu, \nu) = u(x_0 + t\mu, y_0 + t\nu, \mu, \nu), \tag{2}$$

which only depends on t. This function ϕ can be manipulated to obtain

$$\begin{aligned} \frac{d\phi}{dt} &= u_x\mu + u_y\nu = s \cdot \nabla u \qquad \text{and,} \\ f(x, y, \mu, \nu, u) &= f(x_0 + t\mu, y_0 + t\nu, \mu, \nu, \phi(t)) = f(t, \phi(t)). \end{aligned}$$

We recognize the first equation as the directional derivative. Also note that f is non-linear here, in this article f will be non-linear for the one-dimensional methods. Combining these, we observe that equation (1) changes to $\phi'(t) = f(t, \phi(t))$, which is an ODE of a single variable. As for the initial condition, this is just $\phi(0) = u(x_0, y_0, \mu, \nu) = g(x_0, y_0)$. Thus when looking along a single line in a two-dimensional domain, we obtain an ODE.

Similarly, when our domain is one-dimensional, so an interval, we also get such an ODE. In one dimension there are only 2 directions, to the left and to the right. If we take the direction to the right (or upwind), then $s = +1$ and the gradient reduces to a derivative of x. Since the inward boundary is only 1 point, we get initial condition $u(x_0) = g(x_0)$. Thus the one dimensional problem described in this paper will be over an interval $[x_0, x_N]$ for the equation:

$$\begin{aligned} u'(x) &= f(x, u(x)) \\ u(x_0) &= u_0. \end{aligned} \tag{3}$$

For this problem, it is assumed that $f(x, w) \in C^r$ in $[x_0, x_N] \times \mathbb{R}$ with $r \geq 1$ such that f is Lipschitz continuous. This also means that $u(x) \in C^{r+1}[x_0, x_N]$.

In the past numerous methods have been developed and analyzed for solving this differential equation, such as Euler methods and similar one-step methods, Runge-Kutta

methods, multi-step methods and more. Hairer and Wanner [5] detailed some of these methods.
In this paper, Galerkin methods will be analysed for solving (3). In previous work Hulme [7] and Lesaint/Raviart [8] developed continuous and discontinuous Galerkin methods respectively to solve this transport problem in one dimension. This article will analyse and compare some of these Galerkin methods. For the extension to the second dimension, Demkowizch [3] developed a discontinuous Petrov-Galerkin method. There is still on-going development on these types of Galerkin methods, recently Bause and Köcher [2] developed a method with post-processing to improve the Galerkin approximation.

2 Methods

The goal in Galerkin methods is to find an approximate solution to the weak form of the differential equation. Such a weak form is obtained by multiplying the differential equation by a test function and integrating over the domain. In this is done in the one-dimensional case as described by equation (3), we obtain:

$$\int_{x_0}^{x_N} u'(x)v(x)dx = \int_{x_0}^{x_N} f(x,u(x))v(x)dx$$
$$R(u,v) = \int_{x_0}^{x_N} (u' - f(x,u(x)))v(x) = 0 \quad \text{for all } v \in V. \tag{4}$$

The solution $u \in U$ should satisfy $R(u,v) = 0$ for all $v \in V$ with V the space of test functions. In Galerkin methods we choose subspaces $U_h \subset U$ and $V_h \subset V$ and look for a solution $u_h \in U_h$, which holds for all $v_h \in V_h$. For example, since the actual solution is continuous, we can choose U_h to be a polynomial space.

In this paper, we will seek solutions to a modified weak form. First, we define a uniform mesh $x_n = x_0 + nh, \quad 0 \leq n \leq N$ and subsequently we introduce a mesh dependent weak form.

$$R_h(u,v) = \sum_{n=0}^{N-1} (u(x_{n+}) - u(x_{n-}))v(x_n) + \int_{x_n}^{x_{n+1}} [u'(x) - f(x,u(x))]v(x)dx \tag{5}$$

Where we let $u(x_{0-}) = u_0$. This weak form allows discontinuous solutions with $(u(x_{n+}) - u(x_{n-}))$ denoting the jump term. Using this weak form, a solution can be found per interval. For each method, we will approximate the solution by a polynomial of degree k on each interval which we will denote as $I_n = [x_n, x_{n+1}]$. This polynomial approximation means that an error is introduced between the actual solution and the approximate solution, since we only construct a solution in the finite-dimensional space P_k, while the actual solution exists as a continuous function on I_n. For basis functions $\phi_{n,j}$ of P_k the approximate solution satisfies

$$y(x) = \sum_{j=1}^{k+1} b_j \phi_{n,j}(x) \quad 0 \leq n \leq N-1, \quad x \in I_n. \tag{6}$$

In this paper multiple variations of Galerkin methods for solving this weak form will be analysed. First of all, Galerkin methods and Petrov-Galerkin methods can be considered. In Galerkin methods we have that $U = V$, so the trial space where we find our solution

is the same as the test space. Meanwhile, in Petrov-Galerkin methods it is the case that $U \neq V$. An advantage of Petrov-Galerkin is that we have more flexibility in the method, since both the test space and trial space can be modified to yield better stability and smaller errors. In Galerkin methods it is an advantage that both u and v can be expanded as basis from the same space. In continuous methods we require that the approximate solution u_h is continuous, while in discontinuous methods the final solution can have discontinuities. The continuous method places restrictions on the solutions on the subdomains, such that the overall solution is continuous. This means that discontinuous methods have more degrees of freedom.

2.1 Discontinuous Galerkin Method

First a discontinuous method will be given for solving the problem given by equation (3). In such a discontinuous method, the approximate solution can have discontinuities at the grid points of the mesh. The solution is approximated by a polynomial as given in (6). Since we don't require continuity, the approximate solution is a piecewise polynomial function. The test functions will also be piecewise polynomial functions, so our method is a Galerkin method. Then, using the weak form as defined by (5), we require that the solution in this method needs to satisfy the following equation on each interval.

$$(y(x_{n-}) - y(x_{n+}))v(x_n) + \int_{x_n}^{x_{n+1}} (y'(x) - f(x, y(x)))v(x)dx = 0 \quad \text{for all } v \in P_k \quad (7)$$

This is a consistent method, since if we take $y = u$, then the jump term vanishes and y will be a solution to the initial weak form given by (4). The integral can't be calculated exactly in general, so an interpolatory quadrature formula is used.

$$\int_{x_n}^{x_{n+1}} q(x)dx = h\sum_{i=1}^{k+1} w_i q(x_{n,i}) + O(h^{p+1}) \quad (8)$$
$$x_{n,i} = x_n + \xi_i h \quad 1 \leq i \leq k+1$$

Here w_i and ξ_i are the weigths and abscissae associated with the quadrature. The function $q(x)$ should be sufficiently smooth for the quadrature to provide a good approximation. Since there are $k+1$ nodes, we have error order $k+1 \leq p \leq 2k+1$ depending on quadrature. This only holds if $r \geq 2k+2$, since then the integrand in (7) is sufficiently smooth. Furthermore, the quadrature is exact for polynomials up to degree p. The introduction of this quadrature rule has the consequence that the method is no longer consistent. To see this, let's first fill in the quadrature rule into (7).

$$(y(x_{n-})-y(x_{n+}))v(x_n)+h\sum_{i=1}^{k+1} w_i(y'(x_{n,i})-f(x_{n,i}, y(x_{n,i})))v(x_{n,i}) = 0 \quad \text{for all } v \in P_k. \quad (9)$$

It is clear that for $y = u$, we don't obtain the same weak form as in (4). Instead, the quadrature rule introduces a consistency error of order p.

The solution $y(x)$ can be found per interval by filling in (6) and letting v be the basis functions $\phi_{n,i}, \quad 1 \leq i \leq k+1$ of P_k, since equation (7) will hold for any function in P_k if it holds for each basis function. Then a system of equations is obtained which can be solved to obtain the coefficients of the polynomial approximation. For purposes of further stability analysis, it would be desirable to obtain a one-step method. Lesaint and Raviart

[8] showed that the discontinuous Galerkin method as described in this section is equal to an implicit Runge-Kutta scheme. If we define $x_{n-} = x_n$ and $x_{n+} = x_{n,1}$, the following scheme is obtained

$$
\begin{aligned}
&y_{n,i} = y_n + h\sum_{j=1}^{k+1} a_{ij} f(x_{n,j}, y_{n,j}) \quad 1 \le i \le k+1 \\
&y_{n+1} = y_n + h\sum_{j=1}^{k+1} w_j f(x_{n,j}, y_{n,j}) \quad \text{where} \\
&a_{i1} = w_1 \quad 1 \le i \le k+1 \\
&a_{ij} = \int_0^{\xi_i} l_j(x)dx - w_1 l_j(\xi_1) \quad 1 \le i \le k+1, \quad 2 \le j \le k+1 \\
&l_j(x) = \prod_{i=2, i\neq j}^{k+1} \frac{x-\xi_i}{\xi_j - \xi_i}, \quad 2 \le j \le k+1.
\end{aligned}
\tag{10}
$$

By calculating the matrix A in (10), we obtain a non-linear system on each interval I_n, which can be solved to obtain to obtain values at the points $y_{n,i}$. The solution $y(x)$ on this interval will be the polynomial interpolation of the points $y_{n,i}$. Of course, we wish to know how accurate and stable this approximate solution is, so the following sections will describe a stability and error analysis.

2.1.1 Stability analysis

Since the Runge-Kutta scheme of (10) is a one-step method, we can determine the stability of the discontinuous Galerkin method by analyzing if the one-step method is stable. In this section we will prove that the one-step method is strongly A-stable.

Definition 1. *A one-step method is A-stable if all it's solutions tend to 0 as as $N \to \infty$, when the method is applied with fixed step size h to the differential equation $u' = \lambda u$ with $Re(\lambda) < 0$.*

When a one-step method is applied to the differential equation $u' = \lambda u$, it can be rewritten in the form $y_{n+1} = E(\lambda h)y_n$ for $0 \le n \le N$, where $E(\lambda h)$ is a rational function. Instead of showing A-stability, we will show that the one-step method is strongly A-stable. The definition for strong A-stability depends on constraints for the function $E(\lambda h)$.

Definition 2. *A one-step method is strongly A-stable if $y_{n+1} = E(\lambda h)y_n$, $|E(\lambda h)| < 1$ for $Re(\lambda h) < 0$ and $|E(\lambda h)| \to 0$ for $Re(\lambda h) \to -\infty$.*

This function $E(\lambda h)$ approximates $\exp(\lambda h)$. Lesaint and Raviart [8] showed that the one-step method has accuracy of order $p = 2k + 1$ if we choose Gauss-Radau weigths and abscissae. Then, because $E(\lambda h)$ is of order $2k + 1$ and is a rational approximation to $\exp(\lambda h)$, it must be equal to $P_{k+1,k}(\lambda h)$, the Padé rational approximation of $\exp(\lambda h)$. This Padé approximant is a rational function, with the numerator being a polynomial of degree k and the denominator being a polynomial of degree $k + 1$. It is the best rational approximation to a function. Axelsson [1] showed that $P_{k+1,k}(\lambda h)$ satisfies the conditions in the definition of strong A-stability, so our method is strongly A-stable.

2.1.2 Error bounds

In the discontinuous method two error terms arose, namely an error term for the polynomial approximation and for the quadrature. In this section error bounds will be given for the approximate solution found by (10). First, the following theorem will hold.

Theorem 1. *Assume $f(x, y(x))$ is Lipschitz continuous in $[x_0, x_N] \times \mathbb{R}$ and denote L the the Lipschitz constant for f in that region. Given our solution by the discontinuous Galerkin method as described by* (10) *and the actual solution to the problem given in* (4)*, then the following bound holds: $|u_n - y_n| \leq Kh^p$.*

Proof. The truncation error τ_n on each interval equals

$$\tau_n = u_{n+1} - u_n - h\sum_{i=1}^{k+1} w_i f(x_{n,i}, y(x_{n,i})) = \int_{x_n}^{x_{n+1}} f(s, u(s))ds - h\sum_{i=1}^{k+1} w_i f(x_{n,i}, y(x_{n,i})), \tag{11}$$

where we used the one-step method of (10). Now using our quadrature rule this implies $|\tau_n| \leq Bh^{p+1}$. Henrici [6] proved that this implies that $|u_n - y_n| \leq Kh^p$ and the theorem is proven. □

The following theorem will give a general error bound for the approximate solution $y(x)$.

Theorem 2. *Assume the assumptions of theorem 1 hold. Then the following error bound is obtained:* $\max_{x_0 \leq x \leq x_N} |u(x) - y(x)| \leq Ch^{\min(p,k+1)}$.

Proof. First of all, we take $x \in I_n$ and introduce the polynomial interpolation of $u(x)$ using the quadrature points.

$$u_h(x) = \sum_{i=1}^{k+1} l_j(x) u(x_{n,i}) \tag{12}$$

For this approximation we know that $|u_h(x) - u(x)| \leq O(h^{k+1})$. Now using the triangle inequality we obtain $\max_x |u(x) - y(x)| \leq \max_x |u(x) - u_h(x)| + \max_x |u_h(x) - y(x)|$. The polynomial approximation $y(x)$ is also just the interpolating polynomial through the points $y_{n,i}$ as given by (10). Thus we obtain

$$\max_x |u_h(x) - y(x)| = \max_x |\sum_{i=1}^{k+1} l_i(x)(u_{n,i} - y_{n,i})| \leq \max_x \sum_{i=1}^{k+1} |l_i(x)||u_{n,i} - y_{n,i}| \tag{13}$$

Now using (10) and the Lipschitz continuity of $f(x, u(x))$

$$\begin{aligned} |u_{n,i} - y_{n,i}| &\leq |u_n - y_n| + h|\sum_{j=1}^{k+1} a_{ij}[f(x_{n,i}, u(x_{n,i})) - f(x_{n,i}, y(x_{n,i}))]| \\ &\leq |u_n + y_n| + hL|u(x) - y(x)| \sum_{j=1}^{k+1} |a_{ij}| \leq |u_n + y_n| + hLB|u(x) - y(x)|. \end{aligned} \tag{14}$$

Here we choose B such that $\max_i \sum_{j=1}^{k+1} a_{ij} \leq B$. This is filled into (13) to get

$$\max_x |u_h(x) - y(x)| \leq (|u_n + y_n| + hLB|u(x) - y(x)|) \max_x \sum_{i=1}^{k+1} |l_i(x)| \leq F|u_n - y_n| + hLBF \max_x |u(x) - y(x)|,$$

(15)

where F is defined such that $F \geq \max_x \sum_{i=1}^{k+1} |l_i(x)|$. Using the triangle equality, (15), and theorem 1 the final result is obtained.

$$\max_x |u(x) - y(x)| \leq O(h^{k+1}) + F|u_n - y_n| + hLBF \max_x |u(x) - y(x)| \tag{16}$$
$$\max_x |u(x) - y(x)| \leq \frac{F}{1 - hLBF}|u_n - y_n| + O(h^{k+1}) \leq Ch^{\min(k+1,p)}$$

For the step size it is required that $h \leq (LBF)^{-1}$. Since this holds for any interval this will also hold on the whole domain. □

2.2 Continuous Petrov-Galerkin Method

Following the discontinuous method given in the previous sections, a continuous method will now be given. Again, just like in the discontinuous method, $y(x)$ will be approximated on each interval I_n by a polynomial of degree k as given by (6). However, now we also require that the approximate solution is continuous over the whole domain. Thus, we have a continuity requirement at the grid points.

$$y(x_{0+}) = u_0 \tag{17}$$
$$y(x_{n+}) = y(x_{n-}) \quad 1 \leq n \leq N-1$$

In this continuous method we use the weak form as given by (5) and because of the continuity the jump terms vanish. Thus, we require the solution to satisfy the following equation

$$\int_{x_n}^{x_{n+1}} (y'(x) - f(x, y(x)))v(x)dx = 0 \quad \text{for all } v \in P_k. \tag{18}$$

This method is consistent, since the solution u to the initial problem of (4) is also a solution of equation (18). The test functions v are polynomials of degree k on each interval, but don't need to be continuous. So the test functons are piecewise polynomial functions. The solution y does need to be continuous, and equals a polynomial of degree k on each interval. As a result, the test and trial space are different and the method is a Petrov-Galerkin method.

To find an approximate solution $y(x)$ to equation (18) we create a system of equations by filling in equation (6) into equation (18) and letting v be the first k basis function to obtain a system of equations.

$$\int_{x_n}^{x_{n+1}} \sum_{j=1}^{k+1} b_j \phi'_{n,j}(x)\phi_{n,i}(x)dx = \int_{x_n}^{x_{n+1}} f(x, \sum_{j=1}^{k+1} b_j \phi_{n,j}(x))\phi_{n,i}(x)dx \quad 1 \leq i \leq k, \quad 0 \leq n \leq N-1 \tag{19}$$

$$\sum_{i=j}^{k+1} b_j \int_{x_n}^{x_{n+1}} \phi'_{n,j}(x)\phi_{n,i}(x)dx = \int_{x_n}^{x_{n+1}} f(x, \sum_{j=1}^{k+1} b_j \phi_{n,j}(x))\phi_{n,i}(x)dx \quad 1 \leq i \leq k, \quad 0 \leq n \leq N-1$$

This is a system of k equations with $k+1$ unknowns. For the final equation we fill in (6) into continuity requirement (17) to obtain

$$\sum_{i=1}^{k+1} b_{n,i}\phi_{n,i}(x_n) = y(x_n) \quad 0 \leq n \leq N-1. \tag{20}$$

Finally, an interpolatory quadrature formula is introduced to approximate the integral on the right hand side of (19). The integral on the left contains only polynomials and can thus be calculated exactly and thus we don't need the quadrature there.

$$\int_{x_n}^{x_{n+1}} q(x)dx = h\sum_{i=1}^{k} w_i q(x_{n,i}) + O(h^{p+1})$$
$$x_{n,i} = x_n + \xi_i h \quad 1 \le i \le k$$

Note that this time there are k nodes, instead of k+1 in the discontinuous method. We require $q(x)$ to be 2k times differentiable for the quadrature to have error $O(h^{p+1})$, so we should have that $r \ge 2k$. In this case, $k \le p \le 2k-1$ depending on quadrature and for polynomials up to degree p the quadrature is exact. Again this quadrature means a consistency error of order p is introduced. Combining this quadrature rule with (19) and (20) we obtain the following system on each interval I_n.

$$\begin{aligned}
A^{(n)}b &= c^{(n)}(b) \quad \text{with} \\
A^{(n)}_{1,j} &= \phi_{n,j}(x_n) \quad 1 \le j \le k+1 \\
A^{(n)}_{i,j} &= \int_{x_n}^{x_{n+1}} \phi'_{n,j}(x)\phi_{n,i-1}(x)dx \quad 2 \le i \le k+1, \quad 1 \le j \le k+1 \qquad (21)\\
c^{(n)}_1(b) &= y_n \\
c^{(n)}_i(b) &= h\sum_{m=1}^{k} w_m f\Big(x_{n,m}, \sum_{j=1}^{k+1} b_j \phi_{n,j}(x_{n,j})\Big)\phi_{n,i-1}(x) \quad 2 \le i \le k+1
\end{aligned}$$

In this form it is required to calculate a matrix $A^{(n)}$ on each interval. However, for appropriate choice of basis it will be shown that only one matrix needs to be calculated for this method. This is done via integration over the reference element $K = [0,1]$. Let's first define a basis $\hat{\phi}_i$ on this element. Then for each interval I_n, let $\phi_{n,i}$ be the shifted version of ϕ_i. If we introduce the change of variable $\hat{x} = \frac{x-x_n}{x_{n+1}-x_n}$, then we have $\hat{\phi}_i(\hat{x}) = \phi_{n,i}(x)$. Furthermore, $d\hat{x} = dx/(x_{n+1}-x_n) = dx/h$ such that the matrix of equation (21) becomes

$$\begin{aligned}
A^{(n)}_{1,j} &= \phi_{n,j}(x_n) = \hat{\phi}_j(0) \qquad 1 \le j \le k+1 \\
A^{(n)}_{i,j} &= \int_{x_n}^{x_{n+1}} \phi'_{n,j}(x)\phi_{n,i-1}(x)dx = h\int_0^1 \hat{\phi_j}'(\hat{x})\hat{\phi_{i-1}}(\hat{x})d\hat{x} \qquad (22)\\
&2 \le i \le k+1, \quad 1 \le j \le k+1, \quad 0 \le n \le N-1.
\end{aligned}$$

Since the matrix doesn't depend on the interval, it is the same on each interval so we only have to calculate it once. Combing (21), (22), we get the following system of equations:

$$\begin{aligned}
A_{1,j} &= \hat{\phi}_j(0) \quad 1 \le j \le k+1 \qquad (23)\\
A_{i,j} &= h\int_0^1 \hat{\phi_j}'(x)\hat{\phi_{i-1}}(x)dx \qquad 2 \le i \le k+1, \quad 1 \le j \le k+1 \\
c^{(n)}_1(b) &= y_n \\
c^{(n)}_i(b) &= h\sum_{m=1}^{k} w_m f\Big(x_{n,i}, \sum_{j=1}^{k+1} b_j \phi_{n,j}(x)\Big)\phi_{n,i-1}(x) \quad 2 \le i \le k+1 \quad 0 \le n \le N-1.
\end{aligned}$$

So we get the non-linear system $\mathbf{Ab} = c^{(n)}(\mathbf{b})$ on each interval I_n. By solving for $\mathbf{b}$ we obtain the coefficients for the basis polynomials, and taking the solution y as a linear combination of the basis functions with these coefficients, we obtain the solution. In the next sections, a stability and error analysis will be described.

2.2.1 Stability analysis

In this section we will prove that the method as described in the previous section is strongly A-stable. However, if we wish to have a similar analysis as for the discontinuous method, our solution needs to be described as a one-step method. Hulme [7] showed that this Galerkin method produces the same solution as a collocation and interpolatory quadrature method, under the restriction that the step size h is sufficiently small. In this case, the Galerkin solution satisfies

$$hy'(x_{n,i}) = hf(x_{n,i}, y(x_{n,i})) \quad 1 \leq i \leq k, \quad 0 \leq n \leq N-1, \tag{24}$$

which is the requirement for the solution to collocate at the points $x_{n,i}$. Combined with the continuity requirement (17), we obtain that solution must satisfy:

$$\begin{aligned} y(x) &= y_n + \int_{x_n}^{x} y'(s)ds = y_n + \int_{x_n}^{x} \sum_{i=1}^{k} l_i(s) f(x_{n,i}, y(x_{n,i}))ds \\ &= y_n + \sum_{i=1}^{k} \int_{x_n}^{x} l_i(s)ds f(x_{n,i}, y(x_{n,i})) \quad x_n \leq x \leq x_{n+1}, \\ l_i(s) &= \prod_{j=1, i\neq j}^{k} \frac{x - x_{n_j}}{x_{n,i} - x_{n,j}}. \end{aligned} \tag{25}$$

The first equality follows from the continuity, while the second equality is true since $y'(s)$ is a polynomial of degree k-1 which passes through k collocation points and thus can be interpolated by lagrange polynomials. Then (24) is used and we obtain the result. Using this, it is clear that we acquire a one-step method by taking $x = x_{n+1}$.

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=1}^{k} w_i f(x_{n,i}, y(x_{n,i})) \quad \text{with} \\ w_i &= h^{-1} \int_{x_n}^{x_{n+1}} l_i(s)ds \end{aligned} \tag{26}$$

Subsequently, let's show that this one-step method is strongly A-stable. Again, for the problem of $u' = \lambda u$ we may write the approximate solution as $y_{n+1} = E(\lambda h)y_n$, where $E(\lambda h)$ is a rational function. For strong A-stability, definition 2 is used. In the obtained method, the quadrature can still be chosen freely. We choose to use the Gauss-Legendre quadrature, since this is the most accurate quadrature and is of order $O(h^{2k})$. Ehle [4] has proven that for this quadrature, $E(\lambda h) = P_{n,n}(\lambda h)$, where $P_{n,n}(\lambda h)$ is the nth diagonal Padé approximation to $\exp(\lambda h)$. Furthermore, he showed that $P_{n,n}(\lambda h)$ satisfies the conditions of strong A-stability in definition 2. Thus, given one-step method (26) and Gauss-Legendre quadrature, the continuous Petrov-Galerkin method is strongly A-stable.

2.2.2 Error bounds

In this section error bounds will be derived for the approximate solution that was found using this Petrov-Galerkin method. We note that errors arise in the method from two sources, namely a consistency error from the quadrature and an error since we approximate with a polynomial. First of all we repeat theorem 1 of section 2.1.2, but in this case with the one-step method of equation (26). With similar analysis it can be shown that $|u_n - y_n| \leq Kh^p$. Let's now give the main theorem.

Theorem 3. *Assume $f(x, y(x))$ is Lipschitz continuous in $[x_0, x_N] \times \mathbb{R}$ and denote L the the Lipschitz constant for f in that region. Let h be sufficiently small such that equations* (26) *and* (25) *hold. Given the solution $y(x)$ of the Petrov-Galerkin method described by* (25) *and the actual solution $u(x)$ to the problem given in* (5)*, then the following error bound holds:* $\max_{x_0 \leq x \leq x_N} |u(x) - y(x)| \leq Ch^{\min(p,k+1)}$.

Proof. First of all we let $x \in I_n$ and we define $u_h(x)$, the k point polynomial approximation to $u(x)$.

$$u_h(x) = u_n + \sum_{i=1}^{k} f(x_{n,i}, u(x_{n,i})) \int_{x_n}^{x} l_i(s)ds \tag{27}$$

This is defined similarly to (25) and again holds since u satisfies the differential equation and is continuous at the grid points. Now the triangle equality is used to split up the error into two parts.

$$\max_x |u(x) - y(x)| \leq \max_x |u(x) - u_h(x)| + \max_x |u_h(x) - y(x)| \tag{28}$$

The first part is just the polynomial interpolation error for which we have $|u(x) - u_h(x)| \leq O(h^{k+1})$. For the second part, we use $y(x)$ as defined in (25) to get

$$\begin{aligned}
\max_x |u_h(x) - y(x)| &\leq |u_n - y_n| + \max_x |\sum_{i=1}^{k} \int_{x_n}^{x} l_i(s)ds f(x_{n,i}, u(x_{n,i})) - f(x_{n,i}, y(x_{n,i}))| \\
&\leq |u_n - y_n| + L \max_x |u(x) - y(x)| \sum_{i=1}^{k} \max_x |\int_{x_n}^{x} l_i(s)ds| \\
&\leq |u_n - y_n| + hBL \max_x |u(x) - y(x)|
\end{aligned} \tag{29}$$

Here the Lipschitz continuity of f with Lipschitz constant L was used. Furthermore, we let $\max_x |\int_{x_n}^{x} l_i(s)ds| \leq hB$. Now combining (28) and (29) we obtain

$$\begin{aligned}
\max_x |u(x) - y(x)| &\leq O(h^{k+1}) + |u_n - y_n| + hBL \max_x |u(x) - y(x)| \\
\max_x |u(x) - y(x)| &\leq O(h^{k+1}) + \frac{1}{1 - hBL} |u_n - y_n|,
\end{aligned} \tag{30}$$

where we require that $h \leq (BL)^{-1}$. From theorem 1 we know that $|u_n - y_n| \leq Kh^p$ to finally obtain

$$\max_x |u(x) - y(x)| \leq Ch^{\min(p,k+1)}. \tag{31}$$

Since this holds on every interval, it holds on the whole domain and the result is proven. □

3 Numerical studies

3.1 Implementation

This section is devoted to the implementation of the schemes in Matlab and the efficiency of said schemes. Generally, there are more degrees of freedom in discontinuous methods than continuous ones. This can lead to larger systems of equations and thus more time spent computing a solution. The discontinuous Galerkin method in fact has a degree of

freedom more than the continuous Petrov-Galerkin method, since with the quadrature we get to choose $k+1$ nodes instead of k. However, the systems are of similar size, since both require calculating a matrix of size $k+1$. This is because in the continuous Petrov-Galerkin method, we have an equation dedicated to the continuity requirement. Also, both methods require solving a non-linear system on each interval.

Additionally, it was shown that the discontinuous method equals an implicit Runge-Kutta scheme given by (10). This means that in each iteration we only need to re-evaluate the function f and do a matrix multiplication. In the continuous Petrov-Galerkin method the function that has to be iterated involves more variables and takes longer to evaluate. Thus, for the schemes given in this article, the discontinuous method actually is more efficient. When the step size decreases and/or the polynomial order increases, this becomes quite noticeable.

3.2 Numerical error analysis

In this section a comparison is made between the errors found in the theory and the errors found in the numerical schemes. To do this we choose functions $f(x, u(x))$ for which an analytic solution to the differential equation is known. The problem we will consider is

$$u'(x) = 2u, \qquad u(0) = 1 \qquad\qquad 0 \leq x \leq 1. \tag{32}$$

For this differential equation we know that the solution is given by $u(x) = \exp(2x)$. Now we apply both the discontinuous Galerkin method (10) and the continuous Petrov-Galerkin method (23). We choose $h = 1/N$ where $N = 2, 4, 8, 16, 32$ and $k = 1, \cdots, 6$. Just like in the error analysis, the maximum norm is used. For the discontinuous method, the $k+1$ point Gauss-Radau weights and abscissae are used. The basis functions don't need to be specified since we implement the method as a Runge-Kutta scheme.

In the continuous method the k point Gauss-Legendre weights and abscissae are used. The basis functions need to be specified on the reference element $[0, 1]$. The first 2 basis functions are the linear functions $\hat{\phi}_1(x) = 1 - x$ and $\hat{\phi}_2(x) = x$. The subsequent basis functions are of increasing order and are required to be 0 at $x = 0$ and $x = 1$. For example, the third basis function is $\hat{\phi}_3(x) = (1 - x)x$. This is done such that the coefficients of the linear basis functions denote the value of the approximation at the left and right boundary. The first 3 basis function are given in figure 2.

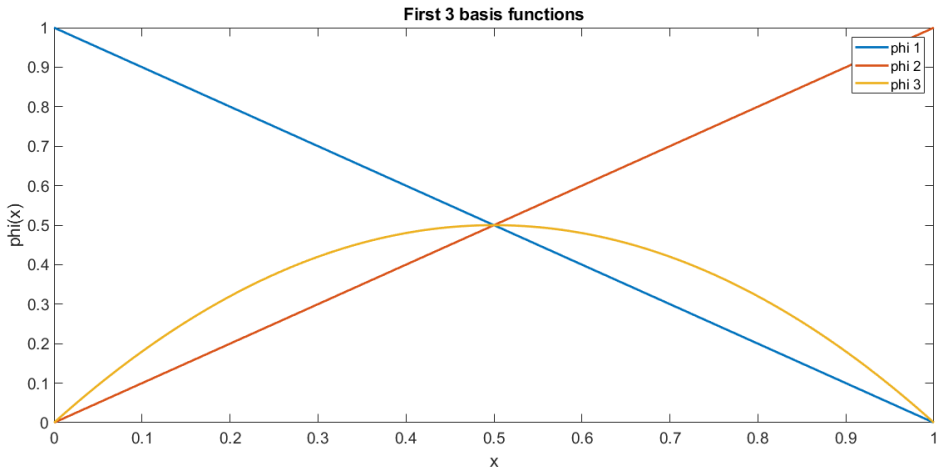

FIGURE 2: First few basis functions $\hat{\phi}_i$

Now for these quadrature rules and basis choices the numerical experiments are done. The numerical errors can be seen in tables 1 and 2.

TABLE 1: Numerical errors for the discontinuous Galerkin method.

h\k	1	2	3	4	5	6
1/2	1.7	2.6e-1	2.2e-2	1.4e-3	7.5e-5	3.4e-6
1/4	6.0e-1	3.9e-2	1.7e-3	5.5e-5	1.5e-6	8.3e-8
1/8	1.7e-1	5.3e-3	1.2e-4	1.9e-6	1.7e-8	4.1e-8
1/16	4.5e-2	7.0e-4	7.6e-6	6.5e-8	5.2e-9	3.0e-9
1/32	1.2e-2	9.0e-5	4.8e-7	8.6e-9	1.3e-8	1.2e-8

TABLE 2: Numerical errors for the continuous Petrov-Galerkin method.

h\k	1	2	3	4	5	6
1/2	1.8	4.9e-2	2.2e-3	1.1e-4	4.7e-6	2.0e-7
1/4	4.3e-1	7.0e-3	1.9e-4	4.5e-6	1.2e-7	7.6e-8
1/8	1.2e-1	9.1e-4	1.1e-5	1.2e-7	4.9e-8	1.7e-7
1/16	3.1e-2	1.1e-4	8.2e-7	1.1e-7	3.8e-7	3.0e-7
1/32	8.1e-3	1.3e-2	2.4e-7	2.9e-7	2.4e-7	3.3e-7

As can be seen in the tables there is convergence in both step size h and polynomial degree k. In general this seems to obey the error bound Ch^{k+1} as found in both methods. However, we see for higher orders of k when the step size decreases that the error actually seems to increase again slightly. This is mostly present in the continuous Petrov-Galerkin method, however it can also be seen a bit in the discontinuous Galerkin method. For this method it however seemed that if h decreases even more that the error decreases again so the problem is minor here.

There are a few possible causes as to why this would happen. First of all the implementation can be a factor. The methods were implemented in Matlab such that we obtain a non-linear system of equations to solve. To solve this system, the built-in function fsolve was used. However, this is only one type of solver and it may be so that other solvers

perform better. In hindsight it would have been better to test with different solvers like for example Newton's method. Also initial conditions could have been considered more since they can influence if the solver reaches a good solution. Since fsolve was used in both methods, this can explain some of the behaviour of both methds.

Secondly, the choice of basis functions can have an influence. Since our discontinuous method was implemented via an implicit Runge-Kutta scheme, this can't really be the problem there. For the continuous method we did choose the basis as described earlier this section. To see how this influencs the solution, we introduce the condition number. For a linear problem $Ax = b$ the condition number is defined as

$$\kappa(A) = \|A\|\|A^{-1}\|. \tag{33}$$

The condition number influences the sensitivity of the solution. In problems with high condition number small errors can greatly be increased. In our case, if we take $f(x, u(x))$ to be linear, then we also get a linear system where the matrix A contains integrals of the polynomial basis functions. Thus for inappropriate choice of basis the condition number can become large and as a result some accuracy is lost. For our choice of basis and $k = 6$, a quick numerical test shows that $\kappa(A) = 4.7 \cdot 10^5$. This is quite large and it can explain the weird behaviour, which especially happens for higher k. This problem can be resolved by choosing a different basis, for example bases of Legendre polynomials for their orthogonality.

4 Conclusion

In this paper both a discontinuous Galerkin method and a continuous Petrov-Galerkin method were analyzed and compared. They are very similar in approximating the solution on each interval by a polynomial of certain degree k. There are a few differences for each method. First of all, quite obviously, the continuous method yield a continuous solution. This can be desirable, since the actual solution u is also continuous given our constraints of continuity on f. In the discontinuous method, we have a modified weak form which allows discontinuities. This also means that there is more freedom in finding the solution and the discontinuities allow us to choose the approximate solution to fit better.

The implementation of both methods is also a bit different. In the discontinuous method, it was shown that the Galerkin method equals a implicit Runge-Kutta scheme. For the continuous method a non-linear system was by more regular Galerkin means. The implicit Runge-Kutta scheme had simpler function evaluations and thus the discontinuous method was more efficient.

For both methods we have shown that they are strongly A-stable, so in that regard there is no difference. However, the discontinuous method had a quadrature of order $p = 2k + 1$, while the continuous method had order $p = 2k$. This is due to the fact that the Petrov-Galerkin method had to satisfy a continuity requirement. This requirement also results in the discontinuous method having a degree of freedom more.

Furthermore, we have shown the error bound $\max_x |u(x) - y(x)| \leq Ch^{\min(p,k+1)}$ for the continuous Petrov-Galerkin method. For the discontinuous method the following error bounds were obtained $\max_x |u(x) - y(x)| \leq Ch^{\min(p,k+1)}$. Due to the order of the methods, the error bound is again of a higher order and thus the discontinuous method is slightly more accurate, however the polynomial approximation dominates the error so the influence

of this is not that high.

To conclude, the continuity constraint ensures that the solution in the continuous method is continuous, but at the cost of an order of accuracy. Also the discontinuous method was more efficient in implementation.

For further research more methods could be analyzed such that a broader comparison can be made. This was the initial idea, but due to time constraints it was not achieved. Furthermore, the methods can be extended to the second dimension and then a comparison can also be made between the one and two dimensions.

References

[1] Owe Axelsson. A class ofa-stable methods. *BIT Numerical Mathematics*, 9(3):185–199, 1969.

[2] Markus Bause, Uwe Köcher, Florin A Radu, and Friedhelm Schieweck. Post-processed galerkin approximation of improved order for wave equations. *arXiv preprint arXiv:1803.03005*, 2018.

[3] Leszek Demkowicz and Jayadeep Gopalakrishnan. A class of discontinuous petrov–galerkin methods. part i: The transport equation. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1558–1572, 2010.

[4] Byron L Ehle. High order a-stable methods for the numerical solution of systems of de's. *BIT Numerical Mathematics*, 8(4):276–278, 1968.

[5] Ernst Hairer, Syvert P Nørsett, and Gerhard Wanner. *Solving ordinary differential equations. 1, Nonstiff problems*. Springer-Vlg, 1991.

[6] Peter Henrici. Discrete variable methods in ordinary differential equations. 1962.

[7] Bernie L Hulme. Discrete galerkin and related one-step methods for ordinary differential equations. *Mathematics of Computation*, 26(120):881–891, 1972.

[8] Pierre Lesaint and Pierre-Arnaud Raviart. On a finite element method for solving the neutron transport equation. *Publications mathématiques et informatique de Rennes*, (S4):1–40, 1974.

A Matlab impementation of the methods

A.1 Matlab code discontinuous Galerkin method

```
%piecewise linear!!
clear all
% subintervals
a = 0; b = 1;
N = 4;
h = (b-a)/N;

% grid points
x = linspace(a,b,N+1);
u = zeros(N+1,1);
% initial condition
u(1) = 1;

k = 6;
A = zeros(k+1,k+1);

% Gauss-Legendre quadrature
syms q
ksi = double(vpasolve(legendreP(k,q) == 0,q));
b = zeros(k,1);
for i=1:k
    b(i) = (2*(1-ksi(i)^2))/((k+1)^2*legendreP(k+1,ksi(i))^2);
end
ksi = 0.5*(ksi +1);
b = 0.5*b;

syms xx

% basis
Mphi = [1-xx,xx,xx.*(1-xx),xx.*(1-xx).*(0.5-xx),xx.*(1-xx)
    .*(0.33-xx).*(0.67-xx),xx.*(1-xx).*(0.25-xx).*(0.5-xx).*(0.75-
    xx),xx.*(1-xx).*(0.2-xx).*(0.4-xx).*(0.6-xx).*(0.8-xx)];
Mphi = Mphi(1:k+1);

% creation matrix A
for i=1:k+1
    for j=1:k+1
        if i == 1
            A(i,j) = subs(Mphi(j),xx,0);
        else
            A(i,j) = h*int(Mphi(i-1)*diff(Mphi(j)),0,1);
        end
    end
end

% per inteval find solution b as with function right hand side
```

```
c = zeros(k+1,1);
coef = zeros(N,k+1);
for n=1:N
    %solve
    xc = x(n) + ksi*h;
    uini = u(n);
    fun = @(coef) A*coef - fc(xc,x(n),x(n+1),uini,k,Mphi,b,coef,h);
    coef(n,:) = fsolve(fun,ones(k+1,1));
    u(n+1) = coef(n,2);
end

% plot solution u as combination of basis functinos on each domain
hold on
title('Numerical solution CPG')
xlabel('x')
ylabel('y(x)')
diff = zeros(N,1);
for n=1:N
    xp = linspace(x(n),x(n+1),100);
    yp = 0;
    for i=1:k+1
        yp = yp+coef(n,i)*subs(Mphi(i),xx,(xp - x(n))/(x(n+1)-x(n)));
    end
    yp2 = exp(2*xp);
    diff(n) = max(abs(yp -yp2));
    plot(xp,yp);
end
hold off
max(diff)

% function for calculating right hand side of system equations
function y2 = fc(xc,x1,x2,uini,k,Mphi,w,coef,h)
syms xx
y2 = zeros(k+1,1);
y2(1) = uini;
for i = 1:k
    sumg = 0;
    for m = 1:k
        xnm = (xc(m) - x1)/(x2 - x1);
        sumg = sumg+w(m)*f(xc(m),double(subs(dot(coef,Mphi),xx,xnm)))*double(subs(Mphi(i),xx,xnm));
    end
    y2(i+1) = h^2*sumg;
end
end

```

```
%function f in u' = f(x,u)
function y = f(x,u)
y = 2*u;
end
```

A.2 Matlab code continuous Petrov-Galerkin method

```
clear all
% subintervals
a = 0; b = 1;
N = 4;
h = (b-a)/N;
xb = linspace(a,b,N+1);
ub = zeros(N+1,1);
%inital condition
ub(1) = 1;

%numerical integration, order polynomial and weights and
    abscissae
k = 1;

% Gauss-Radau weights and abscissae
syms q
ksi = double(vpasolve((legendreP(k,q)+legendreP(k+1,q))/(1+q) ==
    0,q));
b = zeros(k+1,1);
b(1) = 2/(k+1)^2;
for i=2:k+1
    b(i) = (1-ksi(i))/((k+1)^2 * legendreP(k,ksi(i))^2);
end
ksi = 0.5*(ksi +1);
b = 0.5*b;

% Gauss-Legendre
% syms q
% ksi = double(vpasolve(legendreP(k+1,q) == 0,q));
% b = zeros(k+1,1);
% for i=1:k+1
%     b(i) = (2*(1-ksi(i)^2))/((k+2)^2*legendreP(k+2,ksi(i))^2);
% end
% ksi = 0.5*(ksi +1);
% b = 0.5*b;

u = zeros(N,k+1);
x = zeros(N,k+1);

A = zeros(k+1,k+1);
A(:,1) = b(1);

% create matrix A with lagrange polynomials
```

```
for j=2:k+1
    lagr = @(xx) 1;
    for i=2:k+1
        if i ~= j
            g = @(xx) (xx - ksi(i))/(ksi(j)-ksi(i));
            lagr = @(xx) lagr(xx).*g(xx);
        end
    end
    for i = 1:k+1
        if k==1
            A(i,j) = ksi(i) - b(1)*lagr(ksi(1));
        else
            A(i,j) = integral(lagr,0,ksi(i)) - b(1)*lagr(ksi(1));
        end
    end
end

for n=1:N
    x(n,:) = xb(n)+ksi'*h;
    % solve nonlinear equation
    unew = ub(n)*ones(k+1,1);
    fun = @(un) ub(n) + h*A*f(x(n,:).',un) - un;
    unew = fsolve(fun,unew);

    % update boundary points and u vector
    ub(n+1) = ub(n) + h*dot(b,f(x(n,:).',unew));
    u(n,:)= unew;
end

% plot solution
hold on
title('Numerical solution DG')
xlabel('x')
ylabel('y(x)')
diff = zeros(N+1,1);
for i = 1:N
    xx = linspace(xb(i),xb(i+1),100);
    yy = lagrange(xx,x(i,:),u(i,:));
    plot(xx,yy)
    yp2 = exp(2*xx);
    diff(n) = max(abs(yy -yp2));
end
hold off
max(diff)

% function f in u' = f(x,u)
function y = f(x,u)
y = 2*u;
end
```