Universidade Estadual de Campinas
Instituto de Matemática, Estatística e
Computação Científica
Departamento de Matemática Aplicada

Tese de Doutorado

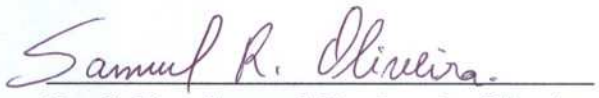
Tratamento das Equações de Einstein-Yang-Mills para Soluções Numéricas com Simetria Esférica Auto-Gravitante e Simetria Axial no Espaço-Tempo de Minkowski

Autor: Luis Alberto D'Afonseca
Orientador: Samuel Rocha de Oliveira

Tratamento das Equações de Einstein-Yang-Mills para Soluções Numéricas com Simetria Esférica Auto-Gravitante e Simetria Axial no Espaço-Tempo de Minkowski

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por **Luis Alberto D'Afonseca** e aprovada pela comissão julgadora.

Campinas, 28 de agosto 2007



Prof. Dr. Samuel Rocha de Oliveira
Orientador

Banca Examinadora:

1. Prof. Dr. Marcelo Evangelista Araújo
2. Prof. Dr. Marcelo Moraes Guzzo
3. Prof. Dr. Petrônio Pulino
4. Prof. Dr. Ricardo Biloti
5. Prof. Dr. Samuel Rocha de Oliveira

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, Como requisito parcial para a obtenção do Título de DOUTOR em Matemática Aplicada.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Júlia Milani Rodrigues

D´Afonseca, Luis Alberto

D131t Tratamento das equações de Einstein-Yang-Mills para soluções numéricas com simetria esférica auto-gravitante e simetria axial no espaço-tempo de Minkowski / Luis Alberto D`Afonseca -- Campinas, [S.P. :s.n.], 2007.

Orientador : Samuel Rocha de Oliveira

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Einstein, Equações de. 2. Campos de calibre (Física). 3. Equações – Soluções numéricas. I. Oliveira, Samuel Rocha de. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Título em inglês: Set up of Einstein-Yang-Mills equations for numerical solutions of self-gravitating spherical symmetric fields and axial simmetric fields on Minkowski space-time.

Palavras-chave em inglês (Keywords): 1. Einstein´s field equations. 2. Gauge fields (Physics). 3. Equations – Numerical solutions.

Área de concentração: Física-Matemática

Titulação: Doutor em Matemática Aplicada

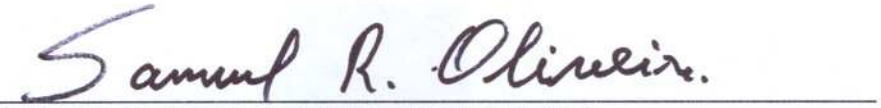
Banca examinadora: Prof. Dr. Ricardo Biloti (IMECC-UNICAMP)
Prof. Dr. Marcelo Evangelista Araújo (UFRJ)
Prof. Dr. Marcelo Moraes Guzzo (IFGW-UNICAMP)
Prof. Dr. Petrônio Pulino (IMECC-UNICAMP)
Prof. Dr. Samuel Rocha de Oliveira (IMECC-UNICAMP)

Data da defesa: 28-08-2007

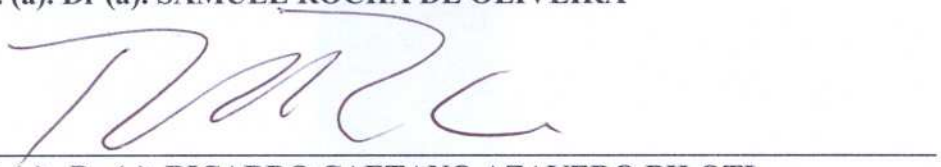
Programa de pós-graduação: Doutorado em Matemática Aplicada

Tese de Doutorado defendida em 28 de agosto de 2007 e aprovada

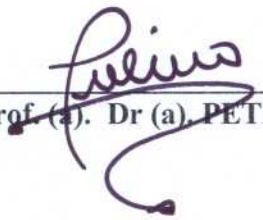
Pela Banca Examinadora composta pelos Profs. Drs.



Prof. (a). Dr (a). SAMUEL ROCHA DE OLIVEIRA

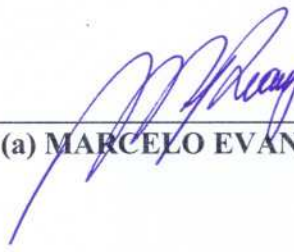


Prof. (a). Dr (a). RICARDO CAETANO AZAVEDO BILOTI



Prof. (a). Dr (a). PETRÔNIO PULINO

Prof. (a). Dr (a). MARCELO MORAES GUZZO



Prof. (a) Dr. (a) MARCELO EVANGELISTA ARAÚJO

Dedico essa tese a

Valéria Mattos da Rosa

Ela também esteve perdida
no mesmo labirinto.

Esse trabalho foi financeiramente apoiado pela FAPESP
(Fundação de Amparo a Pesquisa do Estado de São Paulo).

Resumo

Nesse trabalho delineamos a teoria clássica para o campo de Einstein-Yang-Mills e elaboramos um conjunto particular de equações para obtermos soluções numéricas. Estudamos dois casos com simetria espaço-temporal: Simetria esférica com campo auto-gravitante e simetria axial no espaço-tempo de Minkowski.

Utilizamos métodos numéricos das linhas para fazer a evolução temporal dos campos discretizados. No caso com simetria esférica, os campos são discretizados por diferenças finitas e no caso da simetria axial comparamos as discretizações por métodos Pseudo-Espectrais e por diferenças finitas. Para evolução temporal um método auto-adaptativo de Runge-Kutta é empregado.

Na simulação dos campos de Yang-Mills auto-gravitantes com simetria esférica mostramos a evolução da implosão e explosão de uma casca energética sem formação de buraco negro nem de corpo estável.

No caso com simetria axial além da implosão e explosão de pulsos de cores diferentes dos campos de Yang-Mills, geramos também várias soluções dinâmicas em que vemos o transiente do intercâmbio de energia entre essas cores.

Sumary

In this work we outline the classic theory of Einstein-Yang-Mills fields and work out a set of particular equations suited for numerical simulations. We consider two special cases with space-time symmetries: self-gravitating spherical symmetric and axially symmetric field on a Minkowski space-time.

We use the numerical method of lines for time evolution of discretized fields. On the spherical symmetric case, the fields are discretized by finite differences and on the axial symmetric case we compare the field discretization by the pseudo-spectral method and finite differences method. For time stepping we use a self-adaptive Runge-Kutta method.

In the simulation of Yang-Mills self-gravitating fields with spherical symmetry we show the evolution of implosion and explosion of a energetic shell without black hole or stable body formation.

In the axial symmetric case besides implosion and explosion of pulses of different colours of Yang-Mills fields, we also generate several dynamic solutions that display the transient of the energy exchange among these colours.

Sumário

Notações

Ao longo desse trabalho vamos considerar diferentes espaços e cada um deles possui uma estrutura interna própria. Para evitar ambigüidades, vamos empregar as seguintes notações para os índices das componentes dos elementos de cada um dos espaços:

- $\hat{a}, \hat{b}, \hat{c}$ índices latinos com chapéu indicam as componentes do campo de calibre e assumem os valores 1,2,3;
- α, μ, ν índices gregos indicam as coordenadas do espaço-tempo quadri-dimensional e assumem os valores 0,1,2,3;
- i, j, k quando discutirmos campos com simetria esférica, índices latinos indicam as três coordenadas espaciais, (r, θ, φ);

  na discussão sobre campos axialmente simétricos estes índices correspondem as coordenadas não simétricas, (t, r, z);
- A, B, C índices latinos maiúsculos são usados na decomposição (2+1)+1 para indicar as coordenadas espaciais não simétricas.

Além de distingüir entre as componentes dos elementos dos diversos espaços temos também que distinguir entre os diferentes conceitos de derivação. Para isso usaremos o seguinte conjunto de notações para as derivadas covariantes:

- ∇_μ derivada covariante no espaço-tempo quadridimensional, definida pela métrica $g_{\mu\nu}$;
- ${}^{(3)}\nabla_i$ derivada covariante no espaço-tempo tridimensional, definida pela projeção tridimensional da métrica, γ_{ij};
- $\mathcal{D}_\mu$ derivada covariante do espaço-tempo quadridimensional e do campo de calibre. Essa derivada é definida como

  $$\mathcal{D}_\mu \phi^{\hat{a}} = \nabla_\mu \phi^{\hat{a}} + \varepsilon^{\hat{a}\hat{b}\hat{c}} A^{\hat{b}}{}_\mu \phi^{\hat{c}}$$
- ${}^{(3)}\mathcal{D}_i$ derivada covariante do espaço-tempo tridimensional e do campo de calibre. Essa derivada é definida como

  $${}^{(3)}\mathcal{D}_i = {}^{(3)}\nabla_i \phi^{\hat{a}} + \varepsilon^{\hat{a}\hat{b}\hat{c}} A^{\hat{b}}{}_i \phi^{\hat{c}}$$

1 Introdução

As equações de Einstein e as de Maxwell descrevem com muito sucesso as interações clássicas gravitacional e eletromagnética, respectivamente. A relatividade geral usa a geometria Lorentziana em uma variedade de quatro dimensões, o espaço-tempo, e o eletromagnetismo usa campos definidos em um espaço vetorial (tangente ou co-tangente ao espaço-tempo) com uma álgebra de Lie que representa o grupo de simetria "interna", a chamada invariância de calibre.

O termo invariância de calibre remonta a 1919 em uma tentativa de unificar gravitação e eletromagnetismo desenvolvida por Weyl [21], no entanto o conceito já estava em uso desde 1820, quando o eletromagnetismo foi descoberto e a primeira teoria eletrodinâmica foi proposta [15].

No eletromagnetismo de Maxwell podemos adicionar as derivadas parciais de uma função escalar ao campo de calibre sem alterar suas características físicas. Esta invariância é representada por um grupo isomórfico às rotações em duas dimensões em cada ponto do espaço-tempo, ou seja, esta é uma invariância de calibre local. Representamos este grupo com valores em uma álgebra de Lie de U(1).

Ambas as forças gravitacionais e eletromagnéticas são de longo alcance sendo que a última é descrita pelo grupo U(1). Para descrever as forças nucleares de curto alcance os físicos teóricos pensaram em usar um grupo maior e não abeliano.

Em 1954, Yang e Mills [24] generalizaram a idéia para um grupo de rotações tridimensionais. Os campos são representados por funções em uma álgebra de Lie de SU(2).

Por um lado os campos de Yang-Mills foram bem sucedidos nas descrições quânticas de alguns fenômenos das interações fracas e fortes com o eletromagnetismo, por outro lado as interações de longo alcance destes mesmos campos com a matéria não são compreendidos em sua totalidade até hoje.

Do ponto de vista puramente clássico, em contraste ao quântico, as equações diferenciais parciais da relatividade geral e dos campos de calibre de Yang-Mills apresentam uma complexidade fascinante.

Os campos de Maxwell e de Yang-Mills são dispersivos. Em particular não existem soluções não triviais estáticas e globalmente regulares para nenhum deles. Em contraste, a gravidade é intrinsecamente atrativa. Tanto o é, que existem teoremas com hipóteses razoáveis sobre o conteúdo de matéria, garantindo a inevitabilidade de singularidades na relatividade geral de Einstein.

Assim, pode-se esperar soluções localizadas destes campos se considerarmos os campos de Maxwell e de Yang-Mills com auto-gravitação. Sabemos que não há corpos gravitacionais formados apenas por campos eletromagnéticos. Mas nos anos 1990s foi demonstrada [17] a existência de soluções globais, suaves e estáticas das equacões de Einstein-Yang-Mills. Em outras palavras, a interação não linear repulsiva dos campos de Yang-Mills pode ser suficiente para balancear a sua auto-gravitação atrativa. Em 2000 [25] uma solução oscilatória também foi encontrada.

Mas um dos resultados mais intrigantes foi apresentado nos anos 1980s: um buraco colorido, isto é, uma solução com um horizonte de eventos (típico de buracos negros) que contém campos de Yang-Mills não triviais do lado de fora do horizonte. Esta solução estática que Bartnik e McKinnon [3] apresentaram viola a famosa conjectura de que *buracos negros não tem cabelos*. Em 1990 resultado similar foi obtido numericamente [4].

Outro fenômeno intrigante de colapso gravitacional foi descoberto numericamente por Choptuik et. al. [7, 8]. Eles estudaram os fenômenos críticos na formação de buracos negros a partir de campos de Yang-Mills auto-gravitantes, reproduzindo resultados anteriormente obtidos para campos escalares. Entretanto esses trabalhos consideram apenas o campo de Einstein-Yang-Mills com simetria esférica.

Apresentamos nesse trabalho um pouco da teoria clássica das equações de Einstein-Yang-Mills. Discutimos sobre a imposição de simetrias espaço-temporais nos campos de Yang-Mills e montamos um sistema de equações que pode ser utilizado em investigações numéricas.

Além disto, apresentaremos simulações numéricas para alguns casos particulares, a saber, casos com simetria esférica e auto-gravitação e casos com simetria axial sem gravitação.

Até o momento não temos conhecimento de soluções dinâmicas para o campo de Yang-Mills clássico com simetria axial. Todas as soluções encontradas na literatura se referem

a campos com simetria esférica, cilíndrica ou estacionários.

No Capítulo 2 apresentamos a teoria do campo de Yang-Mills clássico auto gravitante. Chamamos essa teoria de campo de Einstein-Yang-Mills. Não consideramos, entretanto, a existência de um campo de Higgs. Em seguida introduzimos o conceito de simetria espaço-temporal em um campo de calibre, resultado que será posteriormente utilizado para determinar as equações para o campo de Einstein-Yang-Mills axialmente simétrico. Esse capítulo apresenta também um teorema de existência de soluções, algumas escolhas de calibre possíveis e algumas quantidades conservadas.

O Capítulo 3 exibe as equações para o campo de Einstein-Yang-Mills com simetria esférica. Essas equações são construídas a partir do ansatz de Witten, que é o modelo para todas as soluções esfericamente simétricas das equações de Einstein-Yang-Mills. Vários resultados importantes foram obtidos com o estudo das soluções para essas equações. Casos particulares de interesse são as soluções de Bartnik e McKinnon e os fenômenos críticos no colapso gravitacional. Para fins de comparações, fazemos uma evolução de colapso gravitacional de uma casca esférica de Yang-Mills com duas condições iniciais. Observamos a implosão seguida de explosão sem formação da solução localizada.

Na seqüência, o Capítulo 4 constrói as equações de Einstein-Yang-Mills com simetria axial. Nesse capítulo usamos os resultados apresentados no Capítulo 2 para determinar qual deve ser o formato das soluções axialmente simétricas e exibimos as equações de movimento para essas soluções. Depois particularizamos essas equações para o caso de um espaço-tempo plano, obtendo as equações que foram resolvidas numericamente.

Para obter as soluções numéricas utilizamos métodos pseudo-espectrais e diferenças finitas. Esses métodos estão descritos no Capítulo 5. Além da descrição dos métodos apresentamos uma discussão sobre a convergência dos métodos espectrais, considerada exponencial e uma comparação entre os dois métodos. Essa comparação levou a um resultado não esperado. Embora teoricamente o método espectral tenha vantagens significativas e em alguns casos essas vantagens são visíveis nos resultados numéricos, no caso das equações de Yang-Mills usadas nesse trabalho o método de diferenças finitas obteve melhores resultados.

Tendo descrito a formulação teórica dos campos de Yang-Mills e os métodos numéricos empregados, passamos a apresentação dos resultados obtidos. O Capítulo 6 introduz os resultados, detalhando a forma como eles serão apresentados e explicitando as condições iniciais utilizadas. Em seguida o Capítulo 7 apresenta as várias simulações para o caso particular de soluções Abelianas. Esses resultados têm importância apenas como base de

comparações para as simulações completas, pois fisicamente o caso Abeliano equivale ao eletromagnetismo clássico. Finalmente no Capítulo 8, apresentamos as soluções obtidas com as equações de Yang-Mills completas, onde podemos ver o comportamento não linear e as interações entre as componentes internas do campo.

Por completude, apresentamos alguns resultados e informações em apêndices. No Apêndice A apresentamos um resumo dos resultados da teoria de grupos que foram utilizados ao longo do texto. No Apêndice B incluímos as listagens dos programas em Matlab que foram utilizados para obter os resultados apresentados nesse trabalho.

2 A Teoria de Einstein-Yang-Mills

Nesse capítulo vamos apresentar a teoria geral para o campo de Yang-Mills utilizada nesse trabalho. Nosso objetivo é estudar a dinâmica da interação dos campos clássicos de Yang-Mills. Vamos fazer uma revisão da teoria para um campo de calibre auto gravitante. Além disso vamos também discutir a questão da imposição de uma simetria espaço-temporal a um campo de calibre.

2.1 O Campo de Einstein-Yang-Mills

Como não é nossa intenção explorar os aspectos quânticos desses campos, nos restringimos ao campo de Yang-Mills com simetria interna definida pelo grupo SU(2) e não consideramos os campos de Higgs.

Esses campos são descritos matematicamente pelos elementos $(\mathcal{M}, g_{\mu\nu}, A)$, onde $\mathcal{M}$ é uma variedade quadridimensional, com uma geometria Lorentzina definida pela métrica $g_{\mu\nu}$ da Relatividade Geral e A é a conexão do campo. Esta conexão é avaliada na álgebra de Lie do grupo SU(2), ou seja

$$A = A_\mu \, dx^\mu = \tau^{\hat{a}} \, A^{\hat{a}}{}_\mu \, dx^\mu$$

onde $\tau^{\hat{a}}$ são os geradores do grupo SU(2). Esse grupo pode ser representado pelas matrizes bidimensionais unitárias com determinante igual a um. Os geradores de SU(2) que usaremos são, $\tau^{\hat{a}} = \sigma^{\hat{a}}/2i$, onde $\sigma^{\hat{a}}$ são as matrizes de Pauli. No Apêndice A incluímos um resumo dos resultados da teoria de grupos que utilizamos.

Usaremos a notação de componentes para representar os elementos do campo, isto é, usaremos $A^{\hat{a}}_\mu$ denotando a conexão A, exceto quando fizermos transformações de calibre, Neste sentido os índices latinos com chapéu, $\hat{a} = 1, 2, 3$, indicam a componente interna do campo, para uma dada base de geradores. Ao longo do texto vamos usar uma liberdade

de notação para chamar essas componentes de cores do campo. Por outro lado os índices gregos indicam as componentes dos campos em um sistema de coordenadas do espaço-tempo, enquanto que, índices repetidos implicam em somas nos limites adequados tanto para os índices do espaço-tempo quanto para os índices do campo.

2.2 Formulação Lagrangeana

A integral de ação convencional para a teoria de Einstein-Yang-Mills é

$$\int \sqrt{-g}\left(\frac{R}{16\pi G} - \frac{1}{4\gamma^2}F^{\hat{a}}{}_{\mu\nu}F^{\hat{a}\mu\nu}\right)dx^4 \tag{2.1}$$

onde γ é a constante de acoplamento do campo, G é a constante gravitacional, R é o escalar de Ricci, g é o determinante da métrica do espaço-tempo e F é o tensor de força do campo de Yang-Mills que será definido a seguir. Nesse trabalho vamos considerar unidades relativísticas, ou seja vamos definir as constantes G, γ e c, velocidade da luz, como unitárias.

De modo semelhante ao campo eletromagnético, o tensor de força F dos campos de Yang-Mills

$$F = F_{\mu\nu}\,dx^\mu dx^\nu = F^{\hat{a}}{}_{\mu\nu}\,\tau^{\hat{a}}\,dx^\mu dx^\nu$$

é definido em termos das componentes da seguinte forma

$$F^{\hat{a}}{}_{\mu\nu} = \partial_\mu A^{\hat{a}}{}_\nu - \partial_\nu A^{\hat{a}}{}_\mu + \varepsilon^{\hat{a}\hat{b}\hat{c}}A^{\hat{b}}{}_\mu A^{\hat{c}}{}_\nu$$

onde $\varepsilon^{\hat{a}\hat{b}\hat{c}}$ é o símbolo de permutação totalmente anti-simétrico com $\varepsilon^{\hat{1}\hat{2}\hat{3}} = 1$.

Nessa teoria, ao definirmos a derivada covariante temos que levar em conta as transformações espaço-temporais e as transformações de calibre. Portanto, a derivada covariante de um campo arbitrário, $\phi^{\hat{a}}$, deve ser definida como

$$\mathcal{D}_\mu\phi^{\hat{a}} = \nabla_\mu\phi^{\hat{a}} + \varepsilon^{\hat{a}\hat{b}\hat{c}}A^{\hat{b}}{}_\mu\phi^{\hat{c}}$$

aqui ∇_μ é a derivada covariante compatível com a geometria espaço-tempo. Definida dessa forma a derivada é invariante sob transformações de calibre e de coordenadas em geral.

A ação do campo de Einstein-Yang-Mills (2.1) é invariante com respeito a transformações do campo de calibre A, isto é,

$$A_\mu \to \tilde{A}_\mu = uA_\mu u^{-1} + \Big(\partial_\mu u\Big)u^{-1} \Rightarrow F_{\mu\nu} \to \tilde{F}_{\mu\nu} = u\,F_{\mu\nu}\,u^{-1}$$

onde u é um elemento do grupo de calibre, SU(2). Além disso os termos correspondentes ao campo de Yang-Mills também possuem simetria conformal, isso é, para qualquer $\Omega(x) > 0$ transformamos a métrica como

$$g_{\mu\nu} \to \tilde{g}_{\mu\nu} = \Omega(x) g_{\mu\nu}$$

as componentes do campo permanecem invariantes.

Aplicando o princípio variacional na integral (2.1) com respeito a métrica $g_{\mu\nu}$, obtemos as equações de Einstein

$$R_{\mu\nu} - \frac{1}{2} R\, g_{\mu\nu} = 16\pi T_{\mu\nu} \tag{2.2}$$

sendo que o tensor de energia-momento assume a mesma forma que o tensor de energia-momento de um campo eletromagnético

$$T_{\mu\nu} = \frac{1}{2} g_{\mu\nu} F^{\hat{a}}{}_{\sigma\delta} F^{\hat{a}\,\sigma\delta} - 2 F^{\hat{a}}{}_{\mu\lambda} F^{\hat{a}}{}_{\nu}{}^{\lambda} \tag{2.3}$$

A diferença entre esse tensor e o tensor eletromagnético está inserida na definição do tensor de força. Uma propriedade essencial desse tensor é que ele é invariante sob transformações de calibre. Além disso devido a invariância conforme da Lagrangeana este tensor possui traço nulo, $T^{\mu}{}_{\mu} = 0$.

Agora se variarmos a ação (2.1) com relação a $A^{\hat{a}}{}_{\mu}$, obtemos as equações de movimento para o campo de Yang-Mills na métrica $g_{\mu\nu}$

$$\mathcal{D}_{\mu} F^{\hat{a}\,\mu\nu} = \nabla_{\mu} F^{\hat{a}\,\mu\nu} + \varepsilon^{\hat{a}\hat{b}\hat{c}} A^{\hat{b}}{}_{\mu} F^{\hat{c}\,\mu\nu} = 0$$

Como uma conseqüência da definição de F, as identidades de Bianchi são automaticamente satisfeitas, ou seja, o tensor de força sempre satisfaz a identidade

$$\mathcal{D}_{\alpha} \varepsilon^{\alpha\beta\mu\nu} F^{\hat{a}\mu\nu} = 0$$

2.3 Simetrias Espaço-Temporais do Campo

Nosso objetivo é impor uma simetria espaço-temporal sobre um campo de calibre sem impor, a priori, restrições às simetrias internas do campo. A questão não trivial a considerar é que campos que diferem apenas por uma transformação de calibre são fisicamente idênticos. Por essa razão não podemos simplesmente impor que o campo não seja alterado ao aplicarmos uma transformação de coordenadas, como é feito ao considerar outros campos como, por exemplo, a própria métrica. No caso dos campos de

calibre, temos que considerar simétricas aquelas soluções que ao serem transformadas por uma transformação de coordenadas sofram apenas uma mudança de calibre. A teoria geral para fazer essa análise foi desenvolvida por Forgács e Manton em 1980 [10]. Apresentamos abaixo esses resultados aplicados ao nosso caso de interesse.

Para definirmos as simetrias espaço-temporais do campo de Einstein-Yang-Mills temos que levar em conta as isometrias geradas pelo conjunto dos vetores de Killing ξ_m, onde m é um índice usado apenas para identificar os vetores neste conjunto. As simetrias da métrica são definidas diretamente pelas equações de Killing

$$\mathcal{L}_{\xi_m} g_{\mu\nu} = 0$$

Entretanto, como mencionado acima, para definirmos a simetria de um campo de calibre temos que considerar a possibilidade de que, após uma transformação de coordenadas, o campo seja o mesmo a menos de uma transformação de calibre. Neste caso as quantidades físicas permanecem as mesmas e o campo deve ser considerado simétrico. Em outras palavras $\mathcal{L}_{\xi_m} A_\mu = 0$ é uma condição suficiente mas não necessária para a simetria de um campo de calibre.

Temos então que verificar como uma transformação de calibre pode compensar uma transformação de coordenadas. Em uma transformação de coordenadas infinitesimal o tensor A_μ transforma-se segundo a relação

$$\tilde{A}_\mu = A_\mu + \epsilon^m \mathcal{L}_{\xi_m} A_\mu$$

onde ϵ^m são escalares infinitesimais. Embora o índice m apenas identifique os vetores de Killing, ξ_m, e os respectivos coeficientes, ϵ^m, vamos adotar também para esse índice a notação de soma sobre índices repetidos.

Por outro lado, uma transformação de calibre infinitesimal transforma a conexão A_μ de acordo com a expressão

$$\bar{A}_\mu = A_\mu + \epsilon^m \mathcal{D}_\mu W_m$$

onde W_m são escalares do espaço-tempo avaliados na álgebra de Lie do grupo de simetrias de calibre.

Podemos agora determinar quando uma transformação de calibre pode compensar os efeitos de uma transformação de coordenadas. Isso ocorrerá se existirem escalares W_m que anulem os efeitos da transformação de coordenadas, ou seja, satisfaçam a condição

$$\mathcal{L}_{\xi_m} A_\mu = \mathcal{D}_\mu W_m \tag{2.4}$$

Se não for possivel encontrar W_m que satizfaçam (2.4) o campo não posssui a simetria considerada [10].

Se considerarmos que as quantidades W_m se comportam como escalares nas transformações de coordenadas e que sob uma transformação de calibre u se transformam segundo a regra

$$W_m \to \tilde{W}_m = uW_m u^{-1} + (\mathcal{L}_{\xi_m} u)\, u^{-1} \tag{2.5}$$

a equação (2.4) é covariante e independente do calibre. Podemos portanto considerá-la como a definição de simetria espaço-temporal para um campo de calibre.

Temos ainda que levar em conta que qualquer combinação linear dos vetores de Killing, ξ_m, também é um vetor de Killing, $\xi^\mu = \lambda_m \xi^\mu{}_m$. Além disso, se considerarmos a operação

$$[\xi_m, \xi_n]^\mu = \xi^\rho{}_m \left(\partial_\rho \xi^\mu{}_n\right) - \xi^\rho{}_n \left(\partial_\rho \xi^\mu{}_m\right)$$

esse espaço constitui uma álgebra de Lie com a seguinte estrutura algébrica

$$[\xi_m, \xi_n]^\mu = f_{mnp}\, \xi^\mu{}_p$$

sendo que f_{mnp} satisfaz as seguintes relações

$$f_{mnp} = -f_{nmp}$$

$$f_{mnp} f_{pqr} + f_{nqp} f_{pmr} + f_{qmp} f_{pnr} = 0$$

Usando esta informação sobre os vetores de Killing, pode-se provar que os escalares W_m, devem satisfazer a equação de consistência

$$\mathcal{L}_{\xi_m} W_n - \mathcal{L}_{\xi_n} W_m - [W_m, W_n] = f_{mnp} W_p \tag{2.6}$$

Assim temos todas as ferramentas para impor simetrias espaço-temporais nos campos de Yang-Mills.

Um Único Vetor de Killing

Nesse trabalho estamos interessados em um caso particular de simetria espaço-temporal, a simetria axial. Essa simetria é definida por um único vetor de Killing, tipo espaço com órbitas fechadas simples.

Ao considerarmos um único vetor de Killing a condição de simetria pode ser consideravelmente simplificada, pois nesse caso o grupo de simetrias é abeliano e sua álgebra de

Lie é unidimensional. Essa restrição triavializa a condição (2.6) e permite que o escalar W seja anulado por uma transformação de calibre em um sistema de coordenadas adequado.

Para realizarmos essa simplificação, primeiramente escolhemos um sistema de coordenadas adaptado à simetria do espaço-tempo gerada pelo vetor de Killing ξ_μ, ou seja, um sistema onde, $\xi^\mu = \delta^\mu{}_4$. Denotaremos a coordenada simétrica x^4 por φ. Com essa escolha do sistema de coordenadas a equação (2.4) torna-se

$$\partial_\varphi A_\mu = \mathcal{D}_\mu W$$

Impomos agora a transformação de calibre

$$u = \mathcal{P} \exp\left(- \int_0^\varphi W(x^i, y)\, dy \right) \tag{2.7}$$

aqui os índices latinos estão indicando as coordenadas não simétricas, $i = 0, 1, 2$ e $\mathcal{P}$ é o ordenamento da integração, que será empregado para selecionar a ordem de integração. Como veremos abaixo, essa ordenação será importante ao derivarmos u.

Ao aplicarmos a transformação (2.5) com u definido por (2.7) temos

$$\bar{W} = uWu^{-1} + (\mathcal{L}_\xi u)\, u^{-1} \tag{2.8}$$

Para calcularmos a derivada de Lie do elemento u vamos impor que a ordenação $\mathcal{P}$ seja tal que após a derivação, a ordem dos termos seja a seguinte

$$\mathcal{L}_\xi u = \partial_\varphi u = -uW$$

Substituindo na transformação (2.8) temos que

$$\bar{W} = uWu^{-1} - uWu^{-1} = 0$$

Deste modo selecionamos um sistema de coordenadas e um calibre onde a condição de simetria (2.4) se reduz a

$$\partial_\varphi A_\mu = 0 \tag{2.9}$$

É importante notar que neste ponto foi feita uma escolha parcial de coordenadas e de calibre. Para que essa condição permaneça válida não podemos fazer transformações de coordenadas que envolvam a coordenada φ como também não podemos aplicar transformações de calibre que dependam explicitamente dessa coordenada.

2.4 Energia e Momentos

No campo de Yang-Mills, assim como no campo eletromagnético, algumas componentes do tensor de energia momento podem ser interpretadas de modo especial. Essas componentes são a densidade de energia, $\rho = T^{00}$, e a densidade de momento, $J^i = T^{0i}$. Também como no caso das equações de Maxwell podemos definir os campos elétrico, E, e magnético, B, para o campo de Yang-Mills

$$E^i = F^{0i} \qquad\qquad B^k = -\frac{1}{2}\varepsilon^k_{ij}F^{ij}$$

Note que as componente de todos esses campos assumem valores na álgebra de Lie do grupo SU(2).

Os campos definidos acima estão diretamente relacionados. Podemos explicitar essa relação com as seguintes identidades

$$\rho = \frac{1}{2}\left(|E|^2 + |B|^2\right)$$

$$J^{\hat{a}} = \epsilon^{\hat{a}\hat{b}\hat{c}}B^{\hat{b}}E^{\hat{c}}$$

onde $|E|^2 = E_i{}^{\hat{a}}E^{i\hat{a}}$ e $|B|^2 = B_i{}^{\hat{a}}B^{i\hat{a}}$.

2.5 Yang-Mills em Um Espaço-Tempo Plano

Como os resultados numéricos obtidos consideram o campo de Yang-Mills em um espaço-tempo plano e como pode ser bastante complexo obter alguns resultados teóricos na sua forma mais geral apresentamos, nessa seção, alguns resultados que valem para o caso de uma métrica plana.

Escolhas de Calibre

Em geral associa-se às equações de Yang-Mills outras equações por meio de escolhas de calibre. A demonstração de que isso realmente pode ser realizado deve ser feita caso a caso [18]. Estes são alguns exemplos de escolhas de calibre para as equações de Yang-Mills.

Calibre de Lorentz Para impor essa escolha de calibre, acrescenta-se ao sistema de equações de Yang-Mills a equação

$$\partial^\mu A_\mu = 0$$

Com a adição desta equação as equações de movimento do campo podem ser escritas como um sistema hiperbólico, onde cada componente da conexão A_μ satisfaz uma equação de onda não-linear da forma

$$(\partial_t^2 - \Delta)A_\mu = Q_\mu(A, \partial A) + C_\mu(A, A, A) \tag{2.10}$$

Nessa expressão estamos considerando que Q_μ é uma função bilinear e C_μ é uma função trilinear.

Calibre de Coulomb Neste caso adiciona-se a equação

$$\partial_i A_i = 0 \qquad\qquad i = 1, 2, 3$$

Com essa escolha as equações de movimento dividem-se em três equações hiperbólicas da forma (2.10) para as componentes A_i e uma equação elíptica para A_t com a forma

$$-\Delta A_t = Q_0(A, \partial A) + C_0(A, A, A)$$

Calibre de Cronstöm Esse calibre é definido pela inclusão da equação

$$-tA_0 + x_k A_k = 0$$

Calibre Temporal Nesse calibre escolhe-se anular a componente temporal do campo, isso é, impõe-se a equação

$$A_t = 0 \tag{2.11}$$

O sistema resultante consiste de três equações de onda da forma (2.10) para as componentes A_i.

Assume-se que é possível transformar um campo de uma destas escolhas para outra através de uma transformação de calibre u. Isso é, dada uma solução suave das equações de Yang-Mills sempre será possível encontrar uma transformação de calibre, tal que a solução transformada satisfaça a equação agregada. Porém, isso impõem que u satisfaça uma equação diferencial, que no caso do calibre de Lorentz, ou no de Coulomb, é uma equação não linear. Então a possibilidade efetiva de realizar tal transformação fica dependente da existência de uma solução dessa equação não linear.

Por outro lado, para qualquer A dado, sempre podemos obter a transformação que leva ao calibre temporal resolvendo a equação linear

$$\partial_t u = A_t u$$

A demonstração desse fato é bastante simples. Considere uma conexão A_μ e que t seja a coordenada tipo tempo. Ao aplicarmos uma transformação u sobre A_μ sua componente temporal se transforma como

$$\tilde{A}_t = u A_t u^{-1} + \Big(\partial_t u\Big) u^{-1}$$

Se impusermos que $\tilde{A}_t = 0$ obtemos a seguinte equação diferencial para u

$$u A_t u^{-1} + \Big(\partial_t u\Big) u^{-1} = 0$$

rearranjando os termos dessa equação chegamos a

$$\partial_t u = -u A_t$$

que podemos resolver facilmente, obtendo:

$$u = \mathcal{P} \exp\left(-\int_0^t A_t(\tau, x^i)\, d\tau\right)$$

onde $\mathcal{P}$ é a ordem da integração.

Note que nesses cálculos não usamos nenhuma característica de um espaço tempo plano, ou seja, é sempre possível colocar uma solução das equações de Yang-Mills no calibre temporal em qualquer espaço-tempo.

Esse resultado é importante pois em todo o trabalho que desenvolvemos com o campo de Yang-Mills axialmente simétrico, usamos o calibre temporal. O principal argumento para essa escolha é que nesse calibre todas as equações de movimento são hiperbólicas, reduzindo a necessidade de manipular vínculos elípticos apenas à condição inicial.

O calibre de Cronstöm também pode ser sempre alcançado por uma transformação de calibre u, pois para obter esta transformação basta resolver a equação linear

$$(-tA_0 + x_j A_j)u - t\partial_0 u + x_j \partial_j u = 0$$

Existência de Soluções

No caso da métrica de Minkowski a existência da solução é garantida pelo seguinte resultado [18].

Teorema 2.1 (Existência de Soluções) *Para qualquer condição inicial com $A \in H^4$, $E \in H^3$, $B \in H^3$ e que satisfaça os vínculos*

$$\mathcal{D}_i E^i = \partial_i E^i + A_i \times E^i = 0$$

$$B_i = \varepsilon_i^{jk} \left(\partial_j A_k + \frac{1}{2} A_j \times A_k \right)$$

Existe uma única solução para as equações de Yang-Mills, no calibre temporal, para todo o tempo.

Transcrevemos agora os passos necessários para a demonstração desse resultado.

1. Primeiro prova-se a existência em um intervalo $[0, t_0]$, onde t_0 depende apenas da norma $|A|_{2,2} + |E|_{1,2} + |B|_{1,2}$ dos dados iniciais.

2. Então usando a energia sabe-se que $|E|$ e $|B|$ são limitados na norma L^2. Usando o calibre temporal, onde $\partial_t A = E$, prova-se que A também é limitado em L^2. Porém a limitação em L^2 ainda é uma condição muito fraca para um teorema de existência.

3. O passo principal é determinar um limitante em L^∞ para o campo.

   $$\sup_{0 \leq t \leq T} \sup_x \Big(\, |E(x,t)| + |B(x,t)| \, \Big) \leq C$$

   para qualquer $T > 0$.

4. De posse desse limitante prova-se que $|A|_{2,2} + |E|_{1,2} + |B|_{1,2}$ é limitado para qualquer tempo finito. Na verdade tem-se que

   $$A(t) = A(0) + \int_0^t E(t) dt$$

   também é limitado em L^∞. Considere

   $$\mathcal{E}_1 = \int |\partial E|^2 + |\partial B|^2 + |\partial^2 A|^2 dx$$

como o funcional de energia escalado, onde ∂ representa todas as primeiras derivadas. Com o uso das equações de evolução e dos vínculos, obtém-se

$$\frac{d\mathcal{E}_1}{dt} \leq C\mathcal{E}_1$$

onde C depende do limitante em L^∞. Assim $\mathcal{E}_1$ é limitado para qualquer tempo finito, valendo o mesmo para as derivadas maiores. Combinando esse resultado com a existência local completa-se a demonstração.

2.6 Soluções Abelianas Embebidas

O campo de Yang-Mills para o grupo SU(2) possui como soluções particulares as soluções do campo U(1). Essas soluções são chamadas de soluções Abelianas embebidas. De fato, a primeira solução exata para o campo de Yang-Mills clássico obtida por Ikeda e Miyachi [14] em 1962 consistia da solução de Coulomb para o eletromagnetismo [2] embebida no campo SU(2). A primeira solução genuinamente não Abeliana só foi encontrada em 1968 por Wu e Wang [23].

Uma forma de construir explicitamente soluções Abelianas no campo SU(2) é escolher condições iniciais onde duas cores sejam identicamente nulas. Como não estamos considerando fontes para o campo essa escolha implica que essas duas cores permanecerão nulas ao longo de toda a evolução. Embora isoladamente essas soluções sejam de pouco interesse vamos estudá-las em conjunto com as solução não Abelianas para termos parâmetros de comparação para o comportamento não linear.

3 O Campo Esfericamente Simétrico

Os campos esfericamente simétricos são consideravelmente mais simples que os axialmente simétricos, porém, já apresentam comportamento bastante elaborado. Vários resultados de grande interesse físico foram obtidos nesse contexto.

Em 1997, Witten apresentou um ansatz para o campo de Yang-Mills com simetria interna SU(2) [22]. Esse ansatz, atualmente conhecido como ansatz de Witten, contém todas as soluções esfericamente simétricas para o campo de Yang-Mills.

Nesse capítulo apresentamos esse ansatz no contexto da relatividade geral e exibimos as equações para o campo de Einstein-Yang-Mills com simetria esférica.

3.1 Espaço-Tempo Esfericamente Simétrico

Nesta seção apresentamos uma métrica geral para um espaço-tempo esfericamente simétrico. Vamos considerar um sistema de coordenadas esférico, r, θ, φ, onde θ, φ são coordenadas angulares e r é a coordenada radial. Denotaremos a coordenada tipo tempo por t. Estamos supondo que esse sistema de coordenadas cubra todo o espaço-tempo. Sob tais condições a métrica esfericamente simétrica mais geral pode ser descrita pelo seguinte elemento de linha

$$ds^2 = \left(-\alpha^2 + a^2\beta^2\right)\, dt^2 + 2a^2\beta\, dt\, dr + a^2\, dr^2 + b^2 r^2\, d\Omega^2 \tag{3.1}$$

onde $d\Omega^2$ é o elemento de linha da esfera

$$d\Omega^2 = d\theta^2 + \operatorname{sen}\theta^2\, d\varphi^2$$

e os coeficientes da métrica α, β, a, b, dependem apenas das coordenadas temporal, t, e radial, r.

3.2 O Ansatz de Witten

A forma original do ansatz proposto por Witten [22] é

$$A^a{}_0 = \frac{A_0 \delta_{ak} x_k}{r}$$

$$A^a{}_j = \frac{\phi_0 + 1}{r^2} \varepsilon_{jak} x_k + \frac{\phi_1}{r^3} \left(\delta_{aj} r^2 - \delta_{ak} x_k x_j \right) + A_1 \frac{\delta_{ak} x_k x_j}{r^2}$$

Embora descreva uma solução esfericamente simétrica, escrito nessa forma, o ansatz apresenta as componentes do vetor A_μ em coordenadas cartesianas. As funções A_0, A_1, ϕ_0, e ϕ_1 são funções apenas de t e $r = \sqrt{x^2 + y^2 + z^2}$. Enquanto que, os índices se referem a coordenadas cartesianas, tanto no grupo quanto no espaço-tempo. A base usada para os geradores do grupo são as matrizes de Pauli, apresentadas em (A.1).

Em seu trabalho original Witten estava interessado apenas no espaço-tempo plano. Posteriormente outros autores, interessados em soluções para relatividade geral, reescreveram esse ansatz de modo mais adequado para uma métrica geral esfericamente simétrica. Para obter essa nova forma para o ansatz é necessário mudar o sistema de coordenadas para coordenadas esféricas e substituir os geradores do campo SU(2). Os novos geradores são as chamadas projeções esféricas das matrizes de Pauli, τ^a, definidas em (A.2). Essas matrizes formam uma base anti-hermitiana para SU(2) satisfazendo $[\tau^a, \tau^b] = \varepsilon^{abc} \tau^c$, para $a, b, c \in \{r, \phi, \theta\}$. O ansatz é então reescrito da seguinte forma

$$A = A_0 \tau^r \, dt + A_1 \tau^r \, dr + \left(\phi_1 \tau^\theta + (\phi_0 + 1) \, \tau^\phi \right) \, d\theta + \left(\phi_1 \tau^\phi - (\phi_0 + 1) \, \tau^\theta \right) \operatorname{sen}\theta \, d\phi$$

Para chegarmos a forma usada nesse capítulo temos ainda que aplicar uma transformação de calibre, obtendo

$$A = u \tau^r \, dt + v \tau^r \, dr + \left(w \tau^\theta + \tilde{w} \tau^\phi \right) \, d\theta + \left(\cot\theta \tau^r + w \tau^\phi - \tilde{w} \tau^\theta \right) \operatorname{sen}\theta \, d\phi \tag{3.2}$$

As funções u, v, w e $\tilde{w}$ dependem das coordenadas r e t.

Podemos visualizar melhor a estrutura do ansatz de Witten escrevendo-o em forma tensorial

$$A^r{}_\mu = \begin{pmatrix} u \\ v \\ 0 \\ \cos\theta \end{pmatrix} \qquad A^\theta{}_\mu = \begin{pmatrix} 0 \\ 0 \\ w \\ -\tilde{w} \operatorname{sen}\theta \end{pmatrix} \qquad A^\phi{}_\mu = \begin{pmatrix} 0 \\ 0 \\ \tilde{w} \\ w \operatorname{sen}\theta \end{pmatrix}$$

Uma característica marcante deste ansatz é que embora ele seja esfericamente simétrico ele depende explicitamente da coordenada simétrica θ. Isso é possivel devido a forma da equação de simetria para um campo de calibre (2.4). Esse é um exemplo de campo onde W_m não é trivial.

Na construção deste ansatz foram eliminados vários graus de liberdade do campo de Yang-Mills. Entretanto este campo ainda é invariante sob transformações de calibre em U(1) descrita por

$$U = e^{\varphi(t,r)\tau^r} \tag{3.3}$$

onde $\varphi(t,r)$ é uma função escalar das coordenadas r e t.

O tensor de força derivado desta conexão é

$$\begin{aligned} F &= \tau^r\Big(\partial_t v - \partial_r u\Big)\, dtdr \\ &+ \Big[\Big(\partial_t w - u\tilde{w}\Big)\, dt + \Big(\partial_r w - v\tilde{w}\Big)\, dr\Big]\Big(\tau^\theta\, d\theta + \tau^\phi \,\mathrm{sen}\theta\, d\phi\Big) \\ &+ \Big[\Big(\partial_t \tilde{w} - uw\Big)\, dt + \Big(\partial_r \tilde{w} - vw\Big)\, dr\Big]\Big(\tau^\phi\, d\theta - \tau^\theta \,\mathrm{sen}\theta\, d\phi\Big) \\ &- \Big(1 - w^2 - \tilde{w}^2\Big)\,\mathrm{sen}\theta\tau^r d\theta d\phi \end{aligned}$$

que tensorialmente assume a forma

$$F^r{}_{\mu\nu} = \begin{pmatrix} 0 & \partial_t v - \partial_r u & 0 & 0 \\ -(\partial_t v - \partial_r u) & 0 & 0 & 0 \\ 0 & 0 & 0 & -(1 - w^2 - \tilde{w}^2)\,\mathrm{sen}\theta \\ 0 & 0 & (1 - w^2 - \tilde{w}^2)\,\mathrm{sen}\theta & 0 \end{pmatrix}$$

$$F^\theta{}_{\mu\nu} = \begin{pmatrix} 0 & 0 & \partial_t w - u\tilde{w} & -(\partial_t \tilde{w} + uw)\,\mathrm{sen}\theta \\ 0 & 0 & \partial_r w - v\tilde{w} & -(\partial_r \tilde{w} + vw)\,\mathrm{sen}\theta \\ -(\partial_t w - u\tilde{w}) & -(\partial_r w - v\tilde{w}) & 0 & 0 \\ (\partial_t \tilde{w} + uw)\,\mathrm{sen}\theta & (\partial_r \tilde{w} + vw)\,\mathrm{sen}\theta & 0 & 0 \end{pmatrix}$$

$$F^\phi{}_{\mu\nu} = \begin{pmatrix} 0 & 0 & \partial_t \tilde{w} + uw & (\partial_t w - u\tilde{w})\,\mathrm{sen}\theta \\ 0 & 0 & \partial_r \tilde{w} + vw & (\partial_r w - v\tilde{w})\,\mathrm{sen}\theta \\ -(\partial_t \tilde{w} + uw) & -(\partial_r \tilde{w} + vw) & 0 & 0 \\ -(\partial_t w - u\tilde{w})\,\mathrm{sen}\theta & -(\partial_r w - v\tilde{w})\,\mathrm{sen}\theta & 0 & 0 \end{pmatrix}$$

3.3 Einstein-Yang-Mills Esfericamente Simétrico

Vamos agora construir as equações que descrevem o campo de Einstein-Yang-Mills com simetria esférica. Para isso substituímos a métrica (3.1), que descreve um espaço-tempo esfericamente simétrico, nas equações de Einstein (2.2), e o ansatz de Witten (3.2) no tensor de energia momento (2.3). Após algumas manipulações [7] podemos escrever as equações de movimento da teoria de Einstein-Yang-Mills como um sistema de equações de primeira ordem no tempo

$$\partial_t \Pi = \partial_r \left(\beta\, \Pi + \frac{\alpha}{a}\, \Phi \right) + uP - v \left(\beta\, P + \frac{\alpha}{a}\, Q \right) + \frac{\alpha\, a}{b^2\, r^2}\, w \left(1 - w^2 - \tilde{w}^2 \right) \tag{3.4}$$

$$\partial_t P = \partial_r \left(\beta\, P + \frac{\alpha}{a}\, Q \right) - u\, \Pi + v \left(\beta\, \Pi + \frac{\alpha}{a}\, \Phi \right) + \frac{\alpha\, a}{b^2\, r^2}\, \tilde{w} \left(1 - w^2 - \tilde{w}^2 \right) \tag{3.5}$$

$$\partial_r Y = \tilde{w}\, \Pi - wP \tag{3.6}$$

$$\partial_t Y = \frac{\alpha}{a} \left(\tilde{w}\, \Phi - w\, Q \right) + \beta \left(\tilde{w}\, \Pi - w\, P \right) \tag{3.7}$$

onde foram usadas as definições

$$Y = \frac{b^2\, r^2}{2\, \alpha\, a} \left(\partial_t v - \partial_r u \right) \tag{3.8}$$

$$\Pi = \frac{\alpha}{a} \Big(\partial_t w - u\tilde{w} - \beta (\partial_r w - v\tilde{w}) \Big) \tag{3.9}$$

$$\Phi = \partial_r w - v\tilde{w} \tag{3.10}$$

$$P = \frac{\alpha}{a} \Big(\partial_t \tilde{w} + uw - \beta (\partial_r \tilde{w} - vw) \Big) \tag{3.11}$$

$$Q = \partial_r \tilde{w} + vw \tag{3.12}$$

As funções acima foram definidas de modo que as equações de movimento não contivessem, explicitamente, as primeiras derivadas, espacial e temporal, das funções w e $\tilde{w}$.

Temos agora que definir equações de movimento para as grandezas Φ e Q. Para isso derivamos suas definições (3.10) e (3.12).

$$\partial_t \Phi = \partial_{tr} w - \tilde{w} \partial_t v - v \partial_t \tilde{w}$$

$$\partial_t Q = \partial_{tr} \tilde{w} - w \partial_t v - v \partial_t w$$

Isolamos agora as funções $\partial_t v$, $\partial_t w$ e $\partial_t \tilde{w}$ nas expressões (3.8), (3.9) e (3.11), respectivamente. Substituindo essas expressões nas derivadas temporais de Φ e Q temos suas equações evolutivas

$$\partial_t \Phi = \partial_r \left(\frac{\alpha}{a} \Pi + \beta \Phi \right) + uQ - v \left(\frac{\alpha}{a} P + \beta Q \right) + \tilde{w} \, \frac{2\alpha a}{b^2 \, r^2} \, Y$$

$$\partial_t Q = \partial_r \left(\frac{\alpha}{a} P + \beta Q \right) - u\Phi + v \left(\frac{\alpha}{a} \Pi + \beta \Phi \right) + w \, \frac{2\alpha a}{b^2 \, r^2} \, Y$$

A decomposição 3+1 das equações e Einstein produz as equações de evolução para os componentes da métrica

$$\partial_t a = -a\alpha K^r{}_r + \partial_r(\alpha\beta)$$

$$\partial_t b = -\alpha b K^\theta{}_\theta + \frac{\beta}{r} \partial_r (rb)$$

$$\partial_t K^r{}_r = \beta \partial_r K^r{}_r + \alpha K^r{}_r K - \frac{1}{a} \partial_r \left(\frac{\partial_r \alpha}{a} \right) - \frac{2\alpha}{arb} \partial_r \left(\frac{\partial_r (rb)}{a} \right) + 4\pi G \alpha \left(S - \rho - 2 S^r{}_r \right)$$

$$\partial_t K^\theta{}_\theta = \beta \partial_r K^\theta{}_\theta + \alpha K^\theta{}_\theta K + \frac{\alpha}{(rb)^2} - \frac{1}{a(rb)^2} \partial_r \left(\frac{\alpha r b}{a} \partial_r (rb) \right) + 4\pi G \alpha \left(S - \rho - 2 S^\theta{}_\theta \right)$$

assim como os vínculos Hamiltoniano e de momento

$$4 K^r{}_r K^\theta{}_\theta + 2 K^\theta{}_\theta{}^2 - \frac{2}{arb} \left\{ \partial_r \left(\frac{\partial_r (rb)}{a} \right) + \frac{1}{rb} \left[\partial_r \left(\frac{rb}{a} \partial_r (rb) \right) - a \right] \right\} = 16\pi G \rho$$

$$\partial_r K^\theta{}_\theta + \frac{\partial_r (rb)}{rb} \left(K^\theta{}_\theta - K^r{}_r \right) = -4\pi G J_r$$

Nas equações acima ρ é a densidade de energia J_r é a densidade de momento

$$\rho = \frac{1}{4\gamma^2} \left[\frac{4Y^2}{b^4 r^4} + \frac{(1 - w^2 - \tilde{w}^2)^2}{b^4 r^4} + \frac{2}{b^2 r^2 a^2} \left(Q^2 + \Phi^2 + P^2 + \Pi^2 \right) \right]$$

$$J_r = -\frac{1}{\gamma^2 a b^2 r^2} \Big(\, \Pi \, \Phi + P \, Q \, \Big)$$

sendo $S^r{}_r$ e $S^\theta{}_\theta$ as componentes do tensor de energia-momento projetadas na hipersuperfície tipo espaço

$$S^r{}_r = \frac{1}{4\gamma^2} \left[-\frac{4Y^2}{b^4 r^4} - \frac{(1 - w^2 - \tilde{w}^2)^2}{b^4 r^4} + \frac{2}{b^2 r^2 a^2} \left(Q^2 + \Phi^2 + P^2 + \Pi^2 \right) \right]$$

$$S^\theta{}_\theta = S^\phi{}_\phi = \frac{1}{4\gamma^2} \left[\frac{4Y^2}{b^4 r^4} + \frac{(1 - w^2 - \tilde{w}^2)^2}{b^4 r^4} \right]$$

3.4 Equações de Bartnik e McKinnon

Um caso particular de grande interesse foi obtido por Bartnik e McKinnon [3]. Eles consideraram um campo esfericamente simétrico e estático eliminando a dependência da coordenada tipo tempo t das equações para uma métrica esfericamente simétrica e para o ansatz de Witten. Além disso eles fixaram os graus de liberdade restantes.

Para eliminar as liberdades do sistema de coordenadas eles escolheram $\beta = 0$ e $b = 1$. Com essas escolhas o elemento de linha (3.1) torna-se

$$ds^2 = -\alpha^2\, dt^2 + a^2\, dr^2 + r^2\, d\Omega^2$$

O ansatz de Witten (3.2) ainda possui liberdades de calibre, representadas pela transformação (3.3). Impomos então $v = 0$ como uma escolha de calibre. A independência temporal permite que fixemos $\tilde{w} = 0$ sem perda de generalidade. Restringindo o campo de Yang-Mills ao caso de um monopólo-magnético puro, $u = 0$, a conexão A_μ pode ser escrita como

$$A = w\tau^\theta\, d\theta + \left(\cot\theta\tau^r + w\tau^\phi\right) \operatorname{sen}\theta\, d\phi$$

Com estas simplificações as equações de Einstein-Yang-Mills tornam-se, após alguma manipulação, nas equações conhecidas como equações de Bartnik e McKinnon

$$\partial_r\left(\frac{\alpha}{a}\partial_r w\right) = \frac{\alpha a}{r^2} w\left(1 - w^2\right)$$

$$\frac{1}{a}\partial_r a = \frac{1 - a^2}{2r} + 4\pi G r a^2 \rho$$

$$\frac{1}{\alpha}\partial_r \alpha = -\frac{1 - a^2}{2r} + 4\pi G r a^2 S^r{}_r.$$

Nessas equações as funções a, α e w dependem apenas da coordenadas radial r. Como conseqüência direta da escolha $\tilde{w} = 0$ a equação (3.10) se reduz a $\Phi = \partial_r w$.

A densidade de energia para essas soluções é dada por

$$\rho = \frac{1}{2\gamma^2 r^2 a^2}\left[\left(\partial_r w\right)^2 + \frac{a^2}{2r^2}\left(1 - w^2\right)^2\right]$$

Com essas equações Bartnik e McKinnon obtiveram soluções para o campo de Einstein-Yang-Mills que violam a conjectura de que *buracos negros não tem cabelo*.

3.5 Coordenadas de Área

Vamos estudar nessa seção um caso particular do campo de Einstein-Yang-Mills com simetria esférica, onde Choptuik et. al. [7, 8] exibiram fenômenos críticos no colapso gravitacinal [6]. Esses fenômenos aparecem na "tansição de fase" entre formar ou não um buraco negro. As simulações numéricas de Choptuik mostraram dois tipos de transições de fase: um em que o buraco negro se forma com massa finita e outro em que o buraco negro se forma com massa infinitesimal em acordo com uma relação de expoentes críticos [13].

O sistema de coordenadas foi escolhido como polar $K^{\theta}{}_{\theta} = 0$ e de área $b = 1$. Com esta escolha a coordenada r é imediatamente relacionada com a área das esferas bidimensionais centradas na origem. Além disso o campo de Yang-Mills é considerado puramente magnético, isso é, $u = v = \tilde{w} = 0$.

Nesse contexto a métrica é escrita como

$$ds^2 = -\alpha^2\, dt^2 + a^2\, dr^2 + r^2\, d\Omega^2$$

e o tensor de força do campo se reduz a

$$F = \partial_t w\; dt\, d\hat{\Omega} + \partial_r w\; dr\, d\hat{\Omega} - (1 - w^2)\tau^{\hat{3}}\, \mathrm{sen}\theta\; d\theta\, d\phi$$

onde $d\hat{\Omega} = \tau^{\hat{1}} d\theta + \tau^{\hat{2}}\, \mathrm{sen}\theta d\phi$. Este tensor será nulo apenas quando $w = \pm 1$. Isso significa que, neste caso, o campo de Yang-Mills tem exatamente dois estados de vácuo discretos.

As equações de evolução apresentadas na Seção (3.3) se reduzem a

$$\partial_t \Phi = \partial_r \left(\frac{\alpha}{a} \Pi \right) \tag{3.13}$$

$$\partial_t \Pi = \partial_r \left(\frac{\alpha}{a} \Phi \right) + \frac{\alpha a}{r^2} w \left(1 - w^2 \right) \tag{3.14}$$

que podem ser simplificadas em

$$\Phi = \partial_r w \qquad\qquad \Pi = \frac{\alpha}{a} \partial_t w$$

Se w for considerado uma quantidade derivada essas equações podem ser resolvidas diretamente para Π e Φ. Nesse caso w é obtido pela integral

$$w(r,t) = w_0 + \int_0^r \Phi(\tilde{r}, t)\, d\tilde{r}$$

Neste sistema as únicas equações de Einstein relevantes são o vínculo Hamiltoniano

$$\frac{1}{a}\partial_r a + \frac{a^2 - 1}{2r} - \frac{1}{r}\left[\Phi^2 + \Pi^2 + \frac{a^2}{2r^2} \left(1 - w^2 \right)^2 \right] = 0 \tag{3.15}$$

e a condição de fatiamento polar

$$\frac{1}{\alpha}\partial_r\alpha - \frac{a^2-1}{2r} + \frac{1}{r}\left[\Phi^2 + \Pi^2 - \frac{a^2}{2r^2}\left(1-w^2\right)^2\right] = 0 \tag{3.16}$$

A densidade de energia do campo de Yang-Mills nesse sistema é descrito pela expressão

$$\rho = \frac{1}{4\gamma^2 r^2}\left[\frac{2}{a^2}\left(\Phi^2+\Pi^2\right) + \frac{(1-w^2)^2}{r^2}\right] \tag{3.17}$$

3.6 Resultados Numéricos

Usando as equações apresentadas na seção anterior simulamos numericamente o campo de Einstein-Yang-Mills com simetria esférica. As equações evolutivas (3.13) e (3.14) foram resolvidas com um método de Runge-Kutta adaptativo e os vínculos (3.15) e (3.16) foram resolvidos para dada passo no tempo por um método de evolução partindo da origem e avançando em r. Como os valores das funções envolvidas são conhecidos apenas nos pontos da malha escolhemos implementar o método de Euler implícito para resolver os vínculos.

Como condição inicial impusemos um pulso tipo *kink* no campo de Yang-Mills

$$w(r,0) = \left[1 + a\left(1+\frac{br}{\lambda}\right)\exp\left[-2\left(\frac{r}{\lambda}\right)^2\right]\right]\tanh\left(\frac{r_o - r}{\lambda}\right)$$

onde r_o é o centro do pulso e λ é a largura. As constantes a e b são definidas de modo que $w(0,0) = 1$ e $\partial_r w(0,0) = 0$, o que implica nos valores

$$a + 1 = \left[\tanh\left(\frac{r_o}{\lambda}\right)\right]^{-1}$$

$$b = \operatorname{sech}^2\left(\frac{r_o}{\lambda}\right)\left[\tanh\left(\frac{r_o}{\lambda}\right) - \tanh^2\left(\frac{r_o}{\lambda}\right)\right]^{-1}$$

A regularidade da origem foi usada como condição de fronteira. Essa condição implica $\partial_r a = \partial_r \alpha = 0$ e $a = 1$ em $r = 0$. Como essas funções são obtidas por um método de evolução em r, não precisamos impor condições de fronteira no limite exterior do domínio, $r = R_\infty$. As funções Φ e Π são avaliadas pelo método de Runge-Kutta avançando em t. Portanto essas equações demandam condições de fronteira tanto em $r = 0$ quanto em $r = R_\infty$. Na origem é imposta a regularidade do campo, que leva às condições

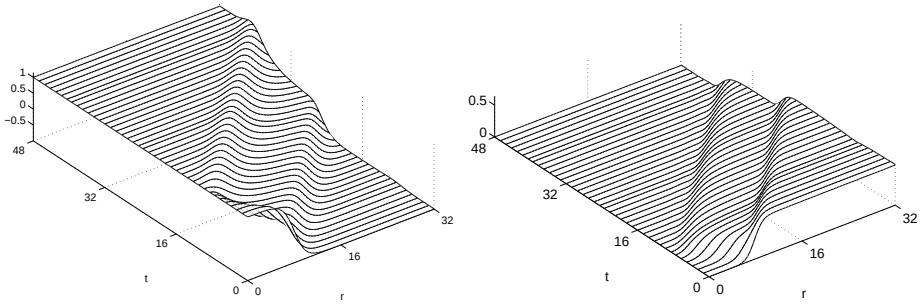
$$\Phi(0,t) = \Pi(0,t) = 0$$

A condição de radiação é obtida pela projeção do campo sobre o vetor nulo que aponta para fora do domínio. Essa condição determina um vínculo entre a derivada temporal e espacial do campo na fronteira

$$\partial_t \Phi = \frac{\alpha}{a} \partial_r \Phi$$

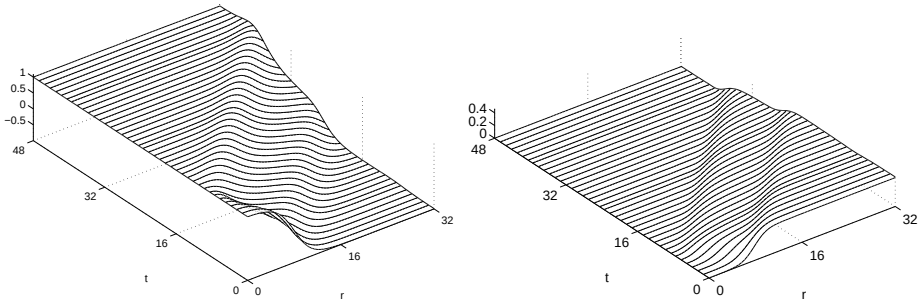
$$\partial_t \Pi = \frac{\alpha}{a} \partial_r \Pi$$

Descreveremos agora algumas soluções para esse sistema geradas numericamente. O primeiro grupo de soluções parte de condições iniciais temporalmente simétricas, isso é, $\Pi = 0$. Enquanto que no segundo grupo a condição inicial é construída de modo que o pulso se dirija para a origem do sistema de coordenadas, esse comportamento é conseguido pela imposicão de $\Pi = \Phi$ em $t = 0$.



(a) w (b) m

Figura 3.1: *Campo e massa da solução com* $\Pi = 0$ *e* $\lambda = 2$

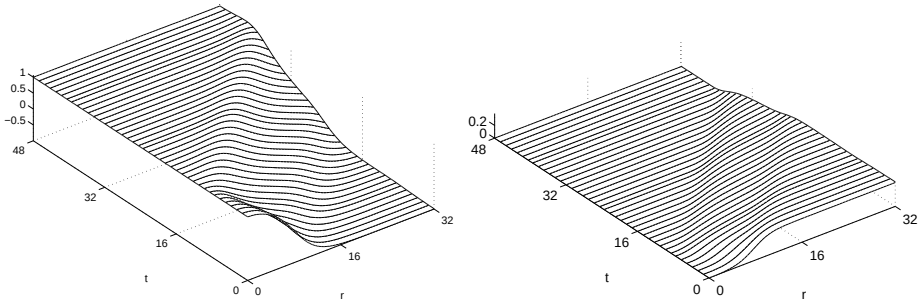


(a) w (b) m

Figura 3.2: *Campo e massa da solução com* $\Pi = 0$ *e* $\lambda = 3$

As Figuras 3.1, 3.2 e 3.3 exibem o campo de Yang-Mills, w, e a função de massa da métrica

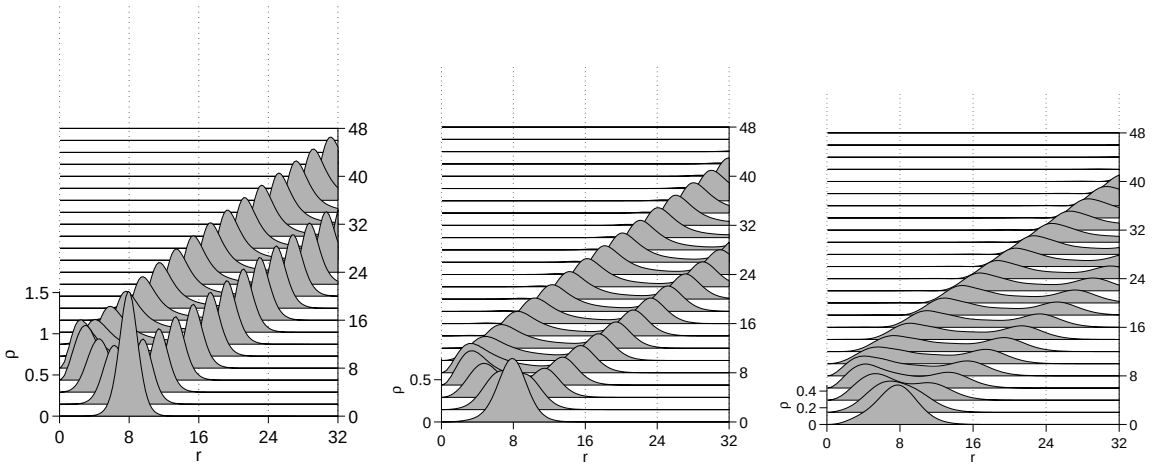
$$m(r,t) = \frac{r}{2}\left(1 - \frac{1}{a^2}\right)$$



(a) w (b) m

Figura 3.3: *Campo e massa da solução com* $\Pi = 0$ *e* $\lambda = 4$

para três soluções com condição inicial temporalmente simétrica. Em todas essas soluções o pulso inicial está centrado em $r = 8$ e a largura dos pulsos, λ, é igual a 2, 3, e 4, respectivamente.



(a) $\lambda = 2$ (b) $\lambda = 3$ (c) $\lambda = 4$

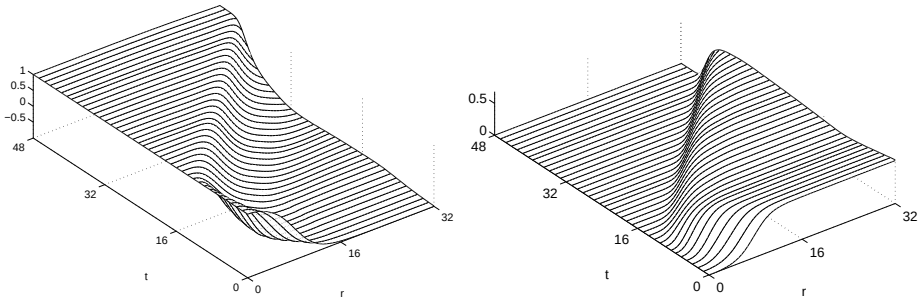
Figura 3.4: *Energia* $\times r$ *para as soluções com velocidade inicial nula,* $\Pi = 0$

A energia do campo (3.17) está ligada às derivadas do campo w. Portando aumentando a largura do *kink* diminuímos a energia contida no campo. Podemos comparar as energias das três soluções nos gráficos da Figura 3.4. Esses gráficos apresentam as integrais da densidade de energia sobre as esferas centradas na origem

$$E_r(r,t) = \int \rho(r,\varphi,\theta,t)\, r^2 d\Omega$$

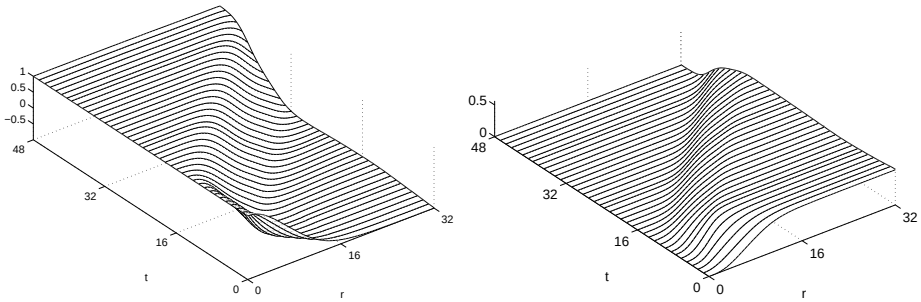
Na primeira solução, com $\lambda = 2$, o pico do pulso da energia alcança 1.5 enquanto que nos casos seguintes ficam em torno de 0.6 e 0.4. Nas três soluções o pulso inicial se divide em dois. Um partindo em direção a origem e outro para o exterior do domínio. O pulso que

se dirige para a origem é refletido em $t \approx 8$ e deixa o domínio em $t \approx 40$. O outro pulso sai do domínio em $t \approx 24$.



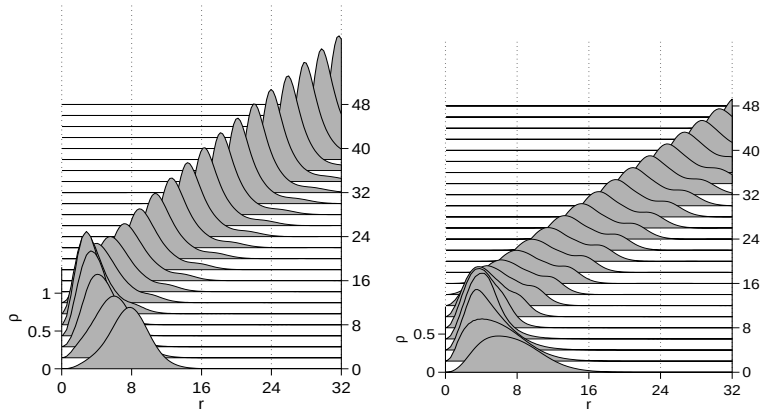
(a) w (b) m

Figura 3.5: *Campo e massa da solução com* $\Pi = \Phi$ *e* $\lambda = 4$



(a) w (b) m

Figura 3.6: *Campo e massa da solução com* $\Pi = \Phi$ *e* $\lambda = 6$



(a) $\lambda = 4$ (b) $\lambda = 6$

Figura 3.7: *Energia* $\times r$ *para as soluções com velocidade inicial na direção da origem,* $\Pi = \Phi$

Um outro conjunto de simulações partem de pulsos que no instante inicial seguem em direção a origem. Essas soluções podem ser vistas nas Figuras 3.5 e 3.6. Assim como no caso anterior essas figuras exibem o campo, w, e a função de massa, m. Esses pulsos também estão centrados em $r = 8$ e tem larguras 4 e 6, respectivamente.

No gráfico da função de massa da Figura 3.5 podemos perceber uma diminuição no valor da função no extremo externo do domínio, aproximadamente em $t = 8$. Essa diminuição indica uma perda de massa devido a erros numéricos. Isso é uma conseqüência do termo de dissipação artificial utilizado para estabilizar as soluções. Quando o pulso se aproxima da origem surge uma região com gradiente relativamente alto. Nessa região ocorre a perda de energia ou equivalentemente de massa.

Os gráficos da Figura 3.7 exibem a integral da densidade de energia sobre as esferas centradas na origem. Em ambos os casos podemos ver claramente o pulso se dirigindo à origem e refletindo para o exterior. Para $\lambda = 4$ o pulso de energia é mais condensado no início e fica claro na evolução que ocorre uma concentração do pulso com o passar do tempo. No caso $\lambda = 6$ a energia está um pouco mais distribuída e não é evidente se o efeito de concentração também ocorre nesse caso.

3.7 Espaço-Tempo Plano

Se considerarmos o ansatz de Witten em um espaço-tempo plano as equações de Yang-Mills se reduzem a equação do potencial $\lambda\phi^4$. Chegamos a esse resultado considerando o espaço-tempo plano descrito pela métrica (3.1), onde escolhemos $\alpha = a = b = 1$ e $\beta = 0$. Nesse caso, as equações do campo de Yang-Mills esfericamente simétrico (3.4-3.7) se reduzem a

$$\partial_t \Pi = \partial_r \Phi + uP - vQ + \frac{w}{r^2}\left(1 - w^2 - \tilde{w}^2\right)$$

$$\partial_t P = \partial_r Q - u\Pi + v\Phi + \frac{\tilde{w}}{r^2}\left(1 - w^2 - \tilde{w}^2\right)$$

$$\partial_r Y = \tilde{w}\Pi - wP$$

$$\partial_t Y = \tilde{w}\Phi - wQ$$

e as funções Π, Φ, P, Q e Y tornam-se

$$\Pi = \partial_t w - u\tilde{w}$$

$$\Phi = \partial_r w - v\tilde{w}$$

$$P = \partial_t \tilde{w} + uw$$

$$Q = \partial_r \tilde{w} + vw$$

$$Y = \frac{r^2}{2}\Big(\partial_t v - \partial_r u\Big)$$

Se fizermos agora as escolhas $u = v = \tilde{w} = 0$ as expressões acima reduzem-se a

$$\partial_t \Pi = \partial_r \Phi + \frac{w}{r^2}\Big(1 - w^2\Big) \qquad \partial_t P = \partial_r Q \qquad \Pi = \partial_t w \qquad \Phi = \partial_r w$$

Rearranjando essas expressões, obtemos a equação de movimento para w

$$\partial_{tt} w = \partial_{rr} w + \frac{1}{r^2}\Big(w - w^3\Big) \tag{3.18}$$

Que é uma equação de onda unidimensional com auto-interação ponderada pela área da esfera.

4 O Campo Axialmente Simétrico

Nos capítulos anteriores foram apresentadas a teoria geral para os campos de Einstein-Yang-Mills e sua simplificação para o caso de um campo esfericamente simétrico. Vamos agora apresentar as equações para o campo com simetria axial.

Inicialmente, com base nos resultados do Capítulo 2, construímos a Lagrangeana reduzida para a teoria de Einstein-Yang-Mills com simetria axial. A partir dessa Lagranceana exibimos as equações de movimento, impondo condições de calibre adequadas. Em seguida, particularizamos essas equações para o caso de um espaço-tempo plano com coordenadas esféricas e determinamos as condições de regularidade das soluções axialmente simétricas. Nos Capítulos 6, 7 e 8 serão apresentadas soluções obtidas numericamente para essas equações.

4.1 A Lagrangeana Reduzida

A simetria axial é definida por um único vetor de Killing ξ^μ. Portanto seguiremos o desenvolvimento descrito na Seção 2.3. Para que nosso sistema de coordenadas seja compatível com a simetria axial escolhemos a coordenada $x^4 = \varphi$ de modo que $\xi^\mu = \delta^\mu{}_\varphi$. A imposição da condição de simetria (2.9) então corresponde a condição

$$\partial_\varphi A^{\hat{a}}_\mu = 0$$

E as transformações de calibre agora não podem depender explicitamente da coordenada φ.

Nessa seção vamos usar os índices latinos com chapéu para indicar as cores do campo, $\hat{a} = 1, 2, 3$. Enquanto que os índices latinos sem chapéu vão indicar as três coordenadas espaço-temporais não simétricas, $x_i = (t, x_2, x_3)$. Note que neste sistema de coordenadas já identificamos o tempo, t, com a componente x_1. Porém, a segunda e terceira

componentes ainda não foram fixadas, pois em todo o desenvolvimento dessa seção essas componentes podem ser consideradas coordenadas arbitrárias, desde que tipo espaço.

Antes de escrevermos a Lagrangeana reduzida tridimensional para o campo de Yang-Mills vamos renomear as componentes da conexão.

$$A_{\mu}^{\hat{a}} = \begin{pmatrix} A_{i}^{\hat{a}}\,(x^{j}) \\ \phi^{\hat{a}}\ \ (x^{j}) \end{pmatrix} \tag{4.1}$$

Essa escolha será justificada quando exibirmos a expressão da Lagrangeana reduzida que possibilita-nos tratar $A^{\hat{a}}{}_{i}$ como um campo de Yang-Mills tridimensional e $\phi^{\hat{a}}$ como um campo escalar.

Nosso primeiro passo para a construção da Lagrangeana reduzida é definir o operador de projeção sobre o espaço quociente gerado pela ação da simetria axial

$$\gamma_{\mu\nu} = g_{\mu\nu} - \frac{1}{s^2}\xi_{\mu}\xi_{\nu}$$

onde $s^2 = \xi^{\mu}\xi_{\mu}$ é a norma do vetor de Killing. Então renormalizamos o vetor de Killing, ξ_{μ}, e definimos o campo vetorial $Z_{\mu\nu}$ obtendo, respectivamente,

$$Y_{\mu} = \frac{1}{s^2}\xi_{\mu}$$

$$Z_{\mu\nu} = \partial_{\mu}Y_{\nu} - \partial_{\nu}Y_{\mu}$$

Com o uso das definições anteriores podemos reescrever a métrica do espaço-tempo quadridimensional em uma forma que será mais adequada a manipulação algébrica que realizaremos

$$g_{\mu\nu} = \begin{pmatrix} \gamma_{ij} + s^2 Y_i Y_j & s^2 Y_i \\ \\ s^2 Y_j & s^2 \end{pmatrix} \tag{4.2}$$

Podemos escrever também a inversa dessa métrica

$$g^{\mu\nu} = \begin{pmatrix} \gamma^{ij} & -\gamma^{ik}Y_k \\ \\ -\gamma^{jk}Y_k & p^2 + s^{-2} \end{pmatrix} \tag{4.3}$$

onde $p^2 = \gamma^{ij}Y_iY_j$. Note que este é exatamente o primeiro passo da decomposição (2+1)+1 do espaço-tempo.

Vamos considerar agora a densidade Lagrangeana para o campo de Yang-Mills

$$\mathcal{L} = \frac{1}{4} F^{\hat{a}\mu\nu} F^{\hat{a}}{}_{\mu\nu} \tag{4.4}$$

Nosso objetivo é isolar os termos, dessa Lagrangeana, que envolvam as componentes $\phi^{\hat{a}}$ da conexão (4.1). Usando o fato que $F^{\hat{a}}{}_{\mu\nu}$ é anti-simétrico podemos escrever

$$\mathcal{L} = \frac{1}{4} F^{\hat{a}ij} F^{\hat{a}}{}_{ij} + \frac{1}{2} F^{\hat{a}i\varphi} F^{\hat{a}}{}_{i\varphi} \tag{4.5}$$

Usando a nomenclatura que definimos para as componentes da conexão, os termos dessa expressão são escritos como

$$F^{\hat{a}}{}_{ij} = \partial_i A^{\hat{a}}{}_j - \partial_j A^{\hat{a}}{}_i + \varepsilon^{\hat{a}\hat{b}\hat{c}} A^{\hat{b}}{}_i A^{\hat{c}}{}_j$$

$$F^{\hat{a}}{}_{i\varphi} = \partial_i \phi^{\hat{a}} + \varepsilon^{\hat{a}\hat{b}\hat{c}} A^{\hat{b}}{}_i \phi^{\hat{c}}$$

Agora se considerarmos ϕ_i um escalar na variedade tridimensional, isso é, $\partial_i \phi = {}^{(3)}\nabla_i \phi$ então, $F^{\hat{a}}{}_{i\varphi}$ será igual à derivada covariante de calibre sobre ϕ_i nesta variedade, ou seja,

$$F^{\hat{a}}{}_{i\varphi} = {}^{(3)}\mathcal{D}_i \, \phi^{\hat{a}} = {}^{(3)}\nabla_i \, \phi^{\hat{a}} + \varepsilon^{\hat{a}\hat{b}\hat{c}} A^{\hat{b}}{}_i \phi^{\hat{c}}$$

Usando a expressão para a inversa da métrica (4.3) podemos subir os índices do tensor $F^{\hat{a}}_{\mu\nu}$. Se escrevermos o tensor $F^{\hat{a}\mu\nu}$ separando a componente associada à coordenada simétrica φ temos as expressões

$$F^{\hat{a}ij} = \gamma^{ik}\gamma^{jl} F^{\hat{a}}{}_{kl} + \left(\gamma^{im}\gamma^{jk} - \gamma^{ik}\gamma^{jm}\right) Y_m \;\; {}^{(3)}\mathcal{D}_k \phi^{\hat{a}}$$

$$F^{\hat{a}i\varphi} = \left[\left(s^{-2} + p^2\right)\gamma^{ik} - \gamma^{im}\gamma^{kn} Y_m Y_n\right] \;\; {}^{(3)}\mathcal{D}_k \phi^{\hat{a}} - \gamma^{ik}\gamma^{lm} Y_m F^{\hat{a}}{}_{kl}$$

Podemos agora avaliar os termos $F^{\hat{a}ij} F^{\hat{a}}{}_{ij}$ e $F^{\hat{a}i\varphi} F^{\hat{a}}{}_{i\varphi}$ que aparecem na Lagrangeana (4.5)

$$F^{\hat{a}ij} F^{\hat{a}}{}_{ij} = \gamma^{ik}\gamma^{jl} F^{\hat{a}}{}_{ij} F^{\hat{a}}{}_{kl} + 2\gamma^{im}\gamma^{jk} Y_m F^{\hat{a}}{}_{ij} \left({}^{(3)}\mathcal{D}_k \phi^{\hat{a}}\right)$$

$$F^{\hat{a}i\varphi} F^{\hat{a}}{}_{i\varphi} = \left(s^{-2} + p^2\right)\gamma^{ik} \; {}^{(3)}\mathcal{D}_k \phi^{\hat{a}} \; {}^{(3)}\mathcal{D}_i \phi^{\hat{a}} - \left[\gamma^{im} Y_m \left({}^{(3)}\mathcal{D}_i \phi^{\hat{a}}\right)\right]^2 - \gamma^{ik}\gamma^{lm} Y_m F^{\hat{a}}{}_{kl} \; {}^{(3)}\mathcal{D}_i \phi^{\hat{a}}$$

Se usarmos γ^{ij} para levantar os índices e eliminarmos o símbolo "$^{(3)}$" das derivadas covariantes tridimensionais podemos escrever a forma reduzida da densidade Lagrangeana (4.4)

$$\mathcal{L} = \frac{1}{4} F^{\hat{a}ij} F^{\hat{a}}{}_{ij} + \frac{1}{2}\left(\frac{1}{s^2} + p^2\right) \; \mathcal{D}^i \phi^{\hat{a}} \; \mathcal{D}_i \phi^{\hat{a}} - \frac{1}{2}\left(Y^i \; \mathcal{D}_i \phi^{\hat{a}}\right)^2 + F^{\hat{a}ij} Y_i \; \mathcal{D}_j \phi^{\hat{a}}$$

Nesta Lagrangeana o acoplamento entre o campo gravitacional, o campo de calibre e o campo escalar é bastante complexo, observe o termo $F^{\hat{a}ij}Y_i\ \mathcal{D}_j\phi^{\hat{a}}$. Entretanto, se nos restringirmos a um espaço-tempo sem rotação, $Y_i = 0$, teremos uma Lagrangeana muito mais simples

$$L = \frac{1}{4}F^{\hat{a}ij}F^{\hat{a}}{}_{ij} + \frac{1}{2s^2}\ \mathcal{D}^k\phi^{\hat{a}}\ \mathcal{D}_k\phi^{\hat{a}} \tag{4.6}$$

Neste caso os A_i comportam-se como campos de Yang-Mills SU(2) tridimensionais e ϕ como um campo escalar tridimensional.

4.2 Equações de Movimento e Escolhas de Calibre

Para simplificarmos a condição de simetria impusemos uma escolha de calibre parcial. Entretanto ainda estamos livres para fazer qualquer escolha que não dependa explicitamente da coordenada φ. Usaremos, então, esta liberdade restante para eliminar as componentes $A_t^{\hat{a}}$ do campo de calibre, impondo o calibre temporal (2.11). A vantagem desta escolha é evitar a necessidade de resolver equações elípticas não-lineares para calcular $A_t^{\hat{a}}$ em cada instante do tempo. Temos agora um campo que pode ser representado pela conexão

$$A_\mu^{\hat{a}} = \begin{pmatrix} 0 \\ A_1^{\hat{a}}\,(x^j) \\ A_2^{\hat{a}}\,(x^j) \\ \phi^{\hat{a}}\ \ (x^j) \end{pmatrix}$$

Com essa escolha de calibre a evolução de todas as componentes do campo podem ser determinadas por equações hiperbólicas. Porém ainda restam um conjunto de equações elípticas que agem como vínculos para as componentes do campo. Entretanto uma propriedade das equações de evolução é que elas preservam esses vínculos, ou seja, se a condição inicial satisfizer os vínculos, eles são automaticamente satisfeitos pela solução para todo o tempo. Deste modo ao resolvermos o problema de Cauchy é necessário impor os vínculos apenas na condição inicial.

Com as escolhas de calibre impostas, transformações que sejam dependentes das coordenadas simétrica e temporal, φ e t não são mais permitidas. Entretanto o calibre ainda não está totalmente determinado. A solução ainda é invariante sob qualquer transformação que dependa apenas das duas coordenadas espaciais restantes, $u = e^{\psi^{\hat{a}}(x_1,x_2)\tau^{\hat{a}}}$. Vamos então usar essa liberdade para simplificar a imposição dos vínculos sobre a superfície inicial, $t = 0$.

Empregando uma escolha de calibre semelhante ao calibre temporal (2.11) restrita a superfície inicial, podemos eliminar uma das componentes da conexão, $A_1^{\hat{a}}$. Para isso basta aplicar a transformação

$$u = \mathcal{P} \exp\left(- \int_0^{x^1} A_1(\xi, x^2)\, d\xi \right) \tag{4.7}$$

Como A_μ não depende de φ e estamos considerando t constante, esta transformação pode ser aplicada sem conflitos com as escolhas anteriores. Deste modo, na condição inicial, as únicas componentes a serem determinadas são $A_2^{\hat{a}}$ e $\phi^{\hat{a}}$.

Nas simulações numéricas vamos impor uma simplificação extra ao campo de Yang-Mills. Vamos fixar que $\phi = 0$, isso pode ser feito com uma escolha adequada da condição inicial. Como ϕ se comporta como um campo escalar no problema reduzido, se escolhermos $\phi = 0$ na condição inicial então ele será nulo em toda a solução.

4.3 Yang-Mills em um Espaço-Tempo Plano

Vamos agora considerar o caso particular de uma métrica plana, ou seja, estamos considerando que o campo de Yang-Mills não gera nenhum campo gravitacional e que também não sofre influência de nenhum campo gravitacional externo.

Para esse caso, escolhemos um sistema de coordenadas esférico onde utilizaremos o cosseno do ângulo com o eixo como coordenada, isso é, utilizaremos $\chi = \cos\theta$ no lugar de θ. Deste modo, o elemento de linha assume a forma

$$ds^2 = -dt^2 + dr^2 + \frac{r^2}{1-\chi^2} d\chi^2 + r^2(1-\chi^2) d\varphi^2$$

Escolhemos trabalhar com o cosseno do ângulo θ pois nosso objetivo é resolver numericamente as equações de movimento resultantes. Com essa escolha reduzimos a necessidade de avaliar numericamente funções trigonométricas.

Fixando o elemento de linha da forma descrita acima, a densidade Lagrangeana para o campo de Yang-Mills (4.4) pode ser escrita na forma (4.6). Além disso temos que $s = 1$, portanto, a Lagrangeana assume a forma

$$L = \frac{1}{4} F^{\hat{a}\, ij} F^{\hat{a}}{}_{ij} + \frac{1}{2} \left(\mathcal{D}^k \phi^{\hat{a}} \right) \left(\mathcal{D}_k \phi^{\hat{a}} \right)$$

As escolhas de calibre que empregaremos aqui são as mesmas que discutimos anteriormente. Vamos usar o calibre temporal, $A_t = 0$ e na condição inicial vamos impor também que $A_\chi = 0$.

Os métodos numéricos de evolução temporal que vamos empregar esperam que as equações a serem resolvidas possuam apenas derivadas primeiras no tempo. Portanto, no decorrer dessa seção substituiremos as derivadas temporais das componentes $A_i^{\hat{a}}$, com $i = r, \chi$, por $N^{\hat{a}}{}_i = \partial_t A^{\hat{a}}{}_i$.

Tendo definido a métrica e o calibre, os vínculos são escritos como

$$\partial_r\Big(r^2 N_r^{\hat{a}}\Big) + r^2\epsilon^{\hat{a}\hat{b}\hat{c}}A_r^{\hat{b}}N_r^{\hat{c}} = -\partial_\chi\Big[(1-\chi^2)N_\chi^{\hat{a}}\Big] - (1-\chi^2)\epsilon^{\hat{a}\hat{b}\hat{c}}A_\chi^{\hat{b}}N_\chi^{\hat{c}} \tag{4.8}$$

Entretanto uma propriedade conhecida das equações de Yang-Mills é que os vínculos são mantidos pelas equações de movimento. Portando é necessário impo-los apenas na condição inicial. Além disso, podemos usar a transformação de calibre (4.7) para eliminarmos uma das componentes desse campo. No caso particular que estamos tratando vamos escolher eliminar a componente $A_\chi^{\hat{a}}$. Desta forma a equação que realmente deve ser resolvida, sobre a superfície inicial, é

$$\partial_r\Big(r^2 N_r^{\hat{a}}\Big) + r^2\epsilon^{\hat{a}\hat{b}\hat{c}}A_r^{\hat{b}}N_r^{\hat{c}} = -\partial_\chi\Big[(1-\chi^2)N_\chi^{\hat{a}}\Big] \tag{4.9}$$

As equações para a evolução das componentes A_r e A_χ são

$$r^2\partial_t N_r^{\hat{a}} = \partial_\chi\left[(1-\chi^2)\left(\partial_\chi A_r^{\hat{a}} - \partial_r A_\chi^{\hat{a}}\right)\right] + F_r^{\hat{a}} + G_r^{\hat{a}} \tag{4.10}$$

$$\partial_t N_\chi^{\hat{a}} = \partial_{rr}A_\chi^{\hat{a}} - \partial_{r\chi}A_r^{\hat{a}} + F_\chi^{\hat{a}} + G_\chi^{\hat{a}} \tag{4.11}$$

onde definimos as quantidades

$$\begin{aligned}
F_r^{\hat{a}} &= \left(1-\chi^2\right)\epsilon^{\hat{a}\hat{b}\hat{c}}\left(2A_\chi^{\hat{b}}\partial_\chi A_r^{\hat{c}} - A_\chi^{\hat{b}}\partial_r A_\chi^{\hat{c}} - A_r^{\hat{b}}\partial_\chi A_\chi^{\hat{c}}\right)\\
F_\chi^{\hat{a}} &= \epsilon^{\hat{a}\hat{b}\hat{c}}\left(2A_r^{\hat{b}}\partial_r A_\chi^{\hat{c}} - A_r^{\hat{b}}\partial_\chi A_r^{\hat{c}} - A_\chi^{\hat{b}}\partial_r A_r^{\hat{c}}\right)\\
G_r^{\hat{a}} &= \left(1-\chi^2\right)\epsilon^{\hat{a}\hat{b}\hat{c}}E^{\hat{b}}A_\chi^{\hat{c}} + 2\chi E^{\hat{a}}\\
G_\chi^{\hat{a}} &= \epsilon^{\hat{a}\hat{b}\hat{c}}A_r^{\hat{b}}E^{\hat{c}}\\
E^{\hat{a}} &= \epsilon^{\hat{a}\hat{b}\hat{c}}A_r^{\hat{b}}A_\chi^{\hat{c}}
\end{aligned}$$

Podemos agora calcular a densidade de energia contida no campo de Yang-Mills de cada cor:

$$\rho^{\hat{a}} = -\frac{1-\chi^2}{r^2}\left[\left(\partial_\chi A_r^{\hat{a}} - \partial_r A_\chi^{\hat{a}} - E^{\hat{a}}\right)^2 + \left(N_\chi^{\hat{a}}\right)^2\right] - \left(N_r^{\hat{a}}\right)^2 \tag{4.12}$$

Do mesmo modo as densidades de momento, de cada cor, são

$$J_r^{\hat{a}} = 2\,\frac{1-\chi^2}{r^2}\Big(\partial_\chi A_r^{\hat{a}} - \partial_r A_\chi^{\hat{a}} - E^{\hat{a}}\Big)N_\chi^{\hat{a}} \tag{4.13}$$

$$J_{\chi}^{\hat{a}} = -2\left(\partial_{\chi}A_{r}^{\hat{a}} - \partial_{r}A_{\chi}^{\hat{a}} - E^{\hat{a}}\right)N_{r}^{\hat{a}} \tag{4.14}$$

As densidades energia e de momentos totais podem ser calculadas pela simples soma das densidades individuais

$$\rho = \rho^{\hat{1}} + \rho^{\hat{2}} + \rho^{\hat{3}}$$
$$J_i = J_i^{\hat{1}} + J_i^{\hat{2}} + J_i^{\hat{3}}$$

A condição de conservação de energia, $\nabla_{\mu}T^{\mu 0} = 0$ pode ser escrita como

$$\partial_t \rho + \partial_r J_r + \frac{2}{r}J_r + \partial_{\chi}J_{\chi} = 0$$

onde

$$\partial_t \rho = \partial_t \rho^{\hat{1}} + \partial_t \rho^{\hat{2}} + \partial_t \rho^{\hat{3}}$$

sendo que

$$\partial_t \rho^{\hat{a}} = 2\,\frac{1-\chi^2}{r^2}\left[\left(\partial_{\chi}A_{r}^{\hat{a}} - \partial_{r}A_{\chi}^{\hat{a}} - E^{\hat{a}}\right)H^{\hat{a}} - N_{\chi}^{\hat{a}}\,\partial_t N_{\chi}^{\hat{a}}\right] - 2N_{r}^{\hat{a}}\,\partial_t N_{r}^{\hat{a}}$$

onde definimos

$$H^{\hat{a}} = \epsilon^{\hat{a}\hat{b}\hat{c}}\,A_{r}^{\hat{b}}\,N_{\chi}^{\hat{c}} - \epsilon^{\hat{a}\hat{b}\hat{c}}\,A_{\chi}^{\hat{b}}\,N_{r}^{\hat{c}} + \partial_{\chi}\left(N_{\chi}^{\hat{a}} - N_{r}^{\hat{a}}\right)$$

Equações para o Campo Abeliano

Um caso particular que vamos estudar com algum cuidado são as soluções Abelianas, veja Seção 2.6. Nesse caso as componentes associadas a duas das cores do campo de Yang-Mills são identicamente nulas, $A^{\hat{2}} = A^{\hat{3}} = 0$. Com essa restrição os vínculos associados a essas cores são trivialmente satisfeitos e o vínculos restante é escrito como

$$\partial_r\left(r^2 N_r^{\hat{1}}\right) = -\partial_{\chi}\left[\,(1-\chi^2)N_{\chi}^{\hat{1}}\,\right] \tag{4.15}$$

enquanto que as equações de movimento tornam-se

$$r^2\partial_t N_r^{\hat{1}} = \partial_{\chi}\left[\,(1-\chi^2)\partial_{\chi}A_r^{\hat{1}}\,\right] - \partial_{\chi}\left[\,(1-\chi^2)\partial_r A_{\chi}^{\hat{1}}\,\right] \tag{4.16}$$

$$\partial_t N_{\chi}^{\hat{1}} = \partial_{rr}A_{\chi}^{\hat{1}} - \partial_{r\chi}A_r^{\hat{1}} \tag{4.17}$$

Podemos também calcular a densidade de energia para esse caso particular

$$\rho^{\hat{1}} = -\frac{1-\chi^2}{r^2}\left[\left(\partial_{\chi}A_r^{\hat{1}} - \partial_r A_{\chi}^{\hat{1}}\right)^2 + \left(N_{\chi}^{\hat{1}}\right)^2\right] - \left(N_r^{\hat{1}}\right)^2 \tag{4.18}$$

Assim como as densidades de momento

$$J_r^{\hat{1}} = 2\,\frac{1-\chi^2}{r^2}\left(\partial_{\chi}A_r^{\hat{1}} - \partial_r A_{\chi}^{\hat{1}}\right)N_{\chi}^{\hat{1}} \tag{4.19}$$

$$J_{\chi}^{\hat{1}} = -2\left(\partial_{\chi}A_r^{\hat{1}} - \partial_r A_{\chi}^{\hat{1}}\right)N_r^{\hat{1}} \tag{4.20}$$

4.4 Condição de Regularidade em Coordenadas Esféricas

O ponto $r = 0$ é uma singularidade do sistema de coordenadas esféricas. Porém fisicamente ele é um ponto ordinário. Além disso, esse ponto é uma fronteira do domínio numérico. Temos então que determinar que condição de fronteira será imposta numericamente na região que corresponde a $r = 0$ para que as soluções sejam regulares nesse ponto.

Vamos impor uma condição de regularidade na origem. Para determinarmos essa condição vamos considerar um campo vetorial suave em coordenadas cartesianas. Determinamos, então, qual a condição sobre esse vetor para que ele seja axialmente simétrico. De posse de um campo vetorial axialmente simétrico e suave em coordenadas cartesianas, podemos transforma-lo para coordenadas esféricas. Analisando as componentes desse vetor podemos determinar a condição de regularidade para o campo vetorial axialmente simétrico em coordenadas esféricas.

A condição de simetria axial é escrita de modo invariante como $\mathcal{L}_\xi V = 0$ onde ξ é o vetor de Killing. Em coordenadas cartesianas e nas proximidades do eixo de simetria podemos escrever ξ como

$$\xi = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$$

Então a equação de simetria para um vetor é

$$\mathcal{L}_\xi V_\mu = \xi^\nu \partial_\nu V_\mu + (\partial_\mu \xi^\nu) V_\nu = 0$$

com alguma manipulação podemos reescrever essa condição como

$$\mathcal{L}_\xi V_\mu = -y\partial_x V_\mu + x\partial_y V_\mu - \delta^y_\mu V_x + \delta^x_\mu V_y = 0$$

Essa condição vetorial pode ser escrita em termos de suas componentes como o sistema

$$\begin{aligned}
y\partial_x V_x - x\partial_y V_x &= V_y \\
y\partial_x V_y - x\partial_y V_y &= -V_x \\
y\partial_x V_z - x\partial_y V_z &= 0 \\
y\partial_x V_t - x\partial_y V_t &= 0
\end{aligned}$$

Uma solução para esse sistema pode ser facilmente obtida. Pode-se verificar que as componentes de um vetor axialmente simétrico em coordenadas cartesianas são

$$\begin{aligned} V_t &= g_0(t, x^2+y^2, z) \\ V_x &= x g_1(t, x^2+y^2, z) + y g_2(t, x^2+y^2, z) \\ V_y &= y g_1(t, x^2+y^2, z) - x g_2(t, x^2+y^2, z) \\ V_z &= g_3(t, x^2+y^2, z) \end{aligned}$$

Como queremos que o campo vetorial seja suave na origem as funções g_i devem ser funções finitas e suaves quando $x^2+y^2+z^2 \approx 0$.

Porém estamos interessados na condição de regularidade para um vetor no sistema de coordenadas (r, χ, φ), que está relacionado com o sistema de coordenadas esférico pela expressão $\chi = \cos\theta$. Transformando o vetor para esse sistema de coordenadas obtemos as componentes

$$\begin{aligned} V_r &= r\left(1-\chi^2\right) g_1 + \chi\, g_3 \\ V_\varphi &= -r^2\left(1-\chi^2\right) g_2 \\ V_\chi &= -r^2\chi\, g_1 + r\, g_3 \end{aligned}$$

onde $g_\mu = g_\mu(t, r^2, r\chi)$ são funções suaves.

Podemos agora observar como estas componentes se comportam nas proximidades da origem. Levando em conta que as funções g são suaves e finitas podemos avaliar os limites quando $r \to 0$

$$\lim_{r\to 0} V_r = \chi\, g_3(t, 0, 0) \tag{4.21}$$

$$\lim_{r\to 0} V_\varphi = \lim_{r\to 0} V_\chi = 0 \tag{4.22}$$

Aplicando essas condições a conexão do campo de Yang-Mills axialmente simétrico temos que

$$\lim_{r\to 0} A^{\hat{a}}_r = \chi\, f^{\hat{a}}(t, 0) \tag{4.23}$$

$$\lim_{r\to 0} \phi^{\hat{a}} = 0 \tag{4.24}$$

onde $f^{\hat{a}}$ são funções suaves de t.

5 Métodos Numéricos

Nesse capítulo discutimos os métodos para a evolução numérica dos campos de Yang-Mills com simetria axial em um espaço-tempo de Minkowski. Fazemos uma comparação entre métodos de diferenças finitas e um método pseudo-espectral. Os métodos pseudo-espectrais foram muito bem sucedidos em várias aplicações de computação científica [11]. O método mais simples de diferenças finitas produziu resultados significantemente melhores quando aplicado a equações mais complexas.

Métodos de diferenças finitas para a solução de equações diferenciais parciais são bem conhecidos. Em qualquer livro de métodos numéricos encontram-se, por exemplo, os coeficientes para as diferenças finitas em uma malha de pontos distribuídos regularmente [16]. Em alguns livros pode-se encontrar os coeficientes para alguns outros casos particulares. Porém, o algoritmo geral que permite o cálculo desses coeficientes para uma distribuição qualquer de pontos é pouco conhecido e vamos apresentá-lo aqui.

Depois dos métodos de diferenças finitas apresentamos com mais detalhes o método pseudo-espectral por ser menos conhecido. Incluímos uma discussão sobre sua convergência e sua implementação computacional.

Em seguida, listamos os métodos de evolução temporal e integração espacial utilizados nesse trabalho. Por fim, apresentamos alguns resultados obtidos com os métodos de diferenças finitas e pseudo-espectral como forma de comparação entre os métodos.

5.1 Método de Diferenças Finitas

O objetivo desse método é obter uma aproximação para a derivada de uma função f amostrada em um conjunto de pontos. Como exemplo, vamos considerar uma partição qualquer, $-1 < x_0 < x_1 < \cdots < x_n = 1$ onde n é o número de sub-intervalos. Denotaremos os valores de f nos pontos x_i por f_i, isso é, $f_i = f(x_i)$.

Para o caso de pontos igualmente espaçados, podemos obter uma aproximação para a derivada f' com até três pontos a partir de uma expressão geral

$$\left.\frac{df}{dx}\right|_{x=x_i} + O(\delta x^N) \approx f'_i = af_{i-1} + bf_i + cf_{i+1} \tag{5.1}$$

onde a, b e c são os coeficientes apropriados para a aproximação de ordem N.[9]

Por exemplo, para o interior da malha, as expressões

$$f'_i \;=\; \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}}, \;=\; \frac{n}{4}\Big(f_{i+1} - f_{i-1}\Big), \tag{5.2}$$

produzem aproximações de diferenças finitas de segunda ordem.

Nos extremos do intervalo é necessário usar fórmulas que considerem apenas os pontos contidos na discretização. Nesses casos são empregadas as fórmulas

$$f'_0 \;=\; \frac{n}{4}\Big(-3f_0 + 4f_1 - f_2\Big),$$

$$f'_n \;=\; \frac{n}{4}\Big(f_{n-2} - 4f_{n-1} + 3f_n\Big).$$

Esse é um esquema muito usado e que tem produzido bons resultados na solução numérica de equações diferenciais [9]. Porém, a restrição de que os pontos sejam igualmente espaçados não é intrínseca ao método. É apenas uma conseqüência da dificuldade em se calcular esquemas para situações mais gerais. O mesmo vale para a ordem de aproximação do método.

Um algoritmo mais geral para a obtenção de fórmulas de diferenças finitas [11] é apresentado a seguir. Esse algoritmo permite que os pesos, ou coeficientes, de esquemas de diferenças finitas sejam calculados para distribuições genéricas de pontos permitindo assim que o método seja aplicado a qualquer distribuição de pontos e com qualquer ordem de aproximação N devidamente generalizada.

Usamos esse método mais geral para obter fórmulas de derivação por diferenças finitas sobre um conjunto específico pontos. Essas fórmulas permitiram que os resultados obtidos por diferenças finitas e pelos métodos pseudo-espectrais pudessem ser comparados diretamente.

Esse algoritmo consiste em um método para obter pesos $C^k_{i,j}$, tais que as aproximações

$$\left.\frac{d^k f}{dx^k}\right|_{x=\xi} \approx \sum_{j=0}^{i} C^k_{i,j} f(x_j) \qquad\qquad k = 0, \ldots, m \qquad i = k, \ldots, n$$

sejam exatas para todos os polinômios até a mais alta ordem possível. Os pontos $x_0, \ldots, x_n$ são pontos arbitrários não repetidos, onde a função f está amostrada. O ponto ξ é onde queremos avaliar a aproximação da derivada de f, não é necessário que esse ponto coincida com um dos pontos da malha. O número m é a ordem da maior derivada que desejamos calcular. E $i+1$ é o número de pontos usados para a aproximação. Obviamente $i \leq m$ e os coeficientes são tais que a ordem da aproximação seja o índice i [11].

Apresentamos agora o pseudo-código desse algoritmo. Todas as quantidades não inicializadas explicitamente têm valores iniciais nulos.

dados $x_1, \ldots, x_n$, ξ **e** m

$C^0_{0,0} = 1$

$\alpha = 1$

para $i = 1$ **até** n

  $\beta = 1$

  para $j = 0$ **até** $i - 1$

    $\beta = \beta(x_i - x_j)$

    para $k = 0$ **até** $\min(i, j)$

      $$C^k_{i,j} = \frac{(x_i - \xi)\, C^k_{i-1,j} - k\, C^{k-1}_{i-1,j}}{x_i - x_j}$$

    fim

  fim

  para $k = 0$ **até** $\min(i, m)$

    $$C^k_{i,j} = \frac{\alpha}{\beta}\Big(k\, C^{k-1}_{i-1,j-1} - (x_{i-1} - \xi)\, C^k_{1-i,j-1}\Big)$$

  fim

  $\alpha = \beta$

fim

O cálculo dos pesos é numericamente estável, o que permite que sejam calculados esquemas de diferenças finitas para derivadas de qualquer ordem desejada. Entretanto, esse esquema acumula erros de truncamento numérico para derivadas de alta ordem. Em outras palavras, o esquema usa matrizes inversíveis em que a razão entre o maior e o menor módulo dos seus auto-valores cresce com a ordem da derivada.

O caso especial $m = 0$ consiste de uma forma rápida para a avaliação de uma inter-

polação polinomial em um único ponto, ξ.

A seguir apresentamos os princípios nos quais a construção desse algoritmo se baseia.

Considere a aproximação polinomial da k-ésima derivada de f no ponto ξ. Essa aproximação é obtida pela derivação do polinômio interpolador, P_n, de grau $n \leq k$ da função nos $n+1$ pontos da malha:

$$\left.\frac{d^k f(x)}{dx^k}\right|_{x=\xi} \approx \left.\frac{d^k P_n(x)}{dx^k}\right|_{x=\xi} = \sum_{j=0}^{n} \left.\frac{d^k \Phi_j(x)}{dx^k}\right|_{x=\xi} f(x_j)$$

onde Φ_j são os polinômios interpoladores de Lagrange de grau j associados à malha, assim definidos: $\Phi_j(x_i) = 1$ se $i = j$ e $\Phi_j(x_i) = 0$ se $i \neq j$.

Assumindo por simplicidade que $\xi = 0$ e considerando também as aproximações obtidas pelos polinômios que interpolam os sub-conjuntos $\{x_j,\ j = 0, \ldots, i\}$ com $i = 0, \ldots, n$, podemos escrever

$$\left.\frac{d^k P_i(x)}{dx^k}\right|_{x=0} = \sum_{j=0}^{i} \left.\frac{d^k \Phi_{i,j}(x)}{dx^k}\right|_{x=0} f(x_j).$$

Note que nessa expressão representamos os polinômios de Lagrange por $\Phi_{i,j}$. Essa notação representa o j-ésimo polinômio do conjunto de polinômios que interpolam os pontos $\{x_j,\ j = 0, \ldots, i\}$. Deste modo o conjunto de pesos procurado é definido por

$$C_{i,j}^k = \left.\frac{d^k \Phi_{i,j}(x)}{dx^k}\right|_{x=0}$$

que são os termos da expansão de Taylor para o polinômio, isto é,

$$\Phi_{i,j}(x) = \sum_{k=0}^{i} \frac{C_{i,j}^k}{k!} x^k. \tag{5.3}$$

Por outro lado sabemos que $F_{i,j}$ é definida pela expressão

$$\Phi_{i,j}(x) = \frac{\prod_{\substack{l=0 \\ l \neq j}}^{i} (x - x_l)}{\prod_{\substack{l=0 \\ l \neq j}}^{i} (x_j - x_l)}$$

Portanto, o problema de obter os pesos $C_{i,j}^k$ é equivalente ao de rearranjar os termos da expressão acima, agrupando os termos de mesma potência de x.

Uma forma conveniente para realizar esse rearranjo é utilizando as seguintes relações de recorrência

$$\Phi_{i,j} = \frac{x - x_i}{x_j - x_i}\Phi_{i-1,j}(x)$$

$$\Phi_{i,i} = \frac{\prod_{l=0}^{i-2} x_{i-1} - x_l}{\prod_{l=0}^{i-1} x_i - x_l}(x - x_{i-1})\Phi_{i-1,i-1}(x)$$

Substituindo a expressão (5.3) nessas relações, obtém-se relações de recorrência para os pesos partindo de $C^0_{0,0} = 1$. O algoritmo simplesmente implementa essa relação de recorrência.

O método de diferenças finitas é bastante conhecido e utilizado. Sua principal vantagem é a simplicidade, tanto para o tratamento teórico, quanto para a implementação computacional. Uma de suas desvantagens é a dificuldade em aplicá-lo a discretizações não uniformes. Esse algoritmo, entretanto, permite que o método seja aplicado em discretizações mais gerais.

5.2 Métodos Pseudo-Espectrais

A formulação dos métodos espectrais foi apresentada originalmente no artigo de Gottlieb e Orszag [12]. O livro texto de Canuto et. al. [5] foca-se em algoritmos para mecânica de fluidos e inclui além dos aspectos práticos, importantes informações teóricas sobre métodos espectrais. Mais recentemente Fornberg [11] publicou um livro tratando unicamente dos métodos pseudo-espectrais e posteriormente Trefethen [19] publicou um livro sobre a implementação desses métodos em Matlab e Weideman [20] disponibilizou uma suite de programas, em Matlab que calculam as matrizes de derivação necessárias para a implementação desse método.

Os métodos espectrais podem ser divididos em dois grandes grupos: os pseudo-espectrais, ou métodos de colocação, e os métodos modais, também conhecidos como do tipo Galerkin. Nos métodos do primeiro grupo, os resíduos são computados apenas nos pontos da malha e os coeficientes são obtidos impondo que o resíduo seja nulo nesses pontos. No segundo grupo o resíduo é ponderado segundo funções de teste e o resultado da integração é igualado a zero. Uma diferença importante entre esses métodos é que, nos métodos de colocação, calcula-se diretamente os valores desejados sobre os pontos nodais, enquanto

que, nos métodos tipo Galerkin calcula-se os coeficientes de uma soma, sendo portanto necessário somar para obter as grandezas desejadas nos pontos nodais.

A convergência dos métodos espectrais ou pseudo-espectrais é exponencial, semelhante à convergência do método de Fourier. Essa propriedade é uma conseqüência direta da teoria do problema de Sturm-Liouville. A taxa de convergência não é fixa e está ligada à regularidade da solução. A convergência exponencial, ou espectral, para uma função suave, implica que se o número de pontos de colocação, ou nós da malha, for dobrado, a magnitude do erro decresce ao menos duas ordens de grandeza. Entretanto, essa rapidez na convergência pode ser facilmente perdida se a solução for pouco suave ou o domínio irregular.

O formalismo dos resíduos ponderados pode ser usado para construir vários métodos numéricos. Diferentes escolhas de funções de base e de teste levam a diferentes métodos. Vamos aplicar esse formalismo para apresentar os métodos pseudo-espectrais. Nesse contexto procura-se uma solução aproximada para uma equação diferencial

$$\mathcal{L}[\ u\ ](x) = 0 \qquad\qquad x \in \Omega$$

sujeita a condições iniciais e de fronteira adequadas. O método dos resíduos ponderados busca esta aproximação em um sub-espaço do espaço de funções ao qual u pertence. Essa aproximação, $\bar{u}$, pode ser construida como uma combinação linear das funcões da base desse sub-espaço

$$\bar{u}(x) = \sum_{i=0}^{n} \alpha_i \Phi_i(x) \tag{5.4}$$

onde Φ_i são as funções da base do sub-espaço considerado. Em geral $\bar{u}$ não pode satisfazer completamente a equação diferencial original. Aplica-se, então, uma forma fraca da equação diferencial

$$\int_\Omega \mathcal{L}[\ \bar{u}\ ](x)\Psi_j(x)\, dx = 0 \qquad\qquad j = 1, \cdots, n \tag{5.5}$$

as funções Ψ_j são chamadas de funções de teste.

Para cada escolha das funções Φ e Ψ temos uma classe diferente de métodos de aproximação. Nos métodos espectrais as funções Φ são funções globais que satisfazem as condições de fronteira da equação diferencial e ou têm suporte compacto. As funções de teste Ψ são escolhidas iguais às funções da base Φ_i.

Por outro lado, os métodos de colocação são construidos ao escolhermos deltas de Dirac como funções de teste

$$\Psi_i(x) = \delta(x - x_i)$$

Com essa escolha o sistema (5.5) se reduz a imposição que o resíduo seja nulo nos pontos amostrados, x_i,

$$\mathcal{L}[\ \bar{u}\ ]\ \Big|_{x=x_i} = 0 \qquad\qquad j = 0, \ldots, n \tag{5.6}$$

O número n de pontos nodais x_i deve coincidir com o número de funções da base para que o número de equações seja o mesmo que o número de incógnitas.

Os métodos pseudo-espectrais são métodos de colocação combinados com funções de base globais. Tipicamente as funções da base são escolhidas como funções trigonométricas ou polinomiais. As funções trigonométricas são especialmente adequadas para resolver problemas com condições de fronteira periódicas. E as funções polinomiais são empregadas na solução de problemas com condições de fronteira mais gerais. Por esse motivo vamos trabalhar apenas com bases polinomiais. Essa base é composta por polinômios interpoladores nos pontos da malha, de modo que, $\Phi_i(x)$ seja igual a unidade quando $x = x_i$ e igual a zero quando $x = x_j$, com $j \neq i$.

5.3 Convergência dos Métodos Pseudo-Espectrais

Na seção anterior apresentamos os métodos espectrais, nessa seção vamos discutir a sua convergência. Na maior parte desse trabalho utilizamos funções de base polinomiais, por isso discutiremos apenas esse caso. A principal questão que temos que abordar é como uma função e suas derivadas podem ser aproximadas por uma interpolação polinomial.

Sabe-se que se os pontos de interpolação forem igualmente espaçados o polinômio interpolador apresentará o fenômeno de Runge [11] Quando o número de pontos cresce os valores do polinômio tendem ao infinito nos extremos do intervalo, ou seja, o polinômio interpolador não converge. Entretanto, se os pontos de interpolação forem as raízes do polinômio de Chebyshev esse fenômeno é minimizado e o polinômio converge.

Apresentamos a seguir um teorema que aborda essa questão e justifica a afirmação de que os métodos espectrais convergem exponencialmente.

Teorema 5.1 *Considere uma função analítica f e um polinômio P_n interpolante em n pontos no intervalo $[-1, 1]$. Se $\mu(x)$ for a função de densidade dos pontos de interpolação no intervalo $[-1, 1]$ podemos formar a função potencial*

$$\phi(z) = -\int_{-1}^{1} \mu(x) \ln|z - x| dx \tag{5.7}$$

Então:

1. *O polinômio $P_n(z)$ converge para $f(z)$ dentro da maior curva equi-potencial de $\phi(z)$ que não contenha nenhuma singularidade da função $f(z)$ e diverge fora dessa curva.*

2. *A taxa de convergência, ou divergência, é exponencial, isso é, se comporta como $\alpha(z)^n$ onde*

   $$\alpha(z) = e^{\phi(z_0 - \phi(z))}$$

   onde z_0 representa um ponto sobre a maior curva equi-potencial que não contém singularidades de $f(z)$. Esse resultado pode ser entendido do mesmo modo como a série de Taylor converge, ou diverge, de acordo com $\alpha(z)^n$ com

   $$\alpha(z) = \left| \frac{z}{z_0} \right|$$

3. *A aproximação de qualquer derivada de $f(z)$ pelo método pseudo-espectral converge, ou diverge, do mesmo modo que o polinômio interpolante.*
   Para demonstrar esse teorema considere que a função $f(z)$ seja analítica em um vizinhança do intervalo $[-1, 1]$. O único polinômio interpolante $P_n(z)$ de $f(z)$ pode ser escrito como

   $$P_n(x) = \sum_{k=0}^{n} f(x_k)\Phi_k(x)$$

   onde as funções $\Phi_k(x)$ são dadas pela fórmula

   $$\Phi_k(x) = \frac{\prod\limits_{\substack{j=0 \\ j\neq k}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j\neq k}}^{n} (x_k - x_j)} \qquad k = 0, \ldots, n \tag{5.8}$$

   Definindo a função

   $$w(x) = \prod_{j=0}^{n} (x - x_j)$$

   podemos reescrever os polinômios Φ_k como

   $$\Phi_k(x) = \frac{w(x)}{(x - x_k)\, w'(x_k)} \qquad k = 0, \ldots, n$$

O erro da interpolação, definido como $R_n(z) = f(z) - P_n(z)$, pode ser escrito como

$$R_n(z) = f(z) - \sum_{k=0}^{n} \frac{w(z) f(x_k)}{(z - x_k) w'(x_k)}$$

Usando agora o cálculo de resíduos, temos que

$$R_n(z) = \frac{w(z)}{2\pi i} \int_C \frac{f(t)}{w(t)(t - z)} dt$$

onde o contorno C engloba o intervalo $[-1, 1]$ e o ponto z, mas não engloba nenhuma singularidade de $f(z)$.

Nosso objetivo é estimar o erro $R_n(z)$, para isso precisamos estimar a função $w(z)$. Primeiramente vamos avaliar o módulo dessa função

$$|w(z)| = \prod_{k=0}^{n} |z - x_k| = \exp\left[\sum_{k=0}^{n} \ln |z - x_k| \right]$$

Notando que, tomando o limite $n \to \infty$ e usando a definição (5.7), temos

$$\frac{1}{n} \sum_{k=0}^{n} \ln |z - x_k| \to \int_{-1}^{1} \mu(x) \ln |z - x| dx = -\phi(z)$$

Para um $\epsilon > 0$ arbitrário, selecionamos a curva C como uma curva equipotencial da função $\phi(z)$, tal que,

$$\phi(z) > \phi(z_0) + \epsilon \qquad\qquad \forall z \in C$$

isso é, C é uma curva interior a curva com a singularidade de $f(z)$ mais próxima, z_0. Temos então

$$\left| \int_C \frac{f(t)\, dt}{w(t)(t - z)} \right| \leq \frac{1}{|w(z)|_{z \in C}} \left[\int_C \left| \frac{f(t)}{t - z} \right| dt \right] \leq c(\epsilon) \exp\left[n \left(\phi(z_0) + \epsilon \right) \right]$$

portanto

$$|R_n(z)| \leq c(\epsilon) \exp\left(n \left[\left(\phi(z_0) + \epsilon \right) - \phi(z) \right] \right)$$

Não podemos melhorar essa estimativa usando $\phi(z_0) - \epsilon$, no lugar de $\phi(z_0) + \epsilon$, pois nesse caso $|R_n(z)| \to 0$ quando $z = z_0$ mas $f(z)$ é singular.

Rearranjando a última expressão e considerando o limite $n \to \infty$ temos que

$$|R_n(z)|^{1/n} \to e^{-\left(\phi(z) - \phi(z_0) \right)}$$

Esse argumento se aplica apenas para valores de z dentro da curva c particular, que foi usada para obter a estimativa. Se z estiver fora do contorno que contém z_0 então $R_n(z)$ diverge. A taxa de divergência é dada pela mesma expressão, pois fica irrelevante considerar o resíduo em $t = z$.

A terceira parte do teorema pode ser facilmente mostrada, considerando-se que $P'_n(z)$ interpola $f'(z)$ em n pontos no intervalo $[-1, 1]$.

Como ilustração desse resultado aplicaremos esse teorema a algumas distribuições de pontos de interpolação. Consideraremos primeiro a interpolação com pontos igualmente espaçados. Neste caso, a densidade de pontos no intervalo $[-1, 1]$ é dada por $\mu = 1/2$, portando a integral (5.7) pode ser avaliada analiticamente. A função potencial ϕ, para esse caso, é dada pela expressão

$$\phi(z) = -\frac{1}{2}\,\mathrm{re}\Big[(1-z)\ln(1-z) - (-1-z)\ln(-1-z)\Big] \tag{5.9}$$

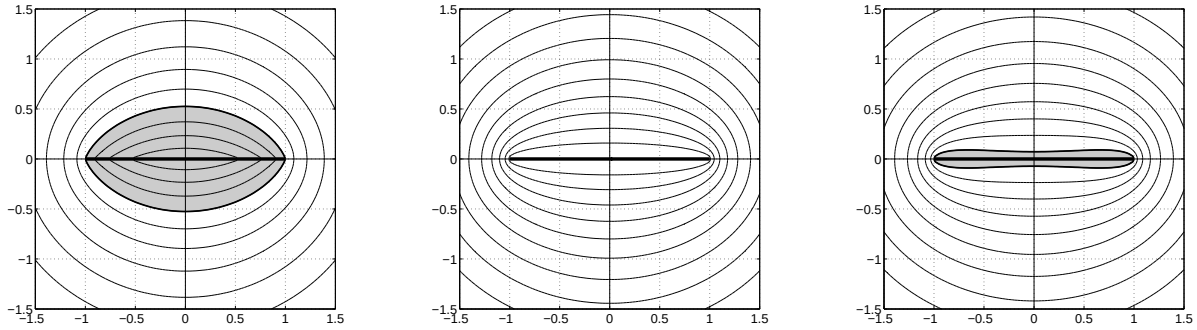


Figura 5.1: *Função potencial $\phi(z)$ para interpolação polinomial no intervalo $[-1, 1]$ considerando n pontos: (a) pontos igualmente espaçados; (b) raízes do polinômio de Chebyshev de grau n ; (c) pontos com densidade $\mu_{0,7}$*

O primeiro gráfico da Figura 5.1 mostra as curvas de nível dessa função. A região em cinza corresponde ao conjunto de pontos que não podem conter singularidades para que a seqüência seja convergente. Se a função a ser interpolada apresentar qualquer singularidade dentro dessa região, sua interpolação apresentará o fenômeno de Runge [11].

Todos os polinômios de Jacobi [1] possuem a mesma densidade de pontos:

$$\mu(x) = \frac{1}{\pi\sqrt{1-x^2}} \tag{5.10}$$

Essa coincidência na função de densidade dos polinômios de Jacobi significa que quando $n \to \infty$ as diferenças entre eles se anulam. Porém, para um n finito, a densidade (5.10) descreve apenas a densidade dos polinômios de Chebyshev. Também neste caso a integral (5.7) pode ser avaliada analiticamente. A função potencial resultante é

$$\phi(z) = -\ln\left|z + \sqrt{z^2 - 1}\,\right|$$

O segundo gráfico da Figura 5.1 mostra as curvas de nível dessa função. Note que, nesse gráfico, a região cinza coincide com o intervalo $[-1, 1]$. A interpretação dessa característica é que os polinômios de Jacobi, e em particular os polinômios de Chebyshev, são os menos exigentes em termos de suavidade da função a ser interpolada. Eles exigem apenas que a função não contenha singularidades no próprio intervalo $[-1, 1]$. Enquanto que outras distribuições de pontos vão exigir que a extensão complexa da função também não contenha singularidades.

As funções de densidade para as raízes do polinômio de Chebyshev de grau n ou os pontos igualmente espaçados são casos particulares de uma família de funções de densidade dada por

$$\mu_\lambda(x) = \frac{C_\lambda}{(1 - x^2)^\lambda} \tag{5.11}$$

a constante C_λ, para $\lambda < 1$, é dada pela fórmula

$$C_\lambda = \frac{\Gamma\left(\frac{3}{2} - \lambda\right)}{\pi^{\frac{1}{2}}\Gamma(1 - \lambda)}$$

Obtemos pontos igualmente espaçados escolhendo $\lambda = 0$ e os pontos de Chebyshev escolhendo $\lambda = 1/2$. O terceiro gráfico da Figura 5.1 exibe a função (5.11) para $\lambda = 0.7$.

A conclusão é que quando aplicamos o método pseudo-espectral em uma função que seja analítica em uma vizinhança do intervalo $[-1, 1]$, a taxa de convergência é da ordem de α^n. Essa taxa distingue os métodos pseudo-espectrais dos métodos de elementos finitos ou diferenças finitas, pois

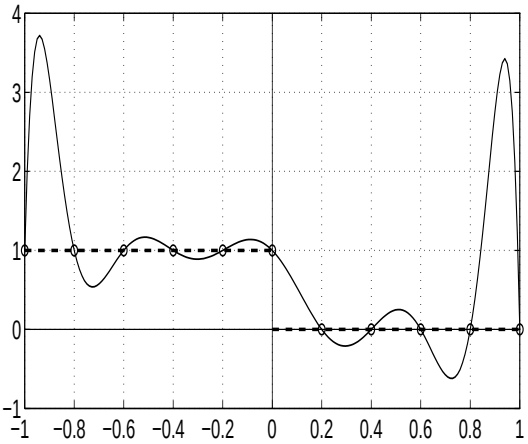


Figura 5.2: *Uma função analítica e uma descontínua indistingüíveis sobre a malha*

para esses métodos a taxa de convergência é da ordem de n^{-p}, onde p é a ordem do método. Assim dizemos que a taxa de convergência n^{-p} é polinomial enquanto que a taxa de convergência α^n dos métodos pseudo-espectrais é exponencial.

Vimos que a convergência dos métodos pseudo-espectrais está diretamente ligada à suavidade da solução. Pórem, numéricamente, a definição de suavidade é ambigua. A Figura 5.2 ilustra esse fato. Nela estão representadas duas funções que coincidem nos pontos de uma malha, marcados por pequenos círculos. Uma dessas funções é a função degrau, que vale 0 se $x \leq 0$ e 1 se $x > 0$. A outra é um polinômio de oitavo grau. Embora ambas as funções sejam idênticas, quando consideramos apenas os valores sobre a malha, uma possui uma descontinuidade tipo salto na origem, enquanto que a outra é analítica em todos os pontos do intervalo.

5.4 Matrizes de Derivação

A matriz de derivação corresponde ao operador de derivação quando estamos manipulando os valores discretizados, sendo muito conveniente tanto para a manipulação teórica quanto para a implementação dos métodos pseudo-espectrais.

Para ilustrar o conceito de matrizes de diferenciação vamos considerar, primeiramente, a derivação numérica de uma função f no intervalo $[-1, 1]$ por diferenças finitas de segunda ordem (5.2). Em seguida apresentamos as matrizes de derivação para os métodos pseudo-espectrais. Podemos representar a aplicação desse esquema de devivação em todos os pontos da malha simultaneamente pela espressão matricial

$$F' = DF$$

onde F é o vetor composto dos valores de f nos pontos x_i e F' é o vetor das aproximações de f' nos mesmos pontos. Nessa representação D é a matrix de derivação empregada, seus elementos são

$$D = \frac{n}{4}\begin{bmatrix} -3 & 1 & -1 & & & & \\ -1 & 0 & 1 & & & & \\ & -1 & 0 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 0 & 1 \\ & & & 1 & -1 & 3 \end{bmatrix}$$

Note que a matriz de derivação é singular. O que está de acordo com a definição da derivada, pois duas funções que sejam diferentes apenas por uma constante, possuem a

mesma derivada. O que implica na matriz de diferenciação possuir um núcleo composto pelos vetores cujas componentes sejam todas iguais.

Para construir a matriz de derivação para os métodos pseudo-espectrais vamos considerar a aproximação de uma função dada pela soma das funções de base (5.4). Essa soma define o polinômio interpolador nos pontos de colocação. Ou seja, para uma função f com valores f_i nos pontos de colocação x_i. A aproximação é determinada pela fórmula

$$\bar{f}(x) = \sum_{i=0}^{n} f_i \Phi_i(x)$$

Para obtermos uma aproximação para a derivada de f vamos derivar a função $\bar{f}$. Pela linearidade do operador de diferenciação temos a seginte aproximação para a derivada de f

$$f'(x) \approx \bar{f}'(x) = \sum_{i=0}^{n} f_i \Phi_i'(x)$$

Como nosso objetivo é empregar o método pseudo-espectral para resolver uma equação diferencial, vamos precisar avaliar a derivada de f apenas nos pontos de colocação. Substituindo os pontos de colocação na aproximação de f' temos

$$f_i' = \bar{f}'(x_i) = \sum_{j=0}^{n} f_j \Phi_j'(x_i)$$

Reescrevendo essa expressão matricialmente obtemos

$$F' = DF$$

onde, como no caso anterior, F é o vetor composto dos valores de f nos pontos x_i e F' é o vetor das aproximações de f' nos mesmos pontos. Porém, nesse caso, as componentes da matriz D são dadas pela fórmula

$$D_{ij} = \Phi_j'(x_i)$$

Repetindo o mesmo processo podemos obter matrizes de derivação para qualquer ordem desejada

$$D_{ij}^{(k)} = \Phi_j^{(k)}(x_i)$$

5.5 Evolução Temporal

Para resolvermos uma equação evolutiva com métodos pseudo-espectrais, podemos discretizar a parte espacial da equação transformando-a em um sistema finito de equações ordinárias no tempo. Considere uma equação, para $f(t, x)$, da forma

$$\frac{\partial f}{\partial t} = \mathcal{L}[f](x)$$

onde $\mathcal{L}$ é um operador diferencial em x. Nesse caso, substituímos as derivadas do operador diferencial por matrizes de derivação, obtendo um sistema de equações ordinárias

$$\frac{dF}{dt} = L(F, x) \tag{5.12}$$

onde F é um vetor em que cada componente é uma função da variável t e representa o valor da função $f(x, t)$ nos pontos de colocação, isso é,

$$F_i(t) = f(t, x_i)$$

Podemos agora aplicar um método de resolução de EDOs para obter os valores de F em diferentes instantes de tempo.

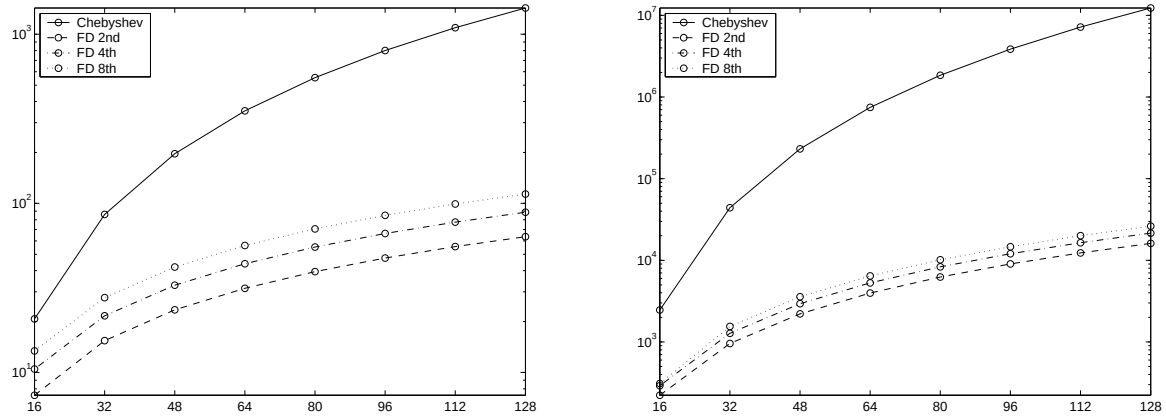
A principal questão ao analisarmos métodos de evolução temporal explícito é sua estabilidade. Se o passo no tempo for instável o erro na evolução temporal cresce exponencialmente inviabilizando o método.

No caso de métodos explícitos para equações diferenciais hiperbólicas a estabilidade do passo no tempo vai impor um vínculo entre a discretização espacial e a temporal que é a condição de CFL [9]. No caso do sistema de equações ordinárias (5.12) esse vínculo vai corresponder à imposição de que os autovalores do operador L estejam na região de estabilidade do método de evolução temporal.

Esta é a maior desvantagem dos métodos pseudo-espectrais. Os autovalores das matrices de diferenciação desses métodos crescem muito mais rápido, em comparação aos da diferenças finitas, quando aumentamos número de pontos de colocação.

Consideremos os $n - 1$ pontos onde um polinômio de Chebyshev de grau n tem valores locais extremos, isto é as raízes das derivadas destes polinômios, mais os pontos de fronteira. Vamos chamar de pontos de Chebyshev estes $n + 1$ pontos.

Note os gráficos da Figura 5.3 onde mostramos mau comportamento dos autovalores das matrizes de diferenciação. Essa característica, provavelmente em conjunto com a não linearidade da equação, inviabilizou o uso dos métodos pseudo-espectrais para resolver,



(a) Primeira derivada (b) Segunda derivada

Figura 5.3: *Comparação do crescimento do módulo dos autovalores das matrizes de derivação do método pseudo-espectral nos pontos de Chebyshev e diferenças finitas de segunda, quarta e oitava ordens*

em alguns casos, as equações de Yang-Mills. Vamos discutir isso com mais detalhes na Seção 5.9.

Uma maneira de evitar esse problema seria o uso de métodos implícitos, ou semi-implícitos, para a evolução temporal. Porém, nas tentativas realizadas para resolver o sistema não-linear, ou linear no caso semi-implícito, o custo computacional tornou as simulações proibitivas. Um caso típico explícito rodava em aproximadamente cinco horas enquanto os testes com implementação implícita foram abortados apos 36 horas sem muito progresso.

Durante o desenvolvimento desse trabalho, vários métodos de evolução foram testados, porém, os melhores resultados foram obtidos com o método de Runge-Kutta de passo adaptativo com a implementação de Cash-Karp [16]. Nesse método, os passos intermediários são construídos de modo que seja possível calcular, simultaneamente, uma aproximação de quarta e uma de quinta ordem, de modo que é possível usar a diferença entre as duas aproximações como uma estimativa do erro local.

Considerando uma equação da forma

$$\frac{d\,y(t)}{dt} = f(t, x)$$

i	a_i	b_{ij}					c_i	$\bar{c}_i$
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{17575}{483884}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{5}{6}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
j		1	2	3	4	5		

Tabela 5.1: *Coeficientes para o método de Runge-Kutta*

e que a solução no tempo t_n é conhecida, podemos calcular a solução no tempo $t_{n+1} = t_n + h$ avaliando os seguintes passos [16]

$$
\begin{aligned}
k_1 &= h\, f(\ t_n, & & x_n\ ) \\
k_2 &= h\, f(\ t_n + a_2 h, & & x_n + b_{21}k_1\ ) \\
k_3 &= h\, f(\ t_n + a_3 h, & & x_n + b_{31}k_1 + b_{32}k_2\ ) \\
k_4 &= h\, f(\ t_n + a_4 h, & & x_n + b_{41}k_1 + b_{42}k_2 + b_{43}k_3\ ) \\
k_5 &= h\, f(\ t_n + a_5 h, & & x_n + b_{51}k_1 + b_{52}k_2 + b_{53}k_3 + b_{54}k_4\ ) \\
k_6 &= h\, f(\ t_n + a_6 h, & & x_n + b_{61}k_1 + b_{62}k_2 + b_{63}k_3 + b_{64}k_4 + b_{65}k_5\ )
\end{aligned}
$$

Com esses valores auxiliares podemos calcular uma aproximação de quinta ordem para a solução em t_{n+1} pela fórmula

$$y_{n+1} = y_n + c_1k_1 + c_2k_2 + c_3k_3 + c_4k_4 + c_5k_5 + c_6k_6$$

Com os mesmos valores intermediários podemos construir uma aproximação de quarta ordem da forma

$$\bar{y}_{n+1} = y_n + \bar{c}_1k_1 + \bar{c}_2k_2 + \bar{c}_3k_3 + \bar{c}_4k_4 + \bar{c}_5k_5 + \bar{c}_6k_6$$

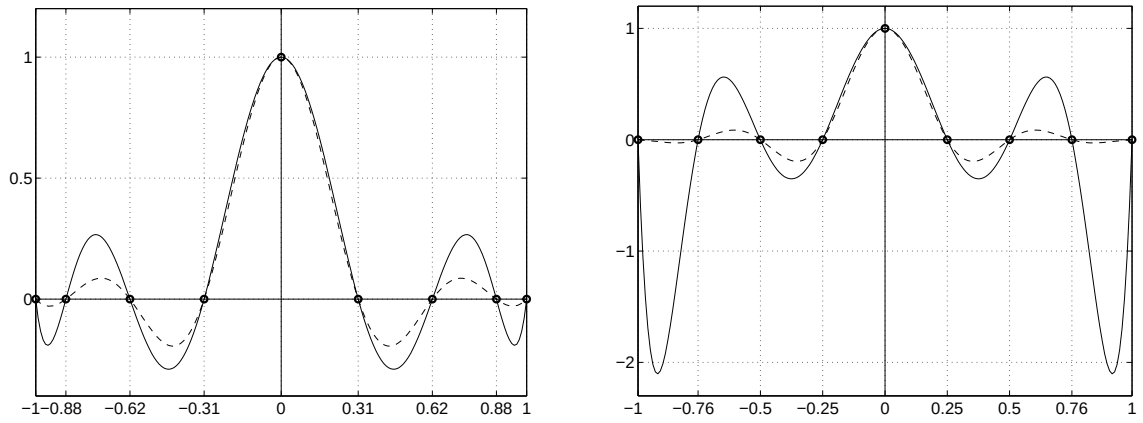
Podemos, então, considerar a diferença entre as duas aproximações como uma estimativa de erro

$$E = y_{n+1} - \bar{y}_{n+1}$$

Os coeficientes dessas formulas estão na Tabela 5.1

5.6 Transformação de Coordenadas

Na seção anterior discutimos sobre os métodos de evolução temporal. E vimos que o rápido crescimento dos autovalores das matrizes de derivação dos métodos pseudo-espectrais são um das suas principais desvantagens. Esse crescimento se deve ao grande acúmulo de pontos nos extremos do intervalo. Uma forma de tentar reduzir esse efeito é aplicar uma transformação na coordenada independente.



(a) $\alpha = 0.9$ (b) $\alpha = 0.9999$

Figura 5.4: *Comparação entre o polinômio interpolador de Lagrange (linha tracejada) e as bases não polinomiais (linha contínua) nos pontos de Chebyshev transformados para dois casos de parâmetros.*

Uma transformação de coordenadas proposta com o intuito de melhorar a distribuição dos autovalores das matrizes de diferenciação [5] é o seguinte

$$x_i = \frac{\arcsin(\alpha y_i)}{\arcsin \alpha}, \quad i = 0, 1, \ldots, n \tag{5.13}$$

nessa expresão x_i são os pontos de Chebyshev e α é o parâmetro que determina o espaçamento dos novos pontos, y_i. Quando $\alpha \to 0$ os pontos transformados tendem aos próprios pontos de Chebyshev, porém quando $\alpha \to 1$ os novos pontos tendem a pontos igualmente espaçados no intervalo $[-1, 1]$.

Após essa transformação as funções de interpolação nos pontos não são mais polinomiais. Os gráficos da Figura 5.4 exibem o polinômio interpolador e a função interpoladora para dois conjuntos de pontos transformados. No primeiro gráfico os pontos foram transformados com $\alpha = 0.9$, enquanto que no segundo a transformação foi com $\alpha = 0.9999$. É fácil perceber que no segundo conjunto de pontos o polinômio interpolador já está bastante comprometido pelo fenômeno de Runge.

Essa transformação reduz o crescimento dos autovalores das matrizes de derivação.

5.7 Integração Sobre o Domínio

Várias informações sobre as soluções de uma EDP podem ser obtidas através de sua integral. Em nosso caso é de grande importância avaliar a energia de uma solução em um determinado instante do tempo. Essa informação será utilizada para medirmos a qualidade das soluções, pois o comportamento da energia total é conhecido, e também, para analisarmos o comportamento da solução com base em uma quantidade invariante.

Nas soluções obtidas por diferenças finitas com pontos igualmente espaçados, calculamos as quadraturas pelo método do trapézio repetido. Embora seja um método bastante simples, apresentou resultados satisfatórios.

Nessa tese,nas soluções calculadas pelo método pseudo-espectral, os valores estão amostrados nos pontos de Chebyshev. Embora seja possível usar, também nesse caso, o método dos trapézios repetidos, existe um método melhor adaptado a esses pontos, o método de Clenshaw-Curtis [19]. Nesse método a integral

$$I = \int_{-1}^{1} f(x)\, dx$$

é aproximada pelo somatório

$$I_n = \sum_{i=1}^{n+1} w_i f(x_i)$$

onde x_i são os pontos de Chebyshev, com os quais já estamos trabalhando

$$x_i = \cos\left(\frac{(i-1)\pi}{n}\right) \qquad\qquad i = 1, \ldots, n+1$$

e w_i são os pesos da integração. Esses pesos são definidos determinando que a aproximação I_n seja exata para polinômios de ordem n. Um algoritmo para avaliar esses pesos é descrito a seguir. Para um determinado n constrói-se inicialmente

$$\theta_i = \frac{(i-1)\pi}{n} \qquad x_i = \cos\theta_i \qquad w_i = 0 \qquad v_i = 1$$

para $i = 1, \ldots, n+1$. Em seguida, se n for par os pesos w_i são avaliados pelo algoritmo

$$w_1 = w_{n+1} = \frac{1}{n^2 - 1}$$

para $k = 1$ **até** $\dfrac{n}{2} - 1$

$$v_i = v_i - \frac{2\cos(\,2\,k\,\theta_{i+1}\,)}{4k^2 - 1} \qquad i = 1, \ldots, n-1$$

fim

$$v_i = v_i - \frac{\cos(\,n\,\theta_{i+1}\,)}{n^2 + 1} \qquad i = 1, \ldots, n-1$$

$$w_i = \frac{2v_{i-1}}{n} \qquad i = 2, \ldots, n$$

Enquanto que, se n for ímpar avaliamos w_i pelo algoritmo

$$w_1 = w_{n+1} = \frac{1}{n^2}$$

para $k = 1$ **até** $\dfrac{n-1}{2}$

$$v_i = v_i - \frac{2\cos(\,2\,k\,\theta_{i+1}\,)}{4k^2 - 1} \qquad i = 1, \ldots, n-1$$

fim

$$w_i = \frac{2v_{i-1}}{n} \qquad i = 2, \ldots, n$$

5.8 Implementação

Nas seções anteriores apresentamos os métodos que foram implementados para resolver e analisar as equações de Yang-Mills. Nessa seção discutimos a forma como esses métodos foram implementados. No Apendice B estão as listagens dos programas desnvolvidos.

Para avaliar as matrizes de derivação usamos um toolkit para o Matlab desenvolvido por Weideman e Reddy [20]. Esse toolkit implementa algumas funções que geram as matrizes de derivação para vários interpolantes, além de outras funcionalidades necessárias para a implementação dos métodos pseudo-espectrais.

É possível implementar o produto da matriz de derivação de Chebyshev por um vetor usando FFT. Essa abordagem é mais eficiente quando o número de pontos de colocação é grande. Porém, como nossa implementação foi em Matlab e usamos um número relativamente pequeno de pontos, o ganho seria pequeno.

Para a solução de equações em mais de uma variável espacial, é necessário aplicar o

método pseudo-espectral independentemente em cada uma das dimensões. De modo que teremos uma matriz de derivação para discretizar cada um dos operadores de derivação parcial. Essa construção do método para várias variáveis corresponde a uma multiplicação tensorial das funções de base unidimensionais. Desse modo a aplicabilidade do método fica restrita a domínios que possuam geometrias muito simples.

5.9 Comparação dos Métodos

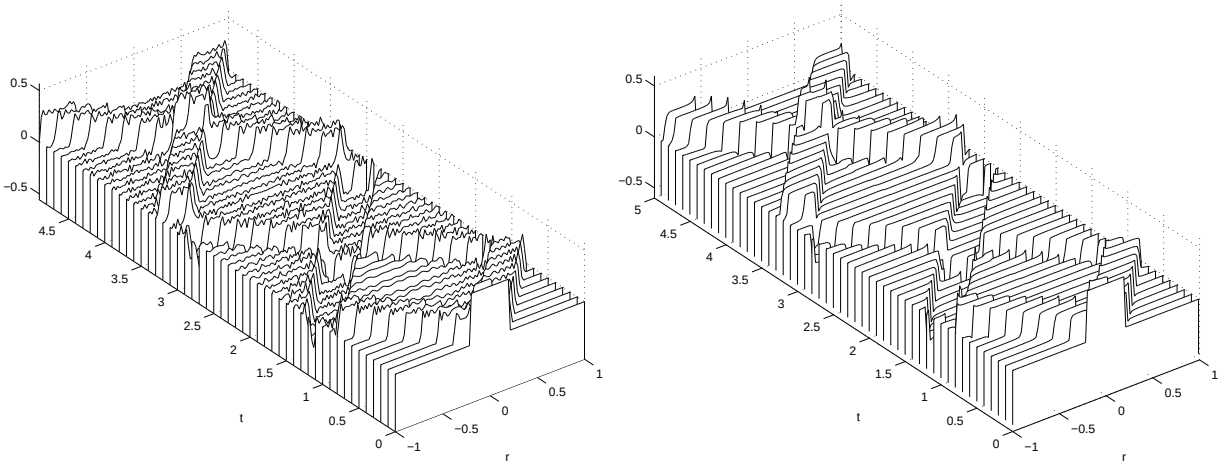
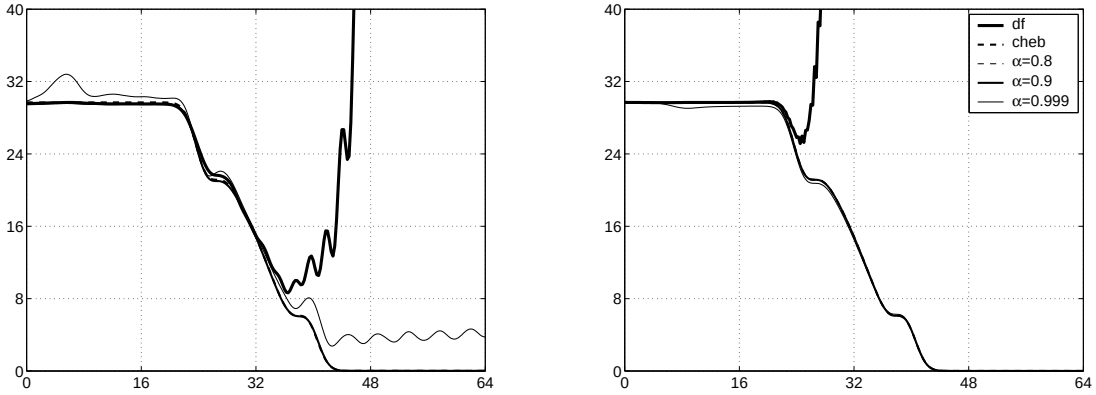


Figura 5.5: *Comparação das soluções da equação de onda linear obtidas por diferenças finitas e pelo método pseudo-espectral*

Embora a teoria seja menos clara quando a solução, ou os coeficientes da equação não são suaves, os métodos pseudo-espectrais também são bem sucedidos nesses casos. Podemos observar esse comportamento na Figura 5.5. Cada gráfico exibe a solução da equação de onda linear obtido por um método. No primeiro, por diferenças finitas e no outro, pelo método pseudo-espectral com pontos de Chebyshev. Em ambos os casos consideramos um domínio limitado com condições de fronteira de Dirichlet. A condição inicial é um pulso quadrado, ou seja, a solução vai conter descontinuidades tipo salto. Em ambos os casos usamos o mesmo número de pontos e o mesmo passo no tempo. É fácil observar que, embora nenhum dos métodos seja capaz de obter a solução desejada, o método pseudo-espectral apresenta uma solução muito menos poluída por modos de alta freqüência.

Entretanto, a equação que estamos interessados em resolver nesse trabalho é consideravelmente mais complexa que a equação de onda linear e as condições de fronteira também não são apenas condições de Dirichlet. Vamos então comparar o desempenho dos métodos



(a) *Malha: 32 × 16 pontos* (b) *Malha: 64 × 32 pontos*

Figura 5.6: *Energia total do campo de Yang-Mills Abeliano ao longo do tempo*

de diferenças finitas e o pseudo-espectral diretamente nas equações de Yang-Mills. Como não possuímos soluções exatas que possam ser utilizadas como gabarito, vamos nos basear no comportamento qualitativo da energia total das soluções.

Como o campo de Yang-Mills é conservativo, a energia total deveria ser constante. Entretanto, o domínio de simulação é finito e em algum momento o pulso de energia deixa esse domínio. Assim, o comportamento esperado para a energia total é permanecer constante pelo tempo em que o pulso estiver totalmente contido no domínio de simulação. Quando o pulso alcançar a fronteira exterior, a energia deve suavemente se anular permanecendo nula depois que o pulso deixar completamente o domínio. Nos Capítulos 6 a 8 discutiremos o comportamento das soluções com mais detalhes.

Um característica importante dos problemas que simulamos é que eles não incluem fontes. Portanto a energia das soluções nunca deve aumentar, isso é, qualquer aumento na energia total é conseqüência direta de erros numéricos.

Legenda	Descrição
df	Diferenças finitas de quarta ordem
cheb	Método pseudo-espectral nos pontos de Chebyshev
α	Método pseudo-espectral nos pontos de Chebyshev transformados segundo a relação (5.13) com o valor para α dado

Tabela 5.2: Descrição dos métodos de discretização espacial comparados

A Figura 5.6 apresenta os gráficos da energia total de um campo de Yang-Mills Abeliano axialmente simétrico com condição inicial da primeira família, veja o Capítulo 6 para obter mais detalhes. O primeiro gráfico exibe a evolução da energia total ao longo do tempo obtida com uma malha de 32 por 16 pontos. Enquanto que, o segundo exibe resultados obtidos com uma malha de 64 por 32 pontos. Cada um desses gráficos exibe a evolução da energia total ao longo do tempo, obtida por vários métodos de discretização espacial. Esses métodos são identificados na legenda do último gráfico e descritos sucintamente na Tabela 5.2.

Observe que a discretização da variável χ deve ser compatível com a da variável r. Basta lembrar que originalmente a coordenada cartesiana $z = r\chi$ ao longo do eixo de simetria induz a seguinte estimativa:

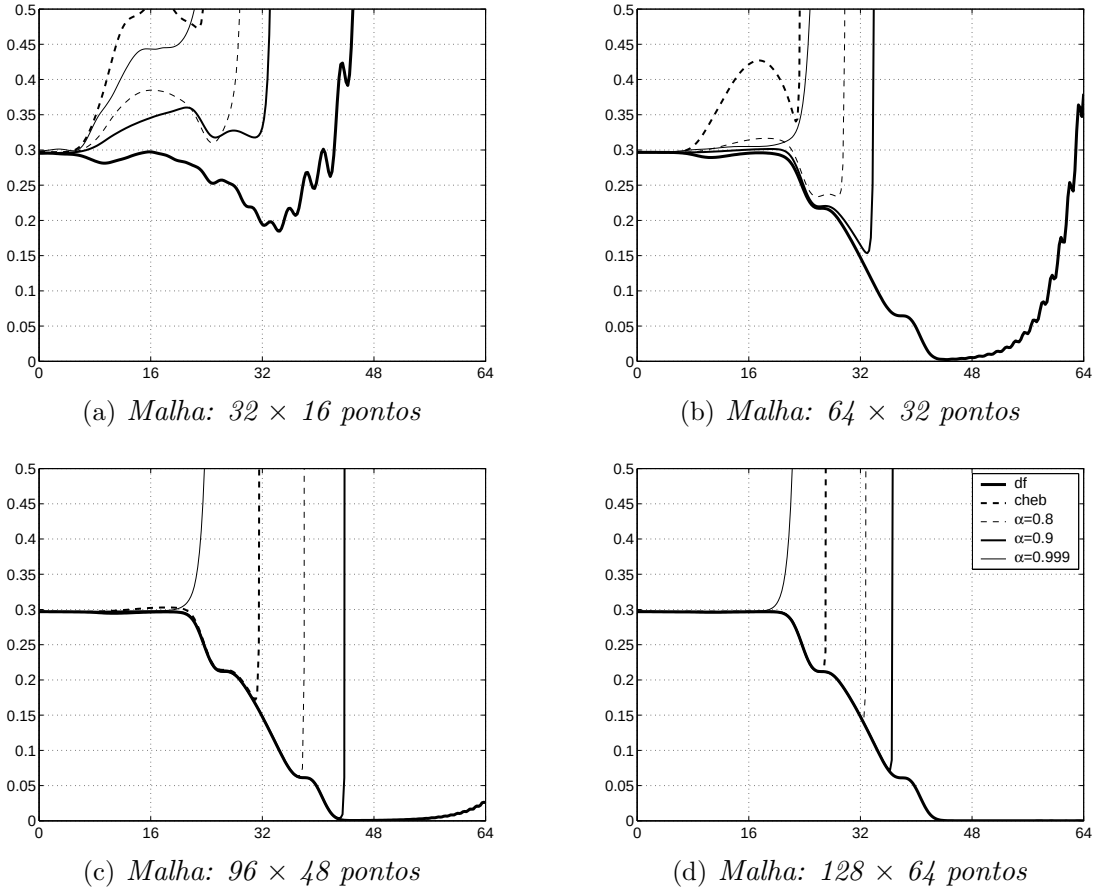
$$\Delta z \approx r\Delta\chi + \chi\Delta r \Rightarrow \Delta\chi \approx \frac{\Delta r}{r}.$$

No primeiro gráfico da Figura 5.6 vemos que o método de diferenças finitas não foi capaz de resolver o problema com apenas 32×16 pontos, gerando resultados completamente espúrios a partir de $t \approx 34$. Esse comportamento pode ser facilmente explicado pelo tamanho dos intervalos da malha, $\Delta r \approx 1$. Entretanto o método pseudo-espectral consegue resolver o problema, tanto nos pontos de Chebyshev, quanto nos pontos transformados com $\alpha = 0.8$ e $\alpha = 0.9$. Sendo que os gráficos que correspondem a essas três formas de discretização espacial são indistinguíveis. Mesmo o método pseudo-espectral com os pontos transformados com $\alpha = 0.999$ que gerou resultados espúrios, obteve resultados bem melhores que o método de diferenças finitas. O gráfico seguinte confirmam o observado no primeiro gráfico. O método pseudo-espectral obtém bons resultados, mesmo o caso $\alpha = 0.999$ se aproxima dos demais casos.

Comparando os métodos com base nos resultados apresentados nos gráficos da Figura 5.6 poderíamos concluir que a relação entre eles é a esperada com base na teoria. Com o pseudo-espectral obtendo melhores resultados com um número bem menor de pontos.

Entretanto quando comparamos os resultados obtidos para a equação de Yang-Mills completa, exibidos nos gráficos da Figura 5.7 encontramos um cenário muito diferente. Esses gráficos exibem essencialmente a mesma informação que os gráficos anteriores.

No primeiro gráfico dessa figura, 32×16, nenhum dos métodos conseguiu resolver as equações de Einstein-Yang-Mills. No caso do método de diferenças finitas isso é esperado devido a pouca resolução da malha. Com o aumento da resolução da malha espacial o



(a) *Malha: 32 × 16 pontos*

(b) *Malha: 64 × 32 pontos*

(c) *Malha: 96 × 48 pontos*

(d) *Malha: 128 × 64 pontos*

Figura 5.7: *Energia total do campo de Yang-Mills ao longo do tempo*

método de diferenças finitas ganha precisão e consegue resolver a equação, na precisão desejada, com a malha de 128×64 pontos. Porém o método pseudo espectral sempre causa uma explosão na energia. Esse comportamento espúrio acontece com todos os valores de α testados. Os melhores resultados foram obtidos com $\alpha = 0.9$. Com esse valor do coeficiente da transformação a solução permanece estável até pouco mais que $t = 40$ na malha de 96×48. Porém na malha de 128×64 a solução se desestabiliza em $t \approx 36$.

Por essa razão as soluções para as equações de Einstein-Yang-Mills não Abelianas apresentadas nos capítulos seguintes foram calculadas com o método de diferenças finitas. Enquanto que as soluções Abelianas foram calculadas com o método espectral.

6 Resultados

Nesse capítulo discutimos como serão apresentadas, nos próximos capítulos, as soluções para o campo de Yang-Mills com simetria axial em um espaço-tempo de Minkowski. As componentes de um campo de calibre não têm significado físico per se. Vamos então descrever o comportamento das soluções a partir de suas densidades de energia.

Descrevemos em seguida quais foram as condições iniciais utilizadas. Essas condições foram determinadas analiticamente e estão dividas em duas famílias. Embora elas sejam apenas casos particulares do problema como um todo, elas foram suficientes para explorar vários aspectos da dinâmica dos campos de Yang-Mills.

6.1 Apresentação dos Resultados

Como os valores das componentes da conexão A_μ são dependentes das escolhas de calibre, preferimos apresentar os gráficos da energia de cada solução obtida. Isso permite uma melhor visualização dos fenômenos físicos representados. Entretanto, a expressão para a densidade de energia em coordenadas esféricas (4.12) contém uma singularidade do tipo r^{-2}, que domina o comportamento da função. Porém esse termo está ligado ao sistema de coordenadas, singular em $r = 0$, e não à densidade de energia. Isso fica explícito ao observarmos que a energia total é obtida pela integral

$$E^{\hat{a}} = \int \rho^{\hat{a}}\, r^2 d\varphi d\chi dr$$

onde a função $\rho^{\hat{a}}$ deve ser multiplicada por r^2 antes de ser integrada. Vamos então exibir os gráficos da densidade de energia multiplicados por r^2 pois assim obtemos uma melhor visualização da dinâmica das soluções.

Uma função que usaremos para visualizar a dinâmica das soluções é a densidade radial de energia. Essa função nada mais é que a integral da densidade de energia sobre as

superfícies com a coordenada r constante em cada instante do tempo, isso é,

$$E_r^{\hat{a}}(r,t) = \int \rho^{\hat{a}}(r,\chi,t)\, r^2 d\chi d\varphi \tag{6.1}$$

6.2 Condições Inicias

As condições iniciais não podem ser especificadas livremente devendo obedecer os vínculos (4.9), que são equações lineares para as funções $N_\chi^{\hat{a}}$. Além de satisfazer os vínculos, queremos que as condições iniciais obedeçam também a alguns princípios físicos, por exemplo, que a energia e momentos sejam finitos.

Por simplicidade escolhemos usar como condições iniciais algumas soluções particulares dos vínculos obtidas analiticamente. Isso impõem uma restrição no conjunto de condições iniciais que iremos trabalhar, mas acreditamos que essas condições são suficientemente gerais para permitir o estudo de muitas características não triviais das soluções de Yang-Mills.

Do ponto de vista numérico, essa escolha elimina a necessidade de resolver numericamente uma equação elíptica para determinar as condições iniciais. Além disso, os vínculos são satisfeitos exatamente no instante inicial, o que contribui para a qualidade das soluções obtidas.

Primeira Família

A primeira família que vamos considerar é determinada pela escolha

$$N_r^{\hat{a}} = N_\chi^{\hat{a}} = 0 \tag{6.2}$$

Com essa escolha os vínculos (4.9) são automaticamente satisfeitos e as componentes $A_r^{\hat{a}}$ podem ser escolhidas arbitrariamente. Além disso, as densidades de momento iniciais são nulas enquanto que as densidades de energia são dadas por

$$\rho^{\hat{a}} = -\frac{1-\chi^2}{r^2}\left(\partial_\chi A_r^{\hat{a}}\right)^2$$

Podemos observar nessa expressão que a densidade de energia da condição inicial depende, do campo, apenas através das derivadas $\partial_\chi A_r^{\hat{a}}$. Portanto, para que nossa solução tenha energia finita, basta que essas derivadas sejam finitas e que se anulem quando r for para infinito. Além disso, qualquer escolha das componentes $A_r^{\hat{a}}$ que não dependa explicitamente da coordenada χ possui energia nula.

Como as componentes $A_r^{\hat{a}}$ são arbitrárias, podemos determina-las por uma expressão de variáveis separadas

$$A_r^{\hat{a}} = R^{\hat{a}}(r)\, X^{\hat{a}}(\chi) \tag{6.3}$$

onde as funções $R^{\hat{a}}(r)$ e $X^{\hat{a}}(\chi)$ são arbitrárias, devendo apenas respeitar condições de fronteira e limitação. Nos experimentos numéricos utilizamos as seguintes funções

$$R^{\hat{a}}(r) = \alpha^{\hat{a}} \exp\left(-\frac{(r - r_0^{\hat{a}})^2}{L^{\hat{a}}}\right) \tag{6.4}$$

$$X^{\hat{a}}(\chi) = \sum_n c_n^{\hat{a}} P_n(\chi) \tag{6.5}$$

onde $\alpha^{\hat{a}}$, $r_0^{\hat{a}}$, $L^{\hat{a}}$ e $c_n^{\hat{a}}$ são constantes e $P_n(\chi)$ são os polinômios de Legendre de ordem n. Os primeiros polinômios dessa família são:

n	$P_n(\chi)$
0	1
1	χ
2	$\frac{3}{2}\chi^2 - \frac{1}{2}$
3	$\frac{5}{2}\chi^3 - \frac{3}{2}\chi$
4	$\frac{35}{8}\chi^4 - \frac{15}{4}\chi^2 + \frac{3}{8}$

Segunda Família

A segunda família que foi construída partiu de uma escolha não simétrica nas cores do campo. Nesse caso escolhemos anular completamente a terceira cor no instante inicial, isso é, impusemos que

$$A_r^{\hat{3}} = N_r^{\hat{3}} = N_\chi^{\hat{3}} = 0 \tag{6.6}$$

Enquanto que $A_\chi^{\hat{3}}$ é nulo na condição inicial por escolha de calibre. Como eliminamos a terceira cor o índice $\hat{a}$ assume apenas os valores 1 e 2.

Dessa forma os vínculos (4.9) para as duas primeiras cores tornam-se

$$\partial_\chi\Big[(1-\chi^2)N^{\hat{a}}_\chi\Big] = -\partial_r\Big(r^2 N^{\hat{a}}_r\Big)$$

Essas equações para os vínculos deixam as cores desacopladas. O acoplamento entre as componentes da primeira e segunda cor aparece apenas na expressão para o terceiro vínculo, que se reduz a uma equação algébrica

$$A^{\hat{1}}_r N^{\hat{2}}_r = A^{\hat{2}}_r N^{\hat{1}}_r$$

Uma solução para o primeiro e segundo vínculo é dada pelas expressões

$$N^{\hat{a}}_r = \partial_\chi\Big[(1-\chi^2)X^{\hat{a}}\Big]R^{\hat{a}} \tag{6.7}$$

$$N^{\hat{a}}_\chi = -X^{\hat{a}}\partial_r\Big(r^2 R^{\hat{a}}\Big) \tag{6.8}$$

Para satisfazer o terceiro vínculo podemos considerar várias escolhas para as componentes $A^{\hat{a}}_r$. Tomando a mais simples temos

$$A^{\hat{a}}_r = \epsilon N^{\hat{a}}_r \tag{6.9}$$

Onde ϵ pode ser um número qualquer.

Com essas escolhas a energia no instante inicial é dada por

$$\rho^{\hat{a}} = -\frac{1-\chi^2}{r^2}\left[\Big(\epsilon\partial_\chi N^{\hat{a}}_r\Big)^2 + N^{\hat{a}2}_\chi\right] - N^{\hat{a}2}_r \tag{6.10}$$

Uma diferença dessa escolha com relação à anterior é que os momentos não se anulam na condição inicial. Suas expressões são

$$J^{\hat{a}}_r = 2\epsilon\frac{1-\chi^2}{r^2}\Big(\partial_\chi N^{\hat{a}}_r\Big)N^{\hat{a}}_\chi \tag{6.11}$$

$$J^{\hat{a}}_\chi = -2\epsilon\Big(\partial_\chi N^{\hat{a}}_r\Big)N^{\hat{a}}_r \tag{6.12}$$

Note que se trocarmos o sinal da constante ϵ o sinal das densidades de momento também é trocado. Podemos usar essa característica para termos algum controle sobre a direção de propagação inicial do pulso.

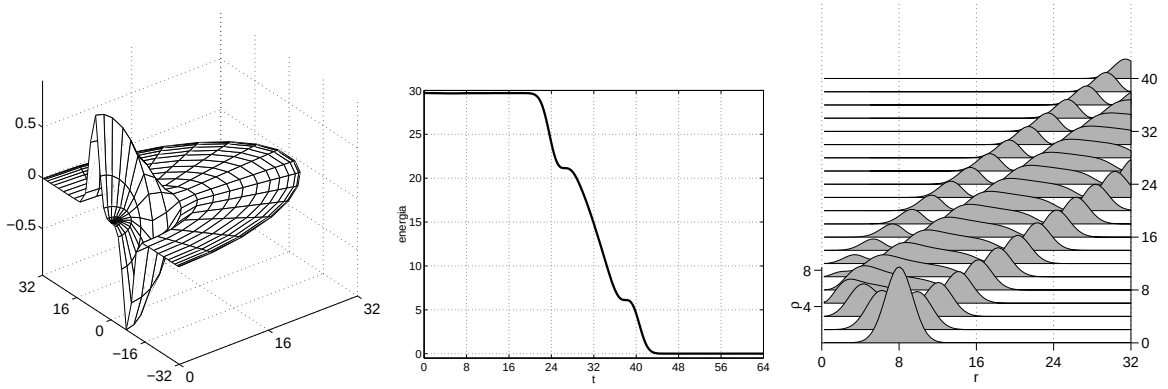
7 Soluções Abelianas

Nesse capítulo vamos considerar soluções em que duas cores são identicamente nulas, $A^{\hat{2}} = A^{\hat{3}} = 0$. Isso reduz as equações de Yang-Mills a equações lineares. Seu comportamento deve ser, portanto, muito mais simples do que o esperado para o conjunto completo de equações. Porém, mesmo sendo um caso particular bastante restrito, as soluções Abelianas são um bom ponto de partida para o estudo das soluções das equações de Yang-Mills.

Como a segunda e terceira cores de todas as soluções apresentadas nesse capítulo serão nulas vamos omitir o índice de cor das componentes do campo.

7.1 Um Pulso Abeliano

Vamos iniciar a apresentação das soluções de Yang-Mills com alguns resultados obtidos partindo de condições iniciais da primeira família (6.2).



(a) *Condição inicial,* A_r (b) *Energia total* (c) *Energia* $\times r$

Figura 7.1: *Solução gerada com* $\alpha = 1$, $r_0 = 8$, $L = 8$, $c = (0, 1)$

A condição inicial para A_r obtida com a escolha

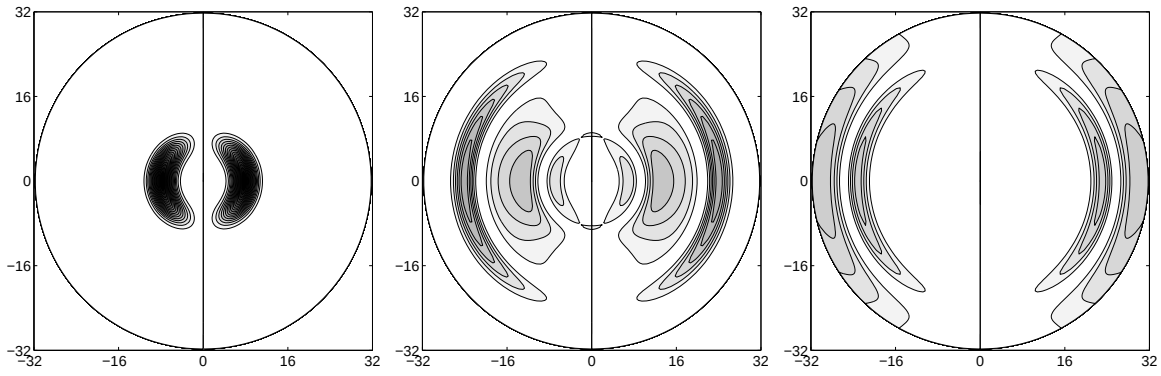
$Ab1$:

α	r_0	L	c_n
1	8	8	$(0, 1)$

está representada no Gráfico (a) da Figura 7.1. Note que nessa família de condições iniciais A_r é a única componente não nula. Os Gráficos (b) e (c), dessa figura, exibem a evolução da energia ao longo do tempo obtida a partir dessa condição inicial. No Gráfico (b) apresentamos a evolução da energia total do campo. Observamos que até $t \approx 20$ a energia total se mantém constante. Esse é o tempo em que o pulso está se movendo no interior do domínio analisado. Em seguida o pulso começa a deixar o domínio e conseqüentemente a energia total se anula. A partir de $t \approx 44$ toda a energia já se deslocou para fora da região simulada.

No Gráfico (c) da Figura 7.1 podemos ver com mais clareza a dinâmica do campo. Nesse gráfico exibimos a densidade radial de energia (6.1). Cada instante do tempo é exibido atrás e acima do anterior. No primeiro plano vemos a distribuição de energia na condição inicial. Essa distribuição é uma Gaussiana de centro em $r = r_0 = 8$. Logo em seguida, vemos que o pulso inicial se divide em um pulso que se propaga em direção a origem, outro que se propaga em direção ao infinito e um pulso intermediário que parece preencher o espaço entre os dois pulsos anteriores. Um comportamento interessante na dinâmica dessa solução é que o pulso que se dirige à origem é repelido antes de atingir o centro.

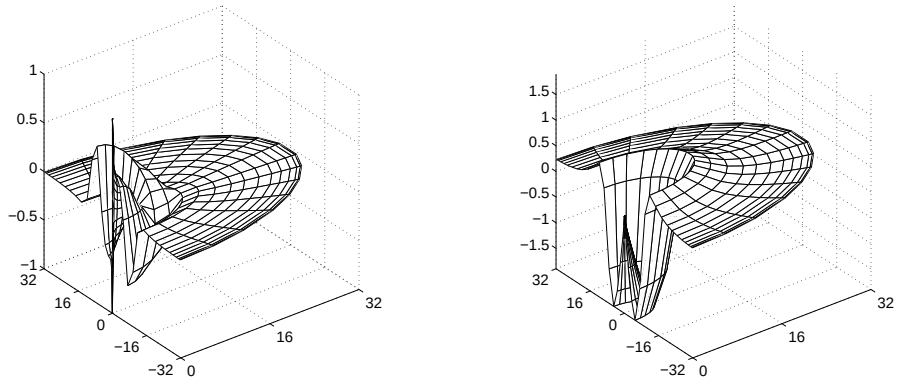
Para termos uma visão espacial da dinâmica desse pulso a Figura 7.2 exibe as curvas de nível da densidade de energia multiplicada por r^2, nos tempos $t = 0, 16, 32$. No primeiro



(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 7.2: *Densidade de energia multiplicada por* r^2

gráfico vemos que na condição inicial a energia está concentrada em um toro em volta do eixo de simetria. Porém com a evolução do campo, esse toro se desfaz dando origem a vários pulsos toroidais alongados, que correspondem aos três pulsos observados no terceiro gráfico da Figura 7.1.



(a) A_r (b) A_χ

Figura 7.3: *Solução estacionária final*

Como vimos após $t \approx 44$ toda a energia do pulso deixa o domínio que estamos considerando. Porém as componentes do campo não se anulam. Elas estabilizam em soluções não nulas de puro calibre. Podemos ver essa solução estacionária final na Figura 7.3. O Gráfico (a) exibe o estado final da componente A_r e o Gráfico (b) da componente A_χ. As demais componente são nulas, a menos de ruídos numéricos. No gráfico da componente A_r vemos um pico sobre a origem. Esse pico se deve à singularidade dessa função, que tende a valores diferentes dependendo da direção com que nos aproximamos da origem. Devemos notar que os métodos numéricos empregados não foram projetados para lidar com descontinuidades. Porém o comportamento qualitativo da solução não parece ter sido poluído por esse pulso. A solução permanece estável até tempos bem maiores que os exibidos nos gráficos, chegando até a $t = 128$.

7.2 Pulsos Abelianos Multipolares

Nos pulsos da primeira família (6.2) a variação dos parâmetros α, r_0 e L não levam a mudanças qualitativas nas soluções obtidas, o que é esperado para soluções de uma equação diferencial linear, enquanto que a variação dos coeficientes c_n gera soluções com características multipolares distintas. Podemos ver isso nitidamente nos gráficos das Figuras 7.4 e 7.5. Em ambas as soluções os parâmetros $\alpha = 1$, $r_0 = 8$ e $L = 8$ são iguais.

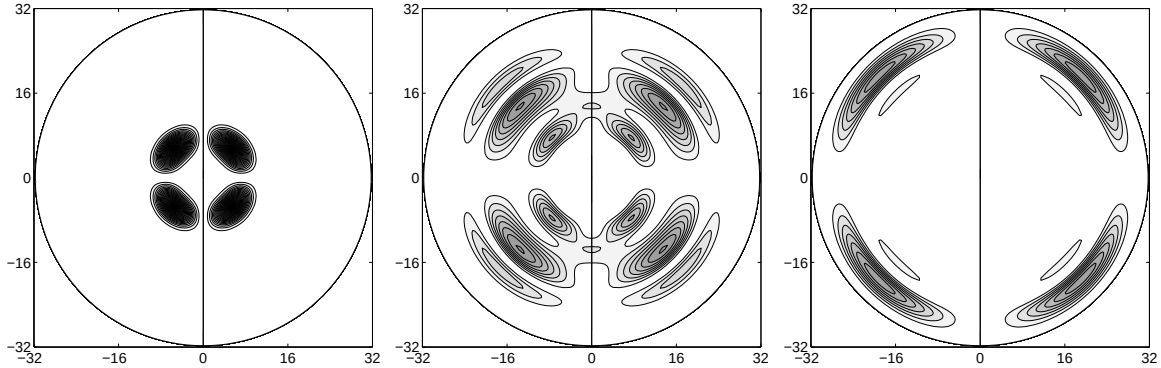


Figura 7.4: *Densidade de energia,* $r^2\rho$*, da solução com* $c = (0, 0, 1)$*, nos instantes* $t = 0, 16, 32$

Porém os coeficientes multipolares são $c = (0, 0, 1)$ e $c = (0, 0, 0, 1)$. As duas figuras exibem a densidade de energia nos instantes de tempo $t = 0, 16, 32$.

*Ab*2 :

α	r_0	L	c_n
1	8	8	$(0, 0, 1)$

*Ab*3 :

α	r_0	L	c_n
1	8	8	$(0, 0, 0, 1)$

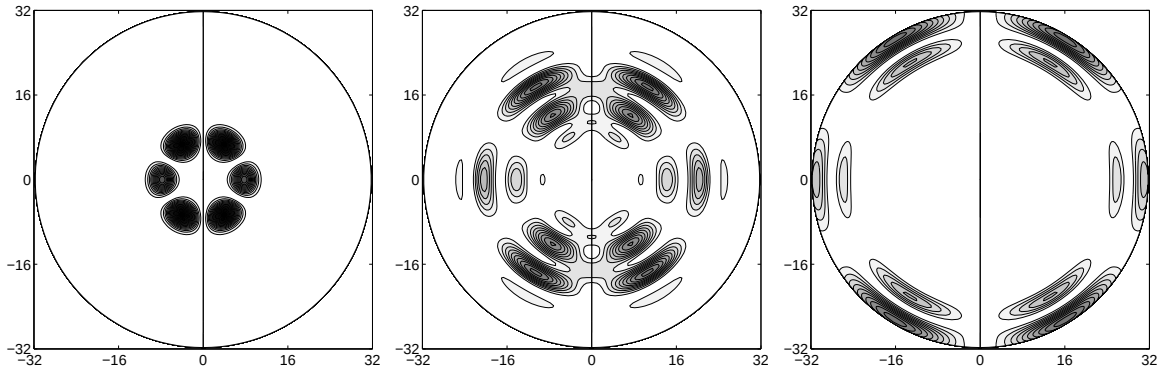
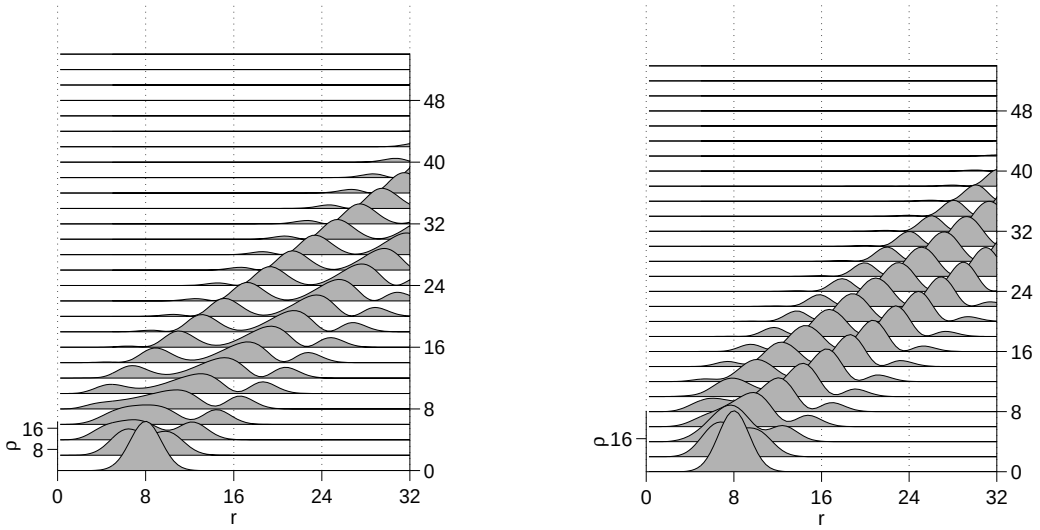


Figura 7.5: *Densidade de energia,* $r^2\rho$*, da solução com* $c = (0, 0, 0, 1)$*, nos instantes* $t = 0, 16, 32$

Outra forma de compararmos essas soluções é observando a evolução das densidades radiais de energia. Esses gráficos são apresentados na Figura 7.6. O gráfico (a) exibe a densidade radial da solução com $c = (0, 0, 1)$ e o gráfico (b) com $c = (0, 0, 1)$. Esses gráficos devem ser comparados também com o gráfico (c) da figura 7.1, que corresponde ao caso $c = (0, 1)$. Lembramos aqui que o caso $c = (1)$ implica em uma solução de puro calibre.



(a) $c = (0, 0, 1)$ (b) $c = (0, 0, 0, 1)$

Figura 7.6: *Energia $\times r$ para soluções com $\alpha = 1$, $r_0 = 8$, $L = 8$ e diferentes componentes multipolares.*

Nesses gráficos podemos observar algumas conseqüências da distribuição mais complexa de energia. Nas soluções com componentes multipolares mais altas, a repulsão na origem parece ser intensificada. Além disso, no caso $c = (0, 1)$ a solução inicial se divide em alguns pulsos e esses pulsos são aparentemente preservados ao longo da simulação. Porém nas soluções com $c = (0, 0, 1)$ e $c = (0, 0, 1)$ esses pulsos apresentam uma dinâmica mais elaborada.

Em todos esses casos o comportamento da energia total é muito semelhante. Além disso, todas essas soluções tendem a soluções estacionárias de puro calibre. A principal diferença é que condições iniciais com multipolos mais altos levam a soluções estacionárias com mais ondulações.

As soluções apresentadas até aqui são simétricas com relação a reflexão sobre o plano $\chi = 0$. Vamos apresentar agora algumas soluções multipolares, onde essa simetria foi removida. Essa quebra de simetria pode ser gerada, por exemplo, com a seguinte escolha de pârametros:

	α	r_0	L	c_n
*Ab*4 :	1	8	8	$\left(0, \frac{1}{2}, \frac{1}{2}\right)$

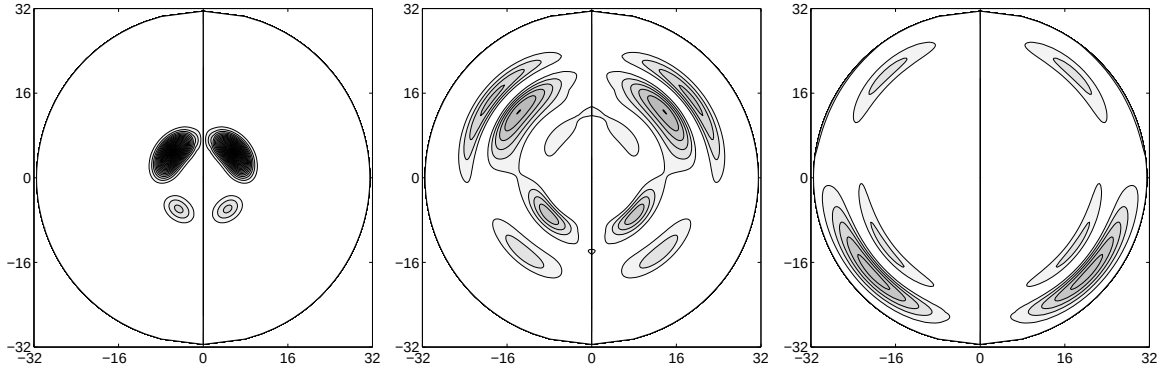


Figura 7.7: *Densidade de energia,* $r^2\rho$*, nos instantes* $t = 0, 16, 32$ *para a solução* $Ab4$

	α	r_0	L	c_n
$Ab5$:	1	8	8	$\left(0, \frac{2}{3}, \frac{1}{3}\right)$

A Figura 7.7 exibe a densidade de energia da solução $Ab4$, nos instantes $t = 0, 16$ e 32. A Figura 7.8 contém os gráficos equivalentes obtidos com a solução $Ab5$. Nesses gráficos fica evidente a assimetria com relação ao plano $\chi = 0$ que, nos gráficos, corresponde a linha horizontal passando por 0.

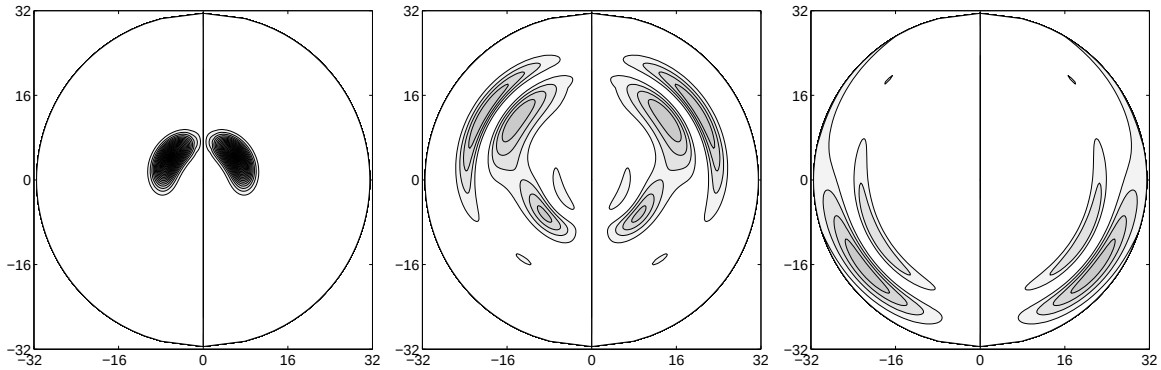
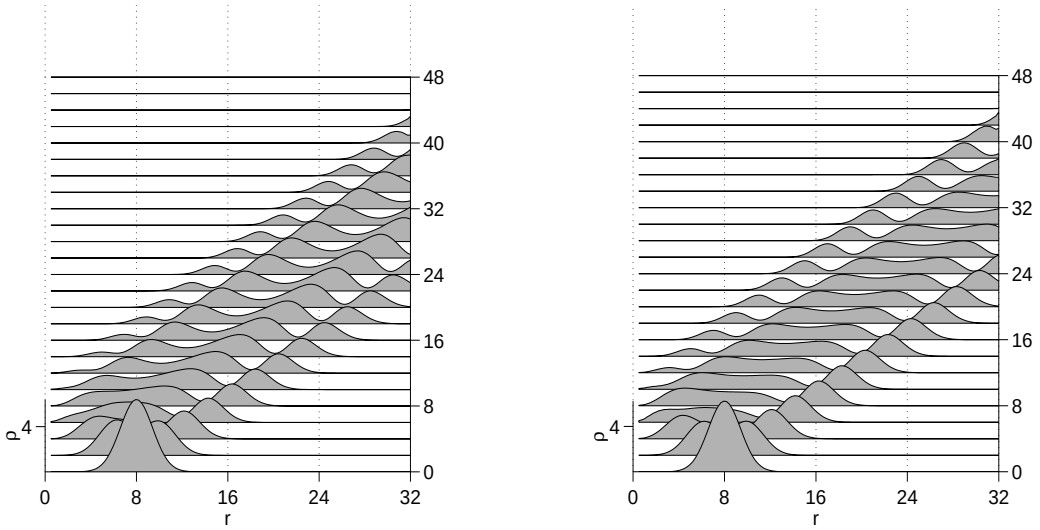


Figura 7.8: *Densidade de energia,* $r^2\rho$*, nos instantes* $t = 0, 16, 32$ *para a solução* $Ab5$

A Figura 7.9 apresenta os gráficos das distribuições da densidade de energia radial ao longo do tempo dessas duas soluções. Nesses gráficos os pulsos iniciais são muito semelhantes. Porém com a passagem do tempo essas distribuições apresentam comportamentos bastate distintos.



(a) $c = \left(0, \frac{1}{2}, \frac{1}{2}\right)$ (b) $c = \left(0, \frac{2}{3}, \frac{1}{3}\right)$

Figura 7.9: *Energia* $\times r$ *nas soluções Ab*4 *e Ab*5.

7.3 Pulsos da Segunda Familia

Apresentamos agora algumas soluções obtidas a partir de condições inicias da segunda familia (6.6). Nessas condições iniciais as componentes N_r, N_χ e A_r são dadas pelas expressões (6.7), (6.7) e (6.7) construídas a partir de duas funções, $R(r)$ e $X(\chi)$, arbitrárias. Os parâmetros dessas funções são semelhantes aos que definem as condições iniciais da primeira família. Porém seus significados não são os mesmos.

Um exemplo dessas soluções foi obtido com o seguinte conjunto de parâmetros

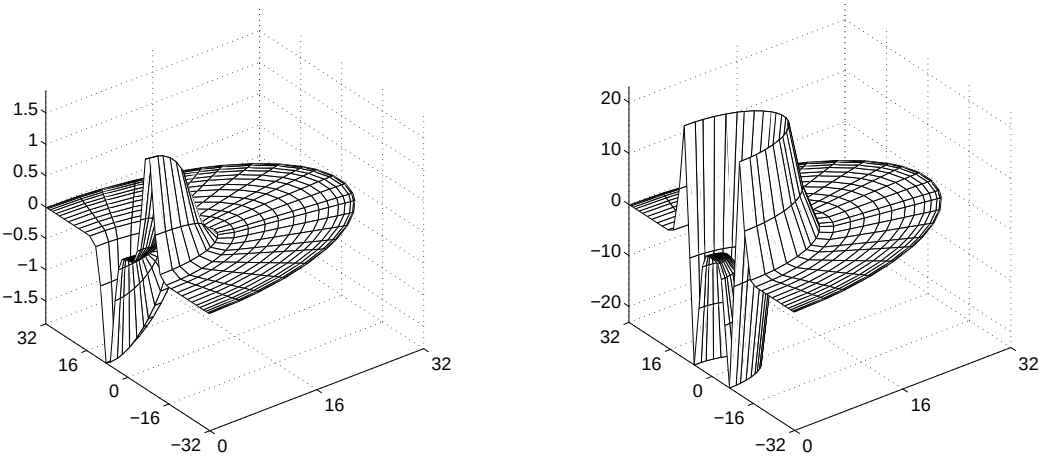
*Ab*6 :

ϵ	α	r_0	L	c_n
0	1	8	8	(1)

As componentes N_r e N_χ geradas por esses parâmetros podem ser vistas na Figura 7.10. A componente A_r é nula devido a escolha do parâmetro ϵ. Isso implica que as densidades de momento iniciais sejam nulas, o que corresponde a uma condição inicial com simetria de reflexão temporal.

Os gráficos da Figura 7.11 exibem as densidades de energia desta solução no instante inicial e em $t = 16$ e 32. Nesses gráficos vemos que a energia está distribuída em dois pulsos aproximadamente concêntricos.

A Figura 7.12 exibe a dinâmica da energia ao longo do tempo. O primeiro gráfico exibe evolução a energia total. Inicialmente a energia é conservada e em seguida a primeira



(a) N_r *em* $t = 0$ (b) N_χ *em* $t = 0$

Figura 7.10: *Componentes da solução Ab*6

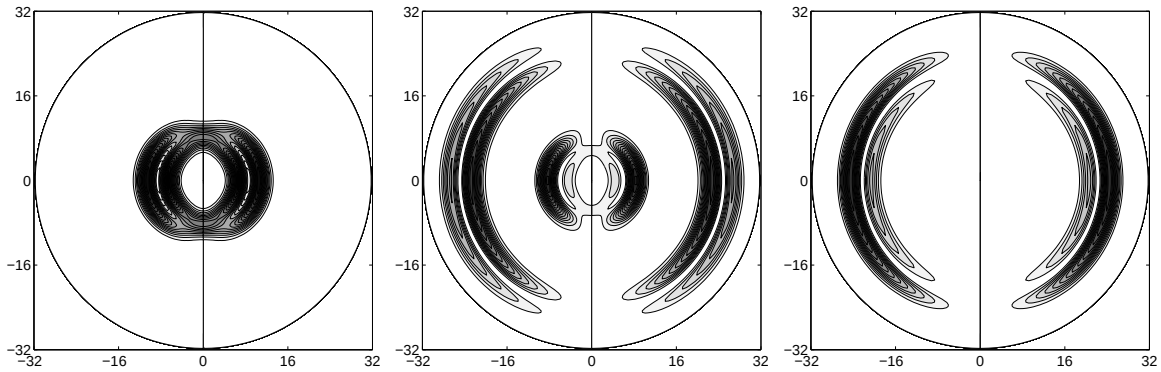


Figura 7.11: *Densidade de energia,* $r^2\rho$ *nos instantes* $t = 0, 16, 32$ *para a solução Ab*6

parte da energia abandona o domínio. Depois ocorre um novo intervalo em que a energia é conservada enquanto a segunda parte do pulso trafega pelo interior domínio. Finalmente toda a energia deixa o domínio. Podemos observar essa mesma dinâmica no segundo gráfico, onde a distribuição de energia ao longo do raio é exibida. Note que no instante inicial a energia está dividida em dois pulsos e que um pico é gerado logo em seguida. Uma diferença dessa solução é que os pulsos viajam em dois conjuntos isolados, enquanto que nas exibidas anteriormente a energia ocupa todo um intervalo do raio.

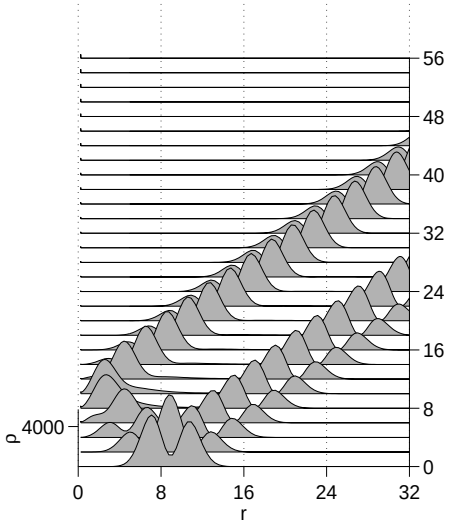
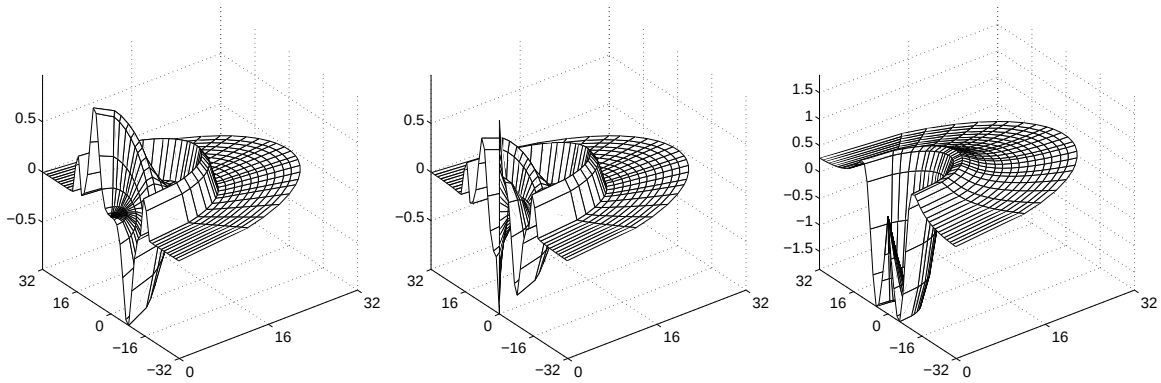


Figura 7.12: *Energia* $\times r$ *da solução Ab*6

7.4 Interação Entre Pulsos Abelianos

Outro caso explorado é composto pela sobreposição de dois pulsos. Vamos utilizar condições iniciais da primeira família (6.2), portanto a forma de construir a sobreposição de dois pulsos é tomar a componente A_r como a soma de funções do tipo (6.3.

$$A_r = R_1(r)X_1(\chi) + R_2(r)X_2(\chi)$$



(a) A_r *em* $t = 0$ (b) A_r *quando* $t \to \infty$ (c) A_χ *quando* $t \to \infty$

Figura 7.13: *Componentes da solução Ab*7

Como primeiro exemplo apresentamos uma solução onde a condição inicial é composta de um pulso de calibre e um pulso energético. Os parâmetros de cada um dos pulsos estão

listados na tabela

	a	α_a	r_{0a}	L_a	c_{na}
*Ab*7 :	1	1	8	8	$(0,1)$
	2	0.5	16	1	(1)

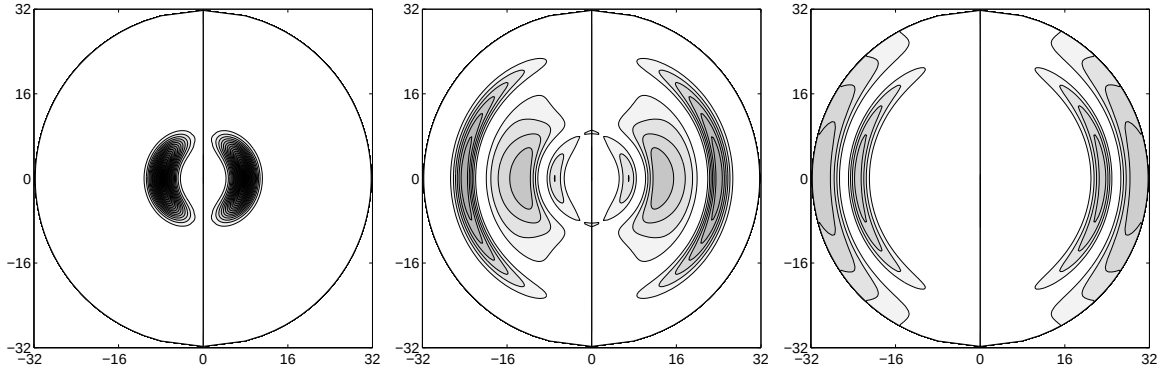
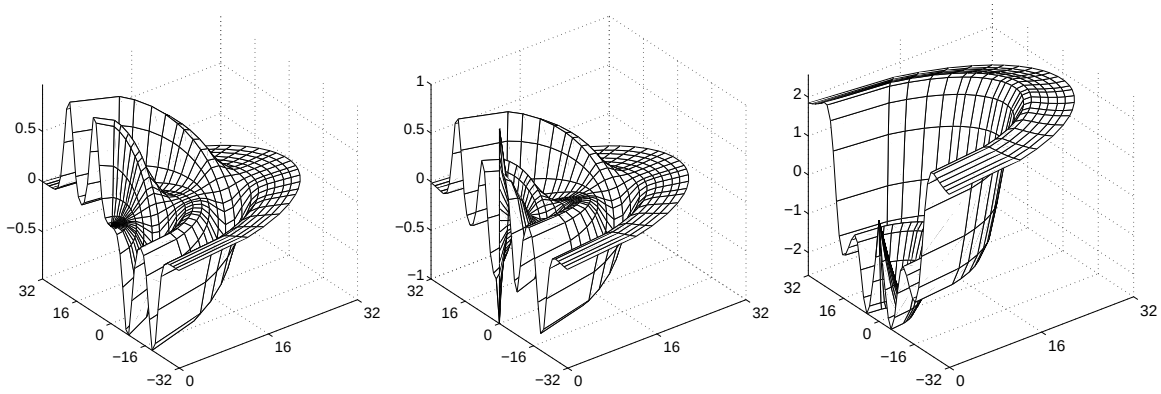


Figura 7.14: *Densidade de energia,* $r^2\rho$*, em* $t = 0, 16, 32$ *da solução Ab*7

O primeiro gráfico da Figura 7.13 exibe a componente A_r no instante inicial. Enquanto que os outros gráficos exibem as componentes A_r e A_χ após a energia do sistema ter abandonado o domínio. Note que o pulso de calibre, na componente A_r, permanece inalterado pela dinâmica do campo.

Na Figura 7.14 exibimos a densidade de energia dessa solução. Comparando esses gráficos com os da Figura 7.2, vemos que, como esperado, o pulso de calibre não altera a dinâmica da energia da solução.



(a) A_r *em* $t = 0$ (b) A_r *quando* $t \to \infty$ (c) A_χ *quando* $t \to \infty$

Figura 7.15: *Componentes da solução Ab*8

Apresentamos agora um exemplo de solução com dois pulsos energéticos. Os parâmetros usados foram

	a	α_a	r_{0a}	L_a	c_{na}
*Ab*8 :	1	1	8	8	$(0,1)$
	2	1	20	8	$(0,1)$

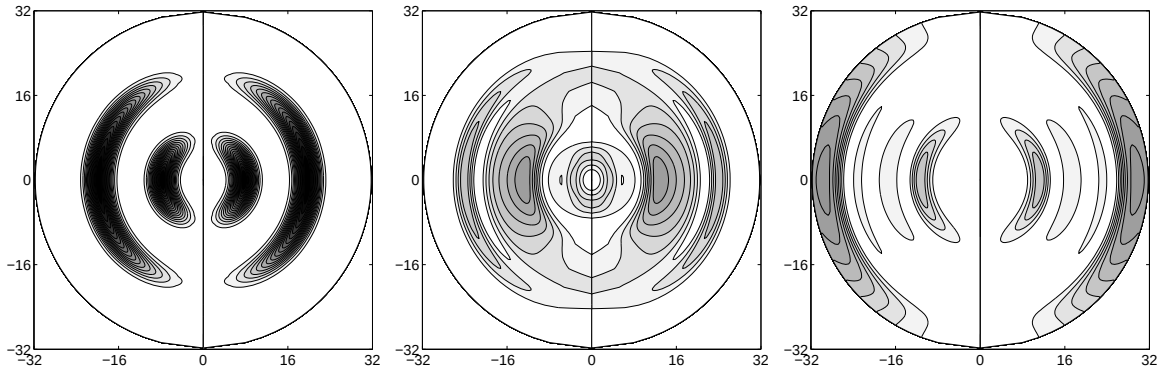


Figura 7.16: *Densidade de energia,* $r^2\rho$*, em* $t = 0, 16, 32$ *da solução Ab*8

A primeiro gráfico da Figura 7.15 exibe a componente A_r no instante inicial. Essa condição inicial corresponde a dois pulsos de energia com formato toroidal alongado. É possível ver a distribuição inicial de energia no primeiro gráfico da Figura 7.16.

O segundo e terceiro gráficos da Figura 7.15 exibem as componentes A_r e A_χ no final da simulação numérica. Como as derivadas temporais dessas componentes são numericamente nulas podemos afirmar, com razoável segurança, que essa corresponde ao estado estacionário final da solução.

A Figura 7.16 exibe a densidade de energia multiplicada por r^2 nos instantes $t = 0, 16, 32$. Podemos ver nesses gráficos a interação entre os dois pulsos.

O comportamento da energia total da solução segue o mesmo comportamento das demais soluções. Inicialmente a energia se conserva, mas quando os pulsos alcançam a fronteira exterior, o campo perde toda sua energia.

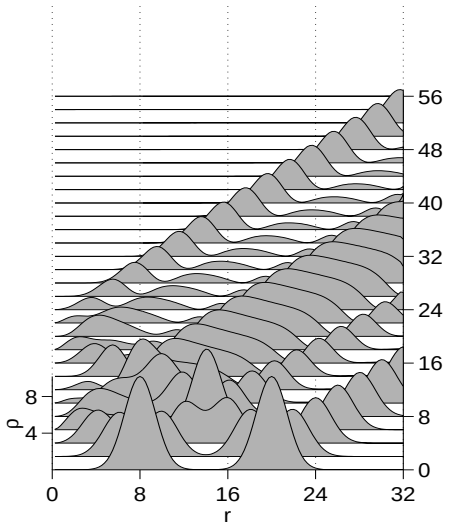


Figura 7.17: *Energia* $\times r$ *da solução Ab*8

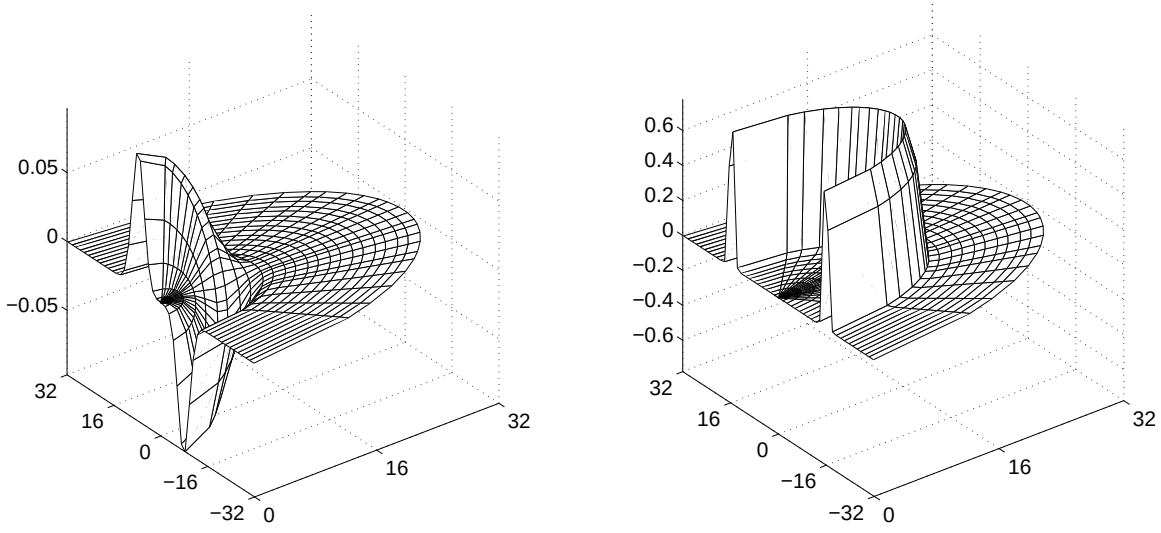
Na Figura 7.17 vemos a evolução da distribuição radial da energia ao longo do tempo. Inicialmente a energia está contida em dois pulsos de formato Gaussiano centrados nos pontos $r = 8$ e $r = 20$. Cada um desses pulsos se divide em dois pulsos, viajando em direções opostas. Em $t \approx 8$ vemos o surgimento de um pico em $r \approx 14$.

8 Soluções Não Abelianas

Esse capítulo contém as soluções das equações de Yang-Mills com simetria axial em um espaço-tempo de Minkowski como acoplamento entre as cores, isto é, nenhuma cor foi eliminada a priori. O fenômeno mais interessante a ser observado nessas soluções é a troca de energia entre as três cores que compõem o campo.

8.1 Interação com Um Pulso de Calibre

Um pulso de puro calibre não possui energia, portanto não poder ser detectado fisicamente. Porém ele é capaz de alterar a dinâmica interna do campo. Nessa seção estudamos a interação entre um pulso de calibre estático e um pulso energético.



(a) *Componente* $A_r^{\hat{1}}$ (b) *Componente* $A_r^{\hat{2}}$

Figura 8.1: *Componentes não nulas da condição inicial*

Vamos considerar como um exemplo, a solução gerada pela condição inicial definida pelos parâmetros:

$YM1$:

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.1	8	8	$(0,1)$
2	1.0	16	1	(1)
3	0.0	0	0	(0)

Na Figura 8.1 vemos os gráficos das componentes $A_r^{\hat{1}}$ e $A_r^{\hat{2}}$ dessa condição inicial. Toda a energia dessa condição inicial está na primeira componente. A componente $A_r^{\hat{3}}$ é identicamente nula e a componente $A_r^{\hat{2}}$ é de puro calibre, não possuindo energia. Várias outras simulações semelhantes foram realizadas e em todas o comportamento qualitativo foi o mesmo que observamos nesse exemplo.

A linha sólida mais escura do gráfico da Figura 8.2 exibe a energia total dessa solução ao longo do tempo. As demais linhas representam as energias contidas em cada uma das cores do campo. No instante inicial vemos que toda a energia está contida na primeira cor, linha sólida mais fina. Porém assim que o pulso na primeira cor começa a interagir com o pulso de calibre da segunda cor, a energia é transportada para a terceira cor. Note que nenhuma energia é transferida para a segunda cor do campo. Ou seja, o pulso de calibre age sobre as componentes do campo transferindo a energia entre as duas outras cores, sem absorver nenhuma energia.

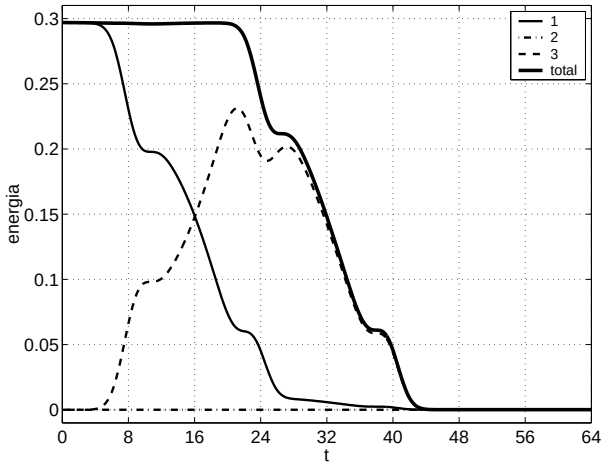
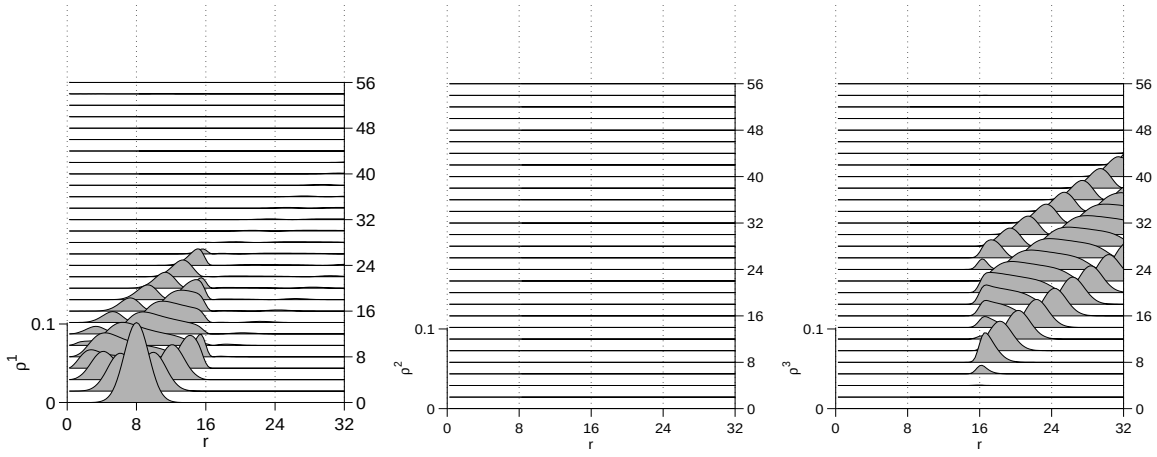


Figura 8.2: $YM1$ - *Energia total*

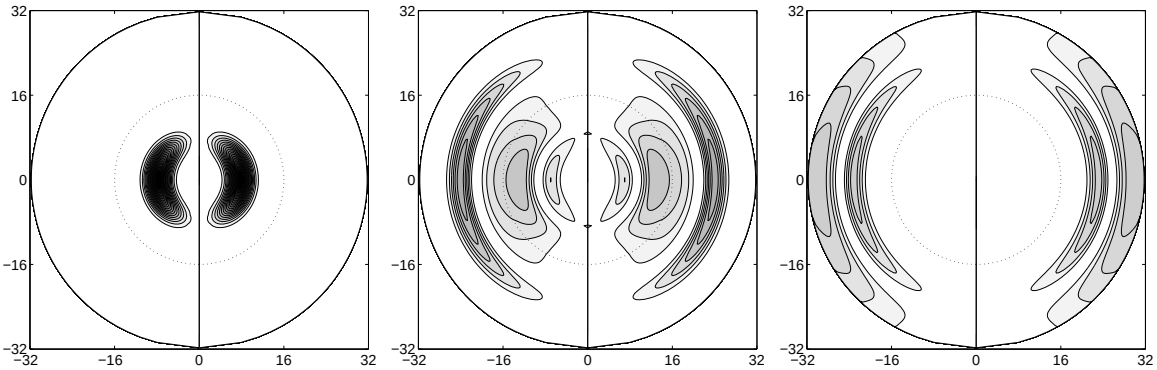
Podemos ver com mais clareza o fenômeno descrito no parágrafo anterior, observando os gráficos da Figura 8.3. Como no capítulo anterior esses gráficos exibem a densidade radial de energia (6.1). Nesses gráficos vemos claramente a energia da condição inicial acumulada em torno de $r = 8$ na primeira cor. Esse pulso começa a se espalhar e quando chega a $r = 16$ sua energia é convertida para a terceira cor. Com o passar do tempo toda a energia que inicialmente estava na primeira cor migra para a terceira. Durante todo esse processo a segunda cor permanece apenas com o pulso de puro calibre.



(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.3: *YM*1 - *Energia* $\times r$

Apesar da migração da energia entre as cores, a dinâmica do campo não é alterada do ponto de vista físico. Na Figura 8.2 podemos ver que a evolução da energia total é qualitativamente idêntico ao caso Abeliano, estudado no capítulo anterior. Mesmo a distribuição espacial da energia total não é alterada pela presença do pulso de calibre. Podemos ver isso nos gráficos da Figura 8.4. Esses gráficos exibem a dinâmica espacial da energia do campo em três instantes no tempo. No primeiro gráfico está a densidade de energia inicial, no segundo essa densidade quando $t = 16$ e no terceiro gráfico quando $t = 32$. A circunferência pontilhada de raio 16 marca a posição do pulso de calibre contido na segunda cor.



(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 8.4: *YM*1 - *Densidade de energia,* $r^2\rho$

Para visualizarmos a dinâmica espacial da energia entre as cores e ao longo do tempo a

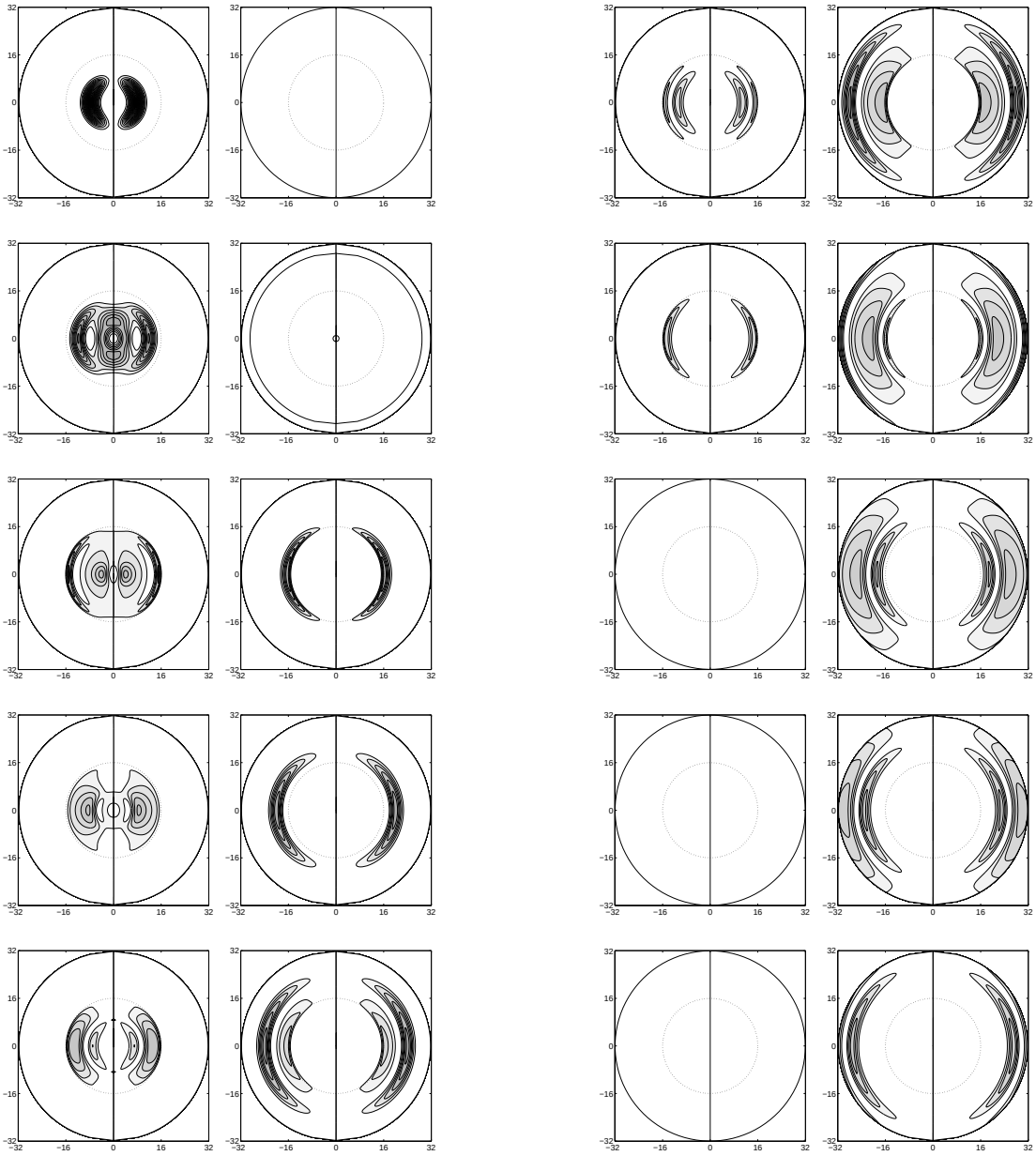


Figura 8.5: *YM*1 - *Densidade de energia na primeira e terceira cores,* $r^2\rho^{\hat{1}}$ *e* $r^2\rho^{\hat{3}}$ *nos tempos* $t = 0, 4, \ldots, 36$

Figura 8.5 exibe uma seqüência de pares de gráficos, onde cada par exibe a distribuição de energia entre a primeira e a terceira cor do campo. Cada par representa o estado do campo em um determinado instante do tempo, $t = 0, 4, \ldots, 36$. Novamente a circunferência pontilhada indica a posição do pulso de calibre. Na seqüência desses gráficos podemos

ver como o pulso de energia, que inicialmente estava todo na primeira cor é transferido totalmente para a terceira cor, quando interage com o pulso de calibre.

8.2 Dupla Interação com Um Pulso de Calibre

Apresentamos nessa seção uma solução onde o pulso energético interage duas vezes com um mesmo pulso de calibre. Vamos observar que em cada interação o comportamento é o mesmo discutido na seção anterior, quando apresentamos a interação de um pulso energético com um pulso de calibre. Para produzir esse resultado colocamos o pulso de calibre entre o pulso energético e seu centro, que se encontra na origem do sistema de coordenadas. Esse arranjo pode ser obtido com os parâmetros:

$YM2:$

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.1	16	8	$(0,0,1)$
2	1	8	1	(1)
3	0.0	0	0	(0)

A Figura 8.6 exibe a evolução da energia total dessa solução. Como no caso anterior, no início toda a energia está na primeira cor. Um pouco antes de $t = 8$ há uma transferência de energia para a terceira cor e logo após $t = 24$ a energia transferida retorna à primeira cor.

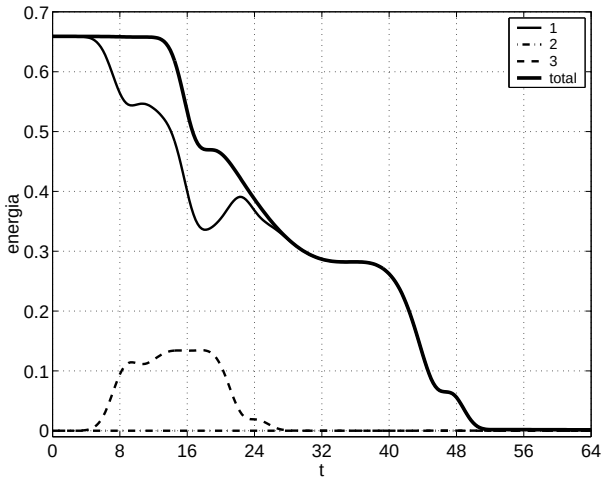
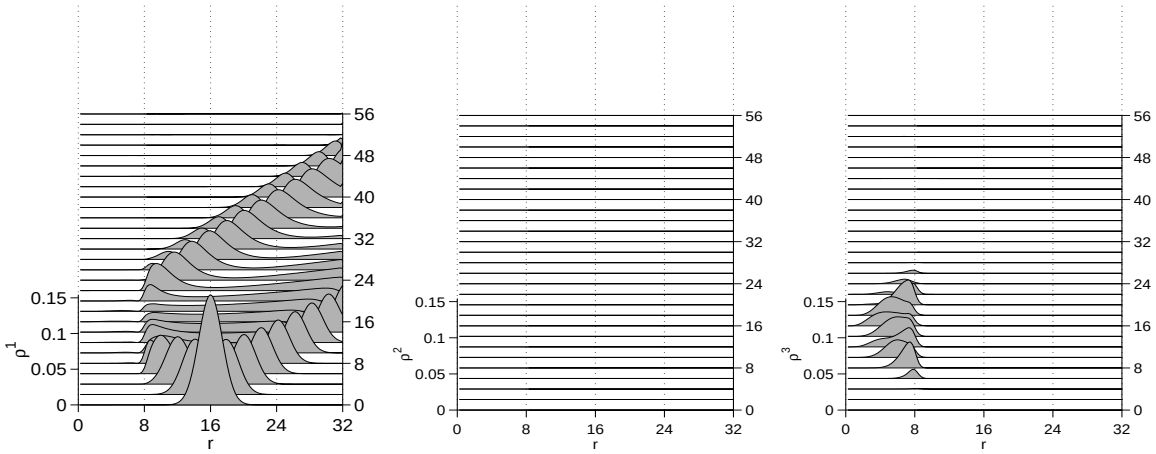


Figura 8.6: *YM2 - Energia total*

Observando a Figura 8.7 podemos ter uma visão melhor da dinâmica dessa solução. Na condição inicial a energia está contida na primeira cor e está distribuída de acordo com uma gaussiana de centro em $r = 16$. Esse pulso inicial se divide em dois pulsos, um seguindo na direção do exterior e outro na direção do centro. O primeiro simplesmente deixa o domínio entre $t = 16$ e $t = 32$, aproximadamente. O segundo pulso, entretanto, segue em direção ao pulso de calibre. Quando esse pulso chega a $r = 8$, o pulso de calibre transfere sua energia para a terceira cor. Essa transferência tem início, aproximadamente, em $t = 8$. No instante $t = 16$ esse pulso chega a origem e é refletido. Porém, ao seguir em direção ao exterior do

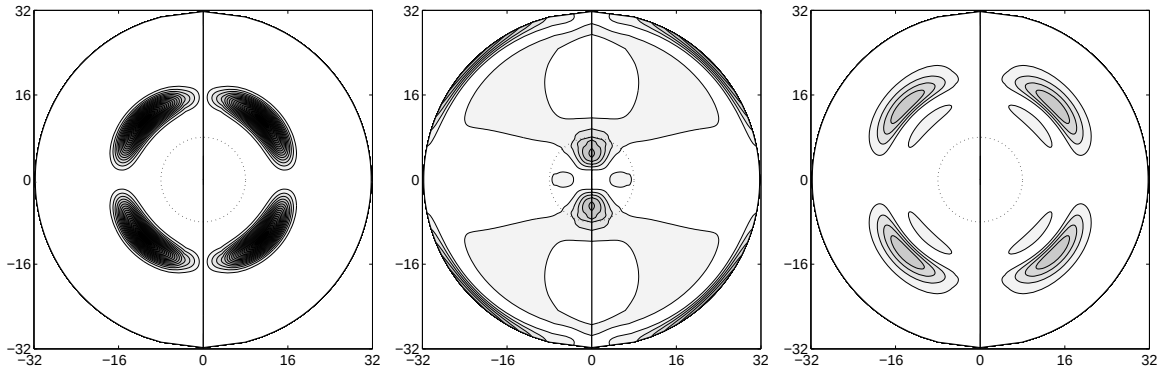
domínio, ele deve passar novamente pelo pulso de calibre. Conseqüentemente sua energia será transferida de volta à primeira cor.



(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.7: *YM2 - Energia* $\times r$

A Figura 8.8 exibe a evolução temporal da distribuição espacial da energia total. Podemos ver claramente, no primeiro gráfico, que a condição inicial tem uma distribuição quadrupolar, que foi imposta pelo parâmetro $c_n^{\hat{1}} = (0, 0, 1)$. Nesse, e em outros exemplos, vemos que a distribuição espacial da energia não tem muita influência sobre a interação com o pulso de calibre.



(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 8.8: *YM2 - Densidade de energia* $r^2\rho$

8.3 Interação com Dois Pulsos de Calibre

Nessa seção vamos considerar um outro arranjo para a interação entre um pulso energético e pulsos de calibre. Vamos posicionar um pulso de calibre exteriormente e outro internamente com relação ao pulso energético. O primeiro caso corresponde a situação apresentada na Seção 8.1 e o outra a Seção 8.2.

Os parâmetros abaixo definem uma condição inicial com um pulso energético na primeira cor e um pulso de calibre em cada uma das outras cores. Sendo que o pulso da segunda cor está centrado em $r = 8$ e o da terceira cor em $r = 24$ de modo que cada um fique de um lado do pulso energético que está centrado em $r = 16$.

$YM3$:

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.1	16	8	$(0, 1)$
2	1	8	1	(1)
3	1	24	1	(1)

A Figura 8.9 apresenta a da evolução energia de cada cor ao longo do tempo. A energia total apresenta o comportamento típico, permanecendo constante no início e se anulando quando o pulso deixa o domínio. Inicialmente toda a energia está contida na primeira cor, portanto o gráfico da energia nessa cor começa com o mesmo valor da energia total. Após algum tempo, essa cor perde energia tanto para a segunda cor, pela interação com o pulso da terceira cor, quanto para a terceira, pela interação com o pulso na segunda. Posteriormente, a energia que foi transferida para a terceira cor retorna a primeira, pelo mesmo processo observado na Seção 8.2.

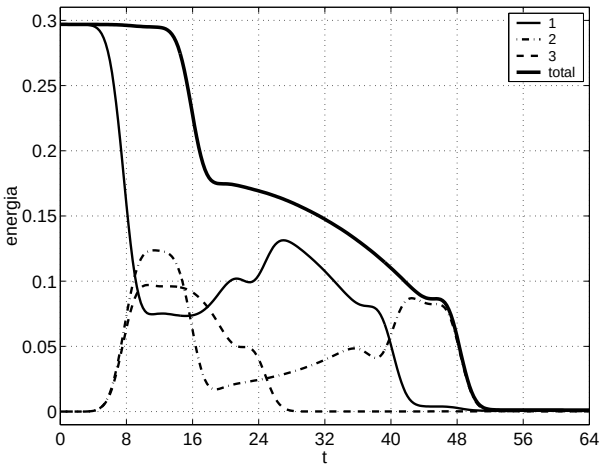
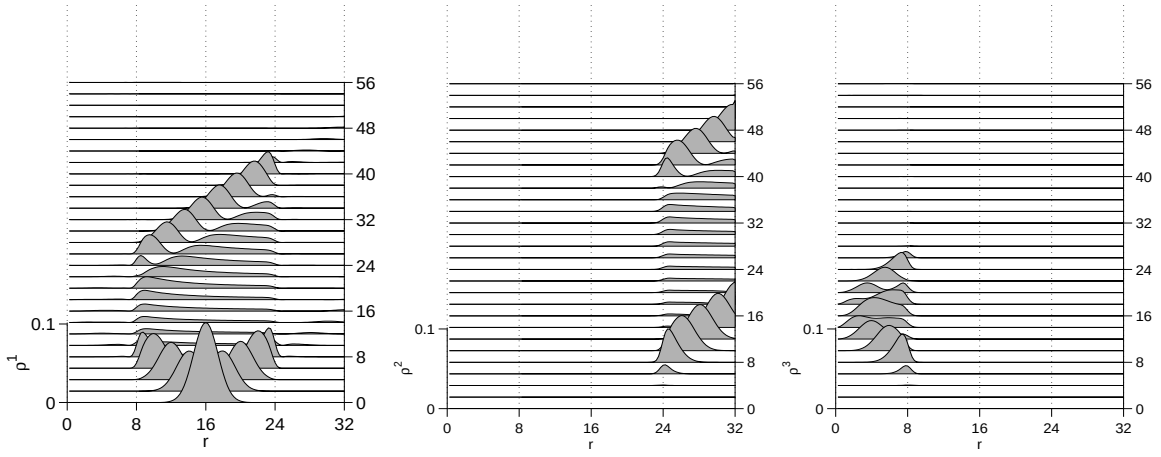


Figura 8.9: *YM3 - Energia total*

Também podemos observar a evolução da dinâmica energética dessa solução nos gráficos da Figura 8.10. Cada um desses gráficos mostra a distribuição da energia contida em uma cor. No instante inicial toda a energia está concentrada em um pulso gaussiano na primeira cor. Com a evolução dessa solução a energia se divide em dois pulsos, ainda



(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.10: *YM*3 - *Energia* $\times r$

restritos a primeira cor. Apenas em $t = 8$ esses pulsos iniciam a interação com os pulsos de calibre colocados nas outras cores. O pulso que se dirige ao centro do sistema de coordenadas interage com o pulso na segunda cor. Por isso vemos sua energia ser transferida para a terceira cor, representada pelo terceiro gráfico da figura. Enquanto isso, o pulso que se dirige para o exterior do domínio interage com o pulso de calibre da terceira cor e é transportado para a segunda.

Em $t = 16$ o pulso que se dirige à origem, agora contido na terceira cor, é refletido para o exterior. Porém em sua trajetória ele vai interagir novamente com o pulso da segunda cor e retornar para a primeira cor. Passado mais algum tempo esse pulso interage com o pulso de calibre da terceira cor e é transferido para a segunda cor. Embora a energia inicialmente esteja restrita a primeira cor, ela vai migrar entre as três cores e deixar o domínio apenas pela segunda cor.

8.4 Interação com Pulsos de Calibre Sobrepostos

Exploramos agora um caso particular da interação com dois pulso de calibre onde esses pulsos estão sobrepostos. Vamos construir uma condição inicial com um pulso energético em $r = 8$ na primeira cor e pulsos de calibre em $r = 16$ nas outras duas cores.

	$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
$YM4$:	1	0.1	8	8	$(0, 1)$
	2	1	16	1	(1)
	3	1	16	1	(1)

O gráfico da Figura 8.11 mostra a evolução da energia total dessa solução. Como nos casos anteriores, na condição inicial a energia está toda concentrada na primeira cor. Porém assim que o pulso energético chega ao pulso de calibre a energia muda de cor.

Nesse caso, entretanto, a transferência não é total. Observe que as curvas que correspondem às energias na segunda e terceira cores coincidem, e que apenas uma parte da energia é transferida. Se considerarmos o instante $t \approx 20$, que corresponde ao ponto de máxima energia na segunda e terceira cores, vemos que apenas 1/6 da energia é transferida para cada uma dessas cores. Após esse instante a energia começa a deixar o domínio.

Podemos acompanhar o transferência da energia também pelos gráficos da Figura 8.12. Uma diferença marcante desses gráficos com relação às soluções anteriores é que a primeira cor não perde toda a sua energia. Como observado na Figura 8.11 apenas um sexto, aproximadamente, da energia é transferida para cada uma das outras cores. Ficando assim ainda na primeira cor algo em torno de dois terços.

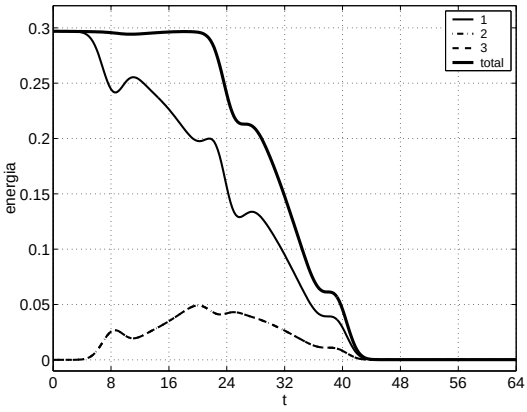
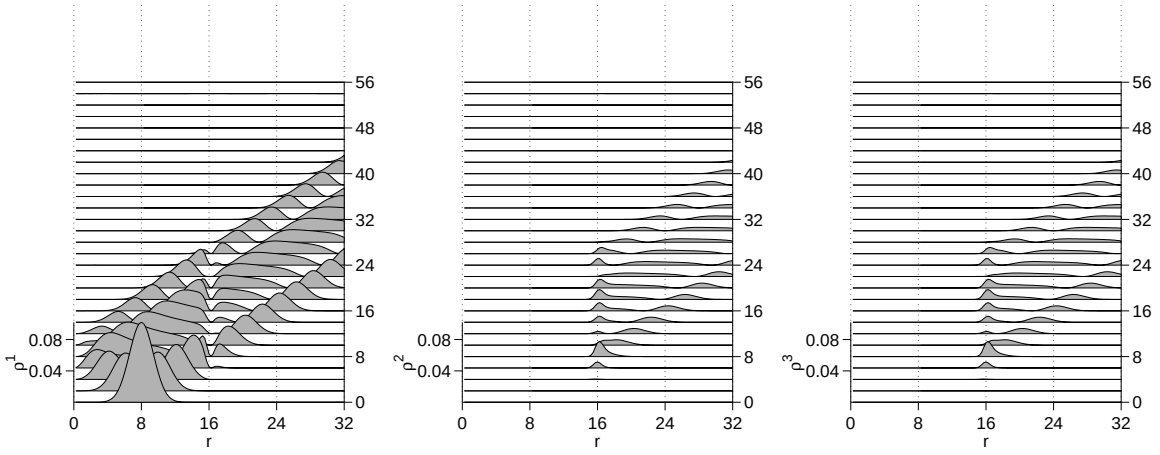


Figura 8.11: *YM4 - Energia total*

Outro ponto marcante dessa solução é que em $r = 16$ a energia na primeira cor é nula. Isso implica que a energia foi totalmente migrada para as outras cores. Porém uma parte dela retorna a cor original logo em seguida.



(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.12: *YM4 - Energia* $\times r$

8.5 Sobreposição de Pulsos Energético e de Calibre

Nessa solução colocamos um pulso energético e um de calibre na mesma posição espacial.

$YM5$:

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.1	16	8	$(0, 1)$
2	1	16	1	(1)
3	0	0	0	(0)

Vamos observar que esse arranjo divide a energia em duas partes aproximadamente iguais entre a primeira e terceira cor. Na Figura 8.13 vemos que isso ocorre antes de $t \approx 4$. Após um período se propagando dentro do domínio em $t \approx 16$ os pulsos em ambas as cores começam a deixar o domínio. Porém isso ocorre mais rápido com o pulso da primeira cor. Enquanto que o pulso na terceira cor deixa o domínio de forma mais abrupta.

Outra característica que podemos observar nesse gráfico das energias é que em $t \approx 32$ ocorre outra troca de energia entre as cores. Essa troca corresponde a energia transportada pelo pulso que se dirigiu inicialmente a origem e foi refletido chegando ao pulso de calibre novamente em $t \approx 32$. Note que nesse processo a energia é inicialmente transferida da primeira para a terceira cor. Porém logo em seguida a energia retorna a primeira cor.

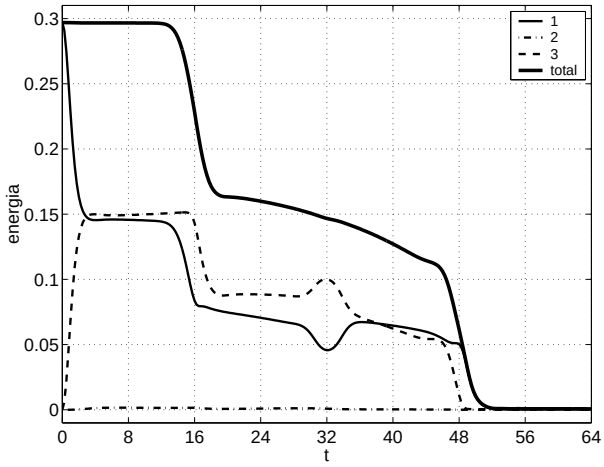
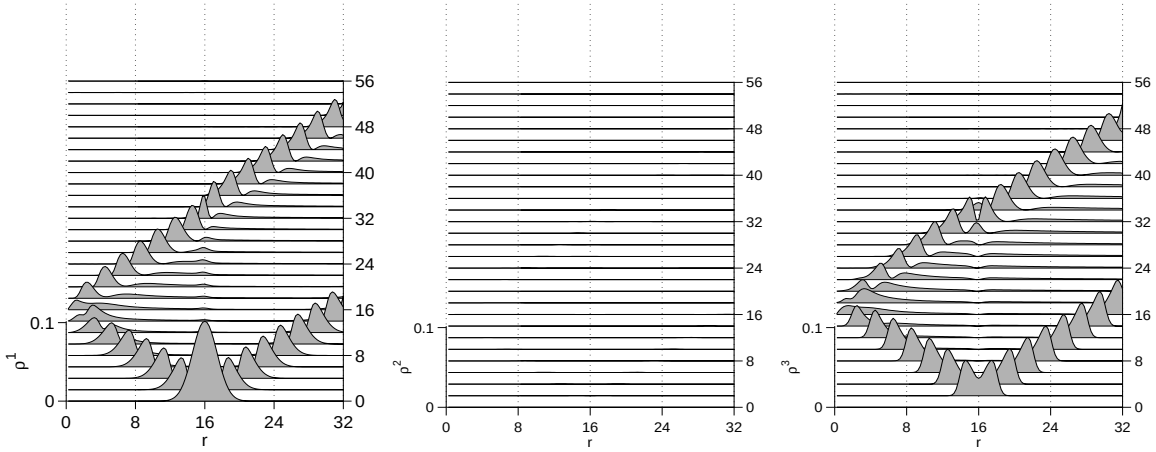


Figura 8.13: $YM5$ - *Energia total*

Na Figura 8.14 estão os gráficos das densidades de energia radial amostradas ao longo do tempo. Como nos casos anteriores na condição inicial toda a energia está na primeira cor, distribuída segundo uma gaussiana de centro em $r = 16$. Como o pulso de calibre está sobreposto ao pulso energético seus efeitos são imediatamente percebidos. Logo no início da evolução, a energia é dividida entre a primeira e a terceira cor. Após essa etapa, os pulsos seguem independentes até $t \approx 32$, quando os pulso que foram refletidos na origem voltam a interagir com o pulso de calibre. Com base nos resultados vistos até o momento podemos afirmar que esses pulsos são transferidos um da primeira para a terceira e o outro no sentido inverso. Podemos confirmar esse fato observando as

curvaturas dos pulsos antes e após a interação. Na primeira cor, o pulso que chega tem derivada maior do lado esquerdo, enquanto que, o que sai, a derivada maior está do lado direito. Uma disposição complementarmente oposta é observada na evolução da energia na terceira cor. Podemos então concluir que o pulso com o lado esquerdo mais íngreme, é transportado da primeira para a terceira cor. Enquanto que, o pulso com o lado direito mais íngreme, é transportado da terceira cor para a primeira.



(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.14: $YM5$ - *Energia* $\times r$

8.6 Interação Entre Dois Pulsos Energéticos

Nas seções anteriores apresentamos várias soluções onde um pulso energético em uma cor interage com um pulso de calibre em outra cor. Nessas soluções pudemos ver com clareza a energia migrar entre as cores do campo. Vamos agora apresentar algumas soluções nas quais dois pulsos energéticos interagem entre si. A solução definida pelos parâmetros abaixo tem, como condição inicial, um pulso de amplitude 0.5 centrado em $r = 8$ na primeira cor e um pulso, também de amplitude 0.5, centrado em $r = 16$ na segunda cor, ambos com a mesma composição multipolar $c = (0, 1)$ e a mesma largura $L = 8$.

$YM6$:

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.5	8	8	$(0, 1)$
2	0.5	16	8	$(0, 1)$
3	0	0	0	(0)

Podemos ver a evolução da energia na Figura 8.15. A evolução da energia total obedece ao padrão esperado permanecendo constante por um pequeno intervalo de tempo e se dispersando completamente em seguida. A energia da primeira e segunda cores são aproximadamente iguais na condição inicial, enquanto que não há nenhuma energia na terceira cor. Nos primeiros momentos da evolução, a primeira e segunda cores perdem, aproximadamente, a mesma quantidade de energia para a terceira cor. Porém em $t \approx 16$ a solução transfere muito mais energia da segunda cor do que da primeira. Ao ponto que pouco antes de $t \approx 20$ a energia contida na terceira cor praticamente se iguala à energia contida na segunda. Porém em $t \approx 40$ a energia da primeira cor é que sofre uma rápida transferência, praticamente se anulando.

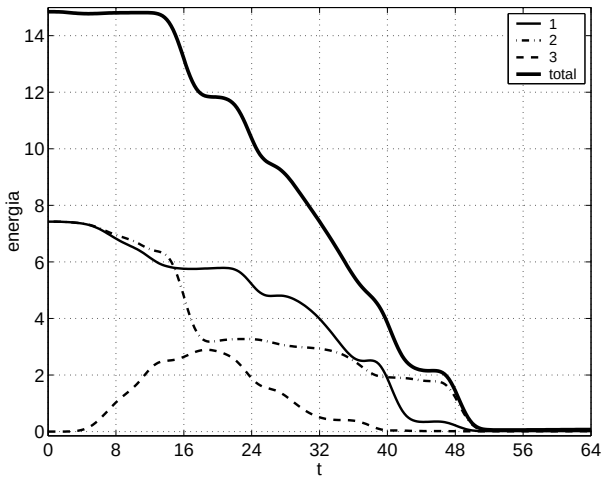
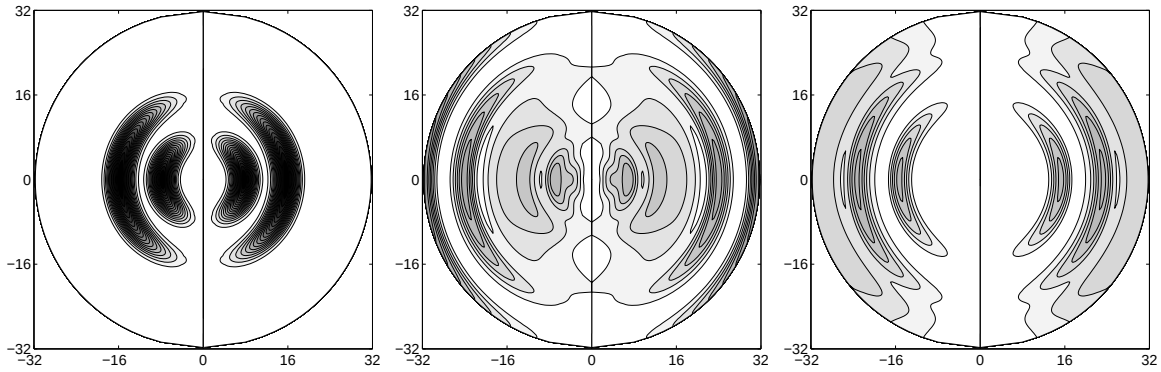


Figura 8.15: *YM*6 - *Energia total*



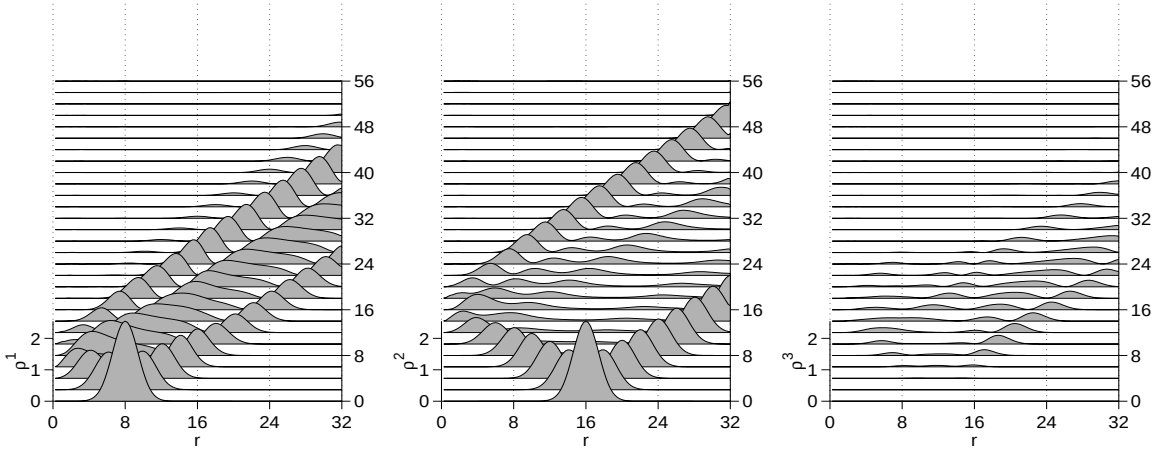
(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 8.16: *YM*6 - *Densidade de energia* $r^2\rho$

Na Figura 8.16 estão os gráficos das densidades de energia total dessa solução nos instantes $t = 0, 16, 32$. No primeiro gráfico aparecem os dois pulsos construídos na condição inicial. Como esperado, a dinâmica da energia total não sofre nenhuma influência da distribuição interna da energia entre as cores.

A interação entre as cores pode ser visualizada com mais clareza nos gráficos da Figura 8.17. No primeiro gráfico está a densidade radial de energia na primeira cor mostrada

ao longo do tempo. Nesse gráfico podemos ver o pulso inicial centrado em $r = 8$ e sua dinâmica com a passagem do tempo. O segundo gráfico exibe a densidade de energia contida na segunda cor, enquanto que o terceiro gráfico exibe a energia na terceira. Nesse último, vemos que no instante inicial não há nenhuma energia nessa componente do campo. Porém, com a passagem do tempo, parte da energia das outras cores é transferida para essa cor. O máximo de energia nessa cor ocorre, aproximadamente quando $t \approx 20$.



(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.17: *YM*6 - *Energia* $\times r$

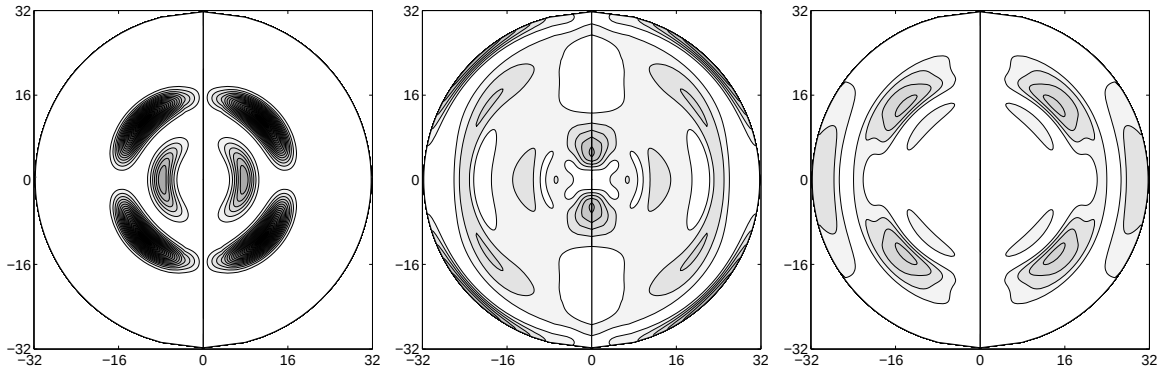
A quantidade de energia transferida para a terceira cor é fortemente dependente da amplitude dos pulsos iniciais, devido aos efeitos não lineares. Na próxima seção apresentaremos uma comparação sistemática dessas grandezas.

Um outro exemplo de interação entre dois pulsos energéticos é definido pelo seguinte conjunto de parâmetros:

$YM7$:

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.5	8	8	$(0, 1)$
2	0.5	16	8	$(0, 0, 1)$
3	0	0	0	(0)

A distribuição da energia determinada por esses parâmetros está nos gráficos da Figura 8.18. O primeiro gráfico exibe a densidade de energia no instante inicial. Essa condição inicial se distingue da anterior pela composição multipolar do pulso na segunda cor, que, no gráfico, é o pulso mais externo. O segundo e terceiro gráficos exibem a distribuição da energia nos instantes $t = 16$ e $t = 32$.



(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 8.18: *YM7 - Densidade de energia* $r^2\rho$

O gráfico da Figura 8.19 exibe a evolução da energia total do campo e de cada uma de suas cores. O comportamento da energia total do campo, como esperado, se mantém constante até $t \approx 14$. Em seguida abandona o domínio em um ritmo quase constante e se anula em $t \approx 50$. A energia da primeira cor começa próxima de 7.5 e tem um comportamento bastante oscilatório até se anular também em $t \approx 50$. Na condição inicial a energia da segunda cor é mais do que o dobro da energia da primeira cor. Isso é uma conseqüência do raio do pulso e de sua composição multipolar. A energia nessa cor quase sempre decresce, porém com velocidades bem diferentes. A terceira cor começa sem nenhuma energia e é energizada pela interação dos pulsos iniciais. O ponto de maior energia nessa cor ocorre em $t \approx 16$.

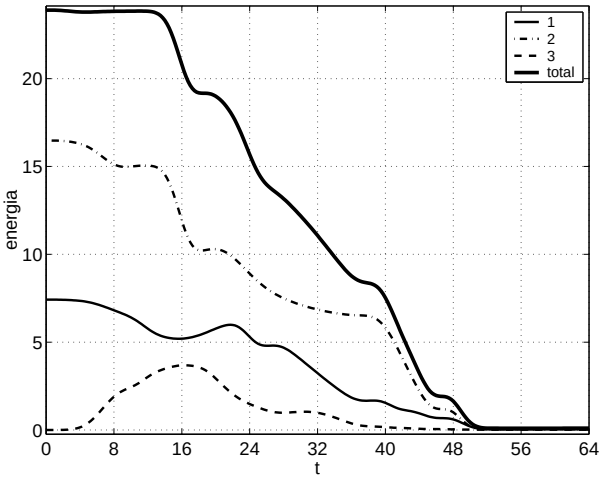
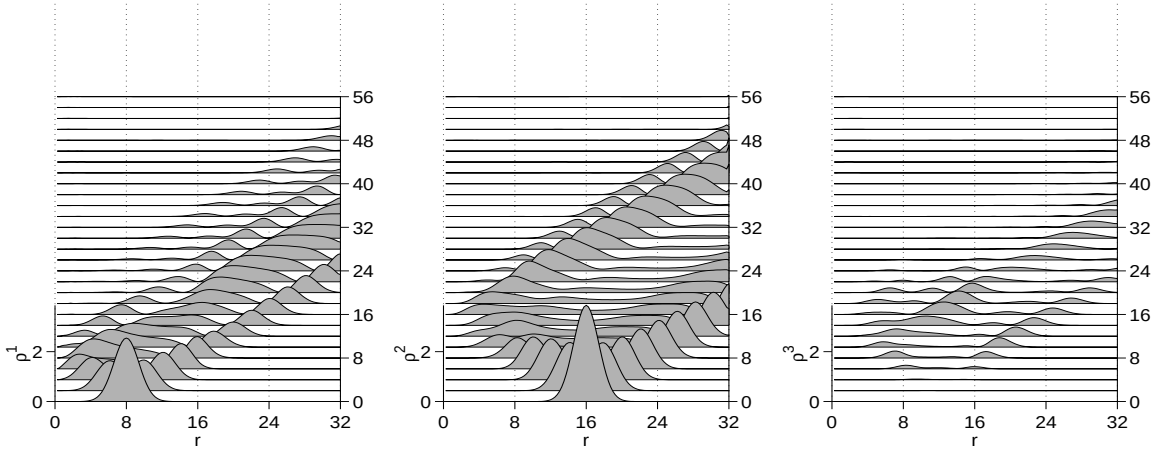


Figura 8.19: *YM7 - Energia total*

Os gráficos da Figura 8.20 exibem a densidade de energia radial dessa solução. O terceiro gráfico exibe a evolução da energia na terceira cor. Vemos que em $t \approx 8$ surgem dois pequenos pulsos de energia em $r \approx 8$ e $r \approx 16$. Esses pulsos ganham intensidade, chegando a seus valores máximos entre $t \approx 16$ e $t \approx 20$. Em seguida, suas amplitudes são reduzidas novamente, até que eles deixam o domínio de simulação bastante dispersos.

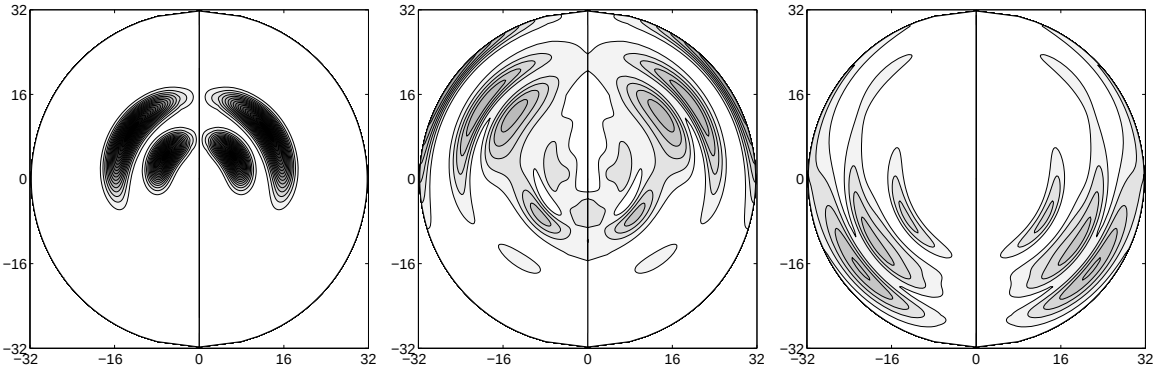


(a) $E_r^{\hat{1}}$ (b) $E_r^{\hat{2}}$ (c) $E_r^{\hat{3}}$

Figura 8.20: *YM*7 - *Energia* $\times r$

8.7 Soluções Multipolares

Obtivemos também soluções não simétricas com relação a reflexão sobre o plano $\chi = 0$, ou $z = 0$ em coordenadas cilíndricas. A seguir estão dois exemplos de resultados desse tipo, soluções 8 e 9.



(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 8.21: *YM*8 - *Densidade de energia* $r^2\rho$

A oitava solução é definida pelos parâmetros

	$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
*YM*8 :	1	0.1	8	8	$(0, 2, 1)$
	2	0.1	16	8	$(0, 2, 1)$
	3	0	0	0	(0)

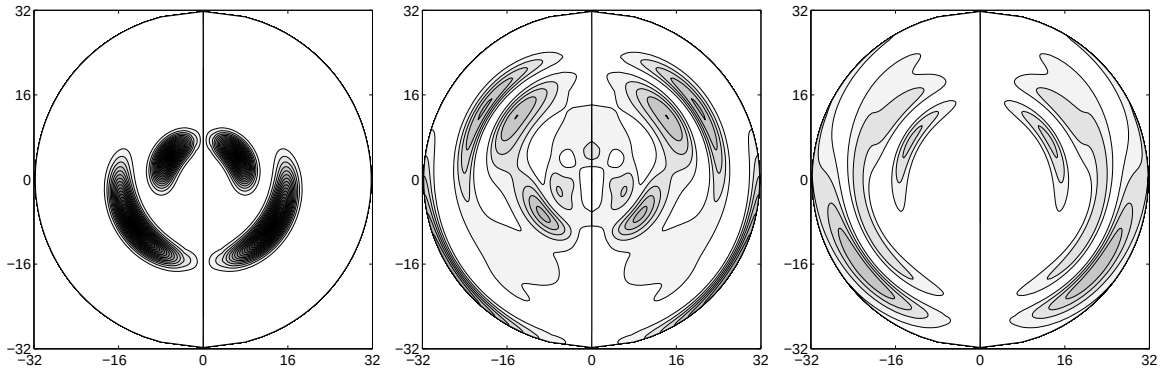
Os pulsos iniciais são cascas de raio 8 e 16, ambos com a mesma composição multipolar $C = (0, 2, 1)$. A densidade de energia obtida com a evolução desses pulsos esta na Figura 8.21.

Outro exemplo de solução multipolar é definida pelos parâmetros

$YM9$:

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	0.1	8	8	$(0, 2, 1)$
2	0.1	16	8	$(0, -2, 1)$
3	0	0	0	(0)

A densidade de energia correspondente a essa solução esta na Figura 8.22.



(a) $t = 0$ (b) $t = 16$ (c) $t = 32$

Figura 8.22: $YM9$ - *Densidade de energia* $r^2\rho$

A dinâmica entre as três cores do campo de Yang-Mills, obtidos por essa simulação, estão nos gráficos da Figura 8.23. O primeiro conjunto de curvas de contorno exibem as densidades de energia de cada cor do campo no instante inicial. As duas primeiras cores são inicializadas com pulsos energéticos, enquanto que a terceira cor começa identicamente nula. Seguido a figura para baixo cada conjunto de gráficos corresponde a densidade de energia em um instante de tempo entre 4 e 36. Na terceira coluna vemos o surgimento e posterior desaparecimento de pequenos pulsos de energia. Esses pulsos são gerados pela interação entre os pulsos das duas primeiras cores.

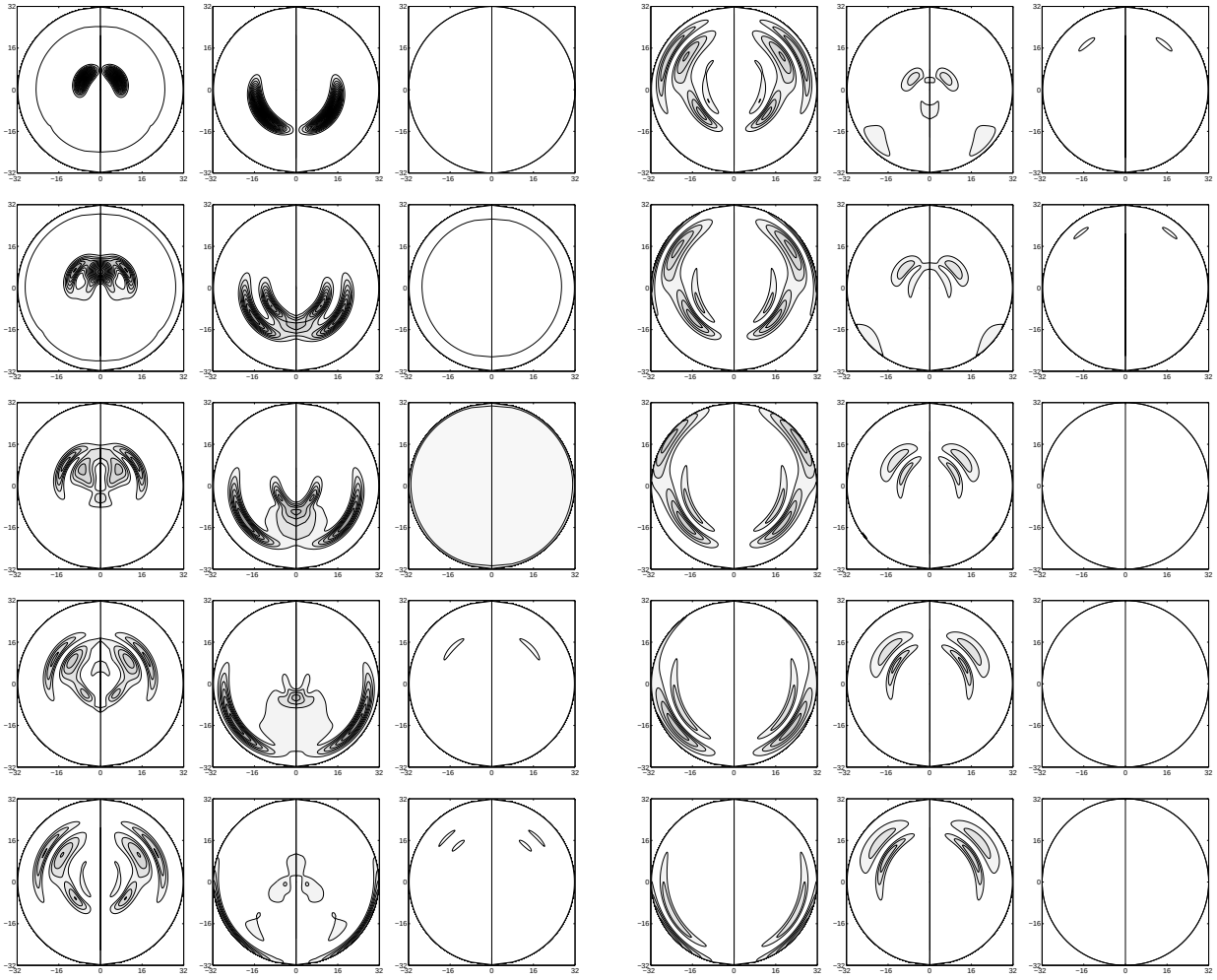


Figura 8.23: *YM*1 - *Densidade de energia nas três cores,* $r^2\rho^{\hat{a}}$ *nos tempos* $t = 0, 4, \ldots, 36$

8.8 Amplitude do Pulso e Transferência de Energia

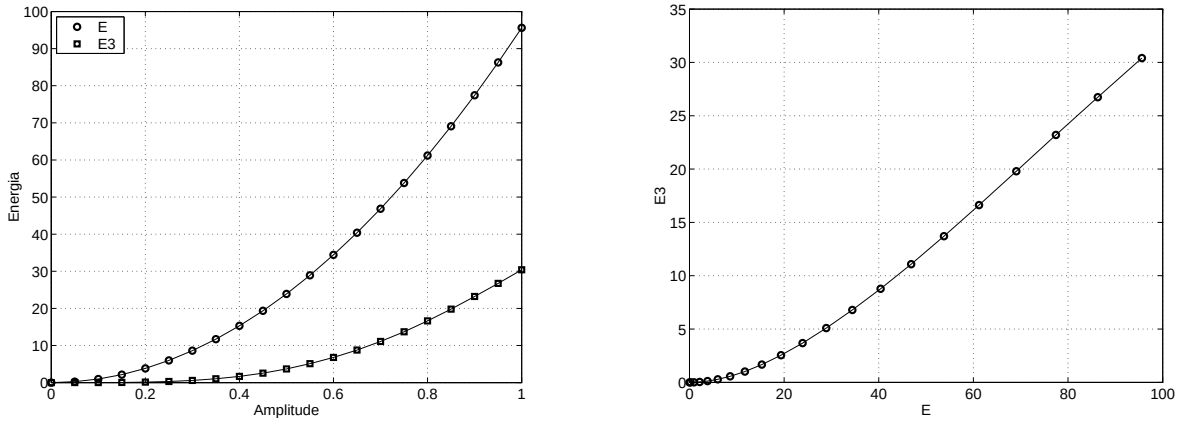
A transferência de energia entre as cores é uma conseqüência da interação não linear entre essas cores. Vamos agora comparar as amplitudes do pulso inicial com as amplitudes das energias total e em cada cor. Vamos comparar os resultados de uma família de 20 soluções obtidas a partir de condições iniciais iguais, exceto pela amplitude do pulso, $\alpha^{\hat{a}}$.

A tabela abaixo exibe os valores utilizados.

$\hat{a}$	$\alpha^{\hat{a}}$	$r_0^{\hat{a}}$	$L^{\hat{a}}$	$c_n^{\hat{a}}$
1	$0.05 : 0.05 : 0.1$	8	8	$(0, 1)$
2	$0.05 : 0.05 : 0.1$	16	8	$(0, 0, 1)$
3	0	0	0	(0)

Os valores de $r_0^{\hat{a}}$, $L^{\hat{a}}$ e $c_n^{\hat{a}}$ são os mesmos para todas as soluções, enquanto que o valor de $\alpha^{\hat{a}}$ varia de 0.05 na primeira solução até 0.1 na última, sendo acrescido de 0.05 de uma solução para a seguinte.

Nessas soluções a energia começa distribuída entre os pulsos na primeira e segunda cores. Com a interação desses dois pulsos ocorre uma transferência de energia dessas duas cores para a terceira cor, que inicialmente não continha nenhuma energia. Os gráficos da Figura 8.24 comparam a energia total no instante inicial, E, com o valor máximo da energia que foi transferida para a terceira componente, $\max(E^{\hat{3}})$.



(a) E *e* $\max(E^{\hat{3}}) \times \alpha^{\hat{a}}$ (b) $\max(E^{\hat{3}}) \times E^{\hat{1}}$

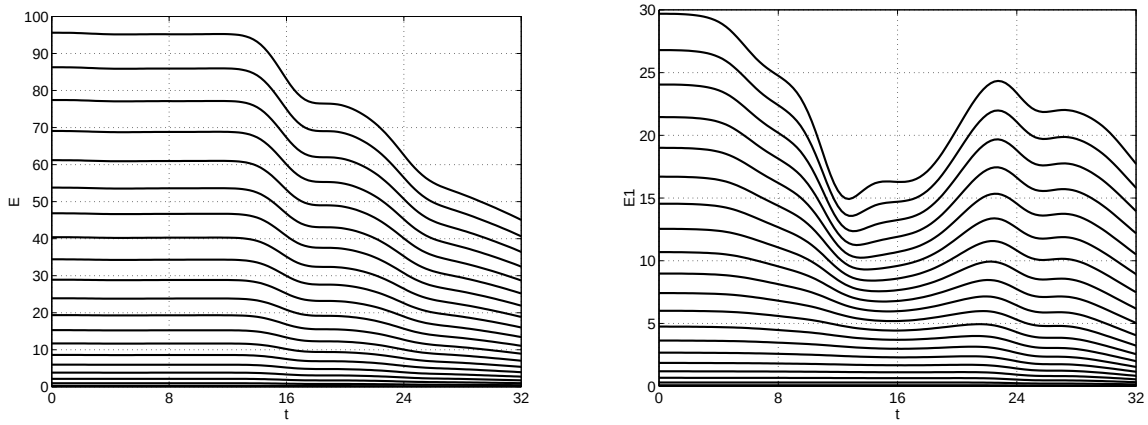
Figura 8.24: *Comparação entre energia e amplitude inicial*

O primeiro gráfico da Figura 8.24 ilustra o crescimento das energias, E e $\max(E^{\hat{3}})$, em comparação com o crescimento da amplitude inicial nas duas primeiras cores, $\alpha^{\hat{a}}$. A linha que representa o crescimento da energia total no instante inicial é exatamente uma quadrática, como esperado pela teoria. Porém a energia transferida para a terceira cor tem um comportamento mais complexo. Para amplitudes maiores o incremento na amplitude parece ter um efeito menos significativo sobre o valor máximo da energia da terceira cor. Isso pode ser observado também no segundo gráfico dessa figura. Esse gráfico compara o valor máximo da energia transferida para a terceira cor, $\max(E^{\hat{3}})$, com o valor da energia

total, E. Vemos nesse gráfico que em soluções pouco energéticas, a transferência de energia é quase nula, mas que quando a energia total aumenta, a energia transferida cresce de forma aparentemente linear.

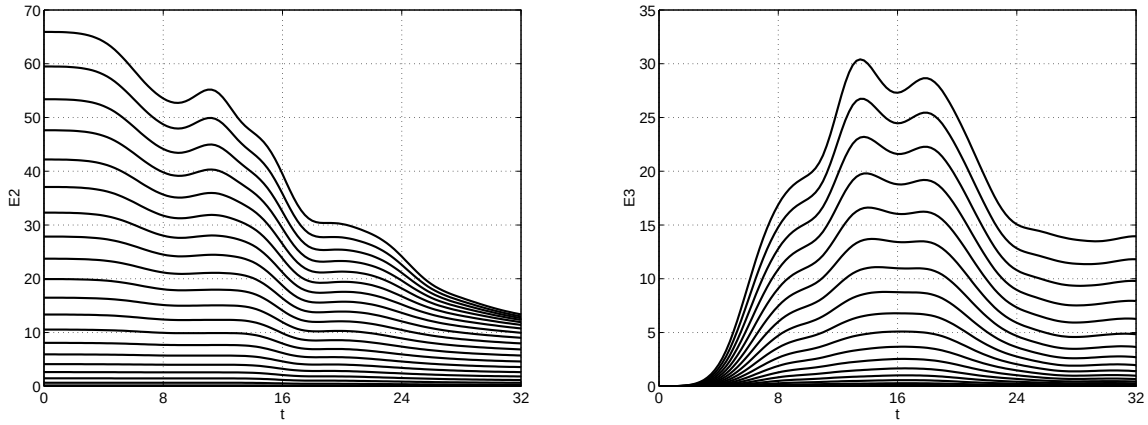
Mais informações sobre a dependência da transferência de energia com relação a amplitude do pulso inicial pode ser obtida quando analisamos a evolução da energia com a passagem do tempo.

O comportamento qualitativo da energia total com a passagem do tempo não sofre nenhuma influência da amplitude inicial, como pode ser observado no primeiro gráfico da Figura 8.25. Enquanto os pulsos trafegam pelo interior do domínio a energia total é constante. Quando os pulsos alcançam a fronteira exterior, a energia total decai terminando



(a) *Energia Total* $\times$ *tempo* (b) $E^{\hat{1}}$ $\times$ *tempo*

Figura 8.25: *Energia* $\times$ *tempo*



(a) $E^{\hat{2}}$ $\times$ *tempo* (b) $E^{\hat{3}}$ $\times$ *tempo*

Figura 8.26: *Energia* $\times$ *tempo*

por se anular. Entretanto, a dinâmica interna da energia entre as cores se torna cada vez mais complexa quando a amplitude inicial cresce. Esse aumento da complexidade pode ser visto no segundo gráfico da Figura 8.25 e nos dois gráficos da Figura 8.26. Nesses gráficos estão desenhadas as energias na primeira, segunda e terceira cor, respectivamente. Para soluções pouco energéticas, as energias na primeira e segunda cor têm uma dinâmica muito semelhante a da energia total. Elas permanecem aproximadamente constantes por algum tempo e depois decaem conforme o pulso deixa o domínio, e quase nenhuma energia é transferida para a terceira cor. Porém, quando a energia inicial cresce, podemos observar que uma parte significativa da energia é transferida para a terceira cor.

9 Conclusões

Como o nosso objetivo é estudar numericamente o campo de Einstein-Yang-Mills axialmente simétrico buscamos uma formulação das equações adequada a esse fim. No caso esfericamente simétrico todas as soluções possíveis são descritas pelo ansatz de Witten. Precisávamos determinar um ansatz adequado para simetria axial. Fizemos isso determinando a condição de simetria para um campo de calibre (2.4) e fazendo as escolhas de calibre apropriadas, como apresentado na Seção 4.2.

As equações diferenciais construídas são compostas por um sistema de seis equações hiperbólicas e três elípticas. As equações elípticas, tratadas como vínculos, são preservadas pela evolução temporal, segundo as equações hiperbólicas. Desse modo precisamos resolver o sistema de equações elípticas apenas no primeiro instante de tempo. Fizemos isso construindo algumas soluções analíticas particulares, descritas na Seção 6.2.

Para resolver numericamente as equações, implementamos uma suite em Matlab. Essa suite é composta de dois métodos de discretização espacial, diferenças finitas e pseudo espectral, e algumas versões do método de Runge-Kutta para a evolução temporal. A suite foi implementada na forma de objetos de forma que os métodos numéricos podem ser trocados independentemente das equações per se. Usamos também métodos de quadratura convencionais necessários para a determinação da energia contida no campo. Consideramos essa suite como uma importante contribuição desse trabalho que pode ser utilizada para explorar outras soluções, bastando apenas trocar os parâmetros apropriados.

A estrutura numérica construída nos permitiu comparar os resultados obtidos com diferenças finitas e o método pseudo espectral. Percebemos que o método de diferenças finitas foi capaz de gerar bons resultados em todas as equações estudadas sendo, entretanto, mais exigente em termos da resolução da malha. O método pseudo espectral foi capaz de gerar ótimos resultados com malhas, aparentemente, grosseiras quando empregado para a

discretização de equações lineares. Porém, quando empregado nas equações não lineares, o método pseudo espectral se desestabilizava não sendo capaz de reproduzir resultados obtidos com diferenças finitas.

O campo de Yang-Mills é muito pouco estudado no universo da física clássica. Existem poucas soluções conhecidas e todas com alto grau de simplificação. Mesmo assim, resultados relevantes foram obtidos. Exemplos significativos para a relatividade geral são as soluções de Bartnik-McKinnon, descritas na Seção 3.4 e conhecidas como Buracos Negros Coloridos, e os estudos de fenômenos críticos no colapso gravitacional.

Na Seção 3.6 apresentamos soluções para o campo de Einstein-Yang-Mills gerados a partir da formulação esfericamente simétrica dos campos. Nessas soluções podemos acompanhar a evolução de uma casca esférica implodindo e se dispersando para o infinito. Assim como o campo gravitacional gerado no processo.

O campo axialmente simétrico permite uma dinâmica muito mais elaborada. Basta notar que o campo de Yang-Mills esfericamente simétrico é descrito por uma única variável em uma dimensão espacial enquanto que para o caso axialmente simétrico estudado são necessárias seis variáveis em duas dimensões espaciais.

No Capítulo 7 foram exibidas várias soluções particulares do campo de Yang-Mills, que correspondem a um campo Abeliano embebido em SU(2). Acompanhamos a evolução temporal de pulsos axialmente simétricos com distribuições iniciais com formato toroidal.

O campo de Yang-Mills SU(2) é composto por três campos internos, que chamamos de cores. Com a simulação numérica desse campo explicitamos a dinâmica dessas cores. Nos resultados apresentados no Capítulo 8 podemos acompanhar a evolução temporal da dinâmica desses campos internos. Observamos que pulsos energéticos podem interagir com pulsos de puro calibre provocando um transferência da energia entre as cores do campo. Também exibimos a interação entre pulsos energéticos de diferentes cores. Nesse caso ocorre uma transferência de energia entre as cores, mas com características muito diferentes em comparação ao caso anterior.

Esse trabalho pode ser considerado como um primeiro passo para o estudo do campo de Einstein-Yang-Mills axialmente simétrico por métodos numéricos. Existem várias opções para dar continuidade e esse trabalho. Do ponto de vista numérico é possível aperfeiçoar a implementação dos métodos ou incluir variações mais adequadas ao problema. Computacionalmente pode-se ganhar muito em eficiência com a reimplementação, ao menos parcial, dos métodos em uma linguagem de programação compilada, como Fortran ou C++.

Por outro lado existe uma variedade muito grande de problemas físicos que podem ser estudados com extensões dessa suite numérica. Uma seqüência possível de trabalho seria: Estudar o campo de Yang-Mills sob o efeito do campo gravitacional de um buraco negro; Fazer uma simulação numérica do problema completo do campo de Yang-Mills axialmente simétrico e auto gravitante; Estudar sobre o colapso desse sistema. Esse último passo poderia, por exemplo, responder questões sobre a formação dos buracos negros coloridos e sua relação com a conjectura de que buracos negros não têm cabelos.

Referências Bibliográficas

[1] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. John Wille & Sons, 10 edition, 1972.

[2] Alfred Actor. Classical solutions of SU(2) Yang-Mills theories. *Rev. Modern Physics*, 51(3):461–525, July 1979.

[3] R. Bartnik and J. McKinnon. Particle like solutions of Einstein Yang-Mills equations. *Phys. Rev. Lett.*, 61:141–144, 1988.

[4] Piotr Bizon. Colored black holes. *Phys. Rev. Lett.*, 64(24):2844–2847, 1990.

[5] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, 1987.

[6] Matthew W. Choptuik. Universality and scaling in gravitational collapse of a massless scalar field. *Phy, Rev. Letters*, 70(1):9–12, 1993.

[7] Matthew W. Choptuik, Tadeusz Chmaj, and Piotr Bizoń. Critical behavior in gravitational collapse of a Yang-Mills field. *gr-qc/9603051*, pages 1–5, 1996.

[8] Matthew W. Choptuik, Eric W. Hirschmann, and Robert L. Marsa. New critical behavior in Einstein-Yang-Mills collapse. *Phys. Rev. D*, 60(12):124011(9), 1999.

[9] C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics*. Springer-Verlag, 2 edition, 1991.

[10] P. Forgács and N. S. Manton. Space-time symmetries in gauge theories. *Commun. Math. Phys.*, 72:15–35, 1980.

[11] Bengt Fornberg. *A Pratical Guide to Pseudospectral Methods*. Cambridge, 1996.

[12] D. Gottlieb and S. A. Orszag. Numerical analysis of spectral methods: Theory and applications. SIAM-CMBS, 1977.

[13] Carsten Gundlach. Critical phenomena in gravitational collapse. *Physics Reports*, 6(376):339–405, 2003.

[14] M. Ikeda and Y. Miyachi. On the static and spherically symmetric solutions of the Yang-Mills field. *Prog. Theor. Phys.*, 27:474–481, 1962.

[15] J. D. Jackson. Historical roots of gauge invariance. *Reviews of Modern Physics*, 73:663–680, 2001.

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - The Art of Scientific computing*. Cambridge University Press, 2 edition, 1992.

[17] Joel A. Smoller, Arthur G. Wasserman, S. T. Yau, and J. B. McLeod. Smooth static solutions of the Einstein/Yang-Mills equations. *Communications in Mathematical Physics*, 143(1):115–147, 1991.

[18] Walter A. Strauss. *Nonlinear Wave Equations*. American Mathematical Society, 1989.

[19] Lloyd N. Trefethen. *Spectral Methods in Matlab*. SIAM, 2000.

[20] J. A. C. Weideman and S. C. Reddy. A matlab differentiation matrix suite. *ACM Transactions on Mathematical Software*, 26(4):465–519, 2000.

[21] Herman Weyl. Eine neue Erweiterung der Relativitätstheorie. *Ann. Phys (Leipzing)*, 59:101–133, 1919.

[22] Edward Witten. Some exact multipseudoparticle solutions of classical Yang-Mills theory. *Phys. Rev. Let.*, 38(3):121–124, January 1977.

[23] T. T. Wu and C. N. Yang. Properties of matter under unusual conditions. *Interscience, New York*, 1968.

[24] C. N. Yang and R. L. Mills. Conservation of isotopic spin and isospin gauge invariance. *Phys. Rev.*, 96:191–195, 1954.

[25] M. Yu. Zotov. Dynamical system analysis for the Einstein-Yang-Mills equations. *Journal of Mathematical Physics*, 41(7):4790–4807, 2000.

O Grupo SU(2) A

A.1 As Matrizes de Pauli

Os geradores de SU(2) que usaremos são, $\tau^{\hat{a}} = \sigma^{\hat{a}}/2i$, onde $\sigma^{\hat{a}}$ são as matrizes de Pauli, satisfazendo as seguintes relações de comutação e anti-comutação

$$\left[\tau^{\hat{a}}, \tau^{\hat{b}}\right] = \varepsilon^{\hat{a}\hat{b}\hat{c}}\tau^{\hat{c}}$$

$$\left\{\tau^{\hat{a}}, \tau^{\hat{b}}\right\} = -\frac{1}{2}\delta^{\hat{a}\hat{b}}I$$

onde I é a matriz unidade de dimensão 2×2.

Os geradores usuais para o grupo SU(2) são as matrizes de *spin* de Pauli. Estas matrizes são definidas como

$$\sigma^{\hat{x}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \sigma^{\hat{y}} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \sigma^{\hat{z}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{A.1}$$

O produto destas matrizes satisfaz a seguinte identidade

$$\sigma^{\hat{a}}\sigma^{\hat{b}} = \delta^{\hat{a}\hat{b}} + i\varepsilon^{\hat{a}\hat{b}\hat{c}}\sigma^{\hat{c}}$$

Entretanto, em algumas ocasiões, pode ser conveniente usar a seguinte combinação linear destas matrizes

$$\sigma^{\pm} = \frac{1}{2}\left(\sigma^{\hat{x}} \pm i\sigma^{\hat{y}}\right)$$

$$\sigma^{+} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad \sigma^{-} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

Os elementos de SU(2) podem ser gerados a partir destas matrizes pela expressão

$$U = \exp\left(-\frac{i\theta}{2}n \cdot \sigma\right) = \cos\frac{\theta}{2}\, I + i\,\mathrm{sen}\frac{\theta}{2}\ (n \cdot \sigma)$$

onde usamos a notação de angulo axial[1] para rotações, isto é, uma rotação de um ângulo θ sobre o eixo n. Estamos considerando que n é um vetor unitário e que $\sigma = \left(\sigma^{\hat{x}}, \sigma^{\hat{y}}, \sigma^{\hat{z}}\right)$. Em termos das componentes temos

$$U = \begin{bmatrix} \cos\dfrac{\theta}{2} - in^{\hat{z}}\,\mathrm{sen}\dfrac{\theta}{2} & -\left(n^{\hat{y}} - in^{\hat{x}}\right)\,\mathrm{sen}\dfrac{\theta}{2} \\ \left(n^{\hat{y}} - in^{\hat{x}}\right)\,\mathrm{sen}\dfrac{\theta}{2} & \cos\dfrac{\theta}{2} - in^{\hat{z}}\,\mathrm{sen}\dfrac{\theta}{2} \end{bmatrix}$$

A.2 Projeções Esféricas

Para trabalhar com campos com simetria esférica em Relatividade Geral, alguns autores definem as chamadas projeções esféricas das matrizes de Pauli

$$\begin{aligned} \sigma^{\hat{r}} &= \sigma^{\hat{x}}\,\mathrm{sen}\theta\cos\varphi + \sigma^{\hat{y}}\cos\theta\cos\varphi - \sigma^{\hat{z}}\,\mathrm{sen}\varphi \\ \sigma^{\hat{\phi}} &= \sigma^{\hat{x}}\,\mathrm{sen}\theta\,\mathrm{sen}\varphi + \sigma^{\hat{y}}\cos\theta\,\mathrm{sen}\varphi + \sigma^{\hat{z}}\cos\varphi \\ \sigma^{\hat{\theta}} &= -\sigma^{\hat{x}}\,\mathrm{sen}\theta + \sigma^{\hat{y}}\cos\theta \end{aligned} \tag{A.2}$$

Explicitamente estas matrizes são

$$\sigma^{\hat{r}} = \begin{bmatrix} \cos\theta & \mathrm{sen}\theta\, e^{-i\phi} \\ \mathrm{sen}\theta\, e^{i\phi} & -\cos\theta \end{bmatrix} \tag{A.3}$$

$$\sigma^{\hat{\phi}} = \begin{bmatrix} -\,\mathrm{sen}\theta & \cos\theta\, e^{-i\phi} \\ \cos\theta\, e^{i\phi} & \mathrm{sen}\theta \end{bmatrix} \tag{A.4}$$

$$\sigma^{\hat{\theta}} = \begin{bmatrix} 0 & -ie^{-i\phi} \\ ie^{i\phi} & 0 \end{bmatrix} \tag{A.5}$$

A.3 Projeções Axiais

Trabalhando com campos com simetria axial, pode-se usar as seguintes projeções das matrizes de Pauli

$$\begin{aligned} \sigma^{\hat{\rho}} &= \sigma^{\hat{x}}\cos\varphi + \sigma^{\hat{y}}\,\mathrm{sen}\varphi \\ \sigma^{\hat{\varphi}} &= -\sigma^{\hat{x}}\,\mathrm{sen}\varphi + \sigma^{\hat{y}}\cos\varphi \\ \sigma^{\hat{z}} &= \sigma^{\hat{z}} \end{aligned}$$

Podemos escrever estas matrizes como

$$\sigma^{\hat{\rho}} = \begin{bmatrix} 0 & e^{-i\varphi} \\ e^{i\varphi} & 0 \end{bmatrix} \qquad \sigma^{\hat{\varphi}} = \begin{bmatrix} 0 & -i\,e^{-i\varphi} \\ i\,e^{i\varphi} & 0 \end{bmatrix}$$

[1] axis-angle

A.4 Normalização

A base realmente usada na literatura são renormalizações das matrizes de Pauli, ou das suas projeções. Estas renormalizações são escolhidas para satisfazerem condições de comutabilidade, tais como

$$\left[\tau^{\hat{a}}, \tau^{\hat{b}}\right] = \beta\, \varepsilon^{\hat{a}\hat{b}\hat{c}} \tau^{\hat{c}}$$

Então o traço do quadrado destas matrizes torna-se

$$\mathrm{tr}\left(\tau^{\hat{a}}\right)^2 = -\frac{\beta^2}{2}$$

Na comunidade que estuda a física de altas energias, a escolha mais comum é $\beta = i$, neste caso

$$\tau^{\hat{a}} = \frac{1}{2}\sigma^{\hat{a}}$$

Mas no caso do campo de EYM alguns autores escolhem $\beta = 1$, obtendo, portanto,

$$\tau^{\hat{a}} = -\frac{i}{2}\sigma^{\hat{a}}$$

e as matrizes τ são anti-simétricas $\left(\tau^{\hat{a}}\right)^{\dagger} = \tau^{\hat{a}}$

Listagens dos Programas B

Incluímos nesse apêndice as listagens dos programas desenvolvidos. Esses programas foram escritos em *Matlab*, pois essa linguagem oferece grande versatilidade e facilidade para o implementador. Esses programas foram implementados de modo que permita a fácil manipulação das equações e métodos. Tentou-se também fazer com que esses programas fossem os mais legíveis possíveis. Todos os arquivos são comentados e incluem um pequeno cabeçalho que é usado pelo *Matlab* como mensagem de ajuda para um eventual usuário do programa.

Do ponto de vista da implementação os próximos passo para o desenvolvimento dessa suite podem ser:

- otimizar a implementação de diferenças finitas,
- aperfeiçoar o salvamento dos resultados parciais,
- ajustar a implementação para obter maior economia de memória,
- testar outros métodos de evolução temporal.

A seqüência das listagens segue, na medida do possível, a ordem em que os programas são chamados. A primeira listagem apresenta o programa principal. Em seguida aparecem as listagens das funções chamadas por esse programa. As duas últimas seções, quadratura e gráficos, contém programas que foram utilizados apenas na geração dos gráficos apresentados nesse trabalho.

B.1 Programas para o Caso Esfericamente Simétrico

Nessa seção estão listados os programas que definem e resolvem as equações de Einstein-Yang-Mills com simetria esférica.

O Programa Principal

Esse é o programa que define todos os parâmetros para a execução dos métodos numéricos. Ele seleciona a equação, os métodos individuais e seus parâmetros. Todas as funções, com exceção apenas das funções de quadratura e gráficos, são chamadas por esse programa.

```
1
% EYMWAVE - Matlab Script - Driver for Wave Solver

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Jul, 26, 2003
% $Id: SphericalWave.m,v 1.16 2007/05/26 03:39:44 akiles Exp $
%-------------------------------------------------------------------------------%

% Spatial Grid
%-------------------------------------------------------------------------------%
%   .Nr - Number of points
%   .Ra - Begining of interval
%   .Rb - End of interval
%
% WaveSolver will create R - Column vector with values of r

% Removing possible angular grid
if exist('SpG'), clear('SpG'); end;

SpG.Nr = 2^8;
SpG.Ra = 2^-4;
SpG.Rb = 32;

% Time Interval
%-------------------------------------------------------------------------------%
%  Tf - Final time
%  dt - Interval between samples
% WaveSolver will create T - Column vector with values of t
TIn.Tf = 64; %% 2*SpG.Rb;
TIn.dt = 2^-2;

% 1D and Spherically Symmetric Wave Equations
%-------------------------------------------------------------------------------%
Eq = 'EqEYM';

% Equation Parameters
EqP.BC = 1;

% Initial Condition
%-------------------------------------------------------------------------------%
InC.system = 'IC_1D_EYM';

% InC.P - Initial Condition Parameters
%
% A  - Amplitude
% Ro - Center
% L  - Width
% D  = 0 - time symmetric
%      1 - ingoing
%     -1 - outgoing

InC.P.A  = 1;
InC.P.Ro = R;
InC.P.L  = L;
InC.P.D  = D;

% Discretization Methods:
%-------------------------------------------------------------------------------%

DMt.MtR    = 'MtFD';
DMt.Points = 'linspace';
DMt.Order  = 2;

% Time evolution methods
%-------------------------------------------------------------------------------%
% Parameters for evolution procedure:
%   Ev.Mt   - Time step method
%      .mU   - Maximum value of U allowed
%      .dt   - Initial time step
%      .P    - Time step method parameters
Ev.Mt = 'EYM_TimeStep';
Ev.mU = 100;
Ev.dt = 2^-12;

% Local error tolerance
Ev.P.tol = 2^-32;

% Solving
%-------------------------------------------------------------------------------%

% Log information
clear ElapsedTime;
```

```
    fprintf('| Started at %s\n', datestr(now) );
86
    WaveReport

    tic;

91  [ U E ] = WaveSolver( Eq, EqP, DMt, Ev, SpG, TIn, InC );

    ElapsedTime = toc;

    % Saving solution
96  PSsave

    h = floor( ElapsedTime / ( 60*60 ) );
    e = ElapsedTime - h*60*60;
    m = floor( e / 60 );
101 s = round( e - m * 60 );

    fprintf('| Elapsed Time = %i:%02i:%02i\n', h, m, s );

    %-------------------------------------------------------------------------%
```

As Equações de Einstein-Yang-Mills

As equações de Einstein-Yang-Mills são divididas em evolutivas e vínculos. As evolutivas são implementadas pelos programas: EqEYM, EqEYM_Energy, EqEYM_Init e EqEYM_SystemSize. Enquanto que os vínculos são resolvidos nos programas: EqEYM_a, EqEYM_alpha e EqEYM_w.

```
    function F = EqEYM( U )

    % EQEYM - Spherical symmetric Einstein-Yang-Mills equations
5   %
    % usage: F = EqEYM( U )

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
10  % since: May, 26, 2007
    % $Id: EqEYM.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
    %-------------------------------------------------------------------------%

    global Global_Dr
15  global Global_D2r

    dissip = evalin( 'base', 'dissip' );

    N = evalin ( 'base', 'SpG.Nr' );
20  U = reshape( U, N, 5 );
    F = zeros  ( size(U) );

    w     = U(:,1);
    Phi   = U(:,2);
25  Pi    = U(:,3);
    a     = U(:,4);
    alpha = U(:,5);

    aa = alpha ./ a;
30
    W = w .* ( 1 - w.^2 );

    r = evalin( 'base', 'SpG.R' );

35  F(:,2) = Global_Dr*(aa.*Pi ) + dissip*Global_D2r*Phi;
    F(:,3) = Global_Dr*(aa.*Phi) + dissip*Global_D2r*Pi + ((alpha.*a.*W)./r)./r;

    % Condicoes de fronteira
    F(1,2) = 0;
40  F(1,3) = 0;

    F(end,2) = -Global_Dr(end,:) * ( aa(end) .* Phi );
    F(end,3) = -Global_Dr(end,:) * ( aa(end) .* Pi  );

45  F = reshape( F, 5*N, 1 );

    %-------------------------------------------------------------------------%
```

```
function E = EqEYM_Energy( U )

% EQEYMENERGY Evaluate the energy density of Einstein-Yang-Mills Wave
%
% usage: E = EqEYM_Energy( U )

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: EqEYM_Energy.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
%-------------------------------------------------------------------------%

R = evalin ( 'base', 'SpG.R' );
N = evalin ( 'base', 'SpG.Nr' );
U = reshape( U, N, 5 );

W = 1 - U(:,1).^2;
P = ( U(:,2).^2 + U(:,3).^2 );
A = U(:,4);

E = ( (W./R).^2 + P./A./A )./R./R;

E = reshape( E, N, 1 );

%-------------------------------------------------------------------------%
```

```
function EqEYM_Init( SpG, DM, EqP )

% EQEYMINIT Initialize spheriscaly symmetric Einstein-Yang-Mills equations
%
% usage: EqEYM_Init( SpG, DM, EqP )
%       SpG - Spatial grid
%       DM. - Differentiation Matrices
%       EqP - Equation parameters
%             .BC - Boundary conditions
%                   0 - Frezzing
%                   1 - Radiation

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: EqEYM_Init.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
%-------------------------------------------------------------------------%

global Global_Dr
global Global_D2r

Global_Dr  = DM.Dr(:,:,1);
Global_D2r = DM.Dr(:,:,2);

%-------------------------------------------------------------------------%
```

```
function [ Un, En ] = EqEYM_SystemSize

% EQEYMSYSTEMSIZE Return number of internal functions
%
% Usage: [ Un En ] = EqEYM_SystemSize

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: May, 26, 2007
% $Id: EqEYM_SystemSize.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
%-------------------------------------------------------------------------%

Un = 5;
En = 1;

%-------------------------------------------------------------------------%
```

```
function a = EqEYM_a( U )

% Calcula a componente w do campo de Yang-Mills com base em Phi
%
% U = [ w Phi Pi a alpha ]
%

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: EqEYM.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
%-------------------------------------------------------------------------%

%% Tolerancia no passo do raio
tol = 2^-24;

%% Numero maximo de iteracoes
max_n = 100;

%% Discretizacao 1D
N = evalin( 'base', 'SpG.Nr' );
R = evalin( 'base', 'SpG.R'  );

%% Preparacao
a = zeros( N, 1 );

%% W = ( 1 - w^2 )^2
W = ( 1 - U(:,:,1).^2 ).^2;

%% P2 = Phi^2 + Pi^2
P2 = U(:,:,2).^2 + U(:,:,3).^2;

%% Primeiro ponto da malha
ao   = 1;
an   = ao;
a(1) = an;

%% Demais pontos da malha
for ii = 2:N

  n  = 1;
  dr = R(ii) - R(ii-1);
  ao = an;

  %% Preditor - Euler
  [ F dF ] = a_function( R(ii-1), W(ii-1), P2(ii-1), ao );

  an = ao + dr * F;

  %% Corretor - Newton
  [ F dF ] = a_function( R(ii), W(ii), P2(ii), an );

  %% Error
  E = abs( an - ao - dr*F );

  while( E > tol && n < max_n )

    an = an - ( an - ao - dr*F ) / ( 1 - dr*dF );

    [ F dF ] = a_function( R(ii), W(ii), P2(ii), an );

    E = abs( an - ao - dr*F );

    n = n + 1;

  end

  if( n == max_n )
    fprintf( '\nMetodo de newton para a nao convergiu em %i!  ', ii );
  end

  a(ii) = an;

end

%-------------------------------------------------------------------------%
function [ F dF ] = a_function( r, W, P2, a )

a2 = a^2;

F  = ( a / r ) * ( P2 +   a2*W/2/r/r -   a2/2 + 0.5 );
dF = ( 1 / r ) * ( P2 + 3*a2*W/2/r/r - 3*a2/2 + 0.5 );
```

```
%-------------------------------------------------------------------------%
```

```
function alpha = EqEYM_alpha( U )

% Calcula a componente w do campo de Yang-Mills com base em Phi
%
% U = [ w Phi Pi a alpha ]
%

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: EqEYM.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
%-------------------------------------------------------------------------%

%% Tolerancia no passo do raio
tol = 2^-24;

%% Numero maximo de iteracoes
max_n = 100;

%% Discretizacao 1D
N = evalin( 'base', 'SpG.Nr' );
R = evalin( 'base', 'SpG.R' );

%% Preparacao
alpha = zeros( N, 1 );

%% a2 = a^2
a2 = U(:,:,4).^2;

%% W = ( 1 - w^2 )^2
W = ( 1 - U(:,:,1).^2 ).^2;

%% P2 = Phi^2 + Pi^2
P2 = U(:,:,2).^2 + U(:,:,3).^2;

%% Primeiro ponto da malha
alpha_o  = 1;
alpha_n  = alpha_o;
alpha(1) = alpha_n;

%% Demais pontos da malha
for ii = 2:N

  n  = 1;
  dr = R(ii) - R(ii-1);
  alpha_o = alpha_n;

  %% Preditor - Euler
  [ F dF ] = alpha_function( R(ii-1), W(ii-1), P2(ii-1), a2(ii-1), alpha_o );

  alpha_n = alpha_o + dr * F;

  %% Corretor - Newton
  [ F dF ] = alpha_function( R(ii), W(ii), P2(ii), a2(ii), alpha_n );

  %% Error
  E = abs( alpha_n - alpha_o - dr*F );

  while( E > tol && n < max_n )

    alpha_n = alpha_n - ( alpha_n - alpha_o - dr*F ) / ( 1 - dr*dF );

    [ F dF ] = alpha_function( R(ii), W(ii), P2(ii), a2(ii), alpha_n );

    E = abs( alpha_n - alpha_o - dr*F );

    n = n + 1;

  end

  if( n == max_n )
    fprintf( '\nMetodo de newton para alpha nao convergiu em %i! ', ii );
  end

  alpha(ii) = alpha_n;

end

%-------------------------------------------------------------------------%
```

```
function [ F dF ] = alpha_function( r, W, P2, a2, alpha )

aux = ( -P2 +a2*W/2/r/r +a2/2 -0.5 ) / r;

F  = alpha * aux;
dF =         aux;

%-------------------------------------------------------------------------------%
```

```
function w = EqEYM_w( U )

% Calcula a componente w do campo de Yang-Mills com base em Phi
%
% U = [ w Phi Pi a alpha ]
%
% w(r,t) = w_o + int_0^r Phi(s,t) ds

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: EqEYM.m,v 1.1 2007/05/26 03:40:47 akiles Exp $
%-------------------------------------------------------------------------------%

%% Discretizacao 1D
N = evalin( 'base', 'SpG.Nr' );
R = evalin( 'base', 'SpG.R'  );

H = R(2) - R(1);

%% Preparacao
w = zeros( N, 1 );
P = U(:,:,2);

%% Primeiro ponto da malha
%% w_o(0,t) = 1;
%% Phi(0,t) = 0;
w(1) = 1;

%% Segundo ponto - trapezio
w(2) = w(1) + H * ( P(2) + P(1) ) / 2;

%% Demais pontos da malha - simpson
for ii = 3:N

  w(ii) = w(ii-2) + H * ( P(ii-2) + 4*P(ii-1) + P(ii) ) / 3;

end

%-------------------------------------------------------------------------------%
```

Condições Iniciais

Para determinar as condições iniciais para o campo de Einstein-Yang-Mills com simetria esférica é necessário resolver os vínculos associados ao problema. O programa de determina a condição inicial, gerenciando esses vínculos é o IC_1D_EYM.

```
function Uo = IC_1D_EYM( SpG, DM, P )

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: IC_1D_EYM.m,v 1.1 2007/05/26 03:44:17 akiles Exp $
%-------------------------------------------------------------------------------%

Uo = zeros( SpG.Nr, 1, 5 );

w   = PulseEYM( SpG, P );
Phi = DM.Dr(:,:,1) * w;
Pi  = P.D * Phi;

Phi(1) = 0;
Pi (1) = 0;
```

```
    Uo(:,:,1) = w;
    Uo(:,:,2) = Phi;
    Uo(:,:,3) = Pi;

    Uo(:,:,4) = EqEYM_a     ( Uo );
    Uo(:,:,5) = EqEYM_alpha( Uo );

%-------------------------------------------------------------------------------%
```

Evolução Temporal

Em cada passo no tempo é necessário calcular Φ e Π a partir do instante anterior e resolver os vínculos para obter os valores das funções a, $alpha$ e w. Os programas a seguir executam essas tarefas.

```
function U = EYM_TimeStep( Uo, T )

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: RungeKuttaAdapt.m,v 1.5 2005/12/30 21:55:13 akiles Exp $
%-------------------------------------------------------------------------------%

U = RungeKuttaAdapt( Uo, T );

N = evalin ( 'base', 'SpG.Nr' );
U = reshape( U, N, 1, 5 );

U(:,:,1) = EqEYM_w     ( U );
U(:,:,4) = EqEYM_a     ( U );
U(:,:,5) = EqEYM_alpha( U );

U = reshape( U, 5*N, 1 );

%-------------------------------------------------------------------------------%
```

```
function EYM_TimeStep_start( Eq, dt, P )

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% $Id: RungeKuttaAdapt_start.m,v 1.4 2004/11/09 00:46:26 akiles Exp $
%-------------------------------------------------------------------------------%

RungeKuttaAdapt_start( Eq, dt, P )

%-------------------------------------------------------------------------------%
```

B.2 Programas para o Caso Axialmente Simétrico

Aqui estão as listagens dos programas usados para as equações de Yang-Mills axialmente simétrico no espaço-tempo de Minkowski.

O Programa Principal

Esse programa é equivalente ao programa principal para o caso esférico.

```
% YMWAVE - Matlab Script - Driver for Yang-Mills Wave Solver
```

```
%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 08, 2003
% $Id: YMWave.m,v 1.22 2007/05/26 03:39:46 akiles Exp $
%-------------------------------------------------------------------------%

% Spatial Grid
%-------------------------------------------------------------------------%
%  .N[rx]  - Number of points on r or x
%  .[RX]a  - Begining of interval
%  .[RX]b  - End of interval
%
% WaveSolver will create
%   R - Column vector with values of r and
%   X - Column vector with values of x = cos(theta)

% "Pulse Center"
Ro = 4;

SpG.Nr = N;
SpG.Ra = 2^-3;
SpG.Rb = 8*Ro;

SpG.Nx =  N/2;
SpG.Xa = -0.999;
SpG.Xb =  0.999;

% Time Interval
%-------------------------------------------------------------------------%
%  Tf - Final time
%  dt - Interval between samples
%
% WaveSolver will create T - Column vector with values of t

TIn.Tf = 2*SpG.Rb;
TIn.dt = 2^-2;

% Yang-Mills Equations
%-------------------------------------------------------------------------%
Equations = { 'EqAbelian' ... %  1 - Abelian Yang-Mills
              'EqYM'      }; %  2 - Yang-Mills

Eq = Equations{ EqID };

% Equation Parameters
% .OC - Outer boundary condition
%   0 - Frezzing
%   1 - Radiation
EqP.OB = 1;

% Initial Condition
%-------------------------------------------------------------------------%
% InC.system - Detemines what system will be solved
Systems = { 'IC_Ab_1' ...  % 1
            'IC_Ab_2' ...  % 2
            'IC_YM_1' ...  % 3
            'IC_YM_2'  };  % 4

InC.system = Systems{ IC };

% InC.P(1:3) - Parameters for each color
%
% P.pulse - M-function to calculate initial conditions
%           all other fields are passed  to this function
%
% A  - Amplitude
% Ro - Center
% L  - Width
% C  - Multipole Coeficients
%
% Yang-Mills ignores
%  D = 0 - time symmetric
%      1 - ingoing
%     -1 - outgoing

% Pulse choises:
Pulses = { 'PulseNull'       ...  %  1 - No Pulse
           'PulseGaussian'   ...  %  2 - Gaussian
           'PulseRGaussian'  ...  %  3 - r * Gaussian
           'PulsePhi4'        };  %  4 - lambda-phi^4

% Abelian or Color = 1
%---------------------
InC.P(1,1).pulse = Pulses{ P11 };
InC.P(1,1).A  = A11;
```

```
89  InC.P(1,1).Ro = R11;
    InC.P(1,1).L  = L11;
    InC.P(1,1).C  = C11;
    InC.P(1,1).epsilon = E1;

94  InC.P(1,2).pulse = Pulses{ P12 };
    InC.P(1,2).A  = A12;
    InC.P(1,2).Ro = R12;
    InC.P(1,2).L  = L12;
    InC.P(1,2).C  = C12;
99
    if( EqID == 2 )

      % Color = 2
      %----------
104   InC.P(2,1).pulse = Pulses{ P21 };
      InC.P(2,1).A  = A21;
      InC.P(2,1).Ro = R21;
      InC.P(2,1).L  = L21;
      InC.P(2,1).C  = C21;
109   InC.P(2,1).epsilon = E2;

      InC.P(2,2).pulse = Pulses{ P22 };
      InC.P(2,2).A  = A22;
      InC.P(2,2).Ro = R22;
114   InC.P(2,2).L  = L22;
      InC.P(2,2).C  = C22;

      % Color = 3
      %----------
119   InC.P(3,1).pulse = Pulses{ P31 };
      InC.P(3,1).A  = A31;
      InC.P(3,1).Ro = R31;
      InC.P(3,1).L  = L31;
      InC.P(3,1).C  = C31;
124
      InC.P(3,2).pulse = Pulses{ P32 };
      InC.P(3,2).A  = A32;
      InC.P(3,2).Ro = R32;
      InC.P(3,2).L  = L32;
129   InC.P(3,2).C  = C32;

    end

    % Discretization Methods:
134 %-------------------------------------------------------------------------------%
    Methods = { 'MtFD'          ...  %  1 - Finite Differences
                'MtChebyshev'   ...  %  2 - Chebyshev
                'MtChebyshevTr' };  %  3 - Transformed Chebyshev

139 DMt.MtR = Methods{ 1 };
    DMt.MtX = DMt.MtR;

    % Parameters used only by finite difference method:
    %   .Order  - Order of FD method
144 %   .Points - M-function to calculate points inputs: ( a, b, Nr )
    PtDistributions = { 'linspace'        ...  %  1 - Equally spaced
                        'PointsChebyshev' };  %  2 - Chebyshev

    DMt.Points = PtDistributions{ 1 };
149 DMt.Order  = 4;

    % Parameter for Transformed Chebychev
    DMt.alpha = 0.9;

154 % Time evolution methods
    %-------------------------------------------------------------------------------%
    EvolutionMethods = {'RungeKutta',        ... % 1 - Kunge-Kutta of Order S
                        'RungeKutta4',       ... % 2 - Kunge-Kutta of 4th Order
                        'RungeKuttaAdapt', ... % 3 - adaptive Kunge-Kutta
159                     'CrankNicolson'      }; % 4 - Crank-Nicolson

    % Parameters for evolution procedure:
    %   Ev.Mt   - Time step method
    %     .mU   - Maximum value of U allowed
164 %     .dt   - Initial time step
    %     .P    - Time step method parameters
    Ev.Mt = EvolutionMethods{ 3 };
    Ev.mU = 100;
    Ev.dt = 2^-8;
169
    % Parameters for time step method
    %   P.Order - for implementation 1 of Kunge-Kutta method
    %    .tol   - for adaptive methods
    Ev.P.Order = 2;
```

```
174   Ev.P.tol    = 2^-20;

      % Solving
      %-------------------------------------------------------------------------%

179   % Log information
      clear ElapsedTime;

      fprintf('| Started at %s\n', datestr(now) );

184   WaveReport

      tic;

      [ U E ] = WaveSolver( Eq, EqP, DMt, Ev, SpG, TIn, InC );
189
      ElapsedTime = toc;

      % Saving solution
      PSsave
194
      h = floor( ElapsedTime / ( 60*60 ) );
      e = ElapsedTime - h*60*60;
      m = floor( e / 60 );
      s = round( e - m * 60 );
199
      fprintf('| Elapsed Time = %i:%02i:%02i\n', h, m, s );

      %-------------------------------------------------------------------------%
```

As Equações de Yang-Mills

Nessa seção estão as funções que avaliam as equações de Yang-Mills. Na primeira subseção estão as que avaliam a equação Abeliana e na próxima subseção as que avaliam a equação completa.

Equação Abeliana

```
      function F = EqAbelian( U )
3
      % EQABELIAN - Abelian Yang-Mills Evolution Equations
      %
      % usage: F = EqAbelian( U )
      %          U - Wave values
8     %          F - Wave Equation values
      %
      % DtNr = (1/r^2) Dx{ (1-x^2)[ DxAr - DrAx ] }
      % DtNx = Drr( Ax ) - Drx( Ar )
      %
13    % DtAr = Nr
      % DtAx = Nx

      %-------------------------------------------------------------------------%
      % Luis Alberto D'Afonseca
18    % since: Nov, 11, 2003
      % $Id: EqAbelian.m,v 1.11 2004/11/17 20:48:46 akiles Exp $
      %-------------------------------------------------------------------------%

      % Global Matrices
23    global g_Dr    % Derivative operator
      global g_Dx    % Derivative operator
      global g_UX    % = 1-x^2
      global g_R2    % = 1/r^2
      global g_OB    % Outer boundary condition
28
      global g_dim   % Grid Dimensions
      global g_iAr   % Ar indices
      global g_iAx   % Ax indices
      global g_iNr   % Nr indices
33    global g_iNx   % Nx indices
      global g_iOBr % Outer boundary index for Ar
      global g_iOBx % Outer boundary index for Ax

      %-------------------------------------------------------------------------%
```

```
38
    Ar = reshape( U(g_iAr), g_dim );
    Ax = reshape( U(g_iAx), g_dim );

    % Enforcing Axial Symmetry
43  Ax(1,:,:) = 0;

    % Equations for Nt
    %-------------------------------------------------------------------------%

48  % First derivatives of A
    DrAx = g_Dr * Ax;
    DxAr = Ar * g_Dx;

    % Equations
53  %-------------------------------------------------------%
    % DtNr = (1/r^2) Dx{ (1-x^2)[ DxAr - DrAx ] }
    % DtNx = Dr( Dr( Ax ) - Dx( Ar ) )

    DxArDrAx = DxAr - DrAx;
58
    DtNr =  g_R2 .* ( g_UX .* DxArDrAx  ) * g_Dx;
    DtNx = -g_Dr  * DxArDrAx;

    % Inner Boundary Condition
63  %-------------------------------------------------------%
    DtNx(1,:) = 0;

    % Output
    %-------------------------------------------------------------------------%
68
    F = [ U([ g_iNr g_iNx ])
          DtNr(:)
          DtNx(:) ];

73  % Outer Boundary Condition
    F( g_iOBr ) = transp( g_OB * Ar );
    F( g_iOBx ) = transp( g_OB * Ax );

    %-------------------------------------------------------------------------%
```

```
    function E = EqAbelian_Energy( U )
3
    % EQABELIANENERGY - Abelian Yang-Mills Energy Evaluation
    %
    % usage: E = EqAbelian_Energy( U )

8   %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Nov, 11, 2003
    % $Id: EqAbelian_Energy.m,v 1.8 2007/05/26 03:39:44 akiles Exp $
    %-------------------------------------------------------------------------%
13
    % Global Matrices
    global g_Dr   %  Derivative operator
    % global g_D2r  %  Derivative operator
    global g_Dx   %  Derivative operator
18  % global g_D2x  %  Derivative operator
    global g_UX   %  = 1-x^2
    % global g_2X   %  = 2x
    % global g_R2   %  = 1/r^2

23  % global g_2r    %  = 2    / r
    global g_2pir  %  = 2 pi / r
    global g_2pixr %  = 2 pi * r

    % global g_OB   % Outer boundary condition
28
    global g_dim  % Grid Dimensions
    global g_iAr  % Ar indices
    global g_iAx  % Ax indices
    global g_iNr  % Nr indices
33  global g_iNx  % Nx indices

    %-------------------------------------------------------------------------%

    Ar = reshape( U(g_iAr), g_dim );
38  Ax = reshape( U(g_iAx), g_dim );
    Nr = reshape( U(g_iNr), g_dim );
    Nx = reshape( U(g_iNx), g_dim );
```

```
%-------------------------------------------------------------------------------%
43
% First derivatives of A
DrAx = g_Dr * Ax;
DxAr = Ar * g_Dx;
% DrAr = g_Dr * Ar;
48 % DxAx = Ax * g_Dx;

% Second derivatives
% D2rAx = g_D2r * Ax;
% D2xAr = Ar * g_D2x;
53
% Mixed derivatives
% DrxAr = g_Dr * DxAr;
% DrxAx = g_Dr * DxAx;

58 % Equations
% DtNr = g_R2.*( g_UX.*( DxAr - DrAx ) ) * g_Dx;
% DtNx = D2rAx - DrxAr;

% Energy Density
63 rho = g_2pir.*g_UX.*( 2*DxAr.*DrAx   ...
                        -DrAx .^2      ...
                        -DxAr .^2      ...
                        -Nx   .^2   ) ...
       - g_2pixr .* Nr.^2;
68
% Fluxes
J  =  2  * g_UX .* g_2pir .* ( DrAx -DxAr );
Jr =  J .* Nx;
Jx = -J .* Nr;
73
% First derivatives of N and J
% DxNr = Nr * g_Dx;
% DxNx = Nx * g_Dx;
% DxJx = Jx * g_Dx;
78 % DrJr = g_Dr * Jr;

% H's
% H = DxNx - DxNr;

83 % DtRho
% DtRho = -J.*H -2*( Nr.*DtNr +g_UX.*g_R2.*Nx.*DtNx );

% Error: Err = DrJr + DxJx + (2/r) Jr + DtRho
% Err = abs( DrJr +DxJx +g_2r.*Jr +DtRho );
88
% Output
E = cat( 3, rho, Jr, Jx );

% Removido: E = cat( 3, rho, Jr, Jx, Err );
93

%-------------------------------------------------------------------------------%
```

```
function EqAbelian_Init( SpG, DM, EqP )

% EQABELIANINIT - Initialize Abelian Yang-Mills Equations
5 %
% usage: EqAbelian_Init( SpG, DM, EqP )
%       SpG - Spatial grid
%       DM. - Differentiation Matrices
%       EqP - Equation parameters
10 %             .BC - Boundary conditions
%                    0 - Frezzing
%                    1 - Radiation

%-------------------------------------------------------------------------------%
15 % Luis Alberto D'Afonseca
% since: Nov, 11, 2003
% $Id: EqAbelian_Init.m,v 1.7 2005/12/30 21:55:14 akiles Exp $
%-------------------------------------------------------------------------------%

20 % Initialize global Matrices
global g_Dr   % Derivative operator
global g_D2r  % Derivative operator
global g_Dx   % Derivative operator
global g_D2x  % Derivative operator
25
```

```
    global g_UX    % = 1-x^2
    global g_2X    % = 2x
    global g_R2    % = 1/r^2

    global g_2r     %  = 2     / r
    global g_2pir   %  = 2 pi / r
    global g_2pixr  %  = 2 pi * r

    global g_OB    % Outer boundary condition

    global g_dim   % Grid Dimensions
    global g_iAr   % Ar indices
    global g_iAx   % Ax indices
    global g_iNr   % Nr indices
    global g_iNx   % Nx indices
    global g_iOBr % Outer boundary index for Ar
    global g_iOBx % Outer boundary index for Ax

    %-------------------------------------------------------------------------%

    Nr = SpG.Nr;
    Nx = SpG.Nx;

    % Coordinates
    R = repmat( reshape(SpG.R,Nr,1), [1, Nx]); % matrix: Nr x Nx
    X = repmat( reshape(SpG.X,1,Nx), [Nr, 1]); % matrix: Nr x Nx

    g_UX = 1-X.^2;
    g_2X = 2*X;
    g_R2 = 1./(R.^2);

    g_2r     = 2    ./R;
    g_2pir   = 2*pi./R;
    g_2pixr = 2*pi *R;

    % Differentiation Matrices
    g_Dr  = DM.Dr(:,:,1);
    g_D2r = DM.Dr(:,:,2);
    g_Dx  = DM.Dx(:,:,1).';
    g_D2x = DM.Dx(:,:,2).';

    % Outer boundary condition
    switch EqP.OB
     case 0, g_OB = zeros(1,Nr);     % 1 - u_t = 0
     case 1, g_OB = -g_Dr(end,:);    % 2 - u_t = -u_r
    end

    %-------------------------------------------------------------------------%

    g_dim = [ Nr Nx ];

    %-----------------------------------------------%

    N = Nr * Nx;

    g_iOBr = (   Nr):Nr:(   N);
    g_iOBx = (N+Nr):Nr:(2*N);

    %-----------------------------------------------%

    g_iAr = (     1):(   N);
    g_iAx = (   N+1):(2*N);
    g_iNr = (2*N+1):(3*N);
    g_iNx = (3*N+1):(4*N);

    %-------------------------------------------------------------------------%
```

```
    function [ Un, En ] = EqAbelian_SystemSize

    % EQABELIANSYSTEMSIZE Return number of internal functions
    %
    % Usage: [ Un En ] = EqAbelian_SystemSize
    %
    %  Un - Number of U functions
    %    1 - Ar
    %    2 - Ax
    %    3 - Nr
    %    4 - Nx
    %
    %  En - Number of E functions
```

```
%   1 - Rho   - Energy Density
%   2 - Jr    - Flux Density in R
%   3 - Jx    - Flux Density in X

% Removido:  4 - Error - Energy Error

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 03, 2004
% $Id: EqAbelian_SystemSize.m,v 1.3 2007/05/26 03:39:44 akiles Exp $
%-------------------------------------------------------------------------%

Un = 4;
En = 3;

%-------------------------------------------------------------------------%
```

Equação de Yang-Mills

```
function F = EqYM( U )

% EQYM - Yang-Mills Evolution Equations
%
% usage: F = EqYM( U )
%        U - Wave values
%        F - Wave Equation values

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 08, 2003
% $Id: EqYM.m,v 1.10 2004/11/17 20:48:46 akiles Exp $
%-------------------------------------------------------------------------%

% Global Matrices
global g_Dr    % Derivative operator
global g_D2r   % Derivative operator
global g_Dx    % Derivative operator
global g_D2x   % Derivative operator
global g_UX    % = 1-x^2
global g_2X    % = 2x
global g_R2    % = 1/r^2
global g_2r    % = 2/r
global g_OB    % Outer boundary condition

global g_dim   % Grid Dimensions
global g_iAr   % Ar indices
global g_iAx   % Ax indices
global g_iNr   % Nr indices
global g_iNx   % Nx indices
global g_iOBr % Outer boundary index for Ar
global g_iOBx % Outer boundary index for Ax

%-------------------------------------------------------------------------%

Ar = reshape( U(g_iAr), g_dim );
Ax = reshape( U(g_iAx), g_dim );

% Enforcing Axial Symmetry
Ax(1,:,:) = 0;

% Equations for Nt
%-------------------------------------------------------------------------%

% First derivatives
DrAr = zeros( g_dim );
DrAx = zeros( g_dim );
DxAr = zeros( g_dim );
DxAx = zeros( g_dim );

% Second derivatives
D2rAx = zeros( g_dim );
D2xAr = zeros( g_dim );

% Mixed derivatives
DrxAr = zeros( g_dim );
DrxAx = zeros( g_dim );
```

```
    for ii = 1:3

      % First derivatives
      DrAr(:,:,ii) = g_Dr * Ar(:,:,ii);
64    DrAx(:,:,ii) = g_Dr * Ax(:,:,ii);
      DxAr(:,:,ii) = Ar(:,:,ii) * g_Dx;
      DxAx(:,:,ii) = Ax(:,:,ii) * g_Dx;

      % Second derivatives
69    D2rAx(:,:,ii) = g_D2r * Ax(:,:,ii);
      D2xAr(:,:,ii) = Ar(:,:,ii) * g_D2x;

      % Mixed derivatives
      DrxAr(:,:,ii) = g_Dr * DxAr(:,:,ii);
74    DrxAx(:,:,ii) = g_Dr * DxAx(:,:,ii);

    end

    % Deltas
79  ind = [ 1 2 3
            2 3 1
            3 1 2 ];

    Delta = Ar(:,:,ind(2,:)) .* Ax(:,:,ind(3,:)) -  ...
84          Ar(:,:,ind(3,:)) .* Ax(:,:,ind(2,:));

    % Calculating G's
    Gr = g_UX .* ( Delta(:,:,ind(2,:)) .* Ax(:,:,ind(3,:)) -    ...
                   Delta(:,:,ind(3,:)) .* Ax(:,:,ind(2,:)) ) +  ...
89       g_2X .* Delta;

    Gx = Ar(:,:,ind(2,:)) .* Delta(:,:,ind(3,:)) -               ...
         Ar(:,:,ind(3,:)) .* Delta(:,:,ind(2,:));

94  % Calculating F's
    Fr = g_UX .* ( Ax(:,:,ind(3,:)) .* DrAx(:,:,ind(2,:)) -        ...
                   Ax(:,:,ind(2,:)) .* DrAx(:,:,ind(3,:)) +        ...
                   Ar(:,:,ind(3,:)) .* DxAx(:,:,ind(2,:)) -        ...
                   Ar(:,:,ind(2,:)) .* DxAx(:,:,ind(3,:)) +        ...
99                 2*( Ax(:,:,ind(2,:)) .* DxAr(:,:,ind(3,:)) -    ...
                       Ax(:,:,ind(3,:)) .* DxAr(:,:,ind(2,:)) ) );

    Fx = Ar(:,:,ind(3,:)) .* DxAr(:,:,ind(2,:)) -       ...
         Ar(:,:,ind(2,:)) .* DxAr(:,:,ind(3,:)) +       ...
104      Ax(:,:,ind(3,:)) .* DrAr(:,:,ind(2,:)) -       ...
         Ax(:,:,ind(2,:)) .* DrAr(:,:,ind(3,:)) +       ...
         2*( Ar(:,:,ind(2,:)) .* DrAx(:,:,ind(3,:)) -   ...
             Ar(:,:,ind(3,:)) .* DrAx(:,:,ind(2,:)) );

109 % Equations
    %-------------------------------------------------------%
    % DtNr = (1/r^2) ( Dx{ (1-x^2)[ DxAr - DrAx ] } +Fr +Gr )
    % DtNx = Drr( Ax ) - Drx( Ar ) +Fx + Gx

114 DtNr = g_R2 .* ( g_UX.*( D2xAr - DrxAx ) +g_2X.*( DrAx - DxAr ) + Fr + Gr );
    DtNx = D2rAx - DrxAr + Fx + Gx;

    % Boundary conditions
    %-------------------------------------------------------%
119
    % Inner Boundary Condition
    DtNx(1,:,:) = 0;

    % Output
124 %-------------------------------------------------------------------------------------%

    F = [ U([ g_iNr g_iNx ])
          DtNr(:)
          DtNx(:) ];
129
    % Outer Boundary Condition
    for ii = 1:3
      F( g_iOBr(:,ii) ) = transp( g_OB * Ar(:,:,ii) );
      F( g_iOBx(:,ii) ) = transp( g_OB * Ax(:,:,ii) );
134 end

    %-------------------------------------------------------------------------------------%
```

```
    function E = EqYM_Energy( U )
```

```
% EQYMENERGY - YM Energy
%
% usage: E = EqYM_Energy( U )

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 08, 2003
% $Id: EqYM_Energy.m,v 1.5 2007/05/26 03:39:45 akiles Exp $
%-------------------------------------------------------------------------%

% Global Matrices
global g_Dr    %  Derivative operator
global g_D2r   %  Derivative operator
global g_Dx    %  Derivative operator
global g_D2x   %  Derivative operator
global g_UX    %  = 1-x^2
global g_2X    %  = 2x
global g_R2    %  = 1/r^2

global g_2r     %  = 2    / r
global g_2pir   %  = 2 pi / r
global g_2pixr  %  = 2 pi * r

global g_OB    % Outer boundary condition

global g_dim   % Grid Dimensions
global g_iAr   % Ar indices
global g_iAx   % Ax indices
global g_iNr   % Nr indices
global g_iNx   % Nx indices

%-------------------------------------------------------------------------%

Ar = reshape( U(g_iAr), g_dim );
Ax = reshape( U(g_iAx), g_dim );
Nr = reshape( U(g_iNr), g_dim );
Nx = reshape( U(g_iNx), g_dim );

%-------------------------------------------------------------------------%

% First derivatives
DrAx = zeros( g_dim );
DxAr = zeros( g_dim );
% DrAr = zeros( g_dim );
% DxAx = zeros( g_dim );

% Second derivatives
% D2rAx = zeros( g_dim );
% D2xAr = zeros( g_dim );

% Mixed derivatives
% DrxAr = zeros( g_dim );
% DrxAx = zeros( g_dim );

for ii = 1:3

  % First derivatives
  DrAx(:,:,ii) = g_Dr * Ax(:,:,ii);
  DxAr(:,:,ii) = Ar(:,:,ii) * g_Dx;
  % DrAr(:,:,ii) = g_Dr * Ar(:,:,ii);
  % DxAx(:,:,ii) = Ax(:,:,ii) * g_Dx;

  % Second derivatives
  % D2rAx(:,:,ii) = g_D2r * Ax(:,:,ii);
  % D2xAr(:,:,ii) = Ar(:,:,ii) * g_D2x;

  % Mixed derivatives
  % DrxAr(:,:,ii) = g_Dr * DxAr(:,:,ii);
  % DrxAx(:,:,ii) = g_Dr * DxAx(:,:,ii);

end

% Deltas
ind = [ 1 2 3
        2 3 1
        3 1 2 ];

Delta = Ar(:,:,ind(2,:)) .* Ax(:,:,ind(3,:)) -  ...
        Ar(:,:,ind(3,:)) .* Ax(:,:,ind(2,:));

% Calculating G's
% Gr = g_UX .* ( Delta(:,:,ind(2,:)) .* Ax(:,:,ind(3,:)) -     ...
%                Delta(:,:,ind(3,:)) .* Ax(:,:,ind(2,:)) ) +  ...
%       g_2X .* Delta;
```

```
% Gx = Ar(:,:,ind(2,:)) .* Delta(:,:,ind(3,:)) -                ...
%      Ar(:,:,ind(3,:)) .* Delta(:,:,ind(2,:));

% Calculating F's
% Fr = g_UX .* ( Ax(:,:,ind(3,:)) .* DrAx(:,:,ind(2,:)) -          ...
%                Ax(:,:,ind(2,:)) .* DrAx(:,:,ind(3,:)) +          ...
%                Ar(:,:,ind(3,:)) .* DxAx(:,:,ind(2,:)) -          ...
%                Ar(:,:,ind(2,:)) .* DxAx(:,:,ind(3,:)) +          ...
%                2 * ( Ax(:,:,ind(2,:)) .* DxAr(:,:,ind(3,:)) -    ...
%                    Ax(:,:,ind(3,:)) .* DxAr(:,:,ind(2,:)) ));

% Fx = Ar(:,:,ind(3,:)) .* DxAr(:,:,ind(2,:)) -                  ...
%      Ar(:,:,ind(2,:)) .* DxAr(:,:,ind(3,:)) +                  ...
%      Ax(:,:,ind(3,:)) .* DrAr(:,:,ind(2,:)) -                  ...
%      Ax(:,:,ind(2,:)) .* DrAr(:,:,ind(3,:)) +                  ...
%      2 * ( Ar(:,:,ind(2,:)) .* DrAx(:,:,ind(3,:)) -            ...
%          Ar(:,:,ind(3,:)) .* DrAx(:,:,ind(2,:)) );

% Equations
% DtNr = g_R2 .* ( g_UX.*( D2xAr - DrxAx ) + g_2X.*( DrAx - DxAr ) + Fr + Gr );
% DtNx = D2rAx - DrxAr + Fx + Gx;

% Energy Density integrated over phi
rho = g_2pir.*g_UX.*( 2 * DxAr  .* ( DrAx + Delta ) -  ...
                      2 * Delta .*   DrAx -            ...
                      Delta.^2 -                       ...
                      DrAx .^2 -                       ...
                      DxAr .^2 -                       ...
                      Nx   .^2  ) ...
      - g_2pixr .* Nr.^2;

% Fluxes
J  =  2 * g_2pir .* g_UX .* ( Delta + DrAx - DxAr );
Jr =  J .* Nx;
Jx = -J .* Nr;

% Gammas
% Gamma_r = Ar(:,:,ind(2,:)) .* Nx(:,:,ind(3,:)) -  ...
%           Ar(:,:,ind(3,:)) .* Nx(:,:,ind(2,:));
% Gamma_x = Ax(:,:,ind(2,:)) .* Nr(:,:,ind(3,:)) -  ...
%           Ax(:,:,ind(3,:)) .* Nr(:,:,ind(2,:));

% First derivatives of N and J
% DxNr = zeros( g_dim );
% DxNx = zeros( g_dim );
% DxJx = zeros( g_dim );
% DrJr = zeros( g_dim );

% for ii = 1:3
%   DxNr(:,:,ii) = Nr(:,:,ii)*g_Dx;
%   DxNx(:,:,ii) = Nx(:,:,ii)*g_Dx;
%   DxJx(:,:,ii) = Jx(:,:,ii)*g_Dx;
%   DrJr(:,:,ii) = g_Dr*Jr(:,:,ii);
% end

% H's
% H = Gamma_r - Gamma_x + DxNx - DxNr;

% DtRho
% DtRho = -J.*H -2*( Nr.*DtNr +g_UX.*g_R2.*Nx.*DtNx );

% Coloured error: cErr = DrJr + DxJx + (2/r) Jr + DtRho
% cErr = DrJr + DxJx + g_2r.* Jr + DtRho;

% Error: Err = "tr" cErr
% Err = abs( sum( cErr, 3 ) );

% Output
%-------------------------------------------------------------------------------%

E = cat( 3, rho, Jr, Jx );
% Removed: E = cat( 3, rho, Jr, Jx, Err );

%-------------------------------------------------------------------------------%
```

```
function EqYM_Init( SpG, DM, EqP )

% EQYMINIT - Initialize YM equation
%
% usage: EqYM_Init( SpG, DM, EqP )
```

```
%       SpG - Spatial grid
%       DM. - Differentiation Matrices
%       EqP - Equation parameters
%             .BC - Boundary conditions
%                   0 - Frezzing
%                   1 - Radiation

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 08, 2003
% $Id: EqYM_Init.m,v 1.8 2005/12/30 21:55:15 akiles Exp $
%-------------------------------------------------------------------------%

% Initialize global Matrices
global g_Dr   % Derivative operator
global g_D2r  % Derivative operator
global g_Dx   % Derivative operator
global g_D2x  % Derivative operator

global g_UX   % = 1-x^2
global g_2X   % = 2x
global g_R2   % = 1/r^2

global g_2r     %  = 2    / r
global g_2pir   %  = 2 pi / r
global g_2pixr  %  = 2 pi * r

global g_OB   % Outer boundary condition

global g_dim  % Grid Dimensions
global g_iAr  % Ar indices
global g_iAx  % Ax indices
global g_iNr  % Nr indices
global g_iNx  % Nx indices
global g_iOBr % Outer boundary index for Ar
global g_iOBx % Outer boundary index for Ax

%-------------------------------------------------------------------------%

Nr = SpG.Nr;
Nx = SpG.Nx;

%-------------------------------------------------------%
% Coordinates
R = repmat( reshape(SpG.R,Nr,1), [1, Nx, 3]); % matrix: Nr x Nx x 3
X = repmat( reshape(SpG.X,1,Nx), [Nr, 1, 3]); % matrix: Nr x Nx x 3

g_UX = 1-X.^2;
g_2X = 2*X;
g_R2 = 1./(R.^2);

g_2r    = 2   ./R;
g_2pir  = 2*pi./R;
g_2pixr = 2*pi *R;

% Differentiation Matrices
g_Dr  = DM.Dr(:,:,1);
g_D2r = DM.Dr(:,:,2);
g_Dx  = DM.Dx(:,:,1).';
g_D2x = DM.Dx(:,:,2).';

% Outer boundary condition
switch EqP.OB
 case 0, g_OB = zeros(1,Nr);     % 1 - u_t = 0
 case 1, g_OB = -g_Dr(end,:);    % 2 - u_t = -u_r
end

%-------------------------------------------------------------------------%

g_dim = [ Nr Nx 3 ];

%-------------------------------------------------------%

N  = Nr * Nx;
N3 =  3 * N;

g_iOBr = reshape( (   Nr):Nr:(  N3), Nx, 3 );
g_iOBx = reshape( (N3+Nr):Nr:(2*N3), Nx, 3 );

%-------------------------------------------------------%

g_iAr = (     1):(  N3);
g_iAx = (  N3+1):(2*N3);
g_iNr = (2*N3+1):(3*N3);
g_iNx = (3*N3+1):(4*N3);
```

```
93  %-------------------------------------------------------------------------%
```

```
 2  function [ Un, En ] = EqYM_SystemSize

    % EQYMSYSTEMSIZE Return number of internal functions
    %
    % Usage: [ Un En ] = EqYM_SystemSize
 7  %
    %  Un - Number of U functions
    %    01:03 - Ar(1:3)
    %    04:06 - Ax(1:3)
    %    07:09 - Nr(1:3)
12  %    10:12 - Nx(1:3)
    %
    %  En - Number of E functions
    %    01:03 - Rho(1:3) - Energy Density
    %    04:06 - Jr (1:3) - Flux Density in R
17  %    07:09 - Jx (1:3) - Flux Density in X

    %       10 - Error    - Energy Error

    %-------------------------------------------------------------------------%
22  % Luis Alberto D'Afonseca
    % since: Oct, 03, 2004
    % $Id: EqYM_SystemSize.m,v 1.3 2007/05/26 03:39:45 akiles Exp $
    %-------------------------------------------------------------------------%

27  Un = 12;
    En = 9;

    %-------------------------------------------------------------------------%
```

Condições Iniciais

Essas são as funções que geram as condições iniciais. As funções para o caso Abeliano estão na primeira subseção. As condições iniciais para a equação completa estão na segunda subseção.

Condições Iniciais para a Equação Abeliana

```
    function Uo = IC_Ab_1( SpG, DM, P )

    % ICAB1- First Family of Initial conditions for Abelian Yang-Mills
 5  %
    % A1r(r,x) = r R(r) X(x)

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
10  % since: Nov, 12, 2003
    % $Id: IC_Ab_1.m,v 1.1 2005/12/30 21:55:15 akiles Exp $
    %-------------------------------------------------------------------------%

    % Creating Uo
15  Uo = zeros( SpG.Nr, SpG.Nx, 4 );

    % A1r is given freely
    for ii = 1:size( P, 2 )

20    Uo(:,:,1) = Uo(:,:,1) + feval( P(1,ii).pulse, SpG, P(1,ii) );

    end

    %-------------------------------------------------------------------------%
```

```
1
    function Uo = IC_Ab_2( SpG, DM, P )

    % ICAB2 - Second Family of Initial Conditions for Abelian Yang-Mills
    %
6   %  Nr =   R(r) Dx[ (1-x^2) L(x) ]
    %  Nx = - L(x) Dr[     r^2  R(r) ]
    %  Ar = epsilon Nr
    %
    %  R(r) = function of r
11  %  L(x) = superposition Legendre Polynomials

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Oct, 11, 2004
16  % $Id: IC_Ab_2.m,v 1.2 2007/05/26 03:39:45 akiles Exp $
    %-------------------------------------------------------------------------%

    % Creating Uo
    Uo = zeros( SpG.Nr, SpG.Nx, 4 );
21
    % Coordinates
    x  = transp( SpG.X );
    ux = 1 - x.^2;

26  r2 = SpG.R.^2;

    spg.R  = SpG.R;
    spg.Nr = SpG.Nr;
    spg.Nx = 1;
31
    % R evaluation
    R = zeros( size( SpG.R ) );

    for ii = 1:size(P,2)
36
      R = R + feval( P(1,ii).pulse, spg, P(1,ii) );

    end

41  % L evaluation
    L = SuperLegendre( x, P(1,1).C );

    % dR = Dr[ r^2 R ]
    % dL = Dx[ (x^2-1) L ]
46  dR = DM.Dr(:,:,1) * ( r2 .* R );
    dL = ( ux.*L ) * transp( DM.Dx(:,:,1) );

    % Nr =    R * dL
    % Nx = - dR *  L
51  Uo(:,:,3) =   kron( R, dL );
    Uo(:,:,4) = - kron( dR, L );

    % Ar = epsilon * Nr
    Uo(:,:,1) = P(1,1).epsilon * Uo(:,:,3);
56

    %-------------------------------------------------------------------------%
```

Condições Iniciais para a Equação de Yang-Mills

```
2   function Uo = IC_YM_1( SpG, DM, P )

    % ICYM1 - Initial condition for Yang-Mills
    %
    % Ar is free and N = 0
7
    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Nov, 02, 2003
    % $Id: IC_YM_1.m,v 1.2 2007/05/26 03:39:45 akiles Exp $
12  %-------------------------------------------------------------------------%

    % Creating Uo
    Uo = zeros( SpG.Nr, SpG.Nx, 12 );
```

```
17  % Ar's are given freely
    for ii = 1:size(P,2)

      Uo(:,:,1) = Uo(:,:,1) + feval( P(1,ii).pulse, SpG, P(1,ii) );
      Uo(:,:,2) = Uo(:,:,2) + feval( P(2,ii).pulse, SpG, P(2,ii) );
22    Uo(:,:,3) = Uo(:,:,3) + feval( P(3,ii).pulse, SpG, P(3,ii) );

    end

    %-------------------------------------------------------------------------%
```

```
    function Uo = IC_YM_2( SpG, DM, P )

 4  % ICYM2 - Second Family of Initial Conditions for Yang-Mills
    %
    % For each color:
    %  Nr =   R(r) Dx[ (1-x^2) L(x) ]
    %  Nx = - L(x) Dr[     r^2  R(r) ]
 9  %  Ar = epsilon Nr
    %
    %  R(r) = function of r
    %  L(x) = superposition Legendre Polynomials

14  %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Oct, 11, 2004
    % $Id: IC_YM_2.m,v 1.2 2007/05/26 03:39:45 akiles Exp $
    %-------------------------------------------------------------------------%
19
    % Creating Uo
    Uo = zeros( SpG.Nr, SpG.Nx, 12 );

    % Coordinates
24  x  = transp( SpG.X );
    ux = 1 - x.^2;

    r2 = SpG.R.^2;

29  spg.R  = SpG.R;
    spg.Nr = SpG.Nr;
    spg.Nx = 1;

    % Only for two colours
34  for ii = 1:2

      % R evaluation
      R = zeros( size( SpG.R ) );

39    for jj = 1:size(P,2)

        R = R + feval( P(ii,jj).pulse, spg, P(ii,jj) );

      end
44
      % L evaluation
      L = SuperLegendre( x, P(ii,1).C );

      % dR = Dr[ r^2 R ]
49    % dL = Dx[ (x^2-1) L ]
      dR = DM.Dr(:,:,1) * ( r2 .* R );
      dL = ( ux.*L ) * transp( DM.Dx(:,:,1) );

      % Nr =    R * dL
54    % Nx = - dR *  L
      Uo(:,:,6+ii) =   kron( R, dL );
      Uo(:,:,9+ii) = - kron( dR, L );

      % Ar = epsilon * Nr
59    Uo(:,:,ii) = P(ii,1).epsilon * Uo(:,:,6+ii);

    end

    %-------------------------------------------------------------------------%
```

B.3 Programas Comuns

Seguem agora os programas comuns aos dois conjuntos de equações estudadas.

Resolvendo as Equações

Todos os métodos numéricos usados para obter as soluções são chamados direta, ou indiretamente, pela função WaveSolver. A função WaveReport apenas exibe informações sobre o problema. Enquanto que PSave salva em disco os resultados obtidos

```
function [ U, E ] = WaveSolver( Eq, EqP, DMt, Ev, SpG, TIn, InC )

% WAVEYMSOLVER - Yang-Mills wave equation solver
%
% Usage: U = WaveAxSolver( Eq, EqP, DMt, RKp, SpG, TIn, InC )
%
%  Eq   - Equations
%  EqP  - Equation parameters
%  DMt  - Discretization Method
%  Ev   - Time Evolution Method
%  SpG  - Spatial Grid
%  TIn  - Time interval
%  InC  - Initial Condition System

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 08, 2003
% $Id: WaveSolver.m,v 1.7 2004/11/09 00:46:27 akiles Exp $
%-------------------------------------------------------------------------%

% Calculate Differentiation Matrices and Spatial grid points
[ SpG.R DM.Dr ] = feval( DMt.MtR, SpG.Ra, SpG.Rb, SpG.Nr, DMt );

if isfield( SpG, 'Nx' )
  [ SpG.X DM.Dx ] = feval( DMt.MtX, SpG.Xa, SpG.Xb, SpG.Nx, DMt );
end

% Exporting grid data
assignin( 'caller', inputname(5), SpG );

% Initialize Equation Function
feval( [Eq '_Init'], SpG, DM, EqP );

% Initial Condition
Uo = feval( InC.system, SpG, DM, InC.P );

% Maximum time step
Ev.P.max = 0.9 * min( diff( SpG.R ) );

% Solve the equation ( If you have luck! )
[ TIn.T U E ] = Evolution( Ev.Mt, Eq, TIn, Uo, Ev );

% Exporting grid data
assignin( 'caller', inputname(6), TIn );

%-------------------------------------------------------------------------%
```

```
% WAVEREPORT - Matlab Script - Show log information

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Oct, 03, 2004
% $Id: WaveReport.m,v 1.13 2007/05/26 03:39:44 akiles Exp $
%-------------------------------------------------------------------------%

% Log information
%-------------------------------------------------------------------------%
fprintf('| Evolution Equation: %s\n', Eq )

fprintf('|\n')
```

```
    fprintf('| Evolution method = %s\n', Ev.Mt    )
    fprintf('| Time samplings:' )
    fprintf('  Tf = %f',    TIn.Tf )
    fprintf('  dt = %f\n', TIn.dt )

    fprintf('|\n')
    fprintf('| Spacial discretisation:\n' )

    fprintf('| R discretisation = %s\n', DMt.MtR )
    fprintf('| Nr = %3i',      SpG.Nr )
    fprintf('  Ra = %7.4f',    SpG.Ra )
    fprintf('  Rb = %7.4f\n', SpG.Rb )

    if( isfield( SpG, 'Nx' ) )

      fprintf('| X discretisation = %s\n', DMt.MtX )
      fprintf('| Nx = %3i',      SpG.Nx )
      fprintf('  Xa = %7.4f',    SpG.Xa )
      fprintf('  Xb = %7.4f\n', SpG.Xb )

    end
    fprintf('|\n')
    fprintf('| Initial Condition: %s\n',  InC.system   )

    wr_size = size( InC.P );

    for wr_ii = 1:wr_size( 1 )

      for wr_jj = 1:wr_size( 2 )

        if( isfield( InC.P(wr_ii,wr_jj), 'pulse' ) )
          pulse = InC.P(wr_ii,wr_jj).pulse;
        else
          pulse = 'EYM_Pulse';
        end

        if( ~strcmp( pulse, 'PulseNull' ) )

          fprintf('| P(%1i,%1i) - %s\n|       ', wr_ii, wr_jj, pulse )
          fprintf('  - A = %7.4f ', InC.P(wr_ii,wr_jj).A   )
          fprintf(' Ro = %7.4f ',  InC.P(wr_ii,wr_jj).Ro )
          fprintf(' L = %7.4f ',   InC.P(wr_ii,wr_jj).L   )

          if( isfield( InC.P(wr_ii,wr_jj), 'D' ) )

            fprintf(' D = %2.0f', InC.P(wr_ii,wr_jj).D );

          end

          if( isfield( InC.P(wr_ii,wr_jj), 'epsilon' ) )

            fprintf(' e = %2.0f', InC.P(wr_ii,wr_jj).epsilon );

          end

          fprintf('\n' )

          if( isfield( InC.P(wr_ii,wr_jj), 'C' ) )

            fprintf('|         C = [ ' )
            fprintf('%4.2f ', InC.P(wr_ii,wr_jj).C )
            fprintf(']\n' )

          end

        end

      end

    end

    if( exist( 'ElapsedTime' ) )

      h = floor( ElapsedTime / ( 60*60 ) );
      e = ElapsedTime - h*60*60;
      m = floor( e / 60 );
      s = round( e - m * 60 );

      fprintf('| Elapsed Time = %i:%02i:%02i\n', h, m, s );

    end

    %-------------------------------------------------------------------------%
```

```
% PSSAVE - Script to save relevant data
%
% It needs some fixed variables

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Nov, 09, 2004
% $Id: PSsave.m,v 1.6 2006/05/24 07:18:28 akiles Exp $
%-------------------------------------------------------------------------------%

% Grid variables:     SpG, TIn
% Initial conditios: InC
% Equation:           Eq EqP
% Numerical Methods: DMt Ev
% Elapsed time:       ElapsedTime
% Solution:           U E

if( isfield( SpG, 'Nx' ) )

  gridsize = sprintf('%ix%i', SpG.Nr, SpG.Nx );

else

  gridsize = sprintf('%i', SpG.Nr );

end

filename = sprintf( 'solutions/%s-%s-%s-%s.mat', ...
                    Eq,                          ...
                    DMt.MtR(3:4),                ...
                    gridsize,                    ...
                    datestr( now, 30 ) );

fprintf( 'Saving data on file %s\n', filename )

save( filename, ...
      'SpG', 'TIn', 'InC', 'Eq', 'EqP', ...
      'DMt', 'Ev', 'ElapsedTime', 'U', 'E')

%-------------------------------------------------------------------------------%
```

Discretização Espacial

Essas funções geram as matrizes de derivação para os métodos de diferenças finitas e pseudo-espectral.

```
function [ X, Dx ] = MtFD( a, b, N, MtP )

% MTFD - Finite Difference Differentiation Matrices
%
% usage: [ X, Dx ] = MtFD( a, b, N, MPr )
%
% Input parameters:
%      a - Beginning of interval
%      b - End of interval
%      N - Number of points
%    MtP - Method Parameters
%      .Order  - Order of differentiation scheme.
%      .Points - M-Function that calculate the grid points. This
%                function must receive the same parameters as
%                Matlab linspace function.
%
% Output parameters
%    X   - Grid points
%    Dx  - Differentiation matrices

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Jul, 31, 2003
% $Id: MtFD.m,v 1.5 2007/05/26 03:39:44 akiles Exp $
%-------------------------------------------------------------------------------%
```

```
    % Calculate grid points
    X = feval( MtP.Points, a, b, N );

    % Force X to be a column vector
    X = reshape( X, length(X), 1 );

    % Scheme band and total size
    B = ceil( MtP.Order / 2 );
    nn = 2*B +1;

    % Differentiation matrices
    Dx = zeros( N, N, 2 );

    %-------------------------------------------------------------------------%

    % Extra points on boundaries
    % E = ceil( nn/2 );
    E = 0;

    % Column index
    col = 1:(nn+E);

    % Left Layer
    for ii = 1:B
      Dx( ii, col, : ) = FDWeights( X(col), X(ii) );
    end

    % Inner points
    col = -B:B;

    for ii = (B+1):(N-B)
      Dx( ii, ii+col, : ) = FDWeights( X(ii+col), X(ii) );
    end

    % Right Layer
    col = (N-B-1-E):N;

    for ii = (N-B+1):N
      Dx( ii, col, : ) = FDWeights( X(col), X(ii) );
    end

    %-------------------------------------------------------------------------%
```

```
    function [ X, Dx ] = MtChebyshev( a, b, N, varargin )

    % MTCHEBYSHEV - Pseudospectral method with Chebyshev basis
    %
    % usage: [ X Dx ] = MtChebyshev( a, b, N )
    %
    %  a - Begining of interval
    %  b - End of interval
    %  N - Number of points
    %
    %  X  - Chebyshev-Gauss-Lobatto points = R(j) = cos( pi*j/N )
    %  Dx - Differentiation matrices

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Jul, 21, 2003
    % $Id: MtChebyshev.m,v 1.3 2004/10/04 23:11:48 akiles Exp $
    %-------------------------------------------------------------------------%

    % Chebyshev differentiation matrices
    [ w Dw ] = chebdif( N, 2 );

    % Coordinate transformation
    c = -( b - a ) / 2;
    X = ( w-1 )*c + a;

    % Transformation of derivatives
    Dx(:,:,1) = Dw(:,:,1) / c;      % First derivative
    Dx(:,:,2) = Dw(:,:,2) / c^2;    % Second derivative

    %-------------------------------------------------------------------------%
```

```
function [ X, Dx ] = MtChebyshevTr( a, b, N, P )

% MTCHEBYSHEV - Pseudospectral method with Transformed Chebyshev basis
%
% usage: [ X, Dx ] = MtChebyshevTr( a, b, N, P )
%
% Input parameters:
%   a - Beginning of interval
%   b - End of interval
%   N - Number of points
%   P - Method Parameters
%     .alpha
%
% Output parameters
%   X   - Grid points
%   Dx  - Differentiation matrices

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Jul, 21, 2003
% $Id: MtChebyshevTr.m,v 1.6 2006/05/24 07:18:28 akiles Exp $
%-------------------------------------------------------------------------------%

asin_a = asin( P.alpha );

% Chebyshev differentiation matrices
[ y Dy ] = chebdif( N, 2 );

w = asin( P.alpha * y ) / asin_a;

% Transformation [-1,1] -> [-1,1]
T =  asin_a   * sqrt( 1 - ( P.alpha * y).^2 ) / P.alpha;
U = -asin_a^2 * y;
V =  asin_a^2 * ( 1 - ( P.alpha * y).^2 ) / P.alpha^2;

Dw(:,:,1) = repmat(T,1,N).*Dy(:,:,1);
Dw(:,:,2) = repmat(U,1,N).*Dy(:,:,1) +repmat(V,1,N).*Dy(:,:,2);

% Transformation [-1,1] -> [a,b]
c = -( b - a ) / 2;

X = ( w-1 )*c + a;

% Transformation of derivatives
Dx(:,:,1) = Dw(:,:,1) / c;      % First derivative
Dx(:,:,2) = Dw(:,:,2) / c^2;    % Second derivative

%-------------------------------------------------------------------------------%
```

```
function R = PointsChebyshev( a, b, n )

% POINTSCHEBYSHEV - Calculate the Chebyshev points
%
% usage: R = PointsChebyshev( a, b, n )
%
% Calculate n Chebyshev-Gauss-Lobatto points on interval [a,b]

%-------------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Aug, 01, 2003
% $Id: PointsChebyshev.m,v 1.3 2004/10/04 23:11:49 akiles Exp $
%-------------------------------------------------------------------------------%

% Calculate the points on interval [-1,1]
x = chebdif( n, 0 );

% Transform to interval [a,b]
R = ( 1-x )*( b-a ) /2 + a;

%-------------------------------------------------------------------------------%
```

Evolução Temporal

Essas são as funções que implementam os métodos de passo temporal. A função Evolution é chamada pela WaveSolver para avaliar a evolução temporal da solução. As demais funções dessa seção, implementam diversos métodos de evolução temporal.

Evolução Temporal

```
function [T,U,E] = Evolution( Mt, Eq, TIn, Uo, Ep )

% EVOLUTION - Evolution of Wave Equations
%
%  Usage: [T,U E] = Evolution( Mt, Eq, TIn, Uo, Ep )
%
%  Mt   - Time step method
%  Eq   - Wave equation M-function
%  TIn  - Time Interval
%    .Tf - Final time (actual value: first n*dt larger than Tf)
%    .dt - Sample solution at t = n*dt
%  Uo   - Initial Condition
%  Ep   - Evolution parameters
%    .mU - Maximum value of U allowed
%    .dt - Initial time step
%    .P  - Time step method parameters
%    .max  - Maximum step size alloed on adaptive methods
%
%  T - Times of Samplings
%  U - Solution
%  E - Energy Density

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Nov, 10, 2003
% $Id: Evolution.m,v 1.8 2005/12/30 21:55:13 akiles Exp $
%-------------------------------------------------------------------------%

% Calculating Time Parameters
%---------------------------------------------------------%

% Interval between samplings
St = TIn.dt;

% Number of samples + Initial condition
Ns = ceil( TIn.Tf / St ) +1;

% Actual final time
Tf = (Ns-1) * St;

% Time values at each sampling
T = 0 : St : TIn.Tf;

% Creating Arrays
%---------------------------------------------------------%

% Getting System Size
foo = [ Eq  '_SystemSize' ];
[ Usize Esize ] = feval( foo );

% Getting Discretization Size
[ Nr Nx Ni ] = size(Uo);

% Constructing the Arrays
U = zeros( Nr, Nx, Usize, Ns );
E = zeros( Nr, Nx, Esize, Ns );

% Echo display
%---------------------------------------------------------%

% Number of echos and Echo index
Ne = min([ Ns 64 ]);
Ec = SampleIndex( Ns, Ne );

fprintf('%s\n',  repmat('_',1,Ne) )
fprintf('=');
ee = 1;
```

```
70  % Initial Condition
    %-------------------------------------------------------------%
    U(:,:,:,1) = Uo;

    Energy = [ Eq, '_Energy' ];
75  E(:,:,:,1) = feval( Energy, Uo(:) );

    % Starting Evolution Method
    %-------------------------------------------------------------%

80  foo = [ Mt, '_start' ];
    feval( foo, Eq, Ep.dt, Ep.P );

    % Evolution
    %-------------------------------------------------------------%
85  for ii = 2:Ns

      % Echo
      if ii == Ec(ee+1), fprintf('='); ee = ee+1; end

90    % Time-Steping
      Uo(:) = feval( Mt, Uo(:), St );

      % Store Solution
      U(:,:,:,ii) = Uo;
95
      % Evaluate Energy Density
      E(:,:,:,ii) = feval( Energy, Uo(:) );

      % Check for instabilities
100   if BlowUp( Uo, Ep.mU )

        fprintf('\nInterrupted at time %f, after %i samplings\n\n', T(ii), ii );

        U = U(:,:,:,1:ii);
105     E = E(:,:,:,1:ii);
        T = T(      1:ii);

        return
      end
110 end

    fprintf('\n');

    %-------------------------------------------------------------------------------%
115
    function B = BlowUp( Uo, mU )
                                        % Uo BlowUp if some element:
    B = min(isfinite(Uo(:))) == 0   | ... % is nan or inf, or
        max(abs      (Uo(:)))   > mU;     % is larger than max
120
    %-------------------------------------------------------------------------------%
```

Runge-Kutta de Ordem s

```
    function U = RungeKutta( Uo, T )

4   % RUNGEKUTTA - Kunge-Kutta Method of order S
    %
    %  Usage: U = RungeKutta( Uo, T )
    %
    %  Uo - Initial condition
9   %  T  - Time interval
    %
    %  U - Solution at t = T

    %-------------------------------------------------------------------------------%
14  % Luis Alberto D'Afonseca
    % since: Jul, 12, 2003
    % $Id: RungeKutta.m,v 1.3 2004/10/04 23:11:47 akiles Exp $
    %-------------------------------------------------------------------------------%

19  global RK_eq;
    global RK_dt;
    global RK_s;

    %-------------------------------------------------------------------------------%
24
```

```
    % Number of steps;
    N = ceil( T/RK_dt );

    % Actual time step
29  dt = T/N;

    U = Uo;

    for jj = 1:N
34
      Uo = U;

      for kk = RK_s:-1:1

39      U = Uo + (dt/kk) * feval( RK_eq, U );

      end
    end

44  %-------------------------------------------------------------------------%
```

```
 1
    function RungeKutta_start( Eq, dt, P )

    % RUNGEKUTTASTART - Kunge-Kutta Method of order S starter
    %
 6  %  Usage: RungeKutta_start( Eq, T, dt, P )
    %    Eq       - Wave equation M-function
    %    dt       - Time step
    %    P.Order - Runge-Kuttar order

11  %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Dez, 21, 2003
    % $Id: RungeKutta_start.m,v 1.3 2004/10/04 23:11:48 akiles Exp $
    %-------------------------------------------------------------------------%
16
    global RK_eq;
    global RK_dt;
    global RK_s;

21  RK_eq = Eq;
    RK_dt = dt;
    RK_s  = P.Order;

    %-------------------------------------------------------------------------%
```

Runge-Kutta de Quarta Ordem

```
    function U = RungeKutta4( Uo, T )

    % RUNGEKUTTA4 - Kunge-Kutta Method of 4th order
 5  %
    %  Usage: U = RungeKutta4( Uo, T )
    %
    %  Uo - Wave initial condition
    %  T  - Time interval
10  %
    %  U  - Solution at t = T

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
15  % since: Dez, 22, 2003
    % $Id: RungeKutta4.m,v 1.3 2004/10/04 23:11:47 akiles Exp $
    %-------------------------------------------------------------------------%

    global RK_eq;
20  global RK_dt;

    %-------------------------------------------------------------------------%

    % Number of steps;
25  N = ceil( T/RK_dt );
```

```
    % Actual time step
    dt = T/N;

30  % Starting the method
    U = Uo;

    for ii = 1:N

35    K1 = dt * feval( RK_eq, U );
      K2 = dt * feval( RK_eq, U +0.5*K1 );
      K3 = dt * feval( RK_eq, U +0.5*K2 );
      K4 = dt * feval( RK_eq, U +    K3 );

40    U = U +( K1 +2*K2 +2*K3 +K4 )/6;

    end

    %-------------------------------------------------------------------------%
```

```
 1
    function RungeKutta4_start( Eq, dt, P )

    % RUNGEKUTTA4START - Kunge-Kutta Method of order 4th starter
    %
 6  %  Usage: RungeKutta_start( Eq, T, dt, P )
    %    Eq       - Wave equation M-function
    %    dt       - Time step

    %-------------------------------------------------------------------------%
11  % Luis Alberto D'Afonseca
    % since: Dez, 22, 2003
    % $Id: RungeKutta4_start.m,v 1.3 2004/10/04 23:11:47 akiles Exp $
    %-------------------------------------------------------------------------%

16  global RK_eq;
    global RK_dt;

    RK_eq = Eq;
    RK_dt = dt;
21
    %-------------------------------------------------------------------------%
```

Runge-Kutta Adaptativo

```


    function U = RungeKuttaAdapt( Uo, T )
 3
    % RUNGEKUTTAADAPT - Adaptive Kunge-Kutta Method
    %
    %  Usage: U = RungeKutta4( Uo, T )
    %
 8  %  Uo - Wave initial condition
    %  T  - Time interval
    %
    %  U  - Solution at t = T

13  %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Jan, 30, 2004
    % $Id: RungeKuttaAdapt.m,v 1.5 2005/12/30 21:55:13 akiles Exp $
    %-------------------------------------------------------------------------%
18
    global RKA_eq;
    global RKA_dt;
    global RKA_tol;
    global RKA_max;
23
    %-------------------------------------------------------------------------%

    % Starting
    t = 0;
28
    if RKA_dt > T
```

```
    RKA_dt = T;
  end

  while T - t > eps

    % Runge Kutta Partial Steps
    K1 = RKA_dt*feval(RKA_eq, Uo );
    K2 = RKA_dt*feval(RKA_eq, Uo +0.200*K1 );
    K3 = RKA_dt*feval(RKA_eq, Uo +0.075*K1 +0.225*K2 );
    K4 = RKA_dt*feval(RKA_eq, Uo +0.300*K1 -0.900*K2 +1.200*K3 );
    K5 = RKA_dt*feval(RKA_eq, Uo -0.203703703703704*K1 ...
                              +2.500000000000000*K2 ...
                              -2.592592592592593*K3 ...
                              +1.296296296296296*K4  );
    K6 = RKA_dt*feval(RKA_eq, Uo +0.029495804398148*K1 ...
                              +0.341796875000000*K2 ...
                              +0.041594328703704*K3 ...
                              +0.400345413773148*K4 ...
                              +0.061767578125000*K5  );

    % Fifth-Order step
    U = Uo +0.097883597883598*K1 ...
           +0.402576489533011*K3 ...
           +0.210437710437710*K4 ...
           +0.289102202145680*K6;

    % Error Estimate
    Er = -0.004293774801587*K1 ...
         +0.018668586093858*K3 ...
         -0.034155026830808*K4 ...
         -0.019321986607143*K5 ...
         +0.039102202145680*K6;

    % Some scaling may be needed for error mesurement!
    R = max(abs(Er(:)));

    % Debug Information
    %-------------------------------------------------------------%
  %   global Db_dt
  %   global Db_R

  %   Db_dt = [ Db_dt RKA_dt ];
  %   Db_R  = [ Db_R  R      ];
    %-------------------------------------------------------------%

    if R < RKA_tol
      % Acept the step
      % and enlarge the step size

      t = t + RKA_dt;

      % End of time interval
      if T - t <= eps, continue; end;

      % Preparing for next step
      Uo = U;

      % New step size
      if R ~= 0
        RKA_dt = 0.9*RKA_dt*(RKA_tol/R)^(0.2);
      end

      if RKA_dt > RKA_max
         RKA_dt = RKA_max;
      end

      if t + RKA_dt > T
        RKA_dt = T - t;
      end

    else
      % Do not acept the step
      % and reduce the step size

      RKA_dt = 0.9*RKA_dt*(RKA_tol/R)^(0.25);

      % if t + RKA_dt == t
      if RKA_dt < 2^-14
        error('Runge-Kutta Adaptive error: Step size to small');
      end

    end

  end
```

```
%-------------------------------------------------------------------------%
```

```
function RungeKuttaAdapt_start( Eq, dt, P )

% RUNGEKUTTAADAPTSTART - Adaptive Kunge-Kutta Method
%
%  Usage: RungeKuttaAdapt_start( Eq, T, dt, P )
%    Eq        - Wave equation M-function
%    dt        - Time step
%    P.tol     - Tolerance

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Jan, 30, 2004
% $Id: RungeKuttaAdapt_start.m,v 1.4 2004/11/09 00:46:26 akiles Exp $
%-------------------------------------------------------------------------%

global RKA_eq;
global RKA_dt;
global RKA_tol;
global RKA_max;

RKA_eq  = Eq;
RKA_dt  = dt;
RKA_tol = P.tol;
RKA_max = P.max;

% Debug Information
%-------------------------------------------------------------------------%

% global Db_dt
% global Db_R

% Db_dt = 0;
% Db_R  = 0;

%-------------------------------------------------------------------------%
```

Pulsos

As condições iniciais são construídas a partir de funções arbitrárias que chamamos de pulsos. A seguir, estão as funções que avaliam os esses pulsos.

```
function Uo = PulseNull( SpG, P )

% PULSENULL - Generates a null pulse
%
% usage: Uo = PulseNull( SpG, P )
%
%  SpG - Spatial grid
%     .R - R points
%     .X - cos(theta)
%  P  - Pulse parameters

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since November, 05, 2003
% $Id: PulseNull.m,v 1.5 2006/05/24 07:18:28 akiles Exp $
%-------------------------------------------------------------------------%

if( isfield( SpG, 'Nx' ) )

  Nx = SpG.Nx;

else

  Nx = 1;

end

% The Pulse
```

```
    Uo = zeros( SpG.Nr, Nx );

    %-------------------------------------------------------------------------%
```

```
    function Uo = PulseGaussian( SpG, P )
 3
    % PULSEGAUSSIAN - Generates a Gaussian pulse initial condition
    %
    % usage: Uo = PulseGaussian( SpG, P )
    %
 8  %  SpG - Spatial grid
    %     .R - R points
    %     .X - cos(theta)
    %  P  - Pulse parameters
    %     .A  - Amplitude
13  %     .Ro - Pulse center
    %     .L  - Pulse width
    %     .e  - Excentricity
    %     .C  - Multipole coeficients
    %
18  %  Uo = A exp( -( R - Ro )^2 /L )

    %-------------------------------------------------------------------------%
    %  Luis Alberto D'Afonseca
    %  since July, 10, 2003
23  % $Id: PulseGaussian.m,v 1.3 2004/10/04 23:11:49 akiles Exp $
    %-------------------------------------------------------------------------%

    % Angular Dependence
    [Nx, Ro, Lx] = PulseAngularDep( SpG, P );
28
    % Radial Dependence
    Nr = length( SpG.R );

    % D = R - Ro
33  D  = repmat( SpG.R, 1, Nx ) - repmat( Ro, Nr, 1 );

    % Pulse on R
    Fr = P.A * exp( -D.^2 / P.L );

38  % The Pulse
    Uo = Fr .* repmat(Lx,Nr,1);

    %-------------------------------------------------------------------------%
```

```
    function Uo = PulseRGaussian( SpG, P )

 4  % PULSERGAUSSIAN - Generates a Gaussian times r pulse initial condition
    %
    % usage: Uo = PulseGaussian( SpG, P )
    %
    %  SpG - Spatial grid
 9  %     .R - R points
    %     .X - cos(theta)
    %  P  - Pulse parameters
    %     .A  - Amplitude
    %     .Ro - Pulse center
14  %     .L  - Pulse width
    %     .e  - Excentricity
    %     .C  - Multipole coeficients
    %
    %  Uo = A r exp( -( R - Ro )^2 /L )
19
    %-------------------------------------------------------------------------%
    %  Luis Alberto D'Afonseca
    %  since July, 10, 2003
    % $Id: PulseRGaussian.m,v 1.1 2006/05/24 07:18:28 akiles Exp $
24  %-------------------------------------------------------------------------%

    % Angular Dependence
    [Nx, Ro, Lx] = PulseAngularDep( SpG, P );

29  % Radial Dependence
    Nr = length( SpG.R );
```

```
    % D = R - Ro
    D  = repmat( SpG.R, 1, Nx ) - repmat( Ro, Nr, 1 );

    % Pulse on R
    Fr = P.A * SpG.R .* exp( -D.^2 / P.L );

    % The Pulse
    Uo = Fr .* repmat(Lx,Nr,1);

    %-------------------------------------------------------------------------%
```

```
    function Uo = PulsePhi4( SpG, P )

    % PULSEPHI4 - Generates a Lambda Phi^4 pulse
    %
    % usage: Uo = PulsePhi4( SpG, P )
    %
    %  SpG - Spatial grid
    %    .R - R points
    %    .X - cos(theta)
    %  P  - Pulse parameters
    %    .A  - Aplitute
    %    .Ro - Pulse center
    %    .L  - Pulse width
    %    .C  - Multipole coeficients
    %
    %  Uo = A * tanh( L*( R - Ro ) )

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since Jul, 10, 2003
    % $Id: PulsePhi4.m,v 1.4 2006/05/24 07:18:28 akiles Exp $
    %-------------------------------------------------------------------------%

    % Angular Dependence
    [Nx, Ro, Lx] = PulseAngularDep( SpG, P );

    % Radial Dependence
    Nr = length( SpG.R );

    % D = R - Ro
    D  = repmat( SpG.R, 1, Nx ) - repmat( Ro, Nr, 1 );

    % Pulse on R
    Fr = tanh( D/P.L );

    % The Pulse
    Uo = P.A * Fr .* repmat(Lx,Nr,1);

    %-------------------------------------------------------------------------%
```

```
    function Uo = PulseEYM( SpG, P )

    % PULSEEYM - Generates a pulse for EYM equations
    %
    % usage: Uo = PulseEYM( SpG, P )
    %
    %  SpG - Spatial grid
    %    .R - R points
    %  P  - Pulse parameters
    %    .Ro - Pulse center
    %    .L  - Pulse width

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % $Id$
    %-------------------------------------------------------------------------%

    Ro = P.Ro;
    L  = P.L;
    R  = SpG.R;

    x  = Ro / L;
```

```
    th = tanh( x );
25  ch = sech( x )^2;
    a =  1 / th - 1;
    b = ch / th / ( 1 - th );

    % Pulse on R
30  Uo = ( 1 + a*( 1 + b*R/L ) .* exp( -2*(R/L).^2 ) ) .* tanh( (Ro-R)/L );

    %-------------------------------------------------------------------------%
```

```
    function [Nx, Ro, Lx] = PulseAngularDep( SpG, P )
 3
    % PULSEANGULARDEP - Angular dependence on initial pulses
    %
    % usage: [Ro Lx] = PulseAngularDep( SpG, P )
    %
 8  % input:
    %  SpG - Spatial grid
    %     .R - R points
    %     .X - cos(theta)
    %  P   - Pulse parameters
13  %     .Ro - Pulse center
    %     .e  - Excentricity
    %     .C  - Multipole coeficients
    %
    % output:
18  %  Nx = length of X or 1 if X doesn't existe
    %  Ro = P.Ro ( 1 +(e^1 -1) x^2 )^(1/2)
    %  Lx = sum( C(i) P_i(x) )

    %-------------------------------------------------------------------------%
23  % Luis Alberto D'Afonseca
    % since Aug, 26, 2003
    % $Id: PulseAngularDep.m,v 1.5 2006/05/24 07:18:28 akiles Exp $
    %-------------------------------------------------------------------------%

28  if ~isfield( SpG, 'X' )        % Spherical symmetric case

      Nx = 1;       % Size of angular coordinate vector
      Lx = 1;       % Multipole polinomial
      Ro = P.Ro;    % Center of pulse, for each angle
33
    else                           % Axissymmetric case

      Nx = length( SpG.X );

38    % Multipolar dependence
      if ~isfield( P, 'C' ), Lx = ones(1,Nx);          % Monopolar pulse
      else                   Lx = SuperLegendre( SpG.X, P.C );
      end

43    Ro = repmat( P.Ro, 1, Nx );

    end

    %-------------------------------------------------------------------------%
```

```
    function P = SuperLegendre( X, C )
 3
    % SUPERLEGENDRE - Superposition of Legendre polinomials
    %
    % Usage: P = SuperLegendre( X, C )
    %
 8  % P = sum_i ( C_i P_i(X) )

    %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Aug, 18, 2003
13  % $Id: SuperLegendre.m,v 1.4 2004/10/05 05:44:33 akiles Exp $
    %-------------------------------------------------------------------------%

    % Initializations
    Nx = length( X );
18  Nc = length( C );
```

```
    X = reshape( X, 1, Nx );

    % Special Cases
23  if      Nc == 1,    P = C * ones( 1, Nx ); return
    elseif Nc == 2,     P = C(1) + C(2)*X;      return
    end;

    % Legendre Polinomials
28  L = zeros( Nc, Nx );

    L(1,:) = 1;
    L(2,:) = X;

33  % Recurrence
    for k = 2:(Nc-1)
      L(k+1,:) = (2*k+1)/(k+1) *X.*L(k,:) - k/(k+1) * L(k-1,:);
    end;

38  % Superposition
    P = sum( repmat( reshape(C,Nc,1), 1,Nx ) .* L );

    %--------------------------------------------------------------------------%
```

Quadraturas

Essas são as funções usadas para calcular as quadraturas da energia das soluções.

```
    function W = ClenCurtWeights( a, b, N )

 4  % CLENCURTWEIGHTS - Weights of Clenshaw-Curtis quatrature
    %
    % usage: W = ClenCurtWeights( a, b, N )
    %
    % Weights for integration of N points on interval [a,b]
 9
    %--------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Nov, 10, 2004
    % $Id: ClenCurtWeights.m,v 1.1 2004/11/11 18:24:17 akiles Exp $
14  %--------------------------------------------------------------------------%

    N = N-1;

    theta = pi*(0:N)'/N;
19
    W = zeros( 1, N+1 );
    V = ones ( N-1, 1 );

    ii = 2:N;
24
    if( mod(N,2) == 0 )

      W(1)   = 1/(N^2-1);
      W(N+1) = W(1);
29
      for k = 1:N/2-1
        V = V - 2*cos( 2*k*theta(ii)) / (4*k^2-1);
      end

34    V = V - cos( N*theta(ii)) / (N^2-1);

    else

      W(1)   = 1/N^2;
39    W(N+1) = W(1);

      for k = 1:(N-1)/2
        V = V - 2*cos( 2*k*theta(ii)) / (4*k^2-1);
      end
44
    end

    W(ii) = 2*V/N;

49  W = (b-a) * W / 2;

    %--------------------------------------------------------------------------%
```

```
function fdW = FDWeights( X, Xi )

% FDWWEIGHTS - Finite Difference Stencil Weights
%
% usage: fdW = FDWeights( X, Xi )
%
% This funciton is called by MtFD.m

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: July, 31, 2003
% $Id: FDWeights.m,v 1.3 2004/10/04 23:11:48 akiles Exp $
%-------------------------------------------------------------------------%

N = length( X );

w = zeros( N, 3 );
w(1,1) = 1;

C1 = 1;
C4 = X(1) -Xi;

for ii = 1:(N-1)

  m = min([ ii 2 ]);

  C2 = 1;
  C5 = C4;
  C4 = X(ii+1) -Xi;

  for jj = 0:(ii-1)

    C3 = X(ii+1) -X(jj+1);
    C2 = C2*C3;

    for kk = m:-1:1
      w(ii+1,kk+1) = C1 * ( kk*w(ii,kk) -C5*w(ii,kk+1) ) / C2;
    end;

    w(ii+1,1) = -C1 * C5 * w(ii,1) / C2;

    for kk = m:-1:1
      w(jj+1,kk+1) = ( C4*w(jj+1,kk+1) -kk*w(jj+1,kk) ) / C3;
    end;

    w(jj+1,1) = C4 * w(jj+1,1) / C3;

  end;

  C1 = C2;

end;

fdW = w(:,2:3);

%-------------------------------------------------------------------------%
```

```
function I = QuadAxialCheb( G, E )

% QUADAXIALCHEB - Quadrature of Axial fuction on Chebyshev-Lobato points
%
% usage: I = QuadAxialCheb( SpG, E )

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Nov, 09, 2004
% $Id: QuadAxialCheb.m,v 1.2 2005/12/30 21:55:14 akiles Exp $
%-------------------------------------------------------------------------%

[ Nr Nx Nf Nt ] = size( E );

% Integration over r
%-------------------------------------------------------------------------%
```

```
18
    R  = repmat( G.R,                                    [ 1 Nx Nf Nt ] );
    Wr = repmat( ClenCurtWeights( G.Ra, G.Rb, Nr )', [ 1 Nx Nf Nt ] );

    Ir = sum( R.*Wr.*E, 1 );
23
    % Integration over x
    %-------------------------------------------------------------------------%

    Wx = repmat( ClenCurtWeights( G.X(1), G.X(end), Nx ), [ 1 1 Nf Nt ] );
28
    Ix = sum( Wx.*Ir, 2 );

    %-------------------------------------------------------------------------%

33  I = squeeze( Ix );

    %-------------------------------------------------------------------------%
```

```
    function I = QuadAxialLinear( G, E )

 4  % QUADAXIALLINEAR - Quadrature of Axial fuction on linear spaced points
    %
    % usage: I = QuadAxialLinear( SpG, E )

    %-------------------------------------------------------------------------%
 9  % Luis Alberto D'Afonseca
    % since: Nov, 11, 2004
    % $Id: QuadAxialLinear.m,v 1.2 2005/12/30 21:55:14 akiles Exp $
    %-------------------------------------------------------------------------%

14  [ Nr Nx Nf Nt ] = size( E );

    % Integration over r
    %-------------------------------------------------------------------------%

19  wwr = repmat( ( G.R(end) - G.R(1) )/( Nr - 1 ), Nr, 1 );
    wwr([1 end]) = wwr([1 end]) / 2;

    R  = repmat( G.R, [ 1 Nx Nf Nt ] );
    Wr = repmat( wwr, [ 1 Nx Nf Nt ] );
24
    Ir = sum( R.*Wr.*E, 1 );

    % Integration over x
    %-------------------------------------------------------------------------%
29
    wwx = repmat( ( G.X(end) - G.X(1) )/( Nx - 1 ), 1, Nx );
    wwx([1 end]) = wwx([1 end]) / 2;

    Wx = repmat( wwx, [ 1 1 Nf Nt ] );
34
    Ix = sum( Wx.*Ir, 2 );

    %-------------------------------------------------------------------------%

39  I = squeeze( Ix );

    %-------------------------------------------------------------------------%
```

```
    function I = QuadAxialLinear( SpG, E )
 3
    % QUADPHI - Quadrature on phi of an axial fuction
    %
    % usage: I = QuadPhi( SpG, E )

 8  %-------------------------------------------------------------------------%
    % Luis Alberto D'Afonseca
    % since: Nov, 19, 2005
    % $Id: QuadPhi.m,v 1.1 2006/05/24 07:18:28 akiles Exp $
    %-------------------------------------------------------------------------%
13
    [ Nr Nx Nf Nt ] = size( E );
```

```
R = repmat( SpG.R, [ 1 Nx Nf Nt ] );

I = 2*pi*R.*E;

%-------------------------------------------------------------------------%
```

Gráficos

Esses são os programas que geram os gráficos exibidos nesse trabalho.

```
function M = ShowWave( varargin )

% SHOWWAVE - Show wave
%
% usage: M = ShowWave( SpG, TIn, U, md, N )
%   SpG  - Spacial grid
%   TIn  - Time interval
%   U    - Wave solution
%   md - Plotting Mode
%        md(1) = 1 - waterfall
%                2 - mesh
%                3 - surf
%                4 - contour
%        md(2) = 1 - simple plot
%                2 - double plot
%        md(3) = 1 - Normal view
%                2 - 2D view
%   N - Maximum number of points to be plotted
%       [ MaxR MaxT ] or [ MaxR MaxX MaxT ] if axial wave
%
% output: M - movie handle if axial

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Jul, 15, 2003
% $Id: ShowWave.m,v 1.10 2006/05/24 07:18:27 akiles Exp $
%-------------------------------------------------------------------------%

% Raise figure
figure(gcf);

% Organizing inputs
%-------------------------------------------------------------------------%

% No input arguments
if( nargin == 0 )

  SpG = evalin('caller','SpG');
  TIn = evalin('caller','TIn');
  U   = evalin('caller','U'  );

  [ md N ] = ParseOptions( varargin, 0, 0 );

% If SpG is the first argument
elseif( isfield( varargin{1}, 'R' ) )

  SpG = varargin{1};
  TIn = varargin{2};
  U   = varargin{3};

  [ md N ] = ParseOptions( varargin, nargin, 3 );

% If U is the first argument
elseif( ndims( varargin{1} ) == 4 )

  SpG = evalin('caller','SpG');
  TIn = evalin('caller','TIn');

  U = varargin{1};

  [ md N ] = ParseOptions( varargin, nargin, 1 );

else

  SpG = evalin('caller','SpG');
  TIn = evalin('caller','TIn');
  U   = evalin('caller','U'  );
```

```
70      [ md N ] = ParseOptions( varargin, nargin, 0 );

    end

    [ Nr Nx ii Nt ] = size( U );
75
    % Axially Symmetric Waves
    %--------------------------------------------------------------------------%

    if Nx > 1
80
      if( length(N) == 2 ), N = N([ 1 1 2 ]); end

      if( nargout == 1 ), M = ShowAxialWave( SpG, TIn, U, md, N );
      else,               ShowAxialWave( SpG, TIn, U, md, N );
85    end

      return
    end

90  % Sampling
    %--------------------------------------------------------------------------%
    Rind = SampleIndex( Nr, N(1) );
    Tind = SampleIndex( Nt, N(2) );

95  RInV = Rind(end:-1:1);

    % Plotting Arrays
    TT = TIn.T(Tind);

100 if md(2) == 2
      % Double plot
      RR = [ -SpG.R(RInV); SpG.R(Rind) ];
      UU = squeeze( U([RInV Rind], 1, 1, Tind ) ).';

105   LR = repmat([ -SpG.R(end) 0 SpG.R(end) ], Nt, 1 );
      LT = repmat( TIn.T', 1, 3 );
      LU = [squeeze(U(end,1,1,:)) squeeze(U(1,1,1,:)) squeeze(U(end,1,1,:))] + 0.1;

    else
110   % Simple plot
      RR = SpG.R(Rind);
      UU = squeeze( U(Rind, 1, 1, Tind ) ).';

      LR = repmat([ 0 SpG.R(end) ], Nt, 1 );
115   LT = repmat( TIn.T', 1, 2 );
      LU = [ squeeze(U(1,1,1,:)) squeeze(U(end,1,1,:)) ];
      LU(:,2) = LU(:,2) + 0.01;

    end
120
    % Colormap
    %--------------------------------------------------------------------------%
    cax = newplot;

125 if md(3) == 2
      aU = UU;
      colormap gray
      brighten(-0.5)
    else
130   aU = abs(UU);
      colormap default
    end

    % Plotting
135 %--------------------------------------------------------------------------%

    switch md(1)
     case 1
      hh = waterfall( RR, TT, UU, aU );
140   set( hh, 'edgecolor','k' )
     case 2

      hold on

145   mesh( RR, TT, UU, aU,      ...
            'MeshStyle', 'row', ...
            'edgecolor', 'k',   ...
            'linewidth', 0.5 );

150   line( LR, LT, LU, 'color','k', 'linewidth', 1 );

      hold off

     case 3
```

```
155     surf( RR, TT, UU, aU );
        shading interp
       case 4
        contour( RR, TT, UU, 20 );
        grid on
160   end

      if md(3) == 2
        view(2);
      else
165     view(3);
      end

      % Labels
      xlabel('r')
170   ylabel('t')

      axis tight
      grid on

175   aRatio = max(abs(MinMax(UU(:,:,1,:)))) / 10;

      %% set( gca, 'dataaspectratio', [ 1 1 aRatio ] );

      set( gca, 'xtick', -32:16:32, ...
180             'ytick',  0:16:128, ...
                'zgrid', 'off' );

      %%        'ztick',  -32:4:32, ...

185   set( gcf, 'nextplot', 'replace' );

      %-------------------------------------------------------------------------%
      %-------------------------------------------------------------------------%
      function [ md, N ] = ParseOptions( inputs, ninputs, used )
190
        % Default values of md and N
        mdef = [  2 1 1 ];
        Ndef = [ 300 50 ];

195     % Check the number of remaining inputs
        switch( ninputs - used )

          % No remaining input
          case 0
200         md = mdef;
            N  = Ndef;

          % One remaining input
          case 1
205         md = inputs{ninputs};
            N  = Ndef;

          % Two remaining inputs
          case 2
210
            I = inputs{ninputs-1};
            N = inputs{ninputs  };

            % Check if I is []
215         if( length(I) == 0 ), md = mdef;
            else,                 md = I;
            end

           % Something is wrong
220        otherwise
             error( 'Wrong number of inputs' )

        end

225   %-------------------------------------------------------------------------%
```

```
      function M = ShowAxialWave( SpG, TIn, U, md, N )
  3
      % SHOWAXIALWAVE - Show Axissymmetric wave
      %
      % usage:
      %   This function is called by ShowWave, or directly by
  8   %   M = ShowAxialWave( SpG, TIn, U, md, N )
      %
```

```
% inputs
%   SpG  - Spacial grid
%   TIn  - Time interval
%   U    - Wave solution
%   md - plot mode
%       md(1) = 1 - not used = mesh
%               2 - mesh
%               3 - surf
%               4 - contour
%       md(2) = 1 - Single plot
%               2 - Double plot
%       md(3) = 1 - Normal view
%               2 - 2D view
%   N - Maximum munger of points to be plotted
%       [ MaxR  MaxX  MaxT ]
%
% output: M - Movie

%-------------------------------------------------------------------------%
% Luis Alberto D'Afonseca
% since: Aug, 05, 2003
% $Id: ShowAxialWave.m,v 1.11 2005/12/30 21:55:14 akiles Exp $
%-------------------------------------------------------------------------%

% Getting sizes
[ Nr, Nx, Nf, Nt ] = size( U );

if( Nf > 12 ), Nf = 12; end;

% Sampling
%-------------------------------------------------------------------------%
Ir = SampleIndex( Nr, N(1) );
Ix = SampleIndex( Nx, N(2) );
It = SampleIndex( Nt, N(3) );
nt = length(It);

UU = permute( U(Ir,Ix,1:Nf,It), [2,1,3,4] );

% Coordinate transformation
%-------------------------------------------------------------------------%

% Points to be ploted
R = SpG.R(Ir);
X = SpG.X(Ix);

% If it is not small domain hide inner boundary crop
if( R(end) > 1 )
  R = R - R(1);
  X = X / abs( X(1) );
end

% Transforming
Th = pi/2 - acos( X );

[ Gr, Gh ] = meshgrid(  R, Th );
[ Gx, Gy ] = pol2cart( Gh, Gr );

% Figure Handling
%-------------------------------------------------------------------------%
cax = newplot;

% Subplot manager
%-------------------------------------------------------------------------%
%  Nsub - number of subplots
%  Nlin - number of lines of plots
%  Ncol - number of columns

Nsub = Nf;

if     ( Nsub == 3 ), Nlin = 1;
elseif( Nsub <= 4 ), Nlin = ceil( Nsub / 2 );
else,                Nlin = ceil( Nsub / 3 );
end

Ncol = ceil( Nsub / Nlin );

% Axis Preparation
%-------------------------------------------------------------------------%

maxR = max( R );

if( maxR == 0 ), maxR = 1; end

for ii = 1:Nsub
  maxU(ii) = max(abs(MinMax( UU(:,:,ii,:) )));
```

```
      if maxU(ii) == 0, maxU(ii) = 1; end
    end

98  [ MinAll MaxAll] = MinMax( UU );

    MaxAbs = max(abs([ MinAll MaxAll ]));

    % Color values
103 if md(3) == 2
      aU = UU;
      colormap gray
    else
      aU = abs(UU);
108   colormap default
    end

    % Double Plot
    if md(2) == 2
113
      Ax = [ -1  1 -1  1 ] * maxR;

      Gx = [ -fliplr(Gx)  Gx ];
      Gy = [  fliplr(Gy)  Gy ];
118
      UU = [ flipdim(UU,2)  UU ];
      aU = [ flipdim(aU,2)  aU ];

    else
123
      Ax = [  0  1 -1  1 ] * maxR;

    end

128 % Plotting
    %-------------------------------------------------------------------------%

    if md(1) == 4

133   colormap gray

      % Layers
      L = linspace( MinAll, MaxAll, 20 );

138 end

    % For each time
    for ii = 1:nt

143   % For each function
      for jj = 1:Nsub

        subplot( Nlin, Ncol, jj );

148     switch md(1)

          case {1,2},
            mesh( Gx, Gy, UU(:,:,jj,ii), aU(:,:,jj,ii) );
            axis ( [ Ax -maxU(jj) maxU(jj) ] );
153         axis square;
            caxis( [ 0 maxU(jj) ] );

          case 3,
            surf( Gx, Gy, UU(:,:,jj,ii), aU(:,:,jj,ii) );
158         shading interp
            axis ( [ Ax -maxU(jj) maxU(jj) ] );
            axis square;
            caxis( [ 0 maxU(jj) ] );

163       case 4,
            contourf(  Gx, Gy, UU(:,:,jj,ii), L );
            axis( Ax );
            axis equal;

168     end

        % 2D view?
        if( md(3) == 2 )
          view(2)
173       axis equal tight
          caxis( [-MaxAbs MaxAbs ] )
        end

      end
178
      % If requested return movie
```

```
    if nargout == 1, M(ii) = getframe;
    else             drawnow
    end
183
  end

  set(gcf,'nextplot','replace')

188 %-------------------------------------------------------------------------%
```

```
  2 function [ TT, RR, GG, BB ] = ShowEnergy( varargin )

  % SHOWENERGY Evaluates and show total energies
  %
  % For now only for EqAbelian and EqYM
  7 %
  % usage: [ T, R, G, B ] = ShowEnergy( SpG, TIn, E, DMt )
  %   SpG  - Spacial grid
  %   TIn  - Time interval
  %   E    - Wave Energy
 12 %
  % output:
  %   T - Total energy
  %   R - Energy on red   component
  %   B - Energy on blue  component
 17 %   G - Energy on green component


  %-------------------------------------------------------------------------%
  % Luis Alberto D'Afonseca
 22 % since: Apr, 26, 2005
  % $Id: ShowEnergy.m,v 1.4 2007/05/26 03:39:43 akiles Exp $
  %-------------------------------------------------------------------------%

  % Organizing inputs
 27 %-------------------------------------------------------------------------%

  % No input arguments
  if( nargin == 0 )

 32   SpG = evalin( 'caller', 'SpG' );
      TIn = evalin( 'caller', 'TIn' );
      E   = evalin( 'caller', 'E'   );
      DMt = evalin( 'caller', 'DMt' );

 37 elseif( nargin == 1 )

      E = varargin{1};

      SpG = evalin( 'caller', 'SpG' );
 42   TIn = evalin( 'caller', 'TIn' );
      DMt = evalin( 'caller', 'DMt' );

  elseif( nargin == 3 )

 47   SpG = varargin{1};
      TIn = varargin{2};
      E   = varargin{3};

      DMt = evalin( 'caller', 'DMt' );
 52
  else

    fprintf('Wrong number of paramaters\n')

 57 end

  %-------------------------------------------------------------------------%

  % Sizes
 62 [ Nr Nx Nf Nt ] = size( E );

  % Is it YM or Abelian?
  if( Nf == 1 | Nf == 3 | Nf == 4 )

 67   % Is Abelian
      ym = 0;

  elseif( Nf == 3 | Nf == 9 | Nf == 10 )
```

```
72      % Is YM
        ym = 1;

    else

77      fprintf('Wrong number of functions\n')

    end

    % % Grid Spacing
82  % if( strcmp( DMt.MtR, 'MtFD' ) & strcmp( DMt.Points, 'linspace' ) );
    %   MQ = 'QuadAxialLinear';
    % else
    %   MQ = 'QuadAxialCheb';
    % end
87  % % Evaluation and display of Energy
    % R = - feval( MQ, SpG, E(:,:,1,:) );

    R = -Quadrature( SpG, E(:,:,1,:), DMt );

92  % Graphic preparation
    figure(gcf)

    clf

97  t = TIn.T(1:Nt);

    if( ym )

      %   G = - Feval( MQ, SpG, E(:,:,2,:) );
102   %   B = - Feval( MQ, SpG, E(:,:,3,:) );

      G = -Quadrature( SpG, E(:,:,2,:), DMt );
      B = -Quadrature( SpG, E(:,:,3,:), DMt );

107   T = R + G + B;

      [ Ymin Ymax ] = MinMax( T );

      hc = plot( t, R, 'k-', ...
112              t, G, 'k-.', ...
                 t, B, 'k--' );

      set( hc, 'LineWidth', 2 );

117   hold on;

      h = plot( t, T, 'k' );

      hold off;
122
      legend( [ hc; h ], '1', '2', '3', 'total' )

    else

127   T = R;

      h = plot( t, R, 'k' );

      [ Ymin Ymax ] = MinMax( R );
132
    end

    set( h,    'LineWidth', 3 );
    set( gca, 'FontSize', 15, 'xtick', 0:8:2^8 );
137
    xlabel('t');
    ylabel('energia');

    Ymax = min( Ymax, 2*T(1) );
142
    M = Ymax / 100;

    Tmax = t( end );

147 axis( [ 0 Tmax -M Ymax+M ] );

    grid on;

    % Outputs
152 if( nargout == 1 )

      TT = T;

    elseif( nargout == 4 )
```

```
157
      TT = T;
      RR = R;
      GG = G;
      BB = B;
162
    end
```

```
 2  function M = ShowHistory( varargin )

    % SHOWHISTORY - Show history of an axissymmetric  wave
    %
    % usage:
 7  %   M = ShowHistory( SpG, TIn, U, md, N )
    %
    % inputs
    %   SpG  - Spacial grid
    %   TIn  - Time interval
12  %   U    - Wave solution
    %   md - plot mode
    %       md(1) = 1 - not used = mesh
    %               2 - mesh
    %               3 - surf
17  %               4 - contour
    %       md(2) = 1 - Single plot
    %               2 - Double plot
    %       md(3) = 1 - Normal view
    %               2 - 2D view
22  %   N - Maximum munger of points to be plotted
    %        [ MaxR MaxT ] or [ MaxR MaxX MaxT ] if axial wave


    %-------------------------------------------------------------------------%
27  % Luis Alberto D'Afonseca
    % since: Aug, 05, 2003
    % $Id: ShowHistory.m,v 1.1 2006/05/24 07:18:27 akiles Exp $
    %-------------------------------------------------------------------------%

32  % Organizing inputs
    %-------------------------------------------------------------------------%

    % No input arguments
    if( nargin == 0 )
37
      SpG = evalin('caller','SpG');
      TIn = evalin('caller','TIn');
      U   = evalin('caller','U'  );

42    [ md N ] = ParseOptions( varargin, 0, 0 );

    % If SpG is the first argument
    elseif( isfield( varargin{1}, 'R' ) )

47    SpG = varargin{1};
      TIn = varargin{2};
      U   = varargin{3};

      [ md N ] = ParseOptions( varargin, nargin, 3 );
52
    % If U is the first argument
    elseif( ndims( varargin{1} ) == 4 )

      SpG = evalin('caller','SpG');
57    TIn = evalin('caller','TIn');

      U = varargin{1};

      [ md N ] = ParseOptions( varargin, nargin, 1 );
62
    else

      SpG = evalin('caller','SpG');
      TIn = evalin('caller','TIn');
67    U   = evalin('caller','U'  );

      [ md N ] = ParseOptions( varargin, nargin, 0 );

    end
72
    if( length(N) == 2 ), N = N([ 1 1 2 ]); end
```

```
    % Getting sizes
77  [ Nr, Nx, Nf, Nt ] = size( U );

    % Sampling
    %-------------------------------------------------------------------------%

82  Nsub = min( [ Nt N(3) ] );

    Ir = SampleIndex( Nr, N(1) );
    Ix = SampleIndex( Nx, N(2) );
    It = SampleIndex( Nt, Nsub );
87
    UU = permute( U(Ir,Ix,1,It), [2,1,3,4] );

    % Coordinate transformation
    %-------------------------------------------------------------------------%
92
    % Points to be ploted
    R = SpG.R(Ir);
    X = SpG.X(Ix);

97  % If it is not small domain hide inner boundary crop
    if( R(end) > 1 )
      R = R - R(1);
      X = X / abs( X(1) );
    end
102
    % Transforming
    Th = pi/2 - acos( X );

    [ Gr, Gh ] = meshgrid(  R, Th );
107 [ Gx, Gy ] = pol2cart( Gh, Gr );


    % Axis Preparation
    %-------------------------------------------------------------------------%
112
    % Color values
    if md(3) == 2
      aU = UU;
      colormap gray
117 else
      aU = abs(UU);
      colormap default
    end

122 % maxR = max( R );
    % if( maxR == 0 ), maxR = 1; end

    % Only for the disertation figures
    maxR = 32;
127
    [ MinAll MaxAll ] = MinMax( UU );

    MaxAbs = max(abs([ MinAll MaxAll ]));

132 % Double Plot
    if md(2) == 2

      Ax = [ -1  1 -1  1 ] * maxR;

137   Gx = [ -fliplr(Gx)  Gx ];
      Gy = [  fliplr(Gy)  Gy ];

      UU = [ flipdim(UU,2)  UU ];
      aU = [ flipdim(aU,2)  aU ];
142
    else

      Ax = [  0  1 -1  1 ] * maxR;

147 end

    if md(1) == 4

      % Layers
152   L = linspace( MinAll, MaxAll, 20 );

    end

    figN = gcf;
157
    % Plotting
```

```
%-------------------------------------------------------------------------%

% For each function
for jj = 1:Nsub

  figure( jj )
  clf

  % Color values
  if md(3) == 2
    colormap gray
  else
    colormap default
  end

  switch md(1)

   case {1,2},
    mesh( Gx, Gy, UU(:,:,1,jj), aU(:,:,1,jj), 'edgecolor', 'k' );
    axis ( [ Ax -MaxAbs MaxAbs ] );
    axis square;
%    caxis( [ 0 MaxAll ] );

   case 3,
    surf( Gx, Gy, UU(:,:,1,jj), aU(:,:,1,jj) );
    shading interp
    axis ( [ Ax -MaxAbs MaxAbs ] );
    axis square;
%    caxis( [ 0 MaxAll ] );

   case 4,

    colormap gray

    contourf(  Gx, Gy, UU(:,:,1,jj), L );
    axis( Ax );
    axis square;
  end

  % 2D view?
  if( md(3) == 2 )
    view(2)
    axis equal tight
    caxis( [ -MaxAbs MaxAbs ] )
  end

  set( gca, 'xtick', -32:16:32, ...
            'ytick', -32:16:32, ...
            'fontsize', 15 );

  fprintf( 'F[%02i] -> t = %4.1f\n', jj, TIn.T( It( jj ) ) );

end

set(gcf,'nextplot','replace')

%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
function [ md, N ] = ParseOptions( inputs, ninputs, used )

  % Default values of md and N
  mdef = [  4 2 1 ];
  Ndef = [ 100 50 ];

  % Check the number of remaining inputs
  switch( ninputs - used )

    % No remaining input
    case 0
      md = mdef;
      N  = Ndef;

    % One remaining input
    case 1
      md = inputs{ninputs};
      N  = Ndef;

    % Two remaining inputs
    case 2

      I = inputs{ninputs-1};
      N = inputs{ninputs  };

      % Check if I is []
      if( length(I) == 0 ), md = mdef;
```

```
            else,                       md = I;
            end

247         % Something is wrong
            otherwise
              error('Wrong number of inputs')

        end
252
    %-------------------------------------------------------------------------%
```