INTRODUÇÃO À COMPUTAÇÃO CIENTÍFICA

Vº Trabalho: PVI

Tobias J. D. E. Rosa

2022

1º Período

# 

# Resumo

O presente trabalho introduz o método numérico de Runge Kutta de 4ª ordem e apresenta uma formulação para o desenvolvimento de um programa em Fortran para solução de problemas de valor inicial, (PVI). Na sequência, são apresentados diversos PVI’s e as respectivas soluções a partir do programa desenvolvido. Por fim, dois problemas são escolhidos para serem comparados com outra ferramenta de solução de EDO, a biblioteca “odeint” do Python, sendo estes o problema da dinâmica de um par engrenado em regime transitório considerando duas situações: (a) Apenas as engrenagens e (b) Engrenagens e os eixos.

# Solução do Problema

Ao modelar problemas de engenharia, frequentemente nos deparamos com as equações diferenciais ordinárias, EDO’s. Estas equações possuem uma família de soluções, e o que irá diferenciar a solução procurada das demais soluções será a condição inicial, ou seja, os valores físicos conhecidos quando a variável do domínio é zero (tempo, posição, temperatura etc.). Esse tipo de problema é conhecido como problema de valor inicial PVI. As soluções para as EDO’s podem ser obtidas analiticamente, no entanto, para grande parte das aplicações o modelo matemático se torna mais complicado, nestas circunstâncias, a solução do PVI por método numérico é mais aconselhável (RUGGIERO, 2008). Os problemas de valor iniciais geralmente são apresentados da seguinte forma:

É importante ressaltar que nos casos em que as equações forem de ordem maior, deverá ser aplicada a redução de ordem.

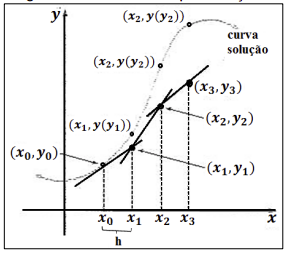
**Método de Runge Kutta de 4ª ordem**

Dado um ponto inicial , o método de Runge Kutta consiste em determinar o próximo ponto a partir de um incremento constante chamado passo, representado por , em que e onde é a inclinação da reta tangente à curva solução, definido pelas equações a seguir.

O próximo ponto é, portanto:

O procedimento é repetido sucessivamente até , cuja o número de pontos é definido em função do passo que é consequentemente a precisão do resultado, ou seja, quanto menor o incremento, maior a precisão. Uma representação geométrica do método de Runge Kutta de 4ª Ordem é ilustrada na Fig. 1.

**Figura 1** - Interpretação geométrica do método de Runge Kutta



**Implementação computacional**

A grande vantagem do método numérico de Runge Kutta é a possibilidade de resolver facilmente EDO’s que teriam solução analítica mais complexa. Entretanto, ao explorar fatores como maior número de graus de liberdade, maior precisão (Diminuição do passo) ou intervalos de tempos maiores, esse processo se torna totalmente inviável de se calcular manualmente, em função do número de cálculos realizados. Nestas circunstâncias, uma alternativa é a implementação computacional do método de Runge Kutta a partir do desenvolvimento de um programa em Fortran cuja estrutura é apresentada na Fig. 2.

**Figura 2** - Estrutura do programa.



Como ilustrado na Fig. 2, de modo geral o programa requer o modelo matemático e um arquivo de entrada com as respectivas condições iniciais, para que seja fornecida a solução em um arquivo de saída com a quantidade de pontos discretos desejada.

Para cumprir os objetivos deste trabalho, o programa desenvolvido deve ser capaz de resolver EDO’s para uma quantidade genérica de graus de liberdade . Como a solução exige que as equações sejam resolvidas acopladas, o programa deve ser da forma vetorial em que a dimensão do vetor equivale ao número de graus de liberdade .

O programa foi dividido em 5 arquivos, da seguinte forma:

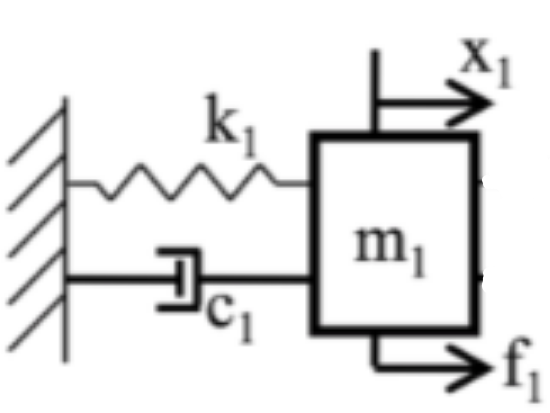
1. Entrada:No arquivo de entrada o usuário deve definir o número de graus de liberdade, a quantidade de pontos desejada, o intervalo do domínio e um vetor de condições iniciais.
2. Programa principal**:** No programa principal são definidas uma primeira função para o modelo matemático e uma segunda função que leva a função do modelo matemático no método numérico do módulo.
3. Módulo: O módulo comporta o método numérico de Runge Kutta de quarta ordem e quando solicitado pela função equivalente no programa principal, calcula os pontos da solução e grava em um arquivo de saída.
4. Saída: O objetivo de gravar os dados em um arquivo de saída não é para que sejam analisados neste arquivo, mas para que sejam acessados por outro programa que pode representar estes dados em formato de gráficos, por exemplo. Visando utilizar o programa Gnuplot para esta análise, devemos gerar os dados no arquivo de saída com uma formatação adequada a leitura do programa. O formato escolhido foi o de tabela, havendo uma coluna para cada grau de liberdade. Vale lembrar que essa formatação é definida no próprio programa principal.
5. Plot: É responsável por ler os dados no arquivo de saída e gerar os gráficos desejados.

**Modelagem matemática dos problemas:**

Para certificar que o programa é eficaz para uma quantidade genérica de graus de liberdade, serão propostos diferentes problemas a serem resolvidos, com diferentes graus de liberdade e condições iniciais, utilizando um único módulo, alterando apenas as funções no programa principal e os dados de entrada.

Problema 1: Sistema massa-mola-amortecedor simples.

**Figura 3** - Sistema massa-mola.

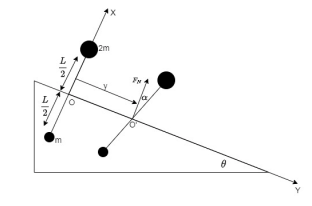


Modelo matemático:



Problema 2: Haltere descendo escorregando sobre um plano inclinado (eixo y) enquanto uma força excita um movimento de rotação em torno do ponto fixado no eixo y.

**Figura 4** - Haltere transladando e rotacionando.



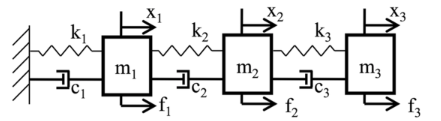
Modelo matemático:



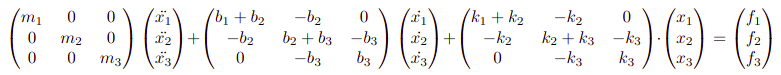


Problema 3: Massa mola em série com 3 massas e 6 graus de liberdade considerando posição e velocidade de cada massa, idêntico ao modelo proposto em sala de aula.

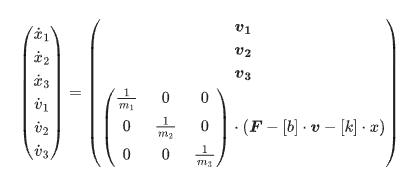
**Figura 5** - Sistema massa-mola em série com 3 graus de liberdade.



Modelo matemático:

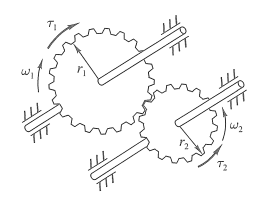


A redução de ordem fica da seguinte forma:



Problema 4: Par de engrenagens com 2 graus de liberdade adaptado de Close (2001).

**Figura 6** - Par de engrenagens.



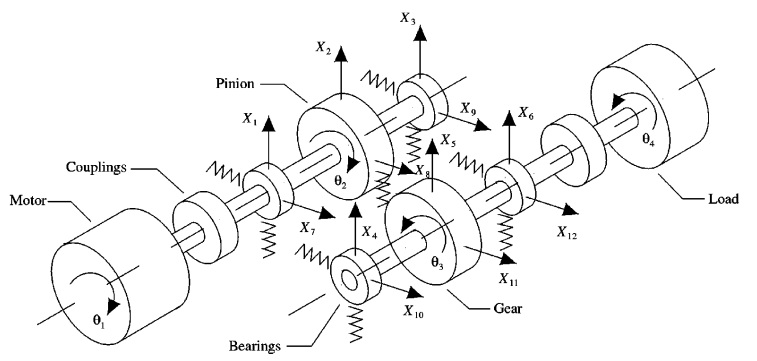
Modelo matemático:

|  | (2) |
| --- | --- |

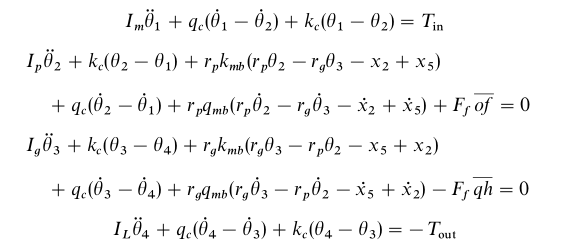
|  | (3) |
| --- | --- |

Problema 5: Par de engrenagens com 8 graus de liberdade adaptado de Howard (2001).

**Figura 7** - Modelo de par de engrenagens considerando os eixos.



Modelo matemático:



onde , “ e conforme apresentado por Rao (2003)

Na implementação do modelo foram desconsiderados os deslocamentos de translação, sendo assim os valores de x e de y são considerados zero. Tal hipótese é equivalente a dizer que as deflexões nos eixos são iguais a zero, e como os eixos são de aço e tem comprimento de apenas 100 mm estando submetido a um torque de apenas 4 Nm, essa hipótese pode ser adotada com segurança, já que o interesse é obter as posições angulares, e para isso, estão sendo consideradas as distorções dos eixos.

**Apresentação e Discussão de Resultados**

**Problema 1:**

Graus de liberdade: 2

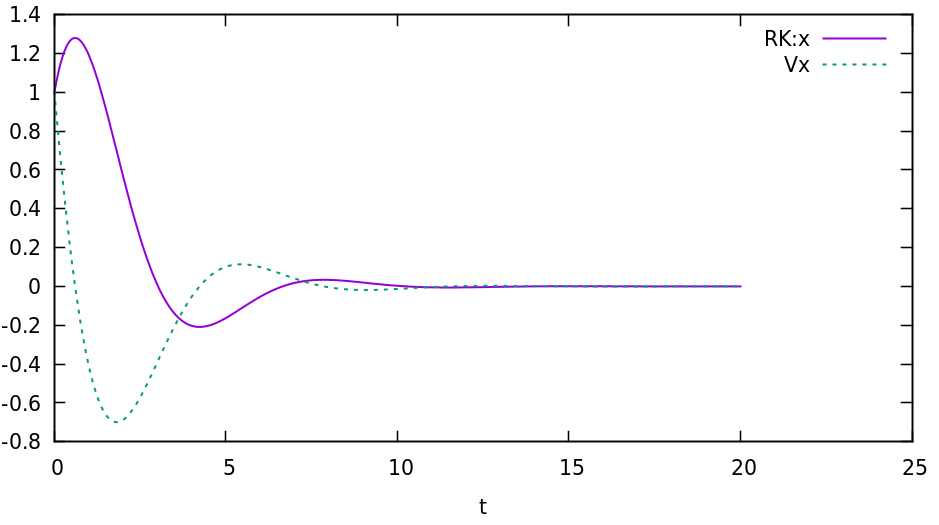
Intervalo de tempo: [0 - 20] s

Número de pontos: 1000

Condições iniciais:

Parâmetros utilizados: , , e

**Figura 8** - Velocidade e posição do um sistema massa-mola-amortecedor.



**Problema 2:**

Graus de liberdade: 4

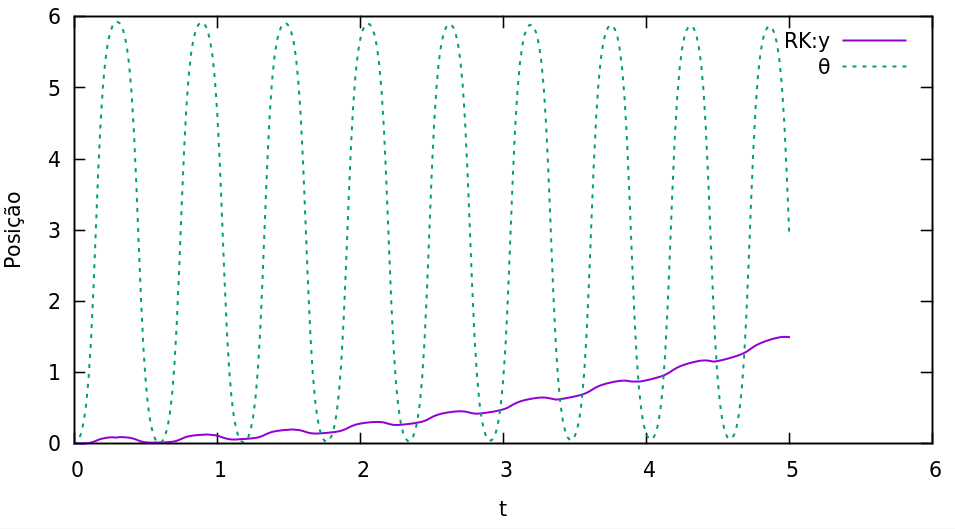
Intervalo de tempo: [0 - 5] s

Número de pontos: 300

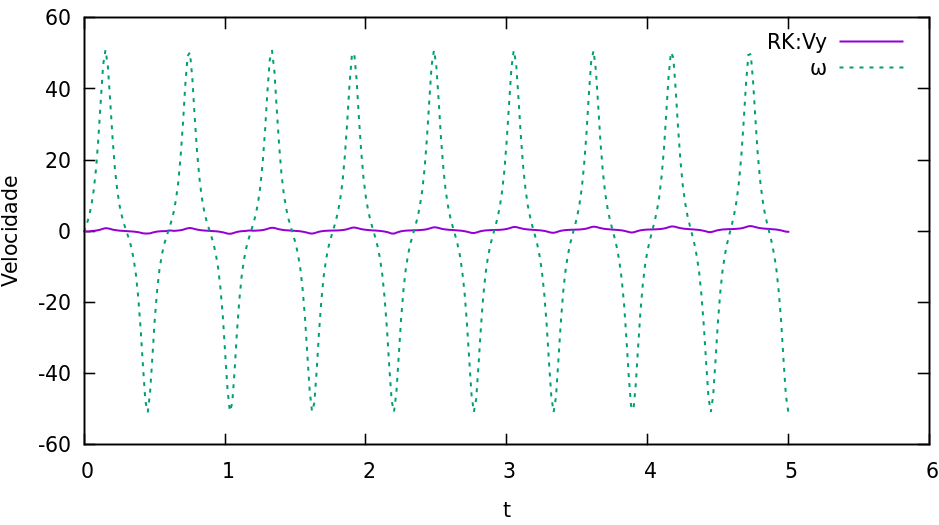
Condições iniciais: , , ,

Parâmetros utilizados: , , , e

**Figura 9** - Posição do haltere.



**Figura 10** - Velocidade do haltere.



**Problema 3:**

Graus de liberdade: 6

Intervalo de tempo: [0 - 5] s

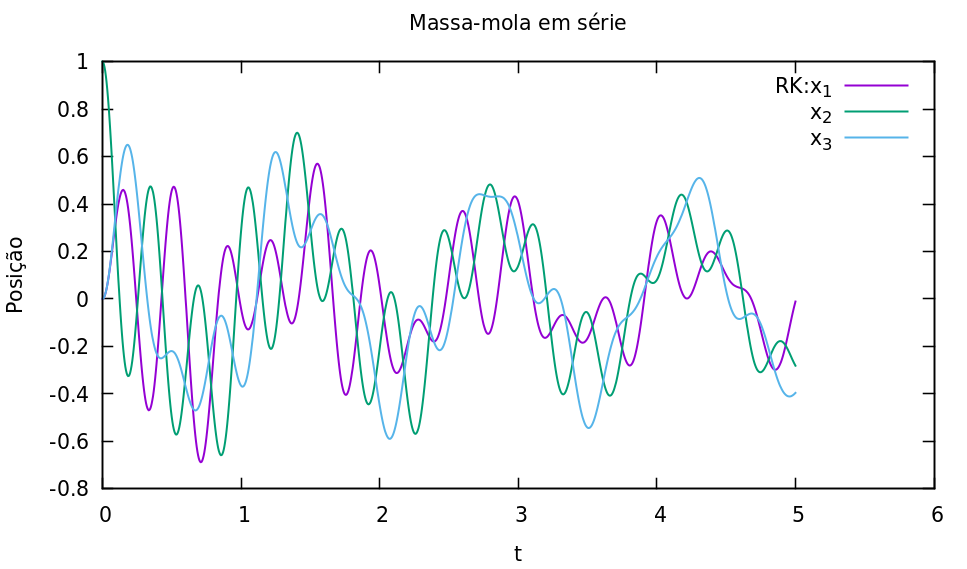
Número de pontos: 1000

Condições iniciais: , , , , e

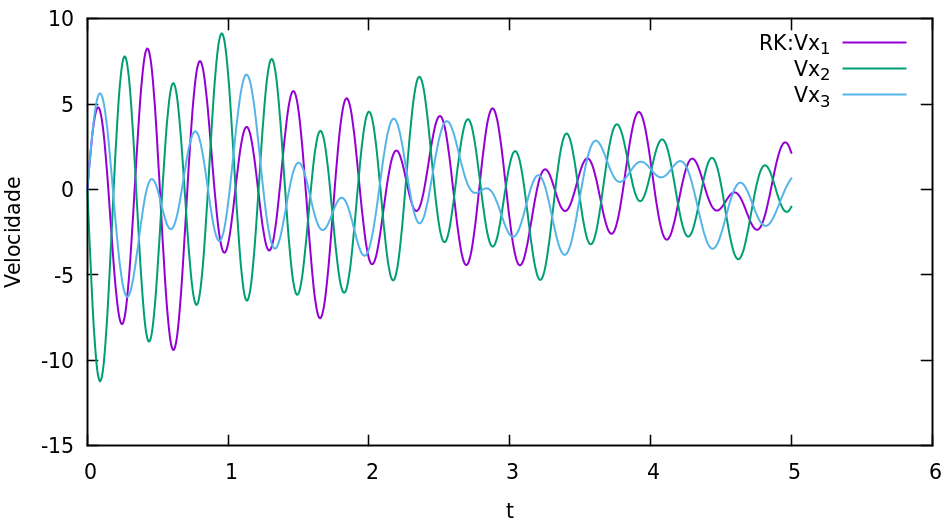
Parâmetros utilizados:

, , e

**Figura 11** - Posição das massas do sistema massa-mola.



**Figura 12** - Velocidades das massas do sistema massa-mola.



**Problema 4:**

Graus de liberdade: 4

Intervalo de tempo: [0 - 0,005] s

Número de pontos: 1000

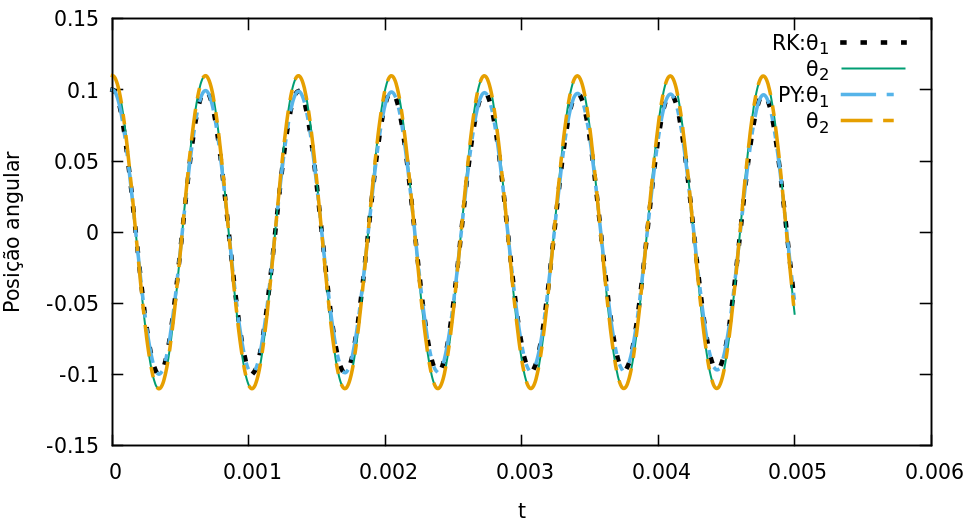
Condições iniciais: e e

Parâmetros utilizados:

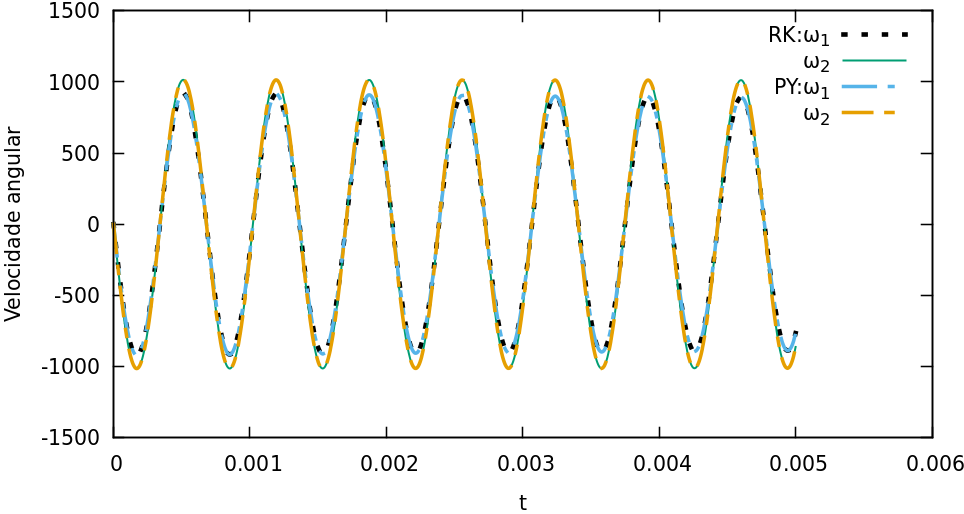
, , , , e N

, , , , e

**Figura 13** - Posição angular das engrenagens.



**Figura 14** - Velocidade angular das engrenagens



**Problema 5:**

Graus de liberdade: 8

Intervalo de tempo: [0 - 0,1] s

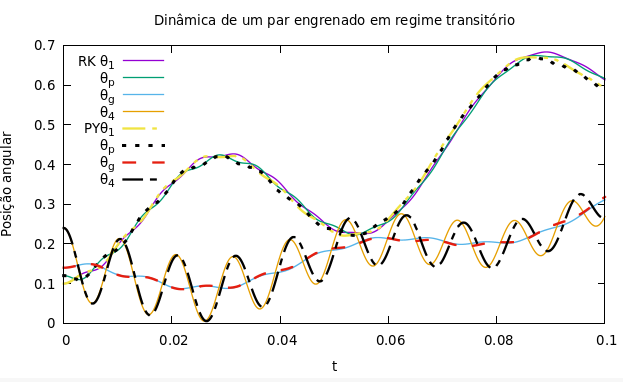
Número de pontos: 10000

Condições iniciais: , , e

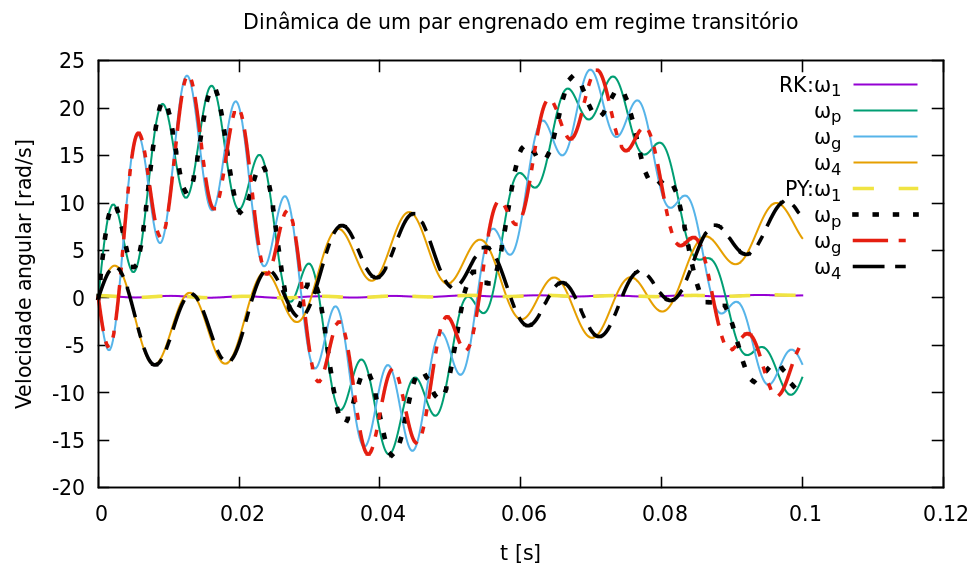
, /s , e

Parâmetros utilizados: , , , , , , , e N, , , e , e .

**Figura 15** - Posição angular das engrenagens e dos eixos.



**Figura 16** - Velocidade angular das engrenagens e dos eixos.



Para os problemas 4 e 5 as soluções obtidas no programa de Runge Kutta de 4ª ordem desenvolvido, chamado de “RK” foram confrontadas com a solução obtida a partir da biblioteca de solução de EDO’s do Python, chamada “odeint”, ilustrado no gráfico como “PY”. Os dados obtidos através do python são salvos em um arquivo de texto automaticamente na mesma pasta em que se encontra o programa em Fortran, facilitando o uso da ferramenta Gnuplot que permite plotar em um único gráfico dados obtidos em programas diferentes.

Os gráficos comparativos das Figuras 15 e 16 permitem observar que a medida em que se aumenta o tempo é acrescentado um erro no deslocamento e principalmente na velocidade angular, sendo mais expressivo na engrenagem movida e no eixo de saída.

**Conclusões**

As soluções dos problemas 1 e 3, apesar dos parâmetros utilizados não serem parâmetros de uma aplicação real, permitiram observar o gráfico em uma escala maior de tempo, pois o regime transitório é muito maior que nos casos dos pares de engrenagem. Entretanto, modelos de sistemas dinâmicos em pares engrenados geralmente são observados em um intervalo muito curto de tempo, entrando rapidamente em regime permanente. Esta é uma característica particular do sistema de engrenagens que requer uma análise restrita a um intervalo muito curto de tempo, o que leva a uma conclusão que o engrenamento quando fabricado de materiais rígidos como aço e suas ligas, têm uma influência muito pequena na resposta de um sistema mecânico como um todo. Levando esse aspecto em consideração, podemos dizer que o programa desenvolvido se mostrou eficaz ao resolver diversos problemas de 2 a 8 graus de liberdade, permitindo inclusive, entender os fenômenos da dinâmica em pares engrenados.

# Referências

CLOSE, Charles M.; FREDERICK, Dean K.; NEWELL, Jonathan C. *Modeling and analysis of dynamic systems*. John Wiley & Sons, 2001.

HOWARD, Ian; JIA, Shengxiang; WANG, Jiande. *The dynamic modelling of a spur gear in mesh including friction and a crack*. Mechanical systems and signal processing, v. 15, n. 5, p. 831-853, 2001.

RAO, S. S., *Mechanical Vibrations*. 4th Edition, Pearson, 2003.

[RUGGIERO, Marcia A. Gomes; LOPES, Vera Lucia da Rocha. *Cálculo numérico: aspectos teóricos e computacionais*. São Paulo: Pearson Makron Books, 2008.](https://www.zotero.org/google-docs/?0BCUCt)

CÓDIGOS-FONTE

# programa\_pvi.f90

!gfortran modulo\_rk4.f90 programa\_pvi.f90 ; ./a.out problema5 ; gnuplot plot\_problema5a.plt ; gnuplot plot\_problema5b.plt

!gfortran modulo\_rk4.f90 programa\_pvi.f90 ; ./a.out problema4 ; gnuplot plot\_problema4a.plt ; gnuplot plot\_problema4b.plt

!gfortran modulo\_rk4.f90 programa\_pvi.f90 ; ./a.out problema3 ; gnuplot plot\_problema3a.plt ; gnuplot plot\_problema3b.plt

!gfortran modulo\_rk4.f90 programa\_pvi.f90 ; ./a.out problema2 ; gnuplot plot\_problema2a.plt ; gnuplot plot\_problema2b.plt

!gfortran modulo\_rk4.f90 programa\_pvi.f90 ; ./a.out problema1 ; gnuplot plot\_problema1a.plt

program pvi

use rk\_4

implicit none

real, dimension (:,:), allocatable::Y

real, dimension (:), allocatable::tspan,y0

character(20)::argument

real::h

integer:: n,gl

!Aqui só alterar o nome da function para o nome do modelo desejado:

! Ex: problema1, problema2, problema3, problema4, problema5

interface

function problema5(t,y) result(dydt)

real, dimension(size(y))::y,dydt

real::t

end function

end interface

call get\_command\_argument(1, argument)

open(1,file=trim(argument)//".txt",status='old')

open(2,file="Resultado\_Runge\_Kutta.txt",status='unknown')

read(1,\*)

read(1,\*)gl

read(1,\*)

read(1,\*)n

allocate(Y(n+1,gl),tspan(2),y0(gl))

read(1,\*)

read(1,\*)tspan

read(1,\*)

read(1,\*)y0

h=(tspan(2)-tspan(1))/n

! Aqui só alterar o nome da function para o nome do modelo desejado:

! Ex: problema1, problema2, problema3, problema4, problema5

Y=rk4(problema5,tspan,y0,h,n)

end program pvi

! MODELO PAR DE ENGRENAGENS CONSIDERANDO EIXOS

function problema5(t,y) result(dydt)

real::rp,rg,r1,r4,E1,E2,nu1,nu2,l1,l2,l3,l4,T\_in,T\_out,qc,qmb, &

&Ff,m1,mp,mg,m4,G1,G2,G3,G4,Im,Ip,Ig,IL,jp1,jp2,jp3,jp4, &

&k1,k2,k3,k4,kc,kmb

real, dimension (8)::y,dydt

! ---DADOS---

rp=0.030/2 ! Raio da engrenagem motriz

rg=0.060/2 ! Raio da engrenagem movida

r1=0.012/2 ! Raio do eixo de entrada

r4=r1 ! Raio do eixo de saída

E1=209e9 ! Módulo de elasticidade do aço 1020

E2=69e9 ! Módulo de elasticidade do alumínio

nu1=0.33 ! Coeficiente de poisson para o alumínio

nu2=0.29 ! Coeficiente de poisson para o aço 1020

l1=0.1 ! Comprimento do eixo

l2=0.006 ! Espessura da engrenagem

l3=0.006 ! Espessura da engrenagem

l4=l1 ! Comprimento do eixo

qc = 0.06 ! Coeficiente de atrito nos mancais dos eixos

qmb = 0.06! Coeficiente de atrito do engrenamento

Ff = 88 ! Força de contato nos dentes das engrenagens (Norma AGMA)

T\_in = 4 ! Torque do motor Cruiser

T\_out = 3.92 ! Torque no eixo de saída

pi = 3.14159 ! Valor de pi

! ---MODELO---

m1=(pi\*r1\*\*2)\*l1\*7860 ! Massa do eixo de entrada

mp=(pi\*rp\*\*2)\*l2\*2700 ! Massa da engrenagem motriz

mg=(pi\*rg\*\*2)\*l3\*2700 ! Massa da engrenagem movida

m4=m1 !Massa do eixo de saída

G1=E1/(2\*(1+nu1)) ! Módulo de elasticidade transversal do eixo de entrada

G4=G1 ! Módulo de elasticidade transversal da engrenagem motriz

G2=E2/(2\*(1+nu2)) ! Módulo de elasticidade transversal da engrenagem movida

G3=G2 ! Módulo de elasticidade transversal do eixo de saída

Im = (1000\*m1\*pi\*r1\*\*2)/2 ! Momento de inércia do eixo de entrada

Ip = (1000\*mp\*pi\*rp\*\*2)/2 ! Momento de inércia da engrenagem motriz

Ig = (1000\*mg\*pi\*rg\*\*2)/2 ! Momento de inércia da engrenagem movida

IL = (1000\*m4\*pi\*r4\*\*2)/2 ! Momento de inércia do eixo de saída

Jp1=(pi\*r1\*\*4)/2 ! Momento de inércia polar do eixo de entrada

Jp2=(pi\*rp\*\*4)/2 ! Momento de inércia polar da engrenagem motriz

Jp3=(pi\*rg\*\*4)/2 ! Momento de inércia polar da engrenagem movida

Jp4=(pi\*r4\*\*4)/2 ! Momento de inércia polar do eixo de saída

k1 = G1\*Jp1/l1 ! Rigidez do eixo de entrada

k2 = G2\*Jp2/l2 ! Rigidez da engrenagem motriz

k3 = G3\*Jp3/l3 ! Rigidez da engrenagem movida

k4 = G4\*Jp4/l4 ! Rigidez do eixo de saída

kc=k1 ! =k4 ! Rigidez dos eixos (Nomenclatura do modelo)

kmb=1/(1/k2+1/k3) ! Rigidez do Engrenamento segundo Rao (2001) pag13

! ---REDUÇÃO DE ORDEM---

dydt(1)=y(5)

dydt(2)=y(6)

dydt(3)=y(7)

dydt(4)=y(8)

dydt(5)=(T\_in - kc\*y(1) + kc\*y(2) - qc\*y(5) + qc\*y(6))/Im

dydt(6)=(-Ff\*rp+kc\*y(1)-kc\*y(2)+kmb\*rg\*rp\*y(3)-kmb\*rp\*\*2\*y(2)+qc\*y(5)-qc\*y(6)+qmb\*rg\*rp\*y(7)-qmb\*rp\*\*2\*y(6))/Ip

dydt(7)=(Ff\*rg-kc\*y(3)+kc\*y(4)-kmb\*rg\*\*2\*y(3)+kmb\*rg\*rp\*y(2)-qc\*y(7)+qc\*y(8)-qmb\*rg\*\*2\*y(7)+qmb\*rg\*rp\*y(6))/Ig

dydt(8)=(-T\_out + kc\*y(3) - kc\*y(4) + qc\*y(7) - qc\*y(8))/IL

end function problema5

!PROBLEMA PAR DE ENGRENAGENS DESCONSIDERANDO OS EIXOS

function problema4(t,y) result(dydt)

real, dimension (2)::b,Tq,m,k,v,x,a,I

real, dimension (4)::y,dydt

! ---DADOS---

b=(/0.06, 0.06/) ! Amortecimento

Tq=(/4.0, 3.92 /) ! Torque na entrada e saída

m=(/0.01145, 0.0458/) !massa das engrenagens

k=(/343796.2355, 5500739.7684/) ! Rigidez das engrenagens

I=(/0.004047, 0.0647/) !Momento de inercia das engrenagens

! ---REDUÇÃO DE ORDEM---

dydt(1)=y(3)

dydt(2)=y(4)

dydt(3)=(Tq(1)-b(1)\*y(3)-k(1)\*y(1))/I(1)

dydt(4)=(-Tq(2)-b(2)\*y(4)-k(2)\*y(2))/I(2)

end function problema4

! PROBLEMA MASSA-MOLA SIMPLES

function problema1(t,y) result(dydt)

real::t,x,b,v,k,m,f

real, dimension(2)::y,dydt ! grau de liberdade é propriedade de cada função

! ---DADOS---

b=1 ! Amortecimento

m=1 ! Massa

k=1 ! Rigidez

f=0 ! Força f(t)

! ---REDUÇÃO DE ORDEM---

x=y(1)

v=y(2)

dydt(1)=v

dydt(2)=(f-b\*v-k\*x)/m

end function problema1

!! PROBLEMA DO HALTERE TRANSLAÇÃO E ROTAÇÃO

function problema2(t,y) result(dydt)

! MODELO MATEMÁTICO

real::m,mu,alpha,alpha1,g,theta,L

real, dimension(4)::y,dydt ! grau de liberdade é propriedade de cada função

! ---DADOS---

m=0.05 ! Massa das partículas

mu=0.1 ! !oeficiente de atrito

g=9.81 ! Aceleração gravidade

theta=10\*3.141592/180 !inclinação da rampa fixa

L=0.01 ! Comprimento da haste (corpo do haltere)

! ---REDUÇÃO DE ORDEM---

alpha=y(3)

alpha1=y(4)

dydt(1)=y(2)

dydt(2)=-L\*alpha1\*\*2\*mu\*cos(alpha)/6 + L\*alpha1\*\*2\*sin(alpha)/6 - &

& g\*mu\*sin(alpha)\*sin(alpha + theta)/3 + g\*mu\*cos(theta) + g\*sin(alpha) - g\*sin(alpha + theta)\*cos(alpha)/9

dydt(3)=y(4)

dydt(4)=2\*g\*sin(alpha + theta)/(3\*L)

end function problema2

!PROBLEMA MASSA-MOLA EM SÉRIE

function problema3(t,y) result(dydt)

real, dimension (3)::y0,b,f,m,k,v,x,z ! INSERIR A DIMENSÃO

real, dimension (size(y0),size(y0))::Mat\_k,Mat\_b

real, dimension (2\*size(y0))::dydt,y

real::h

integer:: n,gdl,i,j

! ---DADOS---

n=size(y0)!número de massas

gdl=2 !Graus de liberdade para cada massa ex: posiçaõ e velocidade

do i=1,n

x(i)=y(i) ! Posição inicial

v(i)=y(n+i) ! Velocidade inicial

end do

! ---DEFININDO MATRIZES DE PARÂMETROS---

b=(/0.2, 0.2, 0.2 /)

f=(/0.0, 1.0, 0.0 /)

m=(/1.0, 1.0, 1.0 /)

k=(/100.0, 100.0, 100.0/)

!Primeira linha

i=1

Mat\_k(i,i)=k(i)+k(i+1)

Mat\_b(i,i)=b(i)+b(i+1)

Mat\_k(i,i+1)=-k(i+1)

Mat\_b(i,i+1)=-b(i+1)

! Linhas intermediárias

do i=2,n-1

Mat\_k(i,i-1)=-k(i)

Mat\_b(i,i-1)=-b(i)

Mat\_k(i,i)=k(i)+k(i+1)

Mat\_b(i,i)=b(i)+b(i+1)

Mat\_k(i,i+1)=-k(i+1)

Mat\_b(i,i+1)=-b(i+1)

end do

! Última linha

i=n

Mat\_k(i,i-1)=-k(i)

Mat\_b(i,i-1)=-b(i)

Mat\_k(i,i)=k(i)

Mat\_b(i,i)=b(i)

! ---MODELO---

z=f-matmul(Mat\_b,v)-matmul(Mat\_k,x)/m

! ---REDUÇÃO DE ORDEM---

do i=1,n

dydt(i)=v(i)

dydt(n+i)=z(i)

end do

do i=1,gdl\*n

end do

end function problema3

# modulo\_rk4.f90

module rk\_4

implicit none

contains

function rk4(arg\_f,tspan,y0,h,len) result(val)

real, dimension(:)::tspan

real,dimension(:)::y0

real, dimension(len+1)::t

real, dimension(len+1,size(y0))::y,val,k1,k2,k3,k4,k

real::h

integer::i,len

interface

function arg\_f(arg\_x,arg\_y) result(val)

real, dimension(size(y0))::arg\_y,val

real::arg\_x

end function

end interface

do i=1,size(y0)

y(1,i)=y0(i)

end do

t(1)=tspan(1)

do i=1,len

k1(i,:)=arg\_f(t(i),y(i,:))

k2(i,:)=arg\_f(t(i)+h/2,y(i,:)+k1(i,:)\*h/2)

k3(i,:)=arg\_f(t(i)+h/2,y(i,:)+k2(i,:)\*h/2)

k4(i,:)=arg\_f(t(i)+h,y(i,:)+k3(i,:)\*h)

k(i,:)=(k1(i,:)+2\*k2(i,:)+2\*k3(i,:)+k4(i,:))/6

t(i+1)=t(i)+h

y(i+1,:)=y(i,:)+k(i,:)\*h

end do

do i=1,len+1

write(2,\*)t(i),y(i,:)

end do

val=y

end function rk4

end module rk\_4