Universidad de Valladolid

Trabajo Fin de Máster

Máster en Investigación en Matemáticas

Facultad de Ciencias

Curso académico 2016/2017

Método Galerkin Descontínuo para Equação de Lotka-McKendrick

Alumno: Anastácio Pascoal Epandi Canhanga
Tutor: Luis María Abia Llera

Autorización del tutor

Luis M. Abia Llera, Catedrático de Matemática Aplicada de la Universidad de Valladolid,

CERTIFICA: que la presente Memoria *Método Galerkin Descontínuo para Equação de Lotka-McKendrick* ha sido realizada por D. Anastácio Pascoal Epandi Canhanga en la Universidad de Valladolid y constituye la memoria de su trabajo fin de Máster, preceptiva para la finalización de sus estudios del Máster en Investigación en Matemáticas de dicha Universidad.

Valladolid, 19 de julio de 2017

Fdo.:

Luis M. Abia Llera

Conteúdo

Resumo

Neste trabalho se aplica o método Galerkin Descontínuo de elementos finitos ao modelo de Lotka-McKendrick que descreve a dinâmica de uma população estruturada por idade. Ilustramos a convergência de primeira ordem quando se utiliza polinómios lineares a partes.

Introdução

As equações em derivadas parciais aparecem na modelação matemática em diversas áreas da ciência, como a Física, a Química e a Biología passando pelas aplicações da dinâmica de fluídos, electromagnetismo, a engenharia de materiais, a astrofísica, a economía, etc.. No caso geral, estas equações são muito complicadas pelo que encontrar uma solução analítica ou resolve-las utilizando apenas os métodos analíticos (por exemplo, aplicando a transformada de Laplace e Fourier ou em busca de soluções em série de potências) é impossível ou pouco práctico computacionalmente.

Portanto, para resolver equações deste tipo, há uma única maneira: tratar de encontrar a solução numérica que se aproxima à solução analítica desconhecida do problema. O Método de Elementos Finitos é uma classe particular de métodos numéricos que nas últimas décadas tornou-se bastante popular na aproximação da solução de equações em derivadas parciais. Este método foi introduzido nos anos 40 e já se transformou numa classe mais geral e mais potente das técnicas para a solução numérica de equações em derivadas parciais que são normalmente utilizadas na Engenharia e nas Matemáticas.

Recentemente, o método de Galerkin Descontinuo está cada vez mais atractivo no seio da comunidade científica no campo dos métodos numéricos. O Método de Galerkin Descontinuo de Elementos Finitos, foi introduzido no princípio dos anos 70 para cálculo da solução numérica de sistemas hiperbólicos de primeira ordem. Ao mesmo tempo, mas de forma independente deste método foi introduzido como uma técnica não convencional para aproximação de equações elípticas de segunda ordem.

Nas últimas décadas o mesmo método ganhou um novo interesse através das vantagens que tem comparativamente ao tradicional Método de Galerkin de Elementos Finitos. O Método Galerkin Descontinuo é conservativo de elemento por elemento, suporta facilmente a aproximação local de alta ordem, já que a ordem de aproximação pode variar ao longo da malha[2].

Estas e outras diversas propriedades do Método de Galerkin Descontinuo, o fazem num forte candidato para uma ampla série de problemas que aparecem nas aplicações, conforme já dito em [2] .

O obejectivo deste trabalho, é de resolver através de Método Galerkin Descontinuo, a Equação de Lotka-McKendrick. No capítulo 1, faz-se uma revisão à teoria clássica do modelo de Lotka-McKendrick mediante o uso da equação de renovação. O capítulo 2 enumera resultados básicos da teoria sobre espaços de funções utizados, e em particular, os espaços de Sobolev. O terceiro capítulo descreve o método Galerkin Descontínuo para equação de Lotka-McKendrick. E um anexo final contém os programas de Matlab utilizados.

Capítulo 1

Teoria linear básica

A teoria linear que vamos desenvolver a posterior, se pode aplicar ao estudo de uma população isolada, num habitat invariável, em que dois indivíduos se diferenciam somente pela idade. De forma particular, poderemos supor que não há diferença de sexo. Desta feita, poderemos utizar parâmetros intrínsecos que só sejam função da idade e que não dependam, nem do tempo, nem do tamanho da população. As principais características da população que afectam a sua evolução são as taxas de fertilidade e mortalidade.

O nosso primeiro objectivo, será conhecer o sistema que descreve uma população com estas características, conhecido como modelo de **Lotka-McKendrick**. O qual, poderemos analizar através da equação de renovação, para tal, precisamos introduzir uma série de parâmetros.

1.1 Introdução dos parâmetros básicos

Conforme foi dito anteriormente, a evolução da população objecto de estudo se escreve mediante a função

$$u(a,t), \quad a \in [0, a_\dagger], \quad t \geq 0,$$

Sendo $a_\dagger$ a idade máxima que um indivíduo da população pode alcançar. Assim, o número de indivíduos que em um tempo t, têm a idade compreendida no intervalo $[a_1, a_2]$ é dado por

$$\int_{a_1}^{a_2} u(a,t)da$$

e, além disso, mediante o cálculo de

$$P(t) = \int_0^{a_\dagger} u(a,t)da$$

obtemos a população total em um tempo t.

Para descrever a fertilidade da população introduzimos $\beta(a)$, uma função não negativa da idade, que dá conta do número de nascimentos por unidade de tempo que procedem de um número de indivíduos com idade no intevalo $[a, a+da]$. Assim, podemos dizer que

$$\int_{a_1}^{a_2} \beta(a)u(a,t)da$$

proporciona o número de nascimentos por unidade de tempo que procedem de todos indivíduos da população com idades compreendidas entre $[a_1, a_2]$. Como consequência da taxa total de nascimentos por unidade de tempo vem dada por

$$B(t) = \int_0^{a_\dagger} \beta(a)u(a,t)da$$

Da mesma forma podemos introduzir, para descrever a mortalidade, uma função não negativa da idade, $\mu(a)$, que nos dá o número de indivíduos falecidos com idade compreendida entre $[a_1, a_2]$. A taxa total de falecimentos por unidade de tempo vem dada por

$$D(t) = \int_0^{a_\dagger} \mu(a)u(a,t)da$$

A função

$$\Pi(a) = exp(-\int_0^a \mu(\sigma)d\sigma, \quad a \in [0, a_\dagger],$$

representa a probabilidade de sobrevivência. Indica a probabilidade de que um indivíduo, sobreviva à idade a (portanto, $\Pi(a_\dagger) = 0$). Nos modelos de idade máxima $a_\dagger$ finita, a condição $\Pi(a_\dagger) = 0$ requer que a função de mortalidade seja divergente em $[0, a_\dagger)$, o que implica que a função de mortalidade seja não acotada. Alternativamente, por conveniência de modelação, se pode considerar que $a_\dagger = \infty$, em cujo caso, requereremos que a função de densidade $u(a,t)$ se anule no infinito para cada $t \geq 0$.

A expressão $K(a) = \beta(a)\Pi(a)$, com $a \in [0, a_\dagger]$, é a função de maternidade da população e representa o número de descendentes de um indivíduo com idade a. Introduzimos também o número

$$R = \int_0^{a_\dagger} \beta(a)\Pi(a)da$$

ou taxa de reprodução, que dá conta do número de nascimentos que se espera de um indivíduo durante a sua vida reprodutiva. Por último, a esperança de vida, cuja expressão é:

$$L = \int_0^{a_\dagger} \Pi(a)da.$$

L pode entender-se melhor se, notamos que $\mu(a)\Pi(a)da$ é a probabilidade de que um indivíduo sobreviva à idade a e faleça em $[a, a+da]$, pelo que

$$\begin{aligned} L &= \int_0^{a_\dagger} a\mu(a)\Pi(a)da = -\int_0^{a_\dagger} a\frac{d}{da}\Pi(a)da = -a\Pi(a)\mid_0^{a_\dagger} + \int_0^{a_\dagger}\Pi(a)da \\ &= \int_0^{a_\dagger}\Pi(a)da, \end{aligned}$$

já que $\Pi(a_\dagger) = 0$.

1.2 Equação de Lotka-McKendrik

Introduzidas que estão as funções vitais fundamentais de uma população, estabeleceremos a equação que descreve sua evolução através de um equilíbrio do número de indivíduos na população em um dado instante. Partimos da função

$$N(a,t) = \int_0^a u(\sigma,t)d\sigma,$$

que representa o número de indivíduos com menor idade ou igual a a em instante t. Para $h > 0$,

$$N(a+h,t+h) = N(a,t) + \int_t^{t+h} B(s)ds - \int_0^h\int_0^{a+s}\mu(\sigma)u(\sigma,t+s)d\sigma ds \quad (1.1)$$

O segundo termo da direita proporciona o número de nascimentos no intervalo de tempo $[t, t+h]$, por estar incluso no número $N(a+h,t+h)$ (por serem indivíduos de idade menor ou igual a h). Além disso

$$\int_0^{a+s}\mu(a)u(\sigma,t+s)d\sigma$$

é o número de indivíduos que falecem no tempo $t+s$ com idade menor ou igual que $a+s$, o terceiro termo da direita de (1.1) indica o número de indivíduos com idade menor ou igual que a no instante t que falecem no intervalo de tempo $[t, t+h]$.

Se derivarmos com respeito à h e fixamos $h = 0$ teremos:

$$u(a,t) + \int_0^a u_t(\sigma,t)d\sigma = B(t) - \int_0^a \mu(\sigma)u(\sigma,t)d\sigma.$$

Tomando $a = 0$, obtemos $u(0,t) = B(t)$, e derivando

$$u(a,t) + \int_0^a u_t(\sigma,t)d\sigma = B(t) - \int_0^a \mu(\sigma)u(\sigma,t)d\sigma.$$

Com respeito à a teremos:

$$u_t(a,t) + u_a(a,t) + \mu(a)u(a,t) = 0.$$

Desta forma, chegamos ao seguinte sistema:

$$\begin{cases} \text{(i)} & u_t(a,t) + u_a(a,t) + \mu(a)u(a,t) = 0 \\ \text{(ii)} & u(0,t) = \int_0^{a_\dagger} \beta(\sigma)u(\sigma,t)d\sigma \\ \text{(iii)} & u(a,0) = u_0(a) \end{cases}$$

Para que nosso estudo seja coerente biológicamente e para desenvolver matemáticamente, precisamos impor algumas condições sobre as funções $\beta(.)$ e $\mu(.)$:

$$\begin{cases} \beta(.) \text{ não negativa}, & \beta(.) \in L^\infty(0, a_\dagger), \\ \mu(.) \text{ não negativa}, & \mu(.) \in L^1_{loc}([0, a_\dagger)), \\ \int_0^{a_\dagger} \mu(\sigma)d\sigma = +\infty, \\ u_0 \in L^1(0, a_\dagger), u_0(a) \geq 0 \text{ em quase todos pontos de } [0, a_\dagger] \end{cases}$$

1.3 Equação de Renovação

Esta secção é dedicada a busca de uma nova formulação do sistema. Para o qual, introduzimos uma nova variaável

$$q(a,t) = exp(\int_0^a \mu(\sigma)d\sigma)u(a,t),$$

que verifica:

$$\begin{cases} q_t(a,t) + q_a(a,t) = 0 & 0 < a \leq a_\dagger, t > 0, \\ q(0,t) = B(t) & t \geq 0, \\ q(a,0) = exp(\int_0^a \mu(\sigma)d\sigma)u_0(a) = q_0(a) & 0 \leq a \leq a_\dagger \end{cases}$$

$q(t)$ é portanto, a solução em $[0, a_\dagger) \times [0, \infty)$ de um problema de valores iniciais e fronteira para a lei de conservacão

$$q_t(a,t) + q_a(a,t) = 0$$

com dados

$$\begin{cases} q(0,t) = B(t), \text{ em } \{a = 0, t > 0\} \\ q(a,0) = q_0(a), \text{ em } \{a \in [0, a_\dagger], t = 0\} \end{cases}$$

Se supormos que $B(t)$ conhecido, q fica completamente determinada na banda $[0, a_\dagger) \times [0, \infty)$ e vem dada por

$$g(a,t) = \begin{cases} clcrq_0(a-t) \text{ se } a \geq t \\ B(t-a) \text{ se } a < t. \end{cases} .$$

Utilizando

$$q(a,t) = exp(\int_0^a \mu(\sigma)d\sigma)u(a,t),$$

chegamos a seguinte expressão

$$u(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} sea \geq t \\ B(t-a)\Pi(a) \text{ se } a < t. \end{cases}$$

Esta fórmula nos permite obter uma equação integral para a taxa de nascimentos $B(t)$. Como $u(0,t) = B(t)$, se substituirmos

$$u(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} \text{ se } a \geq t \\ B(t-a)\Pi(a) \text{ se } a < t. \end{cases}$$

em

$$u(0,t) = \int_0^{a_\dagger} \beta(\sigma)u(\sigma,t)d\sigma,$$

teremos, para $t \leq a_\dagger$:

$$\begin{aligned} B(t) &= \int_0^{a_\dagger} \beta(a)u(a,t)da \\ &= \int_0^t \beta(a)\Pi(a)B(t-a)da + \int_0^{a_\dagger} \beta(a)\frac{\Pi(a)}{\Pi(a-t}u_0(a-t)da, \end{aligned}$$

e para $t > a_\dagger$:

$$B(t) = \int_0^{a_\dagger} \beta(a)\Pi(a)B(t-a)da.$$

Portanto, $B(t)$ é a solução da equação integral de Volterra de segunda espécie

$$B(t) = F(t) + \int_0^t K(t-s)B(s)ds, \tag{1.2}$$

donde

$$F(t) = \int_t^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t)}u_0(a-t)da = \int_0^\infty \beta(a+t)\frac{\Pi(a+t)}{\Pi(a)}u_0(a)da \tag{1.3}$$

e

$$K(t) = \beta(t)\Pi(t), t \geq 0. \tag{1.4}$$

$$\beta(a) \equiv \Pi(a) \equiv u_0(a) \equiv 0, a \notin [0, a_\dagger].$$

A equação (1.2) é conhecida indistintamente como equação de renovação e como equação de Lotka-McKendrik. Equivale ao sistema obtido na secção anterior e é a ferramenta principal para a análise de existência e unicidade do problema que estamos a considerar. Observemos que $K(t)$ em (1.4) é a função de maternidade, já definida anteriormente.

Teorema 1.3.1 *Se se satisfazem as condições*

(i) $\beta(.)$ *não negativa,* $\beta(.) \in L^\infty(0, a_\dagger)$,

(ii) $\mu(.)$ *não negativa,* $\mu(.) \in L^1_{loc}([0, a_\dagger))$,

(iii) $\int_0^{a_\dagger} \mu(\sigma)d\sigma = +\infty$,

(iv) $u_0 \in L^1(0, a_\dagger), u_0(a) \geq 0$ *em quase todos pontos de* $[0, a_\dagger]$

Então

(i) $K(t) \geq 0$ *em quase todo ponto,* $K(t) = 0$ *se* $t > a_\dagger$, $K \in L^1(\mathcal{R}_+) \cap L^\infty(\mathcal{R}_+)$

(ii) $F(t) \geq 0, F(t) = 0$ *para* $t > a_\dagger, F \in C(\mathcal{R}_+)$

Se, além disso,

$$u_0 \in W^{1,1}(0, a_\dagger) e \mu(.)u_0(.) \in L^1(0, a_\dagger),$$

então $F \in W^{1,\infty}(\mathcal{R}_+)$

Prova:
(i) e a primeira parte de (ii) se deduzem facilmente.
Para comprovar que $F \in C(\mathcal{R}_+)$, tomamos $t_0 \geq 0$ e obtemos:

$$\begin{aligned} F(t) &= \int_t^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t}(u_0(a-t) - u_0(a-t_0))da \\ &\quad + \int_t^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t}u_0(a-t_0)da. \end{aligned}$$

Ao ser,

$$\begin{aligned} &|\int_t^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t}(u_0(a-t) - u_0(a-t_0))da| \leq \\ &|\beta|_{L^\infty} \int_0^\infty |u_0(a-t) - u_0(a-t_0)|da, \end{aligned}$$

Logo

$$|\int_t^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t}(u_0(a-t) - u_0(a-t_0))da| \longrightarrow 0 \quad se \quad t \longrightarrow 0,$$

e portanto, tomando o limite quando t tende a t_0 temos:

$$\lim_{t\to t_0} F(t) = \int_{t_0}^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t_0}u_0(a-t_0)da = F(t_0).$$

Da mesma forma se prova a última afirmação.

1.4 Existência e unicidade de soluções

Teorema 1.4.1 *Se se satisfazem as seguintes condições:*

(i) $\beta(.)$ *não negativa,* $\beta(.) \in L^{\infty}(0, a_{\dagger})$,

(ii) $\mu(.)$ *não negativa,* $\mu(.) \in L^{1}_{loc}([0, a_{\dagger}))$,

(iii) $\int_0^{a_{\dagger}} \mu(\sigma)d\sigma = +\infty$,

(iv) $u_0 \in L^1(0, a_{\dagger}), u_0(a) \geq 0$ *em quase todos pontos de* $[0, a_{\dagger}]$

e além disso

$$u_0 \in W^{1,1}(0, a_{\dagger}) e \mu(.)u_0(.) \in L^1(0, a_{\dagger}),$$

então, a equação de renovação

$$B(t) = F(t) + \int_0^t K(t-s)B(s)ds$$

tem uma única solução $B \in C(\mathcal{R}_+)$ *tal que* $B(t) \geq 0$ *para todo* t, $B \in W^{1,\infty}_{loc}(\mathcal{R}_+)$ *e:*

$$B'(t) = F'(t) + K(t)B(0) + \int_0^t K(t-s)B'(s)ds$$

Prova: Para demonstrar este resultado suponhamos

$$\mid K \mid_{L^1(\mathcal{R}_+)} = \int_0^{\infty} K(s)ds < 1 \tag{1.5}$$

e obtemos a solução da equação pelo procedimento iterativo standar:

$$\begin{cases} B^0(t) = F(t) \\ B^{k+1}(t) = F(t) + \int_0^t K(t-s)B^k(s)ds. \end{cases}$$

Pelo resultado anterior, $B^k \in C([0,T])$ para $T > 0$ e $B^k(t) \geq 0$. Além disso,

$$\mid B^{k+1}(t) - B^k(t) \mid \leq \int_0^t K(t-s) \mid B^k(s) - B^{k-1}(s) \mid ds$$

e

$$\mid B^{k+1} - B^k \mid_{C([0,T])} \leq \mid K \mid_{L^1(\mathcal{R}_+)} \mid B^k - B^{k-1} \mid_{C([0,T])} .$$

Pela equação (1.5), a sucessão $B^k(t)$ converge uniformemente em $[0,T]$ a uma solução $B(t)$ da equação de renovação tal que $B \in C([0,T])$ e $B(t) \geq 0$. Para provar a unicidade de solução, sejam $B(t)$ e $\hat{B}(t)$ duas soluções. Então:

$$\mid B - \hat{B} \mid_{C([0,T])} \leq \mid K \mid_{L^1(\mathcal{R}_+)} \mid B - \hat{B} \mid_{C([0,T])},$$

logo, de novo pela (1.5) $B(t) = \hat{B}(t)$

Se, além disso,

$$u_0 \in W^{1,1}(0, a_\dagger) e \mu(.)u_0(.) \in L^1(0, a_\dagger),$$

então, pelo teorema anterior e pelo procedimento iterativo estandar

$$\begin{cases} B^0(t) = F(t) \\ B^{k+1}(t) = F(t) + \int_0^t K(t-s)B^k(s)ds. \end{cases}$$

temos que $B^k \in W^{1,\infty}(\mathcal{R}_+)$, e fixando $V^k(t) = \frac{d}{dt}B^k(t)$ em quase todo ponto, temos que $V^k \in L^\infty(\mathcal{R}_+)$ e $V^{k+1}(t) = F'(t) + K(t)F(0) + \int_0^t K(t-s)V^k(s)$, portanto,

$$| V^{k+1} - V^k |_{L^\infty(\mathcal{R}_+)} \leq | K |_{L^1(\mathcal{R}_+)} | V^k - V^{k-1} |_{L^\infty(\mathcal{R}_+)} .$$

Utilizando uma vez mais (1.5), a sucessão V^k converge em $_{L^\infty(\mathcal{R}_+)}$ a $V(t) = \frac{d}{dt}B(t)$ em quase todo ponto e

$$B'(t) = F'(t) + K(t)B(0) + \int_0^t K(t-s)B'(s)ds$$

em seguida

$$V^{k+1}(t) = F'(t) + K(t)F(0) + \int_0^t K(t-s)V^k(s)ds$$

Se (1.5) não se cumpre, tomemos α tal que

$$\int_0^\infty exp(-\alpha t)K(t)dt < 1$$

e consideremos

$$\bar{B}(t) = exp(-\alpha t)B(t), \bar{F}(t) = exp(-\alpha t)F(t) e \bar{K}(t) = exp(-\alpha t)K(t).$$

A equaç— ao de renovação se transforma em

$$\bar{B}(t) = \bar{F}(t) + \int_0^t \bar{K}(t-s)\bar{B}(s)ds.$$

Como $\bar{K}(t)$ satisfaz (1.5), esta equação pode resolver-se seguindo o argumento anterior.

Teorema 1.4.2 *Sob as hipóteses dos resultados anteriores e supondo, além disso que:*

$$u_0(0) = \int_0^\infty \beta(a)u_0(a)da,$$

com $u(a,t)$ definida por

$$u(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} sea \geq t \\ B(t-a)\Pi(a) sea < t. \end{cases}$$

Sendo $B(t)$ a solução de (1.2)-(1.4), se tem:

$$u \in C([0,a_\dagger] \times \mathcal{R}_+), u(a,t) \geq 0, \mu(.)u(.,t) \in L^1(0,a_\dagger) \forall t > 0. \tag{1.6}$$

Existem

$$\frac{\partial u}{\partial t}(a,t), \frac{\partial u}{\partial a}(a,t) em\ quase\ todo\ ponto\ de\ [0,a_\dagger] \times [0,\infty] \tag{1.7}$$

e o nosso problema está resolvido. Além disso, $u(a,t)$ é a única solução no sentido de (1.6),(2.7).

Prova:

$$\begin{aligned} \int_0^{a_\dagger} \mu(a)u(a,t)da &= \int_0^{t\wedge a_\dagger} \mu(a)B(t-a)\Pi(a)da \\ &\quad + \int_{t\wedge a_\dagger}^{a_\dagger} \mu(a)u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)}da \\ &\leq \max_{s\in[0,t]} |B(s)| \int_0^{t\wedge a_\dagger} \mu(a)\Pi(a)da \\ &\quad + \exp\left(\int_0^{(t\vee a_\dagger)-t} \mu(\sigma)d\sigma\right) |u_0|_{C([0,a_\dagger])} \int_{t\wedge a_\dagger}^{a_\dagger} \mu(a)\Pi(a)da \\ &\leq \max_{s\in[0,t]} |B(s)| + \exp\left(\int_0^{(t\vee a_\dagger)-t} \mu(\sigma)d\sigma\right) |u_0|_{C([0,a_\dagger])}. \end{aligned}$$

Notemos que

$$u_0(0) = \int_0^\infty \beta(a)u_0(a)da$$

garante a continuidade de $u(a,t)$ na linha $a = t$. O qual;

$$B(0) = \int_0^\infty \beta(a)u_0(a)da = u_0(0) \tag{1.8}$$

Comprovamos já que uma solução de

$$\begin{cases} \text{(i)} & u_t(a,t) + u_a(a,t) + \mu(a)u(a,t) = 0 \\ \text{(ii)} & u(0,t) = \int_0^{a_\dagger} \beta(\sigma)u(\sigma,t)d\sigma \\ \text{(iii)} & u(a,0) = u_0(a) \end{cases}$$

deve ser da forma

$$u_0(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} sea \geq t \\ B(t-a)\Pi(a) sea < t. \end{cases}$$

com $\beta(t)$ satisfazendo (1.2)-(1.4). A unicidade de solução deste último problema nos leva à de

$$\begin{cases} \text{(i)} & u_t(a,t) + u_a(a,t) + \mu(a)u(a,t) = 0 \\ \text{(ii)} & u(0,t) = \int_0^{a_\dagger} \beta(\sigma)u(\sigma,t)d\sigma \\ \text{(iii)} & u(a,0) = u_0(a) \end{cases}$$

Vimos que, sob as hipóteses

$$u_0 \in W^{1,1}(0,a_\dagger) e \mu(.)u_0(.) \in L^1(0,a_\dagger)$$

e

$$u_0(0) = \int_0^\infty \beta(a)u_0(a)da,$$

a fórmula nos proporciona uma solução que podemos chamar clássica. Na realidade, esta fórmula é significativa inclusivel se essas condições não se satisfazem;o qual é suficiente para obter uma solução no sentido estabelecido no seguinte teorema.

Teorema 1.4.3 *Se se satisfazem as seguintes condições:*

(i) $\beta(.)$ *não negativa,* $\beta(.) \in L^\infty(0,a_\dagger)$,

(ii) $\mu(.)$ *não negativa,* $\mu(.) \in L^1_{loc}([0,a_\dagger))$,

(iii) $\int_0^{a_\dagger} \mu(\sigma)d\sigma = +\infty$,

(iv) $u_0 \in L^1(0,a_\dagger), u_0(a) \geq 0$ *em quase todo ponto de* $[0,a_\dagger]$

então, $u(a,t)$, *definida por*

$$u(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} sea \geq t \\ B(t-a)\Pi(a) sea < t. \end{cases}$$

verifica as seguintes propriedades:

(i) $u(.,t) \in C([0,a_\dagger]; L^1(0,a_\dagger)), u(a,t) \geq 0$ *em quase todo ponto de* $[0,a_\dagger] \times \mathcal{R}_+$,

(ii) $\mid u(.,t) \mid_{L^1} \leq exp(t \mid \beta \mid_{L^\infty}) \mid u_0 \mid_{L^1}$,

(iii) $u(a,t)$ *é contínua se* $a < t$ *e satisfaz* $\mid u(a,t) = \int_0^{a_\dagger} \beta(\sigma) \mid u(\sigma,t)d\sigma$ *para* $t > 0$,

(iv) $\lim_{h\to 0} \frac{1}{h}[u(a+h,t+h) - u(a,t)] = -\mu(a,t)u(a,t)$ *em quase todo ponto* $[0,a_\dagger] \times \mathcal{R}_+$.

Prova:
Vejamos que se cumpre (ii). Como

$$F(t) = \int_0^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t)}u_0(a-t)da = \int_0^\infty \beta(a+t)\frac{\Pi(a+t)}{\Pi(a)}u_0(a)da,$$

temos $F(t) \leq \mid \beta \mid_{L^\infty} \mid u_0 \mid_{L^1}, K(t) \leq \mid \beta \mid_{L^\infty}$ e como,

$$B(t) = F(t) + \int_0^t K(t-s)B(s)ds,$$

Então,

$$B(t) \leq \mid \beta \mid_{L^\infty} \mid u_0 \mid_{L^1} + \mid \beta \mid_{L^\infty} \int_0^t B(s)ds.$$

Assim, pela desigualdade de Gronwall,

$$B(t) \leq \mid \beta \mid_{L^\infty} exp(t \mid \beta \mid_{L^\infty}) \mid u_0 \mid_{L^1} .$$

Com esta estimação,

$$u(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} sea \geq t \\ B(t-a)\Pi(a) sea < t. \end{cases}$$

nos conduz a

$$\mid u(.,t) \mid_{L^1} = \int_0^t B(t-a)\Pi(a)da + \int_0^\infty \frac{\Pi(a+t)}{\Pi(a)}u_0(a)da \leq$$

$$\leq (\mid \beta \mid_{L^\infty} \int_0^t exp((t-a) \mid \beta \mid_{L^\infty})da + 1) \mid u_0 \mid_{L^1} = exp(t \mid \beta \mid_{L^\infty}) \mid u_0 \mid_{L^1},$$

e já temos a desigualdade desejada. A partir de (ii), obtemos (i) facilmente: com efeito, dado $u_0 \in L^1(0, a_\dagger)$, seja u_0^n uma sucessão tal que u_0^n verifica

$$\begin{cases} u_0^n \in W^{1,1}(0, a_\dagger) \\ \mu(.)u_0^n(.) \in L^1(0, a_\dagger) \\ u_0^n(0) = \int_0^\infty \beta(a)u_0^n(a)da \\ \lim\limits_{n\to\infty} \mid u_0^n - u_0 \mid_{L^1} = 0 \end{cases}$$

E seja u^n a solução de

$$\begin{cases} u_t(a,t) + u_a(a,t) + \mu(a)u(a,t) = 0 \\ u(0,t) = \int_0^{a_\dagger} \beta(\sigma)u(\sigma,t)d\sigma \\ u_0^n(0) = \int_0^\infty \beta(a)u_0^n(a)da \\ u(a,0) = u_0(a) \end{cases}$$

Correspondente a u_0^n. Então, $u^n \in C([0,T]; L^1(0, a_\dagger)$ e por (ii) e a linearidade, temos:

$$\mid u^n(.,t) - u(.,t) \mid_{L^1} \leq exp(t \mid \beta \mid_{L^\infty}) \mid u_0^n - u_0 \mid_{L^1},$$

de forma que u é o limite da sucessão u^n no espaço $C([0,T]; L^1(0, a_\dagger)$, isto é, $u(.,t) \in C([0,T]; L^1(0, a_\dagger), u(a,t) \geq 0$ em quase todo ponto de $[0, a_\dagger] \times \mathcal{R}_+$, como queríamos demonstrar. (iii) e (iv) se seguem facilmente.

1.5 Regularidade de soluções

Pode obter-se maior regularidade da solução $u(a,t)$ se introduzimos hipóteses adicionais. Para algum inteiro $k \geq 0$, vamos supor

- (H1) A função não negativa u_0 pertence a $C^k([0,a_\dagger))$ e tem suporte compacto em $[0,a_\dagger)$;
- (H2) $\beta \in C^{k+1}([0,a_\dagger)), 0 \leq \beta(a) \leq \bar{\beta}$ para alguma constante positiva $\bar{\beta}$ e $\beta(.)$ tem suporte compacto em $[0,a_\dagger)$;
- (H3)$\mu \in C^{k+1}([0,a_\dagger))$.

No âmbito dessas hipóteses, é sabido que o problema

$$u_t + u_a + \mu(a)u = 0, 0 < a < a_\dagger, 0 < t < T, \tag{1.9}$$

$$u(0,t) = \int_0^{a_\dagger} \beta(a)u(a,t)da, 0 < t \leq T, \tag{1.10}$$

$$u(a,0) = u_0(a), 0 \leq a \leq a_\dagger, \tag{1.11}$$

tem uma única solução não negativa, que é global em tempo.

Quando a mortalidade, μ, é não acotada em $a_\dagger$, devemos introduzir hipóteses adicionais sobre a forma da singularidade da referida função, provar a regularidade das soluções do problema. Vamos, portanto, supor

- (H4)Para valores de a próximos a $a_\dagger$, μ toma a forma $\mu(a) = \frac{c}{a_\dagger - a}$, para algum $c \geq k+1$ o $\mu(a) = \frac{c}{(a_\dagger - a)^\alpha}$, para algum $\alpha > 1, c > 0$.
- (H5)Existem constantes positivas $a_* < a_\dagger$ e $\bar{\mu}$ tais que $0 \leq \mu(a) \leq \bar{\mu}$ para $a \in [0,a_*]$ e μ é monótona crescente em $[a_*, a_\dagger]$, sendo

$$\mu(a_*) = \sup_{a \in [0,a_*]} \mu(a).$$

Teorema 1.5.1 *Suponhamos que se verificam as hipóteses (H1)-(H5). Então, para cada $t \geq 0, u(a,t)$ tende a zero quando a tende a $a_\dagger$, e $\frac{\partial^l u}{\partial a^l}$ é limitada para $0 \leq l \leq k+1$. Além disso, para cada $t > 0$, $\frac{\partial^l u}{\partial a^l}$ e $\frac{\partial^l \mu}{\partial a^l} u$, com $0 \leq l \leq k+1$, tendem a zero quando a tende a $a_\dagger$. Se $\mu(a) = \frac{c}{a_\dagger - a}$ proximo de $a_\dagger$, então a última afirmação se verifica só quando $c > k+1$.*

Prova:
Da representação da solução já conhecida

$$u(a,t) = \begin{cases} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} & \text{se } a \geq t \\ B(t-a)\Pi(a) & \text{se } a < t. \end{cases} . \tag{1.12}$$

Com

$$\Pi(a) = exp(-\int_0^a \mu(\sigma)d\sigma)$$

e $B(t)$ a solução da equação de Volterra:

$$B(t) = F(t) + \int_0^t K(t-s)B(s)ds,$$

Com $F(t)$ como em (1.4), temos: $u(a,t) = \Pi(a)H(a,t)$ donde

$$H(a,t) = \begin{cases} u_0(a-t)exp(\int_0^{a-t}\mu(\sigma)d\sigma) \text{ se } a \geq t \\ B(t-a), \text{se } a < t. \end{cases}$$

Se $t = 0$, então $u(a,t) = u_0(a) \longrightarrow 0$ quando $a \longrightarrow a_\dagger$. Para $t > 0$, um cálculo directo prova que $\mu^l(a)\Pi(a)$ e $\frac{d^l\mu}{da^l}\Pi(a), 0 \leq l \leq k+1$, tendem a zero quando $a \longrightarrow a_\dagger$. Por outro lado, para cada $t > 0$,

$$\frac{d^l u}{da^l}(a,t) = \sum_{\nu-0}^{\nu=l} \left(\begin{array}{c} l \\ \nu \end{array} \right) \frac{d^\nu}{da^\nu}[exp(-\int_0^a \mu(\sigma)d\sigma)]\frac{d^{l-\nu}}{da^{l-\nu}}H(a,t), 0 \leq l \leq k+1.$$

Em cada somando, o termo

$$\frac{d^\nu}{da^\nu}[exp(-\int_0^a \mu(\sigma)d\sigma)], 0 \leq \nu \leq k+1$$

contém um factor $\Pi(a)$ que multiplica a uma função de μ e suas derivadas. Como $\mu(a)$, suas potências e suas derivadas sucessivas divergem em $a = a_\dagger$ e o factor $\Pi(a)$ decresce a zero exponencialmente, se conclui que $\frac{\partial^l u}{\partial a^l}, 0 \leq l \leq k+1$, tende a zero quando a tende a $a_\dagger$.

Nota-se que, para $t > 0$ fixo, quando $a_\dagger > t$, a função $H(a,t)$ é derivável com respeito a a tantas vezes como o é $u_0(a)$ e a função $\mu(a)$, e quando a tende a $a_\dagger$ as derivadas sucessivas de $H(.,t)$ tendem à um valor finito. Quando $a_\dagger < t$, e para $0 \leq l \leq k+1$,

$$\frac{d^l u}{da^l}(a,t) = \sum_{\nu-0}^{\nu=l} \left(\begin{array}{c} l \\ \nu \end{array} \right) \frac{d^\nu}{da^\nu}[exp(-\int_0^a \mu(\sigma)d\sigma)]\frac{d^{l-\nu}}{da^{l-\nu}}B(t-a),$$

claro que $B(t)$ é derivável $k+1$ vezes. Mas para cada $T > 0, B(t) \in C([0,T])$ é solução da equação integral de Volterra de segunda espécie

$$B(t) = F(t) + \int_0^t K(t-s)B(s)ds,$$

onde

$$K(t) = \beta(t)\Pi(t),$$

$$F(t) = \int_0^\infty \beta(a)\frac{\Pi(a)}{\Pi(a-t)}u_0(a-t)da.$$

Se $B(t)$ é contínua, $K(t)$ é contínua e $F(t)$ é derivável, então o teorema fundamental do cálculo integral permite obter a derivabilidade de $B(t)$ e além disso, escrever

$$B'(t) = F'(t) + K(t)B(0) + \int_0^t K(t-s)B'(s)ds.$$

De novo, se $B(t)$ é derivável com continuidade, $K(t)$ é derivável e $F(t)$ é derivável duas vezes, $B''(t)$ existe para todo $t > 0$ e além disso

$$B''(t) = F''(t) + K'(t)B(0) + K(t)B'(0) + \int_0^t K(t-s)B''(s)ds.$$

Reiterando este argumento, podemos obter a existência de derivada contínuas até a ordem $k+1$ de $B(t)$, claro que $K(t)$ é k vezes derivável com continuidade e $F(t)$ é $k+1$ vezes derivável com continuidade. A regularidade de $K(t) = \beta(t)\Pi(t)$ é imediata das hipóteses de regularidade de $\beta(t)$ e $\mu(t)$.

Para a reguridade de $F(t)$, observemos que para $t \geq a_\dagger, F(t) \equiv 0$ e que para $t < a_\dagger$, a função

$$F(t) = \int_0^{a_\dagger} exp(-\int_{a-t}^a \mu(\sigma)d\sigma)u_0(a-t)da$$

é derivável.

Para acabar com a prova observemos que se $\mu(a) = \frac{c}{a_\dagger - a}$ nas proximidades de $a_\dagger$, então em tal região de $a_\dagger$ se tem

$$\mu^{k+1}(a)\Pi(a) = \frac{c^{k+1}}{(a_\dagger - a)^{k+1}}\Pi(a)$$

$$= \frac{c^{k+1}}{(a_\dagger - a)^{k+1}}exp(-\int_0^a \frac{c}{a_\dagger - \sigma}d\sigma)$$

$$= \frac{c^{k+1}}{(a_\dagger - a)^{k+1}} \frac{(a_\dagger - a)^c}{{a_\dagger}^c}.$$

Se tem então que se $c > k+1$ então $\mu^{k+1}(a)\Pi(a) \longrightarrow 0$ quando $a \longrightarrow a_\dagger$, mas se $c = k+1$, a função $\mu^{k+1}(a)\Pi(a)$ é limitada em $[0, a_\dagger)$, mas não tende a zero em $a_\dagger$.

O resultado de regularidade da solução $u(a,t)$ que nos interessa se deriva da proposição anterior.

Teorema 1.5.2 *Suponhamos as hipóteses (H1)-(H5). Então a solução $u(a,t)$ de (1.9)-(1.11) pertence a $C^{k+1}([0,a_\dagger) \times [0,T] \setminus \{(a,t) \mid a = t\})$ para algum $T > 0$. Se além disso as condições de compatibilidade (1.13) e (1.14) dadas a continuação se verificam, então a solução $u(a,t)$ de (1.9)-(1.11), pertence a $C^1([0,a_\dagger) \times [0,T])$:*

$$u_0(a) = \int_0^{a_\dagger} \beta(a) u_0(a) da, \tag{1.13}$$

e

$$u_0'(0) = -[\mu(0) + \beta(0)]u_0(0) - \int_0^{a_\dagger} (\beta'(a) - \beta(a)\mu(a)) u_0(a) da. \tag{1.14}$$

Prova:

A proposição anterior nos dá já a regularidade C^{k+1} nos pontos de $([0,a_\dagger) \times [0,T] \setminus \{(a,t) \mid a = t\})$. Para a continuidade nos pontos $a = t$ é necessário que

$$\lim_{(a,t)\to(x,x),a\geq t} u(a,t) = \lim_{(a,t)\to(x,x),a<t} u(a,t),$$

para todo $x \in [0,a_\dagger)$. A representação de $u(a,t)$ na forma (1.12) e a continuidade de $u_0(a)$, e $B(t) = u_0(0,t)$, nos permite calcular respectivamente:

$$\lim_{(a,t)\to(x,x),a\geq t} u(a,t) = \lim_{(a,t)\to(x,x)} u_0(a-t)\frac{\Pi(a)}{\Pi(a-t)} = u_0(0)\Pi(x), \tag{1.15}$$

e

$$\lim_{(a,t)\to(x,x),a<t} u(a,t) = \lim_{(a,t)\to(x,x)} B(t-a)\Pi(a) = B(0)\Pi(x). \tag{1.16}$$

A primeira condição de compatibilidade se obtém de igualar estes dois limites dados que

$$B(0) = \int_0^{a_\dagger} \beta(a) u_0(a) da,$$

Para avaliar as primeiras derivadas de $u(a,t)$ nos pontos $(a,t) = (x,x)$ e sua continuidade em tal ponto, temos

$$\frac{\partial u}{\partial t}(x,x) = \lim_{h\to 0, h>0} \frac{u(x,x+h)-u(x,x)}{h} = \Pi(x) \lim_{h\to 0} \frac{B(h)-B(0)}{h} = \Pi(x)B'(0)$$

Por outro lado

$$\lim_{h\to 0, h<0} \frac{u(x,x+h-u(x,x)}{h} = \lim_{h\to 0} \frac{1}{h}(u_0(-h)\frac{\Pi(x)}{\Pi(-h)} - u_0(0)\Pi(x))$$

$$= u_0(0)\Pi(x) \lim_{h\to 0} \frac{\Pi(h)-1}{h} = -u_0(0)\Pi(x)\mu(0).$$

Para a derivada parcial em relação a a temos, nos pontos (x,x),

$$\begin{aligned} \frac{\partial u}{\partial a}(x,x) &= \lim_{h\to 0, h>0} \frac{u(x+h,x)-u(x,x)}{h} \\ &= \lim_{h\to 0} \frac{u_0(h)\Pi(x+h) - u_0(0)\Pi(x)\Pi(h)}{h\Pi(h)} \\ &= \Pi(x)(u_0'(0) - u_0(0)\mu(x) + u_0(0)\mu(0)). \end{aligned}$$

e

$$\begin{aligned} \lim_{h\to 0, h<00} \frac{u(x+h,x)-u(x,x)}{h} &= \lim_{h\to 0, h<0} \frac{B(-h)\Pi(x+h) - B(0)\Pi(x)}{h} \\ &= \Pi(x)(B'(0) - B(0)\mu(x)). \end{aligned}$$

Note-se que, derivando na equação para $B(t)$, dada por

$$B(t) = F(t) + \int_0^t K(t-s)B(s)ds,$$

Com

$$F(t) = \int_0^{a_\dagger} \beta(a)\frac{\Pi(a)}{\Pi(a-t)}u_0(a-t)da,$$

e pondo $t=0$ resulta em

$$B'(0) = \int_0^{a_\dagger} \beta(a)[-u_0'(a) - \mu(a)u_0(a)da.$$

A continuidade nos pontos da forma (x,x) de $\frac{\partial u}{\partial t}$ e de $\frac{\partial u}{\partial a}$, dadas por

$$\frac{\partial u}{\partial t} = \begin{cases} -u_0'(a-t)\frac{\Pi(a)}{\Pi(a-t)} + u_0(a-t)\Pi(a)\frac{\Pi'(a-t)}{\Pi(a-t)^2}, \text{se } a>t, \\ B'(t-a)\Pi(a), \text{ se } a<t. \end{cases}.$$

e

$$\frac{\partial u}{\partial a} = \begin{cases} u_0'(a-t)\frac{\Pi(a)}{\Pi(a-t)} + u_0(a-t)\frac{\Pi'(a)\Pi(a-t)-\Pi(a)\Pi'(a-t)}{\Pi(a-t)^2}, \text{ se } a>t, \\ -B'(t-a)\Pi(a) + B(t-a)\Pi'(a), \text{ se } a<t. \end{cases}.$$

o que implica de forma directa a condição de compatibilidade (1.14).

Capítulo 2

Generalidades: Espaços de funções

2.1 Espaços de funções contínuas

Por uma questão de simplificação da notação, vamos introduzir o conceito de multi-índice.

Seja N o conjunto dos números inteiros positivos. Chamamos de multi-índice à núpla $\alpha = (\alpha_1, ..., \alpha_n) \in N^n$. Também definimos o inteiro positivo $|\alpha| := \alpha_1 + ... + \alpha_n$ como sendo o tamanho do multi-índice $\alpha = (\alpha_1, ..., \alpha_n) \in N^n$. Assim, usando o multi-índice, podemos simplificar a notação do operador de derivada de alta ordem assim:

$$\frac{\partial^{\alpha_1+...+\alpha_n}}{\partial x_1^{\alpha_1}...\partial x_n^{\alpha_n}} = \frac{\partial^{|\alpha|}}{x_1^{\alpha_1}...\partial x_n^{\alpha_n}} = D^\alpha$$

Seja $\Omega \subset R^n$ um conjunto aberto e limitado e seja $k \in N$. Seja também $\overline{\Omega}$ o fecho de Ω. Denotamos por $\mathcal{C}^k(\Omega)$ o conjunto de todas as funções reais contínuas definidas em Ω tal que $D^\alpha u$ é contínua em Ω para todo $\alpha = (\alpha_1, ..., \alpha_n)$ com $|\alpha| \leq k$.

Assumindo que Ω é um conjunto limitado, $C^k(\overline{\Omega})$ denotará o conjunto de todas as funções $u \in C^k(\Omega)$ tal que $D^\alpha u$ pode ser estendida de forma contínua de Ω para $\overline{\Omega}$. $C^k(\overline{\Omega})$ pode ser equipado com a norma:

$$\| u \|_{C^k(\overline{\Omega})} := \sum_{|\alpha| \leq k} sup_{x \in \Omega} \mid D^\alpha u(x) \mid$$

Definição 2.1.1 *Chamamos de suporte de uma função real contínua u definida em $\Omega \subset \mathcal{R}^n$ ao fecho em Ω do conjunto $\{x \in \Omega : u(x) \neq 0\}$. Se esse conjunto é compacto e é um conjunto do interior de Ω, então dizemos que u tem suporte compacto em relação a Ω. Usamos a notação $supp\, u$ para designar o suporte compacto de u.*

Denotamos por $C_0^k(\Omega)$ o conjunto de todas as $u \in C^k(\Omega)$ que possuem suporte compacto em Ω. Para o caso de $k = \infty$, temos:

$$C_0^k(\Omega) = \cap_{k \geq 0} C_0^k(\Omega) = D(\Omega)$$

2.2 Espaços de funções integráveis

Consideremos agora espaços de funções que são Lebesgue-integráveis. Seja $p \in \mathcal{R}, p \geq 1$, também $\Omega \subset \mathcal{R}$ aberto. Denotamos por $\tilde{L}_p(\Omega)$ o conjunto das funções reais definidas em Ω tal que:

$$\int_\Omega |u(x)|^p dx$$

Consideramos também o conjunto:

$$N = \{u \in \tilde{L}_p(\Omega) : u(x) = 0 \text{ quase sempre em } \Omega\}$$

Pode-se verificar que $\tilde{L}_p(\Omega)$ é um espaço vectorial e N é um subespaço vectorial em $\tilde{L}_p(\Omega)$. Assim, faz sentido a seguinte definição:

Definição 2.2.1 $L_p(\Omega) = \tilde{L}_p(\Omega)/N$.

Ou seja, $L_p(\Omega)$ é o conjunto quociente de $\tilde{L}_p(\Omega)$ por N. Entendemos aqui a seguinte relação de equivalência: f, g pertencem a uma mesma classe de equivalência, se e somente se, $f - g \in N$, ou seja, $f - g = 0$ q.s. (quase sempre) em Ω, o que implica $f = g$ q.s. em Ω. Em resumo, $L_p(\Omega)$ é o conjunto das classes de equivalência de funções que são iguais quase sempre. $L_p(\Omega)$ equipado com a norma:

$$\| u \|_{L_p(\Omega)} = (\int_\Omega |u(x)|^p dx)^{\frac{1}{p}}$$

é um espaço de Banach.

Quando $p = \infty$ temos o espaço $L_\infty(\Omega)$ que consiste de funções u tal que $|u(x)|$ tem supremo essencial em Ω. Ou seja, existe $M \geq 0$ tal que $|u(x)| \leq M$ quase sempre em Ω. $L_\infty(\Omega)$ equipado com a norma:

$$\| u \|_{L_\infty(\Omega)} = ess.sup_{x \in \Omega} |u(x)|$$

é um espaço de Banach.

Um caso especial é quando $p = 2$. O espaço $L_2(\Omega)$ pode ser equipado com o produto interno

$$(u, v)_{L_2(\Omega)} = \int_\Omega u(x)v(x)dx;$$

evidentemente, $\| u \|_{L_2(\Omega)} = \sqrt{(u, u)_{L_2(\Omega)}}$.

Lema 2.2.1 ***(Desigualdade de Cauchy-Schwarz)*** *Sejam $u, v \in L_2(\Omega)$, então $uv \in L_1(\Omega)$ e*

$$|(u,v)| \leq \| u \|_{L_2(\Omega)} \| v \|_{L_2(\Omega)} .$$

Corolário 2.2.1 ***(Desigualdade triangular)*** *Seja $u, v \in L_2(\Omega)$, então $u+v \in L_2(\Omega)$ e*

$$\| (u+v) \| \leq \| u \|_{L_2(\Omega)} + \| v \|_{L_2(\Omega)} .$$

Observação 2 $L_2(\Omega)$ equipado com o produto interno $(.,.)$ é um espaço de Hilbert.

Lema 2.2.2 ***(Desigualdade discreta de Cauchy-Schwarz)*** *Seja $\{a_i\}$ e $\{b_i\}$ duas sequências de N números reais. Então é válida a desigualdade:*

$$\sum_{i=1}^{N} a_i b_i \leq (\sum_{i=1}^{N} a_i^2)^{1/2} (\sum_{i=1}^{N} b_i^2)^{1/2}$$

2.3 Espaços de Sobolev

No estudo do Método de Elementos Finitos, é incontornável o uso da integração por partes.

Proposição 2.3.1 (Fórmula de integração por partes) *Para quaisquer $u, v \in C^1(\overline{\Omega})$, resulta em*

$$\int_\Omega \frac{\partial u}{\partial x_i} v dx = -\int_\Omega \frac{\partial v}{\partial x_i} u dx + \int_{\partial\Omega} \eta_i uv ds, i = 1, ..., n$$

Onde η_i é a i-ésima componente do vector η, normal unitário exterior à $\partial\Omega$.

Proposição 2.3.2 (Fórmula de Green) *Sejam $u \in C^2(\overline{\Omega})$ e $v \in C^1(\overline{\Omega})$, então nos resulta a seguinte fórmula:*

$$\int_\Omega (-\Delta u) v dx = \int_\Omega \nabla u . \nabla v dx - \int_{\partial\Omega} (\eta . \nabla u) v ds,$$

Definição 2.3.1 *O conjunto das funções integraveis localmente, L^1_{loc}, é dado por*

$$L^1_{loc}(\Omega) = \{f : f \in L_1(K) \forall compacto K \subset int(\Omega)\}$$

Definição 2.3.2 *Suponhamos que* $u \in L^1_{loc}(\Omega)$ *e que exista* $\omega_\alpha \in L^1_{loc}(\Omega)$ *que satisfaça*

$$\int_\Omega \omega_\alpha v(x)dx = (-1)^{|\alpha|} \int_\Omega u(x) D^\alpha v(x)dx, \forall v \in D(\Omega).$$

Assim, dizemos que ω_α *é* α*-ésima derivada da função fraca* u *e denotamos por* $\omega_\alpha = D^\alpha u$.

Se u *é suficientemente suave, por exemplo* $u \in C^k(\Omega)$, *então sua derivada fraca* $D^\alpha u$ *para* $|\alpha| \leq K$ *coincide com a derivada parcial no sentido clássico.*

Proposição 2.3.3 *Suponha que* $u \in C^k(\Omega)$ *e seja* $vinD(\Omega)$, *então vale a fórmula:*

$$\int_\Omega D^\alpha u(x)\, v(x)\, dx = (-1)^{|\alpha|} \int_\Omega u(x)\, D^\alpha\, v(x)dx, \quad |\alpha| \leq k, \quad \forall v \in D(\Omega)$$

Demonstração: Podemos supor, sem perda de generalidade, que $\alpha_1 \neq 0$. *Usando a fórmula de integração por partes dada acima e o facto de* $vinD(\Omega)$ *temos,*

$$\begin{aligned}\int_\Omega \frac{\partial^{\alpha_1+\ldots+\alpha_n} u(x)}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}} v(x)\, dx &= \int_\Omega \frac{\partial}{\partial x_1} \left(\frac{\partial^{(\alpha_1-1)+\ldots+\alpha_n} u(x)}{\partial x_1^{\alpha_1-1} \ldots \partial x_n^{\alpha_n}}\right) v(x)\, dx \\ &= -\int_\Omega \left(\frac{\partial^{(\alpha_1-1)+\ldots+\alpha_n} u(x)}{\partial x_1^{\alpha_1-1} \ldots \partial x_n^{\alpha_n}}\right) \frac{\partial v(x)}{\partial x_1} dx\end{aligned}$$

assim, procedendo α_1 *vezes a integral por partes em relação a* x_1, *temos:*

$$\int_\Omega \frac{\partial^{\alpha_1+\ldots+\alpha_n} u(x)}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}} v(x)\, dx = (-1)^{\alpha_1} \int_\Omega \left(\frac{\partial^{(\alpha_2)+\ldots+\alpha_n} u(x)}{\partial x_2^{\alpha_2} \ldots \partial x_n^{\alpha_n}}\right) \frac{\partial^{\alpha_1} v(x)}{\partial x_1^{\alpha_1}} dx$$

fazendo o mesmo processo para todos os $\alpha_i s$ *que são diferentes de* 0, *teremos que*

$$\int_\Omega \frac{\partial^{\alpha_1+\ldots+\alpha_n} u(x)}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}} v(x)\, dx = (-1)^{\alpha_1+\ldots+\alpha_n} \int_\Omega u(x) \frac{\partial^{\alpha_1+\ldots+\alpha_n} v(x)}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}}$$

Lema 2.3.1 (du Bois-Reymond) *Suponha que* $w \in L^1_{loc}(\Omega)$. *Se*

$$\int_\Omega w(x)v(x)dx = 0 \forall v \in D(\Omega)$$

então $w(x) = 0$ *q.s. em* Ω.

Demonstração: Veja [7]

Para ver que a definição de derivada fraca está correcta, devemos mostrar que dada uma função localmente integrável, devemos ter que sua derivada fraca, se existir, é única. Essa constatação é directa usando o lema de du Bois-Reymond. Suponha que para uma $u \in L^1_{loc}(\Omega)$ *existam duas derivadas fracas.* $\dot{u}_1$ *e* $\dot{u}_2$. *Ou seja,*

$$\int_\Omega \dot{u}_1 v(x)dx = (-1)^{|\alpha|}\int_\Omega u(x)D^\alpha v(x)dx.\forall v \in D(\Omega) \tag{2.1}$$

e

$$\int_\Omega \dot{u}_2 v(x)dx = (-1)^{|\alpha|}\int_\Omega u(x)D^\alpha v(x)dx.\forall v \in D(\Omega) \tag{2.2}$$

Substituindo a equação (2.3) da equação (2.2), temos

$$\int_\Omega (\dot{u}_1 - \dot{u}_2)v(x)dx = 0$$

o que implica, pelo lema, que $\dot{u}_1 - \dot{u}_2 = 0$ *q.s. em* $\Omega \Longrightarrow \dot{u}_1 = \dot{u}_2$ *q.s. em* Ω.

Definição 2.3.3 *Seja* $K \in N$ *e* $p \in [1, \infty]$.

Definimos,

$$W_p^k(\Omega) = \{u \in L_p(\Omega) : D^\alpha u \in L_p(\Omega), |\alpha| \leq K\}$$

Neste caso, $W_p^k(\Omega)$ *é chamado de Espaço de Sobolev de ordem* k.

O espaço $W_p^k(\Omega)$ é equipado com a norma de Sobolev:

$$\| u \|_{W_p^k(\Omega)} = (\sum_{|\alpha|\leq k} \| D^\alpha u \|^p_{L_p(\Omega)})^{\frac{1}{p}}$$

quando $1 \leq p \leq \infty$ e

$$\| u \|_{W_p^k(\Omega)} = \sum_{|\alpha|\leq k} \| D^\alpha u \|_{L_\infty(\Omega)}$$

quando $p = \infty$.

Consideramos também a semi-norma,

$$|u|_{W_p^k(\Omega)} = (\sum_{|\alpha|\leq k} \| D^\alpha u \|^p_{L_p(\Omega)})^{\frac{1}{p}}$$

quando $1 \leq p \leq \infty$, assim podemos escrever a norma da forma:

$$\| u \|_{W_p^k(\Omega)} = (\sum_{j=0}^{k} |D^\alpha u|^p_{W_p^j(\Omega)})^{\frac{1}{p}}$$

Desta forma,

$$|u|_{W^k_\infty(\Omega)} = \sum_{|\alpha|=k} \| D^\alpha u \|_{L_\infty(\Omega)}$$

o que permite escrever,

$$\| u \|_{W^k_\infty(\Omega)} = \sum_{j=0}^{k} |u|_{W^j_\infty(\Omega)}$$

o caso especial é quando $p = 2$, o espaço $W^k_2(\Omega)$ é um espaço de Hilbert com produto interno:

$$(u, v)_{W^k_2(\Omega)} = \sum_{|\alpha|\leq k} (D^\alpha u, D^\alpha v).$$

Para este caso usamos uma notação especial, $H^k(\Omega) = W^k_2(\Omega)$. Usando as definições de norma e semi-norma em $W^1_2(\Omega)$, temos

$$H^1(\Omega) = \{u \in L_2(\Omega) : \frac{\partial u}{\partial x_j} \in L_2(\Omega), j, ..., n\}$$

Definição 2.3.4 *O espaço $L^2(\Omega)$ é o espaço das funções quadradas integráveis definidas em Ω ou seja*

$$L^2(\Omega) = \{f : \Omega \longrightarrow \mathcal{R} | f; \int_\Omega f^2 < \infty\}$$

onde f é mensurárel.

Definição 2.3.5 *Seja $\Omega \subset R^n$ um conjunto aberto, conexo e limitado e seja $\partial\Omega$ sua fronteira. Se $n \geq 2$, dizemos que $\partial\Omega$ é uma fronteira **Lipschitziana**, se existe uma cobertura aberta finita $U_1, ..., U_m$ de $\partial\Omega$ tal que para $j = 1, ..., m$, temos:*

1. $\partial\Omega \cap U_j$ é o gráfico de uma função Lipschitziana g_j e;
2. $\Omega \cap U_j$ está em um lado desta curva.

Lema 2.3.2 ***(Desigualdade de Poincaré-Friedrich)*** *Suponha $\partial\Omega$ Lipschitziana e seja $u \in H^1_0(\Omega)$. Então existe uma constante $c_*(\Omega)$, independente de u, tal que*

$$\int_\Omega |u(x)|^2 dx \leq c_* \sum_{i=1}^{n} \int_\Omega |\frac{\partial u}{\partial x_i}(x)|^2 dx$$

ou de outra forma

$$\| u \|^2_{L_2(\Omega)} \leq c_* \mid u \mid_{H^1(\Omega)^2}$$

Demonstração encontra-se em [7]

Teorema 2.3.1 *$L^2(\Omega)$ é um espaço de Hilbert considerando o seguinte produto interno*

$$\langle u, v\rangle_\Omega = \int_\Omega uv$$

Teorema 2.3.2 ***(Desigualdade de Sobolev)*** *Seja $\Omega \subset R^n$ com $\partial\Omega$ Lipschitziana, seja k um inteiro positivo e seja $p \in R$ com $1 \leq p \leq \infty$ tal que*

$$k \geq n \quad quando \quad p = 1$$

$$k > n/p \quad quando \quad p > 1$$

Então existe uma constante $\mathcal{C}$ tal que para toda $u \in W_p^k(\Omega)$

$$\parallel u \parallel_{L_\infty(\Omega)} \geq \mathcal{C} \parallel u \parallel_{W_p^k(\Omega)}$$

Além disso, existe uma função contínua em $L_\infty(\Omega)$ na classe de equivalência de u.

Demonstração: Veja [7]

Corolário 2.3.1 *Seja $\Omega \subset \mathcal{R}^n$ com $\partial\Omega$ Lipschitziana, sejam k e m inteiros positivos satisfazendo $m > k$ e seja $p \in \mathcal{R}$ com $1 \leq \leq \infty$ tal que*

$$k - m \geq n \text{quando} p = 1$$

$$k - m > n/p \text{quando} p > 1$$

Então, existe uma constante C tal que para toda $uinW_p^k(\Omega)$

$$\parallel u \parallel_{W_\infty^m(\Omega)} \leq C \parallel u \parallel_{W_p^k(\Omega)}$$

Além disso, existe uma função que é $C^m(\Omega)$ em $L_p(\Omega)$ na classe de equivalência de u.

Demonstração: Veja [7]

Teorema 2.3.3 ***(Traço)*** *Seja Ω um dominio com fronteira Lipschitziana e p satisfazendo $1 \leq p \leq \infty$. Então existe uma constante C tal que:*

$$\parallel u \parallel_{L_p(\partial\Omega)} \leq C \parallel u \parallel_{L_p(\Omega)}^{1/p} \parallel u \parallel_{W_p^1(\Omega)}^{1/p}, \forall u \in W_p^1(\Omega).$$

Quando $p = 2$, a desigualdade anterior se torna:

$$\| u \|_{L_2(\partial\Omega)} \leq C \| u \|^{1/2}_{L_2(\Omega)} \| u \|^{1/2}_{H^1(\Omega)}, \forall u \in H^1(\Omega).$$

É facil ver que $\| u \|_{L_2(\Omega)} \leq \| u \|_{H^1(\Omega)}$. Por fim temos

$$\| u \|_{L_2(\partial\Omega)} \leq C \| u \|_{H^1(\Omega)}, \forall u \in H^1(\Omega).$$

Por outro lado, é de tamanha importância, introduzirmos também outros ente matemáticos que são bastante utilizados no estudo de métodos de elementos finitos.

Se $u = (u_1, u_2, ..., u_d)$ é uma função vectorial, definimos o gradiente de u como a matriz

$$\underline{\nabla u} = \left[\frac{\partial u_i}{\partial x_j} \right]_{i,j=1,2,...,d}$$

e se $\underline{A} = (a_{ij})$ for uma matriz, então definimos por

$$\nabla . \underline{A} = \left(\sum_{j=1}^{d} \frac{\partial a_{1j}}{\partial x_j}, \sum_{j=1}^{d} \frac{\partial a_{2j}}{\partial x_j}, ..., \sum_{j=1}^{d} \frac{\partial a_{dj}}{\partial x_j} \right).$$

A anterior definição, permite-nos também definir

$$\Delta u = \nabla . \underline{\nabla u}.$$

E para o divergente de um vector $u = (u_1, u_2, ..., u_d)$ se pode utilizar tanto $divu$, bem como $\nabla . u$. Logo,

$$divu = \nabla . u = \sum_{i=1}^{d} \frac{\partial u_i}{\partial x_i}.$$

2.4 Partição de Elementos Finitos

Consideremos $\Omega \subset \mathcal{R}^2$, aberto e limitado com fronteira $\partial\Omega$ Lipchitziana. Denotamos por Γ_D a parte da fronteira sobre a qual temos condições de Dirichlet e Γ_N onde temos condições de Neumann. Essa divisão da fronteira deve ser tal que $\Gamma_D \cap \Gamma_N = \emptyset$ e $\overline{\Gamma}_D \cup \overline{\Gamma}_N = \partial\Omega$.

Seja $\mathcal{P}_h$ uma partição de Ω, ou seja, $\mathcal{P}_h$ é um conjunto finito de subdomínios K tal que:

$$\overline{\Omega} = \cup_{K_i \in \mathcal{P}_h} \overline{K}_i, \text{ e } K_i \cap K_j = \emptyset, \quad i \neq j$$

Vamos definir agora dois coeficientes que conterão informações sobre a forma e tamanho de um elemento K da partição $\mathcal{P}_h$. Nomeadamente: h_k e ρ_k, cuja definição é:

$$h_k = \text{ diâmetro de } K$$
$$\rho_k = sup\{\text{diâmetro de } \mathcal{B}; \mathcal{B} \text{ é bola contida em } K\}$$

É importante observar que estas bolas mencionadas na definição de ρ_k são em relação a norma Euclidiana.

Assim, cada elemento K tem seu h_k e seu ρ_k. Pelo facto de $\mathcal{P}_h$ possuir número finito de elementos K, vamos chamar de h o maior dos h_k em $\mathcal{P}_h$, ou seja:

$$h = \max_{K \in \mathcal{P}_h} h_k$$

Para evitar a degeneração de elementos K de uma partição $\mathcal{P}_h$ quando h tende para zero, faz-se necessária a definição:

Definição 2.4.1 *A família $\{\mathcal{P}_h\}$ de partições $\mathcal{P}_h$ é dita ter forma regular quando $h \longrightarrow 0$, se existe um número $\varrho > 0$ que independe de h e de K tal que:*

$$\frac{h_K}{\rho K} \leq \varrho, \forall K \in \mathcal{P}_h.$$

Com cada elemento K associamos a sua fronteira ∂K e, nesta fronteira, associamos o vector unitário exterior que será denotado por n.

Dada uma partição $\mathcal{P}_h$, denotaremos o conjunto das arestas em $\mathcal{P}_h$ por $\varepsilon_h = \{\gamma_l\}$ com $l = 1, ..., N_h$, onde N_h é um inteiro que depende de h. Essas arestas podem estar em $\partial\Omega$ ou no interior de Ω, assim tem sentido a definição:

$$\Gamma_{int} = (U_{l=1}^{N_h}) \setminus \partial\Omega,$$

ou seja, as arestas que estão no interior de Ω. Assim, vemos que $\gamma_{ij} \subset \Gamma_{int}$ denotando a aresta localizada entre os elementos K_i e K_j, o que por convenção temos $i > j$. Em cada aresta γ, associamos o vector normal unitário exterior n. Se γ estiver associado com K_i adjacente à $\partial\Omega$, ou seja, $\gamma \subset \Gamma_D \cup \Gamma_N$, então o vector normal unitário é definido como $n = n|_i$.

Para um $\gamma_{ij} \subset \Gamma_{int}$, usando a convenção $i > j$, n é escolhido como o vector normal unitário exterior a K_i, ou seja $n|_i$. E também $n|_i = -n|_j$.

2.5 Espaço particionado de Sobolev

De maneira análoga, como fizemos com $H^1(\Omega)$ e $H^2(\Omega)$, definimos $H^s(\Omega)$ com s um inteiro positivo. Definimos também $H^s(S)$, onde S é um aberto e $S \subset \Omega$ pode ser todo $\Omega, K \in P_h$ ou mesmo um $\gamma \in \varepsilon_h$. O espaço $H^s(S)$ denotará o usual espaço de Sobolev com norma respectiva $\| \, . \, \|_{H^s(S)}$. E por fim, definimos o espaço particionado de Sobolev:

$$H^s(P_h) = \{v \in L_2(\Omega); v|_k \in H^s(K), \forall K in P_h\}.$$

cuja norma associada é dada por

$$\| v \|_{H^s(P_h)} = (\sum_{K \in P_h} \| v \|^2_{H^s(K)})^{1/2}.$$

2.6 Salto e média

Nesta secção, achamos imperioso fazer uma abordagem para explicar os ente matemáticos, salto e média.

Ora, consideremos $\mathcal{T}_h$, uma partição de Ω e K um elemento desta partição.

Cada elemento K tem suas faces, que podem ser segmentos de recta ou de planos. O conjunto dessas faces que não estão em $\partial\Omega$, chamaremos de Γ_h, isto é , o conjunto das faces interiores ao domínio Ω. Para cada face e de $\Gamma_h \cup \partial\Omega$, é associado um vector normal exterior unitário n_e. Se por acaso $e \in \partial\Omega$, então n_e será a normal exterior e unitária de $\partial\Omega$.

Se tomarmos $v \in H^1(\mathcal{T}_h)$ teremos então a garantia de que o traço de v está bem definido em todas as faces de $K, \forall K \in \mathcal{T}_h$. No entanto, os dois elementos K_i e K_j compartilham a mesma face e teremos dois traços definidos nessa face, sabendo que a normal n_e é convencionada ser de K_i para K_j. Assim, teremos a média e o salto de v em função da face.

$$[v] = v_i - v_j$$

$$\langle v \rangle = \frac{1}{2}(v_i + v_j)$$

Capítulo 3

Método Galerkin Descontínuo

Consideremos o problema do modelo de Lotka-McKendrick, introduzido no capítulo 1:

$$
\begin{aligned}
&u_t(a,t) + u_a(a,t) + \mu(a)u(a,t) = 0, \quad 0 < a < a_\dagger, \quad t > 0, \\
&u(a,0) = \int_0^{a_\dagger} \beta(a)u(a,t)da, \quad t > 0, \\
&u(a,0) = u_0(a), \quad 0 \leq a < a_\dagger
\end{aligned} \tag{3.1}
$$

Entretanto, assumimos que as funções básicas $\beta(\cdot)$ e $\mu(\cdot)$ satisfazem as suposições seguintes:
$\beta(\cdot)$ é não negativa e pertence à $L^\infty(0, a_\dagger)$,
$\mu(\cdot)$ é não negativa e pertence à $L^1_{loc}(0, a_\dagger)$,
$\int_0^{a_\dagger} \mu(\sigma)d\sigma = +\infty$,
$u_0 \in L^1(0, a_\dagger), u_0(a) \geq 0, a \in [0, a_\dagger]$.

3.1 Método de Elementos Finitos Galerkin Descontínuo

O Método de Elementos Finitos Galerkin Descontínuo é uma variação do Método de Elementos Finitos, no qual as funções de aproximação utilizadas são descontínuas entre os elementos adjacentes, consequentemente, os graus de liberdade de cada elemento são independentes do elemento vizinho. A continuidade da solução entre os elementos é imposta de maneira fraca na formulação variacional, por meio de termos integrais sobre fronteiras dos elementos. No entanto, o Método de Galerkin Descontínuo é um método local, ou seja, permite que as soluções sejam obtidas elemento por elemento. O cálculo necessário para cada elemento é essencialmente feito da mesma forma que no Método de Elementos Finitos Galerkin Contínuo. Análogamente ao caso contínuo, a formulação de Galerkin Descontínuo é obtida quando as funções de interpolação são as mesmas usadas tanto para solução aproximada assim como para as funções de ponderação, esta abordagens têm antecedentes à Cockburn, 2003, encontradas em [2].

Portanto, o Método Galerkin Descontínuo utiliza os mesmos princípios do Método de Elementos Finitos Contínuo, as diferenças se apresentam na construção de formulação variacional, onde não é assumida a continuidade entre os elementos, e na definição do espaço de aproximação utilizado.

O Método Galerkin Descontínuo, possui diversas vantagens, pois é um método localmente conservativo, estável, paralelizável, apresenta também uma alta ordem de precisão, que depende apenas da solução exacta do problema, adapta-se facilmente a geometria complexas.

Vamos no entanto, considerar o Método de Elementos Finitos Galerkin Descontínuo para determinar a solução aproximada do problema 3.1. Vamos começar por introduzir o referido Método para uma equação linear hiperbólica, utilizando um Problema de Valor Fronteira mais simples em relação ao problema 3.1, para à partir daí adequarmos ao mesmo problema. Seja Ω um dominio poligonal convexo de plano $\mathcal{R}^2$ com fronteira Γ e seja $\gamma = (\gamma_1, \gamma_2)$ um vector constante com $|\gamma| = 1$. Consideremos o seguinte Problema de Valor Fronteira:

$$\begin{aligned} u_\gamma + u &= f \quad em \quad \Omega, \\ u &= g \quad em \quad \Gamma_-, \end{aligned} \tag{3.2}$$

onde Γ_- é a fronteira de entrada e é definida por

$$\Gamma_- = \{x \in \Gamma : n(x).\gamma < 0\}.$$

E neste caso, $\mathbf{n}(x)$, denota a unidade externa normal para Γ ao ponto $x \in \Gamma$, e $v_\gamma = \gamma \cdot \nabla v$ é a derivada direcional na direcção de γ. Para introduzir o Método de Elementos Finitos Galerkin Descontínuo, primeiro vamos introduzir algumas anotações importantes para nos ajudar a interpretar o problema. Seja $\mathcal{T}_h$ uma triangulação admissível de Ω com diâmetro h que geralmente satisfaz a condição de ângulo mínimo. Para $K \in \mathcal{T}_\langle$, dividimos a fronteira ∂K de cada triângulo K em parte de entrada ∂K_- e em parte de saída ∂K_+, as quais defimos em:

$$\partial K_- = \{x \in \partial K : \mathbf{n}(x) \cdot \gamma < 0\},$$

$$\partial K_+ = \{x \in \partial K : \mathbf{n}(x) \cdot \gamma \geq 0\}.$$

Consideremos também uma função v que pode ter uma discontinuidade de salto na fronteira entre triângulos, e definimos os limites a esquerda v_- e à direita v_+ por

$$v_- = \lim_{s \to 0^-} \quad v(x + s\gamma),$$

$$v_+ = \lim_{s \to 0^+} \quad v(x + s\gamma),$$

e também definimos o salto $[v]$ entre triângulos adjacentes por

$$[v] = v_+ - v_-.$$

Usaremos também

$$(u, v) = \int_\Omega u\, v\, dx,$$

e

$$< u, v >= \int_\Gamma u\, v\, (\mathbf{n} \cdot \gamma)\, ds$$

Temos também

$$< u, v >_- = \int_{\Gamma_-} u\, v\, (\mathbf{n} \cdot \gamma)\, ds \quad < u, v >_+ = \int_{\Gamma_+} u\, v\, (\mathbf{n} \cdot \gamma)\, ds$$

E assim, a forma fraca para o problema 3.2, temos, encontrar $u \in L^2(\Omega)$ tal que

$$-(u, v_\gamma) + (u, v) + < u, v >_+ \;=\; (f, v) - < g, v >_-, \quad \forall v \in H^1(\Omega). \tag{3.3}$$

No entanto, vamos agora considerar o Método Galerkin Descontínuo de Elemento Finito para a forma fraca 3.3, o qual, está baseado no seguinte espaço de elementos finitos:

$$W_h = \{v \in L^2(\Omega) : v|_K \in P_r(K), \forall K \in \mathcal{T}_h\}.$$

Aqui $P_r(K)$ é o espaço de polonómio de grau $r \geq 0$. E em razão disto, somamos as integrais da forma fraca 3.3, por cada elemento do polinómio e no entanto, estamos a achar $u^h \in W_h$, tal que,

$$\sum_{K \in \mathcal{T}_h} \int_K (-u^h\, v_\gamma + u^h\, v) d\Omega + < u^h, v >_+ \tag{3.4}$$
$$= \sum_{K \in \mathcal{T}_h} \int_K f\, v\, d\Omega - < g, v >_-, \quad \forall v \in P_r(K)$$

Assim, usamos a fórmula de Green para $\int_K u^h\, v_\gamma\, dx$

$$\int_K u^h\, v_\gamma\, d\Omega = \int_{\partial K_-} u^h_+\, v_+\, (\mathbf{n} \cdot \gamma)\, d\Gamma + \int_{\partial K_+} u^h_-\, v_+\, (\mathbf{n} \cdot \gamma)\, d\Gamma - \int_K u^h_\gamma\, v\, d\Omega. \tag{3.5}$$

Somando as integrais dos K, teremos,

$$\sum_{K \in \mathcal{T}_h} \int_K u^h\, v_\gamma\, d\Omega \;=\; \sum_{K \in \mathcal{T}_h} \{ \int_{\partial K_-} u^h_+\, v_+\, (\mathbf{n} \cdot \gamma) d\Gamma + \int_{\partial K_+} u^h_-\, v_+\, (\mathbf{n} \cdot \gamma) d\Gamma - \int_K u^h_\gamma\, v\, d\Omega \}$$

$$= -\sum_{K \in \mathcal{T}_h} \int_K u^h_\gamma\, v\, d\Omega + \sum_{\partial K_- \subset \Gamma_-} \int_{\partial K_-} u^h_+\, v_+\, (\mathbf{n} \cdot \gamma)\, d\Gamma + \sum_{\partial K_+ \subset \Gamma_+} \int_{\partial K_+} u^h_-\, v_+\, (\mathbf{n} \cdot \gamma)\, d\Gamma +$$
$$+ \sum_{\partial K_- \not\subset \Gamma_-} \int_{\partial K_-} [u^h]\, v_+\, (\mathbf{n} \cdot \gamma)\, d\Gamma.$$

Partindo de que,

$$\sum_{\partial K_+ \subset \Gamma_+} \int_{\partial K_+} u_-^h \, v_+ \, (\mathbf{n} \cdot \gamma) \, d\Gamma =< u^h, v >_+,$$

obtemos então, a seguinte expressão do Método de Elemento Finito Galerkin Descontínuo: Encontrar $u^h \in W_h$ tal que

$$\begin{aligned} &\sum_{K \in \mathcal{T}_h} \int_K (u_\gamma^h + u^h) \, v \, d\Omega - \sum_{\partial K_- \subset \Gamma_-} \int_{\partial K_-} u_+^h \, v_+ \, (\mathbf{n} \cdot \gamma) \, d\Gamma \\ &- \sum_{\partial K_- \not\subset \Gamma_-} \int_{\partial K_-} [u^h] \, v_+ \, (\mathbf{n} \cdot \gamma) \, d\Gamma \\ = \; &\sum_{K \in \mathcal{T}_h} \int_K f \, v \, d\Omega - \int_{\Gamma_-} g \, v_+ (\mathbf{n} \cdot \gamma) \, d\Gamma, \quad \forall v \in P_r(K). \end{aligned}$$

Como a função $v \in W_h$ varia independente em cada K, podemos formular alternativamente como se segue: Para $K \in \mathcal{T}_h$, vem u_-^h em ∂K_- achar $u^h \equiv u^h|_{K \in \mathcal{P}_r(K)}$ que,

$$\begin{aligned} &\int_K (u_\gamma^h + u^h) \, v \, d\Omega - \int_{\partial K_-} u_+^h \, v_+ \, (\mathbf{n} \cdot \gamma) \, ds \\ = \; &\int_K f \, v \, d\Omega - \int_{\partial K_-} u_-^h \, v_+ (\mathbf{n} \cdot \gamma) \, ds, \quad \forall v \in P_r(K). \end{aligned} \tag{3.6}$$

Se pode observar aqui que se $\partial K_- \subset \Gamma_-$, então $u_-^h|_{\partial K_-} = g$, isto de acordo com o problema 3.2. e também, se u_-^h é calculado em ∂K_-, deduzimos que $u^h|_K$ é determinado exclusivamente por 3.6. Assim, partimos para determinar u^h a partir do triângulo K com $\partial K_- \subset \Gamma_-$ e então determinamos u^h em triângulos K adjacentes.

Agora, aplicamos então, o Método Galerkin Descontínuo de elemento finito 3.6 à equação de Lotka-McKendrick 3.1. Seja $T > 0$, o tempo final e tenhamos $\Omega = [0, a_\dagger] \times [0, T]$. E a derivada da direção γ, é então determinada por $\gamma = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. E a fronteira de entrada Γ_-, está composta por

$$\Gamma_1 = \{(0, t) : t > 0\} \quad e \quad \Gamma_2 = \{(a, 0) : 0 \leq a < a_\dagger\},$$

ou seja ,

$$\Gamma_- = \Gamma_1 U \Gamma_2.$$

E nesta conformidade, a equação de Lotka-McKendrick pode ser escrita assim:

$$\begin{aligned} u_\gamma(a,t) + \frac{1}{\sqrt{2}} \mu(a) u(a,t) = 0, \text{ em } \Omega, \\ u(a,t) = g(a,t), \text{ em } \Gamma_-, \end{aligned} \tag{3.7}$$

onde

$$g(a,t) = \begin{cases} \int_0^{a_\dagger} \beta(\sigma) u(\sigma, t) d\sigma, \text{ se } (a,t) \in \Gamma_1, \\ u_0(a), \text{ se } (a,t) \in \Gamma_2. \end{cases} \tag{3.8}$$

E portanto, a forma variacional para 3.7, é achar $u \in L^2(\Omega)$ tal como no 3.3 e nos resulta,

$$-(u, v_\gamma) + (\frac{1}{\sqrt{2}}\mu\, u, v) + < u, v >_+ = - < g, v >_-, \quad \forall v \in H^1(\Omega). \tag{3.9}$$

Agora consideramos o método de elemento finito Galerkin Descontínuo usando uma triangulação no dominio $\Omega = [0, a_\dagger] \times [0, T]$. A partir de agora vamos considerar o espaço de funções lineares a partes sobre a triangulaç ao $\mathcal{T}_h$ e pomos

$$W_h = \{v \in L^2(\Omega) : v|_K \in P_1(K), \quad \forall K \in \mathcal{T}_h\}.$$

Denotamos $h = \Delta a = \Delta t$ os diâmetros das patições, de maneiras que $M = a_\dagger/h$ e $N = T/h$ sejam inteiros positivos. Seja $a_j = j\,h, 0 \le j \le J, t^n = n\,h, 0 \le n \le N$ as coordenadas da malha. Os triângulos se denotam K_j^n e $\tilde{K}_j^n$ dependendo se a fronteira de entrada é paralela ao eixo de idade a ou eixo de tempo t.

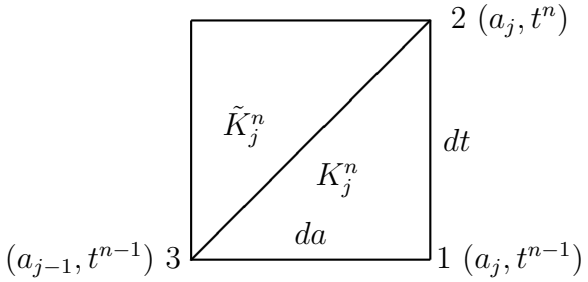

Figura 3.1: triângulo K_j^n

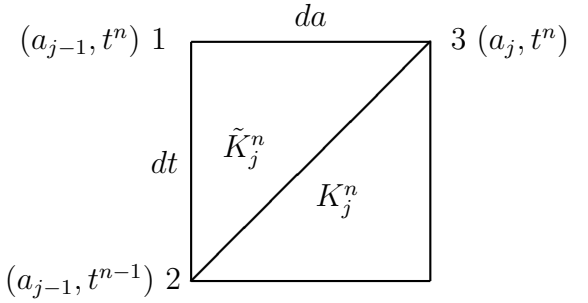

Figura 3.2: triângulo $\tilde{K}_j^n$

O método Galerkin Descontínuo que resulta é: Para $K \in \mathcal{T}_h$, dado u_-^h sobre ∂K_- encontrar $u^h \in P_1(K)$ tal que

$$\int_k (u_a^h + u_t^h + \mu\, u^h) v\, d\Omega + \int_{\partial K_-} u_+^h\, v_+\, d\Gamma = \int_{\partial K_-} u_-^h\, v_+\, d\Gamma, \quad \forall v \in \mathbb{P}_1(K). \tag{3.10}$$

Computamos $u^h \in W_h$ triângulo à triângulo começando com os triângulos da primeira banda que se recorrem da esquerda para direita. A continuação se calcula u^h

aos triângulos da segunda banda e assim sucessivamente. Se pormos $u^h \in P_1(K)$ como

$$u^h = \sum_{i=1}^{3} U_i T_i, \tag{3.11}$$

donde $T_i(a,t)$, $i = 1,2,3$, são as funções bases associadas aos vértices do triângulo K e substituimos (3.11) em (3.10) e pondo $v = T_j$, $j = 1,2,3$, obtemos o sitema linear de equações

$$A\mathbf{U} = \mathbf{b},$$

donde

$$A = (a_{ij})_{i,j=1}^{3}, \quad a_{ij} = \int\int_K (\frac{\partial T_j}{\partial a} + \frac{\partial T_j}{\partial t}) T_i \, da \, dt + A_i^j(K),$$

onde

$$A_i^j(K) = \int\int_K \mu(a) \, T_i(a,t) \, T_j(a,t) \, da \, dt.$$

Por outro lado,

$$B = (b_i)_{i=1}^{3}, \quad b_i = \int_{\partial K_-} u_-^h \, T_i d\Gamma,$$

e

$$U = (U_i)_{3\times 1}$$

Para calcular as integrais sobre cada triângulo K

$$\int_K \frac{\partial T_j}{\partial a} \, T_i \, da \, dt, \quad \int_K \frac{\partial T_j}{\partial t} \, T_i \, da \, dt, \quad \int_K \mu \, T_i \, T_j \, da \, dt.$$

determinamos as correspondentes funções bases. Para os triângulos K_j^n como da figura podemos determinar as seguintes funções bases:

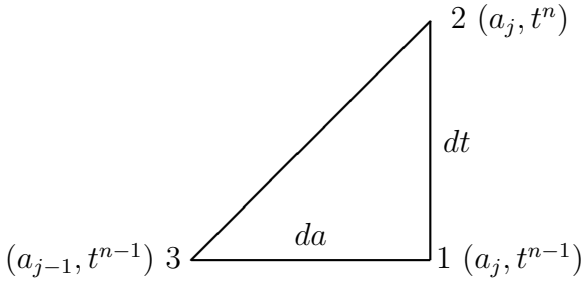

Figura 3.3: triângulo K_j^n

$$T_1(a,t) = \frac{(a - a_j) - (t - t^n)}{dt};$$

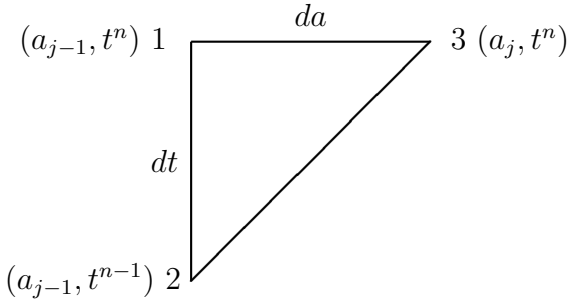

Figura 3.4: triângulo $\tilde{K}_j^n$

$$T_2(a,t) = \frac{t - t^{n-1}}{dt};$$

$$T_3(a,t) = \frac{a_j - a}{da}.$$

Para os triângulos $\tilde{K}_j^n$ como da figura
as funcções bases saõ

$$\tilde{T}_1(a,t) = \frac{(a_j - a) - (t^n - t)}{da};$$

$$\tilde{T}_2(a,t) = \frac{t^n - t}{dt};$$

$$\tilde{T}_3(a,t) = \frac{a - a_{j-1}}{da}.$$

Isto significa, que a partir da expressão 3.10, podemos determinar as i filas e as j colunas da matriz do sistema que dà à solução aproximada.

A expressão (3.12),

$$\int\int_K (\frac{\partial T_1}{\partial t} + \frac{\partial T_1}{\partial a}) T_1(a,t) da dt + \int\int_K \mu(a) T_1 \, T_1 \, dt + \int_{a_{j-1}}^{a_j} T_1 \, T_1 \, da = \frac{h}{3} + A_1^1(K). \tag{3.12}$$

o seu resultado, nos garante o valor da primeira fila, primeira coluna da matriz de rigidez. E, por outro lado, a expressão (3.13), nos garante o valor da segunda fila, primeira coluna e assim sucessivamente, se procede até concluir com todos elementos da matriz.

$$\int\int_K (\frac{\partial T_1}{\partial t} + \frac{\partial T_1}{\partial a}) T_2(a,t) da dt + \int\int_K \mu(a) T_1 . T_2 dt + \int_{a_{j-1}}^{a_j} T_1 . T_2 da = \frac{h}{6} + A_1^2(K) \tag{3.13}$$

Depois de determinados os valores todos, nos resulta então, a matriz:

$$A = \begin{pmatrix} \frac{h}{3}+A_1^1(K) & \frac{h}{6}+A_1^2(K) & A_1^3(K) \\ A_2^1(K) & \frac{h}{6}+A_2^2(K) & -\frac{h}{6}+A_2^3(K) \\ \frac{h}{6}+A_3^1(K) & \frac{h}{6}+A_3^2(K) & \frac{h}{6}+A_3^3(K) \end{pmatrix} \tag{3.14}$$

que corresponde aos cálculos do triângulo K_j^n.

Da mesma maneira, determinamos a matriz do triângulo K, quando este for do tipo $\tilde{K}_j^n$ e resulta,

$$A = \begin{pmatrix} \frac{h}{3}+A_1^1(\tilde{K}) & A_1^2(\tilde{K}) & \frac{h}{6}+A_1^3(\tilde{K}) \\ \frac{h}{6}+A_2^1(\tilde{K}) & \frac{h}{6}+A_2^2(\tilde{K}) & \frac{h}{6}+A_2^3(\tilde{K}) \\ A_3^1(\tilde{K}) & -\frac{h}{6}+A_3^2(\tilde{K}) & \frac{h}{6}+A_3^3(\tilde{K}) \end{pmatrix} \tag{3.15}$$

Como vimos anteriormente que

$$g(a,t) = \begin{cases} \int_0^{a_\dagger} \beta(\sigma)u(\sigma,t)d\sigma, \text{ se } (a,t) \in \Gamma_1, \\ u_0(a), \text{ se } (a,t) \in \Gamma_2. \end{cases},$$

vector B é computado de forma diferente dependendo do K e da condição do fluxo de entrada. Para $\partial K_- \subset \Gamma_2$, temos:

$$B = \begin{pmatrix} \int_{\partial K_-} u_0(a)T_1 da \\ 0 \\ \int_{\partial K_-} u_0(a)T_3 da \end{pmatrix}, \tag{3.16}$$

que corresponde ao caso de triângulo adjacente à fronteira horizontal.

Na mesma análise, para $\partial K_- \subset \Gamma_1$, a condição de fronteira envolve a função desconhecida u_-^h. E assim, para computar, aproximamos $u_-^h(0,t^n)$ usando a regra de Simpson,

$$s(n) = \frac{h}{3-h\beta(0)}\{\beta(a_\dagger)u^h|_{K_M^n}(a_\dagger,t^j) + 4\sum_{j=1}^{M/2}\beta(a_{2j-1})u^h|_{K_{2j-1}^n}(a_{2j-1},t^n) + 2\sum_{j=1}^{(M-2)/2}\beta(a_{2j})u^h|_{K_{2j}^n}(a_{2j},t^n)\}. \tag{3.17}$$

Nesta conformidade, para $\partial K_- \subset \Gamma_1$, temos,

$$B = \begin{pmatrix} \frac{h}{6}(2s(n)+s(n-1)) \\ \frac{h}{6}(2s(n-1)+s(n)) \\ 0 \end{pmatrix}, \tag{3.18}$$

É de salientar que, para $K = K_j^n, n \neq 1$, B é obtido previamente, ou seja, no caso do triângulo interior, temos que,

$$B = \begin{pmatrix} 0 \\ \frac{h}{6}(2U_{j-1}^{n-1}+U_{j-1}^n) \\ \frac{h}{6}(U_{j-1}^{n-1}+2U_{j-1}^n) \end{pmatrix}, \tag{3.19}$$

onde

$$U_{j-1}^n = u^h|_{\tilde{K}_j^{n-1}}(a_{j-1},t^{n-1}) \quad e \quad U_{j-1}^{n-1} = u^h|_{\tilde{K}_j^{n-1}}(a_j,t^{n-1}).$$

Da mesma forma, vimos que, para $K = K_j^n, n \neq 1$, B é obtido previamente do passo da tira. De salientar, que este é no caso de triângulos internos, que nos resulta em,

$$B = \begin{pmatrix} \frac{h}{6}(2\tilde{U}_{j-1}^{n-1} + \tilde{U}_j^{n-1}) \\ \frac{h}{6}(2\tilde{U}_j^{n-1} + \tilde{U}_{j-1}^{n-1}) \\ 0 \end{pmatrix}, \tag{3.20}$$

onde

$$\tilde{U}_{j-1}^{n-1} = u^h|_{K_{j-1}^n}(a_{j-1}t^n) \quad e \quad \tilde{U}_j^{n-1} = u^h|_{K_{j-1}^n}(a_{j-1}, t^{n-1}).$$

Neste contexto, computamos a solução discreta u^h sucessivamente de partição à partição, movendo de triângulo à triângulo da esquerda para direita, primeiro tratamos de determinar a solução computacional de u^h no triângulo K_j^n sendo $j = 1, ..., M$ na dimenção $[0, a_\dagger] \times [t^{n-1}, t^n]$. Seguidamente, determinamos a solução computacional de u^h no triângulo $\tilde{K}_j^n$ sendo $j = 1, ..., M$ e também na dimenção seguinte $[0, a_\dagger] \times [t^n, t^{n+1}]$.

3.2 Experimentos computacionais

Nesta secção, apresentamos os resultados numéricos, dos testes feitos computacionalmente na linguagem de Matlab.

3.2.1 Exemplo 1:

Para este primeiro exemplo, resolvemos o problema 3.1, com os seguintes dados:

$\beta(a) = 20a(1-a)$, como função de fertilidade e $\mu(a) = 10exp(-100(1-a))$, função de mortalidade, $a_\dagger = 1$ e $u(a,0) = u_0(a)$ funções de dado inicial. A solução exacta é do tipo separável $u(a,t) = \omega_0\, \omega(a)\, \exp(\alpha^* t)$, e

$$\omega(a) = \exp(-\int_0^a \mu(\xi)\, d\xi - \alpha^*\, a).$$

α^* está definida pela relação

$$1 = \int_0^1 \beta(a)\, \omega(a)\, da$$

e vale

$$\alpha^* \approx 2.78576939$$

ω_0 está definida pela relação

$$1 = \int_0^1 \omega_0\, \omega(a)\, da$$

e vale

$$\omega_0 \approx 2.969447353$$

Assim, calculamos a aproximação Galerkin descontínua u^h nos triângulos K e $\tilde{K}$, com os dados anteriores e as matrizes A resultantes do problema já referido e resultou nos seguintes erros $E_2(h) = \|u - u^h\|_{L^2}$

J/N	20	40	80	160	320	640
$E_2(h)$	0.5705	0.2855	0.1428	0.0714	0.0357	0.0179

Tabela 3.1: E_2-Erro de convergência linear.

3.2.2 Exemplo 2:

Consideramos, agora os dados $\beta(a) = e$, como função de fertilidade e $\mu(a) = \frac{1}{1-a}$, função de mortalidade bem como $u(a,0) = \omega(a)$ e $a_\dagger = 1$ funções de dado inicial. A soluç ao exacta é

$$u(a,t) = \omega(a)\, e^t = (1-a)\, e^{t-a}.$$

Assim, tal como fizemos no exemplo anterior, neste também calculamos a aproximação Galerkin descontínua u^h nos triângulos K e $\tilde{K}$, com os dados anteriores e as matrizes A resultantes do problema já referido e resultou nos seguintes erros:

J/N	20	40	80	160	320	640
$E_2(h)$	0.0469	0.0235	0.0117	0.0059	0.0029	0.0015

Tabela 3.2: E_2-Erro de convergência linear.

Devemos notar neste exemplo que a função de mortalidade é não acotada no intervalo $[0,1]$.

3.3 Conclusão

Observando os resultados do problema testado em dois exemplos, nota-se claramente que o problema apresenta regularidade, pois que, apresenta um erro de convergência, para estes resultados, é linear, ou seja, a medida que se sucedem os passos, dividindo por dois, o erro também divide-se por dois e sucessivamente, convergindo desta forma para zero.

Bibliografia

[1] Bernner, S.C., Scott, L.R., 1994, *The mathematical theory of finite element methods*. New York, Springer-Verlag.

[2] Cockburn, B., 2003, *Discontinuous Galerkin methods*, ZAMM Z. Angew. Math. Mech. 83, No. 11, pp. 731–754, /DOI 10.1002/zamm.200310088.

[3] Coyle, J. and Nigam, N., 2016, *High-order Discontinuous Galerkin Methods for a class of transport equations with structured populations*, Computers and Mathematics with Applications 72, pp. 768–784.

[4] Hesthaven, J.S., Warburton, T. 2008, *Nodal Discontinuous Galerkin Methods. Algorithms, Analysis, and Applications*, Springer.

[5] Iannelli,M., 1995, *Mathematical Theory of Age-Structured Population Dynamics*, Giardini Editori E Stampatori in Pisa.

[6] Kim, M.-Y. and Selenge, Ts., 2003, *Age-time discontinuous Galerkin method for the Lotka-McKendrick equation*, Commun. Korean Math. Soc. 18, No. 3, pp. 569–580.

[7] Medeiros, L.A., Miranda, M.M., 2000. *Espaços de Sobolev* . Universidade Federal do Rio de Janeiro. Rio de Janeiro.

Apêndice A

Programas de MATLAB

Neste apêndice se incluem os programas de Matlab utilizados. `GD1poblacion.m` é o programa principal e calcula a aproximação Galerkin Descontínua linear a partes. O programa `L2NormRectangularMesh.m` calcula o erro da norma L_2 da aproximação. Este programa e os programas auxiliares `EvalNodalBasisFns.m`, `EvalNodalBasisGrads.m`, `DunavantData1.m` são modificações de programas correspondentes de colecções de programas `femcode` que acompanha "Understanding and Implementing the Finite Element Method"by Mark S. Gockenbach (copyright SIAM 2006).

```
function [u,v] = GD1poblacionpaper(Amax,N,J,prec,usolution)
%
% A     - Edad maxima
% T     - Tiempo maximo
% J     - numero intervalos particion en la variable edad
% N     - numero intervalos particion en la variable tiempo
%             (Atencion: A/J = T/N).
% u0    - funcion dato inicial u0(a)
% mu    - funcion de mortalidad mu(a)
% beta  - funcion de fertilidad beta(a)
%
% u     - solucion Galerkin discontinua en triangulos K
% v     - solucion Galerkin discontinua en triangulos K tilde
%
da = Amax/J;
a = [0:da:Amax];
dt = da; dt6 = dt/6; dt3 = dt/3; dt2 = dt/2; da2 = da/2;
areaK3 = dt*da/6.0;
%N = T*J/A;
T = Amax*N/J;
times = [0:dt:T];
%
% Ejemplo 1
%
```

```
mu = inline('10*exp(-100*(1-a))','a');
beta = inline('20*a.*(1-a)','a');
%
% Ejemplo 2
%
%mu = inline('1./(1-a)','a');
%beta = inline('exp(1) + 0.0*a','a');
u = zeros(J,N,3);
v = zeros(J,N,3);
s = zeros(N+1,1);
usol = zeros(3,1);
vsol = zeros(3,1);
errormax=0.0;
s(1) = feval(usolution,0.0,0.0);
for n = 1:N
    %
    % Bloque 1: Calculo de la solucion Galerkin discontinua
    %           u_h en triangulos K
    for jj = 1:J
     Amu = zeros(3,3); A = zeros(3,3);
     A1 = [dt3, dt6 , 0; 0, dt6, -dt6; dt6, dt6 , dt6];
     c=[a(jj+1),times(n);a(jj+1),times(n+1);a(jj),times(n)];
        [qpts,qwts]=DunavantData(prec,c);
        % obtiene nodos y pesos de cuadratura en triangulo de
        % vertices c
        vv=EvalNodalBasisFcns(c,qpts);
        % evalua las funciones base en nodos de cuadratura
        vmu = feval(mu,qpts(:,1));
        Amu(1,1) = qwts'*(vv(:,1).*vv(:,1).*vmu);
        Amu(1,2) = qwts'*(vv(:,1).*vv(:,2).*vmu);
        Amu(2,1) = Amu(1,2);
        Amu(1,3) = qwts'*(vv(:,1).*vv(:,3).*vmu);
        Amu(3,1) = Amu(1,3);
        Amu(2,2) = qwts'*(vv(:,2).*vv(:,2).*vmu);
        Amu(2,3) = qwts'*(vv(:,2).*vv(:,3).*vmu);
        Amu(3,2) = Amu(2,3);
        Amu(3,3) = qwts'*(vv(:,3).*vv(:,3).*vmu);

        [vva,vvt]=EvalNodalBasisGrads(c,qpts);
        A(1,1) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,1));
        A(1,1) = A(1,1) + da/3;
        A(1,2) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,2));
        A(1,3) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,3));
        A(1,3) = A(1,3) + da/6;
        A(2,1) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,1));
```

```
      A(2,2) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,2));
      A(2,3) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,3));
      A(3,1) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,1));
       A(3,1) = A(3,1) + da/6;
      A(3,2) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,2));
      A(3,3) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,3));
      A(3,3) = A(3,3) + da/3;
      A = A';

      A = A1 + Amu; % Alternative: A = A + Amu
      if (n == 1)
       u03 = feval(usolution, a(jj), times(n));
       u01 = feval(usolution, a(jj+1), times(n));
       u0half=feval(usolution,(a(jj)+a(jj+1))/2,times(n));
     % b = da2*[u01; 0; u03];    % Regla de los trapecios
       b = dt6*[u01+2*u0half  ; 0 ; u03+2*u0half ]% Simpson
      else
     % b = [dt3*v(jj,n-1,1)+dt6*v(jj,n-1,2);
            dt6*v(jj,n-1,1)+dt3*v(jj,n-1,2);
            0];
           b = [dt3*v(jj,n-1,3)+dt6*v(jj,n-1,1);
           0;
           dt6*v(jj,n-1,3)+dt3*v(jj,n-1,1)]; %  exacta
      end
      usol = A\b;
      u(jj,n,:) = usol;
end
%
% Bloque 2: Calculo de la solucion Galerkin discontinua
%           v_h en triangulos K tilde
for jj = 2:J
 Amu = zeros(3,3); A = zeros(3,3);
 A1 = [dt3, 0, dt6; dt6, dt6, dt6; 0, -dt6, dt6];
 c=[a(jj),times(n+1);a(jj),times(n);a(jj+1),times(n+1)];
 [qpts,qwts]=DunavantData(prec,c);
 % obtiene nodos y pesos de cuadratura en triangulo
 % de vertices c
 vv=EvalNodalBasisFcns(c,qpts);
 % evalua las funciones base en los nodos de cuadratura
     vmu = feval(mu,qpts(:,1));
     Amu(1,1) = qwts'*(vv(:,1).*vv(:,1).*vmu);
     Amu(1,2) = qwts'*(vv(:,1).*vv(:,2).*vmu);
     Amu(2,1) = Amu(1,2);
     Amu(1,3) = qwts'*(vv(:,1).*vv(:,3).*vmu);
     Amu(3,1) = Amu(1,3);
```

```
Amu(2,2) = qwts'*(vv(:,2).*vv(:,2).*vmu);
Amu(2,3) = qwts'*(vv(:,2).*vv(:,3).*vmu);
Amu(3,2) = Amu(2,3);
Amu(3,3) = qwts'*(vv(:,3).*vv(:,3).*vmu);


[vva,vvt]=EvalNodalBasisGrads(c,qpts);
A(1,1) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,1));
A(1,1) = A(1,1) + dt/3;
A(1,2) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,2));
A(1,2) = A(1,2) + dt/6;
A(1,3) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,3));
A(2,1) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,1));
A(2,1) = A(2,1) + dt/6;
A(2,2) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,2));
A(2,2) = A(2,2) + dt/3;
A(2,3) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,3));
A(3,1) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,1));
A(3,2) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,2));
A(3,3) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,3));
A = A';

A = A1 + Amu;

b = [dt3*u(jj-1,n,2) + dt6*u(jj-1,n,1);
dt6*u(jj-1,n,2) + dt3*u(jj-1,n,1);
0];
vsol = A\b;
v(jj,n,:) = vsol;

% EL CASO jj=1 (primer triangulo K tilde anexo a la
% frontera)

Amu = zeros(3,3); A = zeros(3,3);
A1 = [dt3, 0, dt6; dt6, dt6, dt6; 0, -dt6, dt6];
c=[a(1),times(n+1);a(1),times(n);a(2),times(n+1)];
[qpts,qwts]=DunavantData(prec,c);
% obtiene nodos y pesos de cuadratura en triangulo
% de vertices c
vv=EvalNodalBasisFcns(c,qpts);
% evalua las funciones base en nodos de cuadratura
vmu = feval(mu,qpts(:,1));
vbeta = feval(beta,qpts(:,1));
Amu(1,1) = qwts'*(vv(:,1).*vv(:,1).*vmu);
Amu(1,2) = qwts'*(vv(:,1).*vv(:,2).*vmu);
```

```
        Amu(2,1) = Amu(1,2);
        Amu(1,3) = qwts'*(vv(:,1).*vv(:,3).*vmu);
        Amu(3,1) = Amu(1,3);
        Amu(2,2) = qwts'*(vv(:,2).*vv(:,2).*vmu);
        Amu(2,3) = qwts'*(vv(:,2).*vv(:,3).*vmu);
        Amu(3,2) = Amu(2,3);
        Amu(3,3) = qwts'*(vv(:,3).*vv(:,3).*vmu);

        [vva,vvt]=EvalNodalBasisGrads(c,qpts);
        A(1,1) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,1));
        A(1,1) = A(1,1) + dt/3;
        A(1,2) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,2));
        A(1,2) = A(1,2) + dt/6;
        A(1,3) = qwts'*((vva(:,1)+vvt(:,1)).*vv(:,3));
        A(2,1) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,1));
        A(2,1) = A(2,1) + dt/6;
        A(2,2) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,2));
        A(2,2) = A(2,2) + dt/3;
        A(2,3) = qwts'*((vva(:,2)+vvt(:,2)).*vv(:,3));
        A(3,1) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,1));
        A(3,2) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,2));
        A(3,3) = qwts'*((vva(:,3)+vvt(:,3)).*vv(:,3));
        A = A';

        A = A1 + Amu;
        %
        % 1-Evaluacion utilizando Simpson de u(0,t_n)
         sum = 0.0; sum1 = 0.0; sum2 = 0.0;
         for jjj = 1:J/2
             sum1 = sum1+feval(beta,a(2*jjj))*u(2*jjj-1,n,2);
         end
         for jjj = 1:J/2-1
             sum2 = sum2+feval(beta,a(2*jjj+1))*u(2*jjj,n,2);
         end
         sum = 4.0*sum1+2.0*sum2+feval(beta,Amax)*u(J,n,2);
         s(n+1) = dt*sum/(3-dt*feval(beta,0));
        b = dt/6*[2*s(n+1) + s(n); 2*s(n) + s(n+1); 0];
        vsol = A\b;
        % vsol = (A - Amodif)\bC;
        v(1,n,:) = vsol;
end

end
```

```
function m=L2NormErrRectangularMesh(fnu,U,V,A,N,J,qo)
```

```
% m=L2NormErrRectangularMesh(fnu,U,V,A,N,J,qo)
%
%   This function computes the L2 norm of the difference
%   between the function u(x,y) and the piecewise
%   polynomial function u_h(x,y) defined on a rectangular
%   mesh of triangles T and the nodal values U and V.
%   The vectors U, V define the nodal values at the nodes of
%   the mesh.
%   qo is the (optional) quadrature order; the default
%   is qo=2*d+2, where d is the degree of the mesh.
%
%   This routine is part of the MATLAB Fem code that
%   accompanies "Understanding and Implementing the Finite
%   Element Method" by Mark S. Gockenbach
%  (copyright SIAM 2006).

% d is the degree of the elements (1=linear, 2=quadratic,etc.).

da = A/J;
dt = da; Tfinal = N*dt;
a = [0:da:A];
times = [0:dt:Tfinal];

d=1;   % grado de los polinomios en cada triangulo

% id is the number of nodes per triangle.

id=round((d+2)*(d+1)/2);

% Handle optional input arguments:

if (nargin < 7)
qo=2*d+2;      % grado de exactitud de formulas de cuadratura
end

m=0;           % acumulador de integrales en cada triangulo

% Create the reference triangle and the quadrature weights
% and  nodes on it.

TR=RefTri(d);
[qpts,qwts]=DunavantData1(qo);
npts=length(qwts);
inodes=getNodes(TR,1);     % inodes = getNodes(TR,d)
```

```
% Evaluate the basis functions at the quadrature nodes:

Vals=EvalNodalBasisFcns(inodes,qpts);

% Loop over the triangles in the mesh

for n = 1:N
    for jj = 1:J

   % Get the coordinate of the vertices

    c=[a(jj+1),times(n);a(jj+1),times(n+1);a(jj),times(n)];

    % Get the quadrature nodes and weights on the triangle:

    [qpts,qwts]=DunavantData1(qo,c);


   % Record the nodal values of the piecewise polynomial:

   u=zeros(id,1);    %

   for j=1:id
      u(j)=U(n,jj,j);
   end

   Vals1=Vals*u;

   % Transform the triangle to the reference triangle:
   % (The object trans is a struct that describes the
   % transformation (matrix J, etc.).  See TransToRefTri
   % for details.)

   trans=TransToRefTri(c);

   % Compute all the quadrature nodes on T:

   z=(trans.z1*ones(1,npts)+trans.J*qpts')';

   % Compute fnu at all the nodes:

   uvals=feval(fnu,z(:,1),z(:,2));

   % Now add the contribution to the integral
```

```
m=m+trans.j*(qwts'*(Vals1-uvals).^2);

 end

 for jj = 1:J

% Get the coordinate of the vertices

 c = [a(jj),times(n+1);a(jj),times(n);a(jj+1),times(n+1)];

 % Get the quadrature nodes and weights on the triangle:

 [qpts,qwts]=DunavantData1(qo,c);

% Record the nodal values of the piecewise polynomial:

v=zeros(id,1);

for j=1:id
   u(j)=V(n,jj,j);
end

Vals1=Vals*u;        % Vals1 son los valores de u_h
                     % (solucion numerica)
                     % en los nodos de cuadratura

% Transform the triangle to the reference triangle:
% (The object trans is a struct that describes the
% transformation (matrix J, etc.).  See TransToRefTri
% for details.)

trans=TransToRefTri(c);

% Compute all the quadrature nodes on T:

z=(trans.z1*ones(1,npts)+trans.J*qpts')';

% Compute fnu at all the nodes:

uvals=feval(fnu,z(:,1),z(:,2));

% Now add the contribution to the integral

m=m+trans.j*(qwts'*(Vals1-uvals).^2);
```

```
    end
end

m=sqrt(m);
end
```

```
function [qpts,qwts]=DunavantData(prec,vert)

% [qpts,qwts]=DunavantData(prec,vert)
%
%   This function computes the quadrature points
%   qpts and the quadrature weights qwts for the
%   Dunavant quadrature rule of precision prec over
%   the triangle with vertices vert.  The array
%   vert is 3 by 2; each row contains the
%   coordinates of one of the vertices.  The
%   precision is restricted to 0<=prec<=20.
%
%   If vert is omitted, the standard reference
%   triangle with vertices (0,0), (1,0), and (0,1)
%   is assumed.

%   This routine is part of the MATLAB Fem code that
%   accompanies "Understanding and Implementing the Finite
%   Element Method" by Mark S. Gockenbach
%  (copyright SIAM 2006).

if prec>20
   error('Precision cannot be more than 20')
end

% Handle the case prec==0 (any one-point rule will do, so I
% might as well use the one-point rule with precision 1):

if prec==0
   prec=1;
end

% The rule with precision p has a total of
%
%   n(i)+n(i+1)+...+n(i+k-1)
%
% points, where i=pptr(p) and k=pptr(p+1)-pptr(p).
%
% Each n(j) is either 1, 3, or 6:
```

```
%
%     If n(j)=1, the point has barycentric coordinates
%     (x,y,z) and the corresponding weight is weights(j).
%
%     If n(j)=3, the points are (x,y,y), (y,x,y), (y,y,x)
%     and the weight for each is weights(j).
%
%     If n(j)=3, the points are (x,y,z), (x,z,y), (y,x,z),
%     (y,z,x), (z,x,y), (z,y,x) and the weight for each
%     is weights(j).
%
%     Here I write x=coord1(j), y=coord2(j), z=coord3(j).

pptr=[1;2;3;5;7;10;13;17;22;28;34;41;49;59;69;80;93;108;125;...
142];

n=[1;3;1;3;3;3;1;3;3;3;3;6;1;3;3;6;1;3;3;3;6;1;3;3;3;...
3;6;1;3;3;6;6;6;3;3;3;3;3;6;6;3;3;3;3;3;6;6;6;1;3;...
3;3;3;3;3;6;6;6;3;3;3;3;3;3;6;6;6;6;3;3;3;3;3;3;6;...
6;6;6;6;1;3;3;3;3;3;3;3;6;6;6;6;6;1;3;3;3;3;3;3;3;...
3;6;6;6;6;6;6;1;3;3;3;3;3;3;3;3;3;6;6;6;6;6;6;1;...
3;3;3;3;3;3;3;3;6;6;6;6;6;6;6;1;3;3;3;3;3;3;3;3;3;...
6;6;6;6;6;6;6;6];

weights=[
1.000000000000000;0.333333333333333;-0.562500000000000;...
0.520833333333333;0.223381589678011;0.109951743655322;...
0.225000000000000;0.132394152788506;0.125939180544827;...
0.116786275726379;0.050844906370207;0.082851075618374;...
-0.149570044467682;0.175615257433208;0.053347235608838;...
0.077113760890257;0.144315607677787;0.095091634267285;...
0.103217370534718;0.032458497623198;0.027230314174435;...
0.097135796282799;0.031334700227139;0.077827541004774;...
0.079647738927210;0.025577675658698;0.043283539377289;...
0.090817990382754;0.036725957756467;0.045321059435528;...
0.072757916845420;0.028327242531057;0.009421666963733;...
0.000927006328961;0.077149534914813;0.059322977380774;...
0.036184540503418;0.013659731002678;0.052337111962204;...
0.020707659639141;0.025731066440455;0.043692544538038;...
0.062858224217885;0.034796112930709;0.006166261051559;...
0.040371557766381;0.022356773202303;0.017316231108659;...
0.052520923400802;0.011280145209330;0.031423518362454;...
0.047072502504194;0.047363586536355;0.031167529045794;...
0.007975771465074;0.036848402728732;0.017401463303822;...
0.015521786839045;0.021883581369429;0.032788353544125;...
```

```
0.051774104507292;0.042162588736993;0.014433699669777;...
0.004923403602400;0.024665753212564;0.038571510787061;...
0.014436308113534;0.005010228838501;0.001916875642849;...
0.044249027271145;0.051186548718852;0.023687735870688;...
0.013289775690021;0.004748916608192;0.038550072599593;...
0.027215814320624;0.002182077366797;0.021505319847731;...
0.007673942631049;0.046875697427642;0.006405878578585;...
0.041710296739387;0.026891484250064;0.042132522761650;...
0.030000266842773;0.014200098925024;0.003582462351273;...
0.032773147460627;0.015298306248441;0.002386244192839;...
0.019084792755899;0.006850054546542;0.033437199290803;...
0.005093415440507;0.014670864527638;0.024350878353672;...
0.031107550868969;0.031257111218620;0.024815654339665;...
0.014056073070557;0.003194676173779;0.008119655318993;...
0.026805742283163;0.018459993210822;0.008476868534328;...
0.018292796770025;0.006665632004165;...
0.030809939937647;0.009072436679404;0.018761316939594;...
0.019441097985477;0.027753948610810;0.032256225351457;...
0.025074032616922;0.015271927971832;0.006793922022963;...
-0.002223098729920;0.006331914076406;0.027257538049138;...
0.017676785649465;0.018379484638070;0.008104732808192;...
0.007634129070725;0.000046187660794;0.032906331388919;...
0.010330731891272;0.022387247263016;0.030266125869468;...
0.030490967802198;0.024159212741641;0.016050803586801;...
0.008084580261784;0.002079352027485;0.003884876904981;...
0.025574160612022;0.008880903573338;0.016124546761731;...
0.002491941817491;0.018242840118951;0.010258563736199;...
0.003799928855302;...
0.033057055541624;0.000867019185663;0.011660052716448;...
0.022876936356421;0.030448982673938;0.030624891725355;...
0.024368057676800;0.015997432032024;0.007698301815602;...
-0.000632060497488;0.001751134301193;0.016465839189576;...
0.004839033540485;0.025840906534650;0.008471091054441;...
0.018354914106280;0.000704404677908;0.010112684927462;...
0.003537909385950];
coord1=[
0.333333333333333;0.666666666666667;0.333333333333333;...
0.600000000000000;0.108103018168070;0.816847572980459;...
0.333333333333333;0.059715871789770;0.797426985353087;...
0.501426509658179;0.873821971016996;0.053145049844817;...
0.333333333333333;0.479308067841920;0.869739794195568;...
0.048690315425316;0.333333333333333;0.081414823414554;...
0.658861384496480;0.898905543365938;0.008394777409958;...
0.333333333333333;0.020634961602525;0.125820817014127;...
0.623592928761935;0.910540973211095;0.036838412054736;...
```

```
0.333333333333333;0.028844733232685;0.781036849029926;...
0.141707219414880;0.025003534762686;0.009540815400299;...
-0.069222096541517;0.202061394068290;0.593380199137435;...
0.761298175434837;0.935270103777448;0.050178138310495;...
0.021022016536166;0.023565220452390;0.120551215411079;...
0.457579229975768;0.744847708916828;0.957365299093579;...
0.115343494534698;0.022838332222257;0.025734050548330;...
0.333333333333333;0.009903630120591;0.062566729780852;...
0.170957326397447;0.541200855914337;0.771151009607340;...
0.950377217273082;0.094853828379579;0.018100773278807;...
0.022233076674090;0.022072179275643;0.164710561319092;...
0.453044943382323;0.645588935174913;0.876400233818255;...
0.961218077502598;0.057124757403648;0.092916249356972;...
0.014646950055654;0.001268330932872;-0.013945833716486;...
0.137187291433955;0.444612710305711;0.747070217917492;...
0.858383228050628;0.962069659517853;0.133734161966621;...
0.036366677396917;-0.010174883126571;0.036843869875878;...
0.012459809331199;0.333333333333333;0.005238916103123;...
0.173061122901295;0.059082801866017;0.518892500060958;...
0.704068411554854;0.849069624685052;0.966807194753950;...
0.103575692245252;0.020083411655416;-0.004341002614139;...
0.041941786468010;0.014317320230681;0.333333333333333;...
0.005658918886452;0.035647354750751;0.099520061958437;...
0.199467521245206;0.495717464058095;0.675905990683077;...
0.848248235478508;0.968690546064356;0.010186928826919;...
0.135440871671036;0.054423924290583;0.012868560833637;...
0.067165782413524;0.014663182224828;...
0.333333333333333;0.013310382738157;0.061578811516086;...
0.127437208225989;0.210307658653168;0.500410862393686;...
0.677135612512315;0.846803545029257;0.951495121293100;...
0.913707265566071;0.008430536202420;0.131186551737188;...
0.050203151565675;0.066329263810916;0.011996194566236;...
0.014858100590125;-0.035222015287949;0.333333333333333;...
0.020780025853987;0.090926214604215;0.197166638701138;...
0.488896691193805;0.645844115695741;0.779877893544096;...
0.888942751496321;0.974756272445543;0.003611417848412;...
0.134466754530780;0.014446025776115;0.046933578838178;...
0.002861120350567;0.223861424097916;0.034647074816760;...
0.010161119296270;0.333333333333333;-0.001900928704400;...
0.023574084130543;0.089726636099435;0.196007481363421;...
0.488214180481157;0.647023488009788;0.791658289326483;...
0.893862072318140;0.916762569607942;0.976836157186356;...
0.048741583664839;0.006314115948605;0.134316520547348;...
0.013973893962392;0.075549132909764;-0.008363153208227;...
0.026686063258714;0.010547719294141];
```

```
coord2=[
0.333333333333333;0.166666666666667;0.333333333333333;...
0.200000000000000;0.445948490915965;0.091576213509771;...
0.333333333333333;0.470142064105115;0.101286507323456;...
0.249286745170910;0.063089014491502;0.310352451033784;...
0.333333333333333;0.260345966079040;0.065130102902216;...
0.312865496004874;0.333333333333333;0.459292588292723;...
0.170569307751760;0.050547228317031;0.263112829634638;...
0.333333333333333;0.489682519198738;0.437089591492937;...
0.188203535619033;0.044729513394453;0.221962989160766;...
0.333333333333333;0.485577633383657;0.109481575485037;...
0.307939838764121;0.246672560639903;0.066803251012200;...
0.534611048270758;0.398969302965855;0.203309900431282;...
0.119350912282581;0.032364948111276;0.356620648261293;...
0.171488980304042;0.488217389773805;0.439724392294460;...
0.271210385012116;0.127576145541586;0.021317350453210;...
0.275713269685514;0.281325580989940;0.116251915907597;...
0.333333333333333;0.495048184939705;0.468716635109574;...
0.414521336801277;0.229399572042831;0.114424495196330;...
0.024811391363459;0.268794997058761;0.291730066734288;...
0.126357385491669;0.488963910362179;0.417644719340454;...
0.273477528308839;0.177205532412543;0.061799883090873;...
0.019390961248701;0.172266687821356;0.336861459796345;...
0.298372882136258;0.118974497696957;0.506972916858243;...
0.431406354283023;0.277693644847144;0.126464891041254;...
0.070808385974686;0.018965170241073;0.261311371140087;...
0.388046767090269;0.285712220049916;0.215599664072284;...
0.103575616576386;0.333333333333333;0.497380541948438;...
0.413469438549352;0.470458599066991;0.240553749969521;...
0.147965794222573;0.075465187657474;0.016596402623025;...
0.296555596579887;0.337723063403079;0.204748281642812;...
0.189358492130623;0.085283615682657;0.333333333333333;...
0.497170540556774;0.482176322624625;0.450239969020782;...
0.400266239377397;0.252141267970953;0.162047004658461;...
0.075875882260746;0.015654726967822;0.334319867363658;...
0.292221537796944;0.319574885423190;0.190704224192292;...
0.180483211648746;0.080711313679564;...
0.333333333333333;0.493344808630921;0.469210594241957;...
0.436281395887006;0.394846170673416;0.249794568803157;...
0.161432193743843;0.076598227485371;0.024252439353450;...
0.043146367216965;0.358911494940944;0.294402476751957;...
0.325017801641814;0.184737559666046;0.218796800013321;...
0.101179597136408;0.020874755282586;0.333333333333333;...
0.489609987073006;0.454536892697893;0.401416680649431;...
0.255551654403098;0.177077942152130;0.110061053227952;...
```

```
0.055528624251840;0.012621863777229;0.395754787356943;...
0.307929983880436;0.264566948406520;0.358539352205951;...
0.157807405968595;0.075050596975911;0.142421601113383;...
0.065494628082938;0.333333333333333;0.500950464352200;...
0.488212957934729;0.455136681950283;0.401996259318289;...
0.255892909759421;0.176488255995106;0.104170855337758;...
0.053068963840930;0.041618715196029;0.011581921406822;...
0.344855770229001;0.377843269594854;0.306635479062357;...
0.249419362774742;0.212775724802802;0.146965436053239;...
0.137726978828923;0.059696109149007];
coord3=[
0.333333333333333;0.166666666666667;0.333333333333333;...
0.200000000000000;0.445948490915965;0.091576213509771;...
0.333333333333333;0.470142064105115;0.101286507323456;...
0.249286745170910;0.063089014491502;0.636502499121399;...
0.333333333333333;0.260345966079040;0.065130102902216;...
0.638444188569810;0.333333333333333;0.459292588292723;...
0.170569307751760;0.050547228317031;0.728492392955404;...
0.333333333333333;0.489682519198738;0.437089591492937;...
0.188203535619033;0.044729513394453;0.741198598784498;...
0.333333333333333;0.485577633383657;0.109481575485037;...
0.550352941820999;0.728323904597411;0.923655933587500;...
0.534611048270758;0.398969302965855;0.203309900431282;...
0.119350912282581;0.032364948111276;0.593201213428213;...
0.807489003159792;0.488217389773805;0.439724392294460;...
0.271210385012116;0.127576145541586;0.021317350453210;...
0.608943235779788;0.695836086787803;0.858014033544073;...
0.333333333333333;0.495048184939705;0.468716635109574;...
0.414521336801277;0.229399572042831;0.114424495196330;...
0.024811391363459;0.636351174561660;0.690169159986905;...
0.851409537834241;0.488963910362179;0.417644719340454;...
0.273477528308839;0.177205532412543;0.061799883090873;...
0.019390961248701;0.770608554774996;0.570222290846683;...
0.686980167808088;0.879757171370171;0.506972916858243;...
0.431406354283023;0.277693644847144;0.126464891041254;...
0.070808385974686;0.018965170241073;0.604954466893291;...
0.575586555512814;0.724462663076655;0.747556466051838;...
0.883964574092416;0.333333333333333;0.497380541948438;...
0.413469438549352;0.470458599066991;0.240553749969521;...
0.147965794222573;0.075465187657474;0.016596402623025;...
0.599868711174861;0.642193524941505;0.799592720971327;...
0.768699721401368;0.900399064086661;0.333333333333333;...
0.497170540556774;0.482176322624625;0.450239969020782;...
0.400266239377397;0.252141267970953;0.162047004658461;...
0.075875882260746;0.015654726967822;0.655493203809423;...
```

```
0.572337590532020;0.626001190286228;0.796427214974071;...
0.752351005937729;0.904625504095608;...
0.333333333333333;0.493344808630921;0.469210594241957;...
0.436281395887006;0.394846170673416;0.249794568803157;...
0.161432193743843;0.076598227485371;0.024252439353450;...
0.043146367216965;0.632657968856636;0.574410971510855;...
0.624779046792512;0.748933176523037;0.769207005420443;...
0.883962302273467;1.014347260005363;0.333333333333333;...
0.489609987073006;0.454536892697893;0.401416680649431;...
0.255551654403098;0.177077942152130;0.110061053227952;...
0.055528624251840;0.012621863777229;0.600633794794645;...
0.557603261588784;0.720987025817365;0.594527068955871;...
0.839331473680839;0.701087978926173;0.822931324069857;...
0.924344252620784;0.333333333333333;0.500950464352200;...
0.488212957934729;0.455136681950283;0.401996259318289;...
0.255892909759421;0.176488255995106;0.104170855336758;...
0.053068963840930;0.041618715196029;0.011581921406822;...
0.606402646106160;0.615842614456541;0.559048000390295;...
0.736606743262866;0.711675142287434;0.861402717154987;...
0.835586957912363;0.929756171556853];

% Get the pointer into the weights and coords arrays:

i=pptr(prec);

% Get the number of entries to examine:

if prec<20
   k=pptr(prec+1)-i;
else
   k=length(weights)-i+1;
end

% Calculate the number of quadrature points:

N=sum(n(i:i+k-1));
qwts=zeros(N,1);
bcoords=zeros(N,3);

% Now get the points:

m=0;
for j=i:i+k-1

   x=coord1(j);
```

```
    y=coord2(j);
    z=coord3(j);

    if n(j)==1
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[x,y,z];
    elseif n(j)==3
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[x,y,y];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[y,x,y];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[y,y,x];
    elseif n(j)==6
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[x,y,z];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[x,z,y];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[y,x,z];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[y,z,x];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[z,x,y];
       m=m+1;
       qwts(m)=weights(j);
       bcoords(m,:)=[z,y,x];
    else
       error(['Error in n(',int2str(j),'); must be 1,3 or 6'])
    end
end

% Now convert to global coordinates:

if nargin<2
    vert=[0 0
```

```
          1 0
          0 1];
end
qpts=zeros(N,2);
for j=1:N
   qpts(j,:)=(vert'*(bcoords(j,:)'))';
end

if nargin==2

   % Compute the area of the triangle:

   x13=vert(1,1)-vert(3,1);
   x23=vert(2,1)-vert(3,1);
   y13=vert(1,2)-vert(3,2);
   y23=vert(2,2)-vert(3,2);
   J=0.5*abs(x13*y23-y13*x23);

else

   % Reference triangle has area 1/2:

   J=0.5;

end

% Adjust the weights

qwts=J*qwts;
```

```
function v=EvalNodalBasisFcns(inodes,enodes)

% v=EvalNodalBasisFcns(inodes,enodes)
%
%  This function evaluates the standard nodal basis functions
%  defined on a triangle with interpolation nodes inodes
%  at the evaluation nodes enodes.  The id by 2 array inodes
%  must store the coordinates of the id interpolation nodes.
%
%  The resulting matrix v is n by id; each column contains
%  the values of one of the basis functions at the n
%  evaluation nodes.

%   This routine is part of the MATLAB Fem code that
%   accompanies "Understanding and Implementing the Finite
```

```
%    Element Method" by Mark S. Gockenbach
%    (copyright SIAM 2006).

% Determine id and d (id=(d+1)*(d+2)/2)):

id=size(inodes,1);
d=round((sqrt(8*id+1)-3)/2);
if (d+2)*(d+1)/2~=id
    error('The number of nodes must be (d+2)(d+1)/2 for ...
           some integer d')
end

% Form the vandermonde-like matrix defined by the
% interpolation nodes:

N=zeros(id);

x=inodes(:,1)';
y=inodes(:,2)';
l=0;
for i=0:d
    for j=0:i
        l=l+1;
        N(l,:)=x.^(i-j).*y.^j;
    end
end

% Form the vandermonde-like matrix containing the evaluation
% nodes:

n=size(enodes,1);
D=zeros(id,n);

x=enodes(:,1)';
y=enodes(:,2)';
l=0;
for i=0:d
    for j=0:i
        l=l+1;
        D(l,:)=x.^(i-j).*y.^j;
    end
end

% Solve the system giving the values:
```

```
v=(N\D)';
end
```

```
function [O1,O2,O3]=EvalNodalBasisGrads(inodes,enodes,w)

% [Vx,Vy]=EvalNodalBasisGrads(inodes,enodes)
%          or
% [Grads,Vx,Vy]=EvalNodalBasisGrads(inodes,enodes,w)
%
%   This function evaluates the partial derivatives of the
%   standard nodal basis functions defined on a triangle
%   with interpolation nodes inodes at the evaluation nodes
%   enodes.  The id by 2 array inodes must store the
%   coordinates of the id interpolation nodes.
%
%   Upon exit, the n by id arrays Vx and Vy contain the
%   partial derivatives with respect to x and y, respectively
%   (as columns) of the id functions at the n points.
%
%   If the optional argument w is included, the gradients of
%   the linear combination of the basis functions (with
%   the weights in the linear combination coming from w) are
%   computed instead.  In this case, the output is a 2 by n
%   array.  Vx and Vy are then returned, if requested, as the
%   second and third arguments.

%   This routine is part of the MATLAB Fem code that
%   accompanies "Understanding and Implementing the Finite
%   Element Method" by Mark S. Gockenbach
%   (copyright SIAM 2006).

% Determine id and d (id=(d+1)*(d+2)/2):

id=size(inodes,1);
d=round((sqrt(8*id+1)-3)/2);
if (d+2)*(d+1)/2~=id
   error('The number of nodes must be (d+2)(d+1)/2 for ...
          some integer d')
end

% Form the vandermonde-like matrix defined by the
% interpolation nodes:

N=zeros(id);
```

```
x=inodes(:,1)';
y=inodes(:,2)';
l=0;
for i=0:d
    for j=0:i
        l=l+1;
        N(l,:)=x.^(i-j).*y.^j;
    end
end

% Form the vandermonde-like matrix defined by the evaluation
% nodes (for partial derivative with respect to x):

n=size(enodes,1);
Cx=zeros(id,n);

x=enodes(:,1)';
y=enodes(:,2)';
l=0;
for i=0:d
    for j=0:i
        l=l+1;
        if (i>0 & j<i)
            Cx(l,:)=(i-j)*x.^(i-j-1).*y.^j;
        end
    end
end

% Form the vandermonde-like matrix defined by the evaluation
% nodes (for partial derivative with respect to y):

Cy=zeros(id,n);

x=enodes(:,1)';
y=enodes(:,2)';
l=0;
for i=0:d
    for j=0:i
        l=l+1;
        if j>0
            Cy(l,:)=j*x.^(i-j).*y.^(j-1);
        end
    end
end
```

```
% Solve the system giving the values:

T=N\[Cx,Cy];
O1=T(:,1:n)';
O2=T(:,n+1:2*n)';

if nargin==3
    Grads=[(O1*w)';(O2*w)'];
    if nargout>1
        O3=O2;
        O2=O1;
    end
    O1=Grads;
end
```

```
function u = usolution(a,t)
%
omega0 = 2.969447356;
alpha = 2.78576939;
c1 = exp(-100)/10;
u=omega0*exp(alpha*t).*exp(-alpha*a).*...
  exp(-c1*(1-exp(100*a)));
end
```

```
function u = usolution2(a,t)
%
u = (1-a).*exp(t).*exp(-a);
end
```