INTRODUÇÃO À COMPUTAÇÃO CIENTÍFICA

IVº Trabalho: Interpolação e Aproximação

Tobias J. D. E. Rosa

2022

1º Período

# 

# Resumo

Este trabalho apresenta brevemente os métodos da spline cúbica para interpolação e dos mínimos quadrados para aproximação e mostra as etapas da criação de um programa que resolve os métodos. Por fim, os códigos fontes em linguagem fortran são disponibilizados.

# Solução do Problema

Ao interpolar utilizando polinômios de grau mais elevado, deparamos com o indesejado fenômeno de Runge. Como forma de contornar esse problema, são apresentadas as funções spline que se resumem em interpolar a função de aproximação a partir de um número menor de pontos por funções porém aumentando-se o número de funções. As funções spline podem ser utilizadas desde que a função de aproximação seja contínua e tenha derivadas contínuas até certa ordem. Entretanto, ao usar spline linear, as derivadas de primeira ordem são descontínuas nos nós, enquanto que a spline cúbica possui derivada contínua até a segunda ordem, o que significa que as curvas serão suaves nos nós, sendo essa uma vantagem da spline cúbica sobre a spline linear. Embora a spline cúbica seja ideal para interpolação, não pode ser usada nos casos de extrapolação, e ainda não é recomendado para os casos em que haja erros não previsíveis como nos casos de resultados de pesquisa ou de experimento físico. Para isso, sugere-se o uso do método dos quadrados mínimos (RUGGIERO, 2008).

**Spline cúbica interpolante:** É um método que interpola de forma em que são definidas funções polinomiais cúbicas para intervalos. Esse método utiliza uma técnica de trabalhar com uma matriz tridiagonal em que a equação para cada linha é obtida da seguinte forma:

A solução do sistema linear da matriz tridiagonal é o vetor .

A função polinomial cúbica respectiva a cada intervalo é da seguinte forma:

Em que os coeficientes ,, , e são obtidos através das seguintes equações:

, , e sendo.

A estratégia particular deste trabalho foi montar uma matriz e um vetor separadamente, a fim de resolver o sistema em que é o número de linhas:

O método de resolução de sistema utilizado foi o de Gauss Seidel, e como entrada, foi utilizada a matriz que é a própria matriz porém, agora o vetor foi inserido como uma coluna adicional à direita.

Uma vez encontrado o vetor solução , foi utilizado um método iterativo para gerar uma função polinomial para cada intervalo . Finalmente, para interpolar basta aplicar um ponto x na função do respectivo intervalo. Os intervalos são definidos a partir dos dados originais do problema, apresentados na Tab. 1.

**Tabela 1** - Dados originais do problema

| **x** | **y** |
| --- | --- |
| 1,59 | 92,75 |
| 4,46 | 79,16 |
| 7,32 | 76,56 |
| 10,29 | 67,39 |
| 13,31 | 65,92 |
| 16,28 | 61,00 |
| 19,33 | 50,76 |
| 22,40 | 49,86 |
| 25,29 | 43,83 |
| 28,17 | 40,74 |
| 31,21 | 38,44 |
| 34,32 | 34,87 |
| 37,22 | 30,39 |
| 40,20 | 28,70 |
| 43,25 | 25,88 |
| 46,25 | 22,72 |
| 49,20 | 21,48 |

Como o objetivo deste trabalho não é realizar a interpolação em um ponto específico, mas sim interpolar uma quantidade de pontos em um intervalo especificado, serão definidas três variáveis: número de pontos, limite inferior e limite superior, possibilitando ao usuário escolher a quantidade de pontos e restringir o intervalo desejado.

Retomando as ideias anteriores, temos funções para intervalos que não necessariamente estão espaçados e que irão receber uma quantidade de pontos que também não estão distribuídos conforme o tamanho dos intervalos.

Para resolver esse problema, foi criada uma rotina do tipo while que compara o ponto a ser calculado com cada intervalo e encontra a sua posição, os quatro coeficientes do polinômio de terceiro grau e a função . Desta forma, é possível interpolar um número infinito de pontos dentro do intervalo escolhido. Para o problema proposto foram utilizados 50 pontos dentro do intervalo de 2 a 40, como apresentado na Tab. 2.

**Método dos quadrados mínimos:** A estratégia utilizada no método dos mínimos quadrados se resumiu basicamente na montagem da matriz A. Observou se que todos os termos da matriz que se repetiam tinham a soma dos coeficientes i+j iguais, a partir disso e do uso da função “sum(vetor)” que possibilita somar os termos de um ou mais vetores ou fazer o produto interno de ambos, foi possível encontrar coeficientes para polinômios de grau maior, inclusive.

# Apresentação e Discussão de Resultados

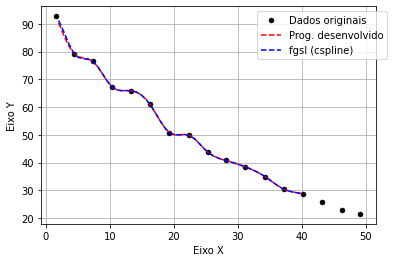
Os resultados relacionados à aproximação pelo método da spline cúbica podem ser observados na Tab. 2.

**Tabela 2** - Conjunto de pontos gerados por interpolação

| **Dados:**  **50 pontos [2 : 40]** | **Programa desenvolvido** | **FGSL** | **Erro relativo** |
| --- | --- | --- | --- |
| **X** | **Y1** | **Y2** | **|Y2-Y1|/Y1** |
| 2.00000 | 90.3142166 | 91.19991 | 0,009806799342818 |
| 2.76000 | 86.0090485 | 87.37768 | 0,015912645516594 |
| 3.52000 | 82.3097916 | 83.21339 | 0,010978018318783 |
| 4.28000 | 79.6100388 | 79.75919 | 0,001873522513595 |
| 5.04000 | 78.2181549 | 77.92108 | 0,00379803001464 |
| 5.80000 | 77.7411804 | 77.38708 | 0,004554862663238 |
| 6.56000 | 77.4368210 | 77.24044 | 0,002536015779883 |
| 7.32000 | 76.5599976 | 76.56000 | 3,13479633602408E-08 |
| 8.08000 | 74.5985336 | 74.69278 | 0,001263381402446 |
| 8.84000 | 71.9629974 | 72.05846 | 0,001326551192266 |
| 9.60000 | 69.2968674 | 69.34487 | 0,000692709523548 |
| 10.36000 | 67.2433701 | 67.23971 | 5,44306449031578E-05 |
| 11.12000 | 66.2148132 | 66.18854 | 0,000396787346068 |
| 11.88000 | 65.9260941 | 65.90071 | 0,000385038736885 |
| 12.64000 | 65.9585495 | 65.94616 | 0,000187837666139 |
| 13.40000 | 65.8936768 | 65.89488 | 1,82597186625072E-05 |
| 14.16000 | 65.3966522 | 65.40348 | 0,00010440595612 |
| 14.92000 | 64.3658829 | 64.37230 | 9,96972264011789E-05 |
| 15.68000 | 62.7413597 | 62.74431 | 4,70232078824705E-05 |
| 16.44000 | 60.4638443 | 60.46331 | 8,83668589359296E-06 |
| 17.20000 | 57.6349297 | 57.63304 | 3,27874087785961E-05 |
| 17.96000 | 54.7055511 | 54.70377 | 3,25579390790985E-05 |
| 18.72000 | 52.1719170 | 52.17105 | 1,66181357683197E-05 |
| 19.48000 | 50.5289726 | 50.52909 | 2,32341949484829E-06 |
| 20.24000 | 49.9738464 | 49.97430 | 9,07674779262468E-06 |
| 21.00000 | 50.0333252 | 50.03382 | 9,88940866953339E-06 |
| 21.76000 | 50.1431313 | 50.14343 | 5,95694748727973E-06 |
| 22.52000 | 49.7395477 | 49.73955 | 4,62408707921627E-08 |
| 23.28000 | 48.5074463 | 48.50743 | 3,36030882719069E-07 |
| 24.04000 | 46.7577057 | 46.75766 | 9,77379007737491E-07 |
| 24.80000 | 44.8975372 | 44.89748 | 1,27401197410299E-06 |
| 25.56000 | 43.3300743 | 43.32999 | 1,9455309357151E-06 |
| 26.32000 | 42.2418365 | 42.24163 | 4,88851851877543E-06 |
| 27.08000 | 41.5023384 | 41.50213 | 5,02140380596533E-06 |
| 27.84000 | 40.9561958 | 40.95611 | 2,09492113036538E-06 |
| 28.60000 | 40.4524536 | 40.45266 | 5,10228630490959E-06 |
| 29.36000 | 39.9177055 | 39.91823 | 1,31395327821255E-05 |
| 30.12000 | 39.3437958 | 39.34442 | 1,58652714438191E-05 |
| 30.88000 | 38.7245560 | 38.72487 | 8,10855003742416E-06 |
| 31.64000 | 38.0529633 | 38.05239 | 1,50658437683149E-05 |
| 32.40000 | 37.3076172 | 37.30594 | 4,49559667939113E-05 |
| 33.16000 | 36.4550056 | 36.45281 | 6,022766870732E-05 |
| 33.92000 | 35.4612198 | 35.45991 | 3,69361236694221E-05 |
| 34.68000 | 34.2956467 | 34.29736 | 4,99567777504162E-05 |
| 35.44000 | 33.0132103 | 33.01887 | 0,000171437432124 |
| 36.20000 | 31.7604008 | 31.76771 | 0,000230135634812 |
| 36.96000 | 30.6882515 | 30.69153 | 0,000106832414352 |
| 37.72000 | 29.9336720 | 29.92464 | 0,000301733779939 |
| 38.48000 | 29.4645901 | 29.44118 | 0,000794516398177 |
| 39.24000 | 29.1372948 | 29.10952 | 0,000953238802389 |
| 40.00000 | 28.8060818 | 28.79616 | 0,000344434209029 |

Os pontos apresentados na Tab. 2, foram utilizados para comparar os resultados da interpolação feita pelo programa desenvolvido e pela biblioteca FGSL. Para isso foi calculado um erro relativo entre as respostas, que apesar de apresentar um aumento do erro à medida que Y aumenta, foi considerado pequeno, na casa de . Uma forma de comparar resultados é sobrepor os resultados graficamente, para isso, os resultados obtidos foram inseridos em um programa em linguagem python e plotados com auxílio da biblioteca matplotlib. Conforme mencionado, é possível observar na Fig. 1 a pequena tendência do erro à medida que os valores de Y aumentam.

**Figura 1** - Conjunto de pontos gerados por interpolação



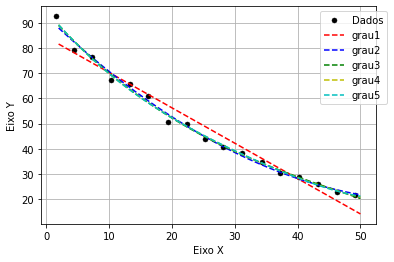
A partir dos mesmos dados de entrada da Tab. 1 foi também testado o programa desenvolvido para aproximação pelo método dos mínimos quadrados. O programa desenvolvido retorna polinômios de até nove graus, e sendo facilmente alterado para solucionar grau maior. Entretanto, não há necessidade de se implementar uma aproximação com polinômio de grau maior, visto que em algumas situações, polinômios de grau menor representam melhor os dados a serem aproximados. Para o problema atual, foram testadas 5 funções polinomiais e comparadas com a tendência dos dados do gráfico, e chegou-se a conclusão que o polinômio de grau 2 pode ser considerado uma boa aproximação para os dados. A extrapolação para x=55 é apresentada na Tab. 3.

**Tabela 3** - Aproximação para x=55 usando diferentes polinômios

| Grau do polinômio | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- |
| f(55) | 6.9961 | 19.8352 | 15.9925 | 18.5507 | 20.5662 |

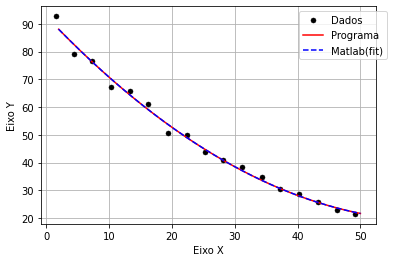
A extrapolação pelo polinômio de grau 3 ficou um pouco fora do previsto a partir da análise visual da curva pela Fig. 1. Os polinômios de grau 2 e grau 3 podem ser boas aproximações, portanto será adotado o polinômio de grau 2.

**Figura 2** - Curvas obtidas pelos diferentes polinômios



Uma vez escolhido o polinômio de grau dois, foi utilizada a ferramenta “fit” do matlab para comparar os resultados, que foram praticamente coincidentes, como apresentado na Fig. 3.

**Figura 3** - curvas de um polinômio de grau 2 por Matlab e pelo programa.



**Conclusões**

De modo geral, foi possível observar o poder das ferramentas de interpolação e aproximação e entender como as ferramentas disponibilizadas pelos softwares consagrados funcionam. Além do mais, os programas desenvolvidos chegaram em resultados próximos aos da biblioteca fgs e da ferramenta do Matlab utilizados.

# Referências

[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

# spline\_cubica.f90

!gfortran spline\_cubica.f90 ; ./a.out entrada\_trabalho\_4

!esse programa usa o método interpola e gera "Resultado\_spline\_cubica.txt"

!Institudo Militar de Engenharia - IME

!Programa utilizado na disciplina de introdução a computação científica

!Aluno: Tobias José Degli Esposte Rosa

program spline

implicit none

real, dimension (:,:), allocatable ::A1,A2

real, dimension (:), allocatable::x,h,vetor\_x,g,y,vetor\_y,a,b,c,d,y\_data

integer:: n\_lin,ctd,i,j

character(20)::argument

real::s,g0,x\_data,n\_pontos,e,x\_ant,lim\_inf,lim\_sup

call get\_command\_argument(1, argument)

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

open(2,file="Resultado\_spline\_cubica.txt",status='unknown')

read(1,\*)

read(1,\*)n\_lin

read(1,\*)

read(1,\*)

read(1,\*)

read(1,\*)lim\_inf,lim\_sup

read(1,\*)

read(1,\*)n\_pontos

read(1,\*)

allocate (A1(n\_lin,n\_lin),A2(n\_lin-2,n\_lin-1),x(n\_lin),y(n\_lin),h(n\_lin-1),vetor\_x(n\_lin),&

& g(n\_lin-2),vetor\_y(n\_lin),a(n\_lin-2),b(n\_lin-2),c(n\_lin-2),d(n\_lin-2),y\_data(n\_lin-2))

read(1,\*)x(:)

read(1,\*)y(:)

close(1)

do i=2,n\_lin

h(i)=x(i)-x(i-1)

end do

do i=2,n\_lin-1

vetor\_y(i-1)=6\*((y(i+1)-y(i))/h(i+1)-(y(i)-y(i-1))/h(i))

!print\*, 'y(',i+1,')=',y(i+1),'y(',i,')=',y(i),'y(',i-1,') = ',y(i-1),'h(',i,') = ',h(i),'vetor\_y(',i-1,') = ',vetor\_y(i-1)

end do

do i=1,n\_lin-2

A1(i,i)=h(i+1)

A1(i,i+1)=2\*(h(i+1)+h(i+2))

A1(i,i+2)=h(i+2)

end do

! MATRIZ INPUT PARA O SISTEMA DE GAUSS JACOBI

do i=1,n\_lin-2

A2(i,n\_lin-1)=vetor\_y(i)

do j=1, n\_lin-2

A2(i,j)=A1(i,j+1)

end do

end do

! ENCONTRA O VETOR SOLUÇÃO DO SISTEMA DA MATRIZ A PELO MÉTODO DE GAUSS SEIDEL

ctd=0 ! CONTA ITERAÇÕES

e=1 !condição para entrar no while

do while (e>1e-5)

do i=1,n\_lin-2

s=0

do j=1,n\_lin-2

if (i/=j) then

s=s+A2(i,j)\*g(j)

end if

end do

x\_ant=g(i)

g(i)=(A2(i,n\_lin-1)-s)/(A2(i,i))

e=((g(i)-x\_ant)\*\*2)\*\*0.5

end do

ctd=ctd+1

end do

!print\*,"vetor g",g,"numero de iterações",ctd,"erro =",e

ctd=0

!AQUI INICIA UM PROCESSO ITERATIVO PARA GERAR UMA QUANTIDADE "n\_pontos" DE PARES ORDENADOS UTILIZANDO SPLINE CÚBICA

do while (ctd<n\_pontos+1)

x\_data=lim\_inf+(lim\_sup-lim\_inf)\*(1/n\_pontos)\*ctd

!DETERMINAÇÃO DOS COEFICIENTES a,b,c e d

do i=1,n\_lin-1

if (i==1) then ! g0 = gn = zero

g0=0

else

g0=g(i-1)

end if

a(i)=(g(i)-g0)/(6\*h(i+1))

b(i)=g(i)/2

c(i)=(y(i+1)-y(i))/h(i+1)+(2\*h(i+1)\*g(i)+g0\*h(i+1))/6

d(i)=y(i+1)

y\_data(i)=a(i)\*(x\_data-x(i+1))\*\*3+b(i)\*(x\_data-x(i+1))\*\*2+c(i)\*(x\_data-x(i+1))+d(i)

end do

s=1 !condição para entrar no while

i=1

do while (s>0)! Encontra o intervalo do x\_data e chama a função respectiva ao intervalo

s=x\_data-x(i)

i=i+1

end do

write(2,\*)x\_data,y\_data(i-2) !,'função',i-1

ctd=ctd+1

end do

close(2)

deallocate (A1,A2,x,y,h,vetor\_x,vetor\_y)

end program spline

# min\_quad.f90

!gfortran min\_quad.f90 ; ./a.out entrada\_trabalho\_4

!esse programa usa o método dos mínimos quadrados e gera "Resultado\_min\_quad.txt"

!Institudo Militar de Engenharia - IME

!Programa utilizado na disciplina de introdução a computação científica

!Aluno: Tobias José Degli Esposte Rosa

program min\_quad

implicit none

real, dimension (:,:), allocatable ::A

real, dimension (:), allocatable::x,y,g

integer:: n\_lin,x\_len,n,i,j,grau

character(20)::argument

real::s,e,x\_anterior

call get\_command\_argument(1, argument)

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

open(2,file="Resultado\_min\_quad.txt",status='unknown')

read(1,\*)

read(1,\*)x\_len

read(1,\*)

read(1,\*)grau

read(1,\*)

read(1,\*)

read(1,\*)

read(1,\*)

read(1,\*)

n\_lin=grau+1

allocate (A(n\_lin,n\_lin+1),x(x\_len),y(x\_len),g(n\_lin))

read(1,\*)x(:)

read(1,\*)y(:)

! GERANDO A MATRIZ A (RESOLVE POLINÔMIO ATÉ GRAU 9)

do i=1,n\_lin

A(i,n\_lin+1)=sum(y\*x\*\*(n\_lin-i))

do j=1, n\_lin

if (i+j==2) then

A(i,j)=sum(x\*\*(2\*grau))

elseif (i+j==3) then

A(i,j)=sum(x\*\*(2\*grau-1))

elseif (i+j==4) then

A(i,j)=sum(x\*\*(2\*grau-2))

elseif (i+j==5) then

A(i,j)=sum(x\*\*(2\*grau-3))

elseif (i+j==6) then

A(i,j)=sum(x\*\*(2\*grau-4))!para resolver um polinomio de grau até 2, bastaria até aqui

elseif (i+j==7) then

A(i,j)=sum(x\*\*(2\*grau-5))

elseif (i+j==8) then

A(i,j)=sum(x\*\*(2\*grau-6))

elseif (i+j==9) then

A(i,j)=sum(x\*\*(2\*grau-7))

elseif (i+j==10) then

A(i,j)=sum(x\*\*(2\*grau-8))

elseif (i+j==11) then

A(i,j)=sum(x\*\*(2\*grau-9))

elseif (i+j==12) then

A(i,j)=sum(x\*\*(2\*grau-10))

elseif (i+j==13) then

A(i,j)=sum(x\*\*(2\*grau-11))

elseif (i+j==14) then

A(i,j)=sum(x\*\*(2\*grau-12))

elseif (i+j==15) then

A(i,j)=sum(x\*\*(2\*grau-13))

elseif (i+j==16) then

A(i,j)=sum(x\*\*(2\*grau-14))

elseif (i+j==17) then

A(i,j)=sum(x\*\*(2\*grau-15))

elseif (i+j==18) then

A(i,j)=sum(x\*\*(2\*grau-16))

elseif (i+j==19) then

A(i,j)=sum(x\*\*(2\*grau-17))

elseif (i+j==20) then

A(i,j)=sum(x\*\*(2\*grau-18))

end if

end do

end do

! ENCONTRA O VETOR SOLUÇÃO DO SISTEMA DA MATRIZ A PELO MÉTODO DE GAUSS SEIDEL

n=0 ! CONTA ITERAÇÕES

e=1!condição para entrar no while

do while (e>1e-10)

do i=1,n\_lin

s=0

do j=1,n\_lin

if (i/=j) then

s=s+A(i,j)\*g(j)

end if

end do

x\_anterior=g(i)

g(i)=(A(i,n\_lin+1)-s)/(A(i,i))

e=((g(i)-x\_anterior)\*\*2)\*\*0.5

end do

n=n+1

end do

do i=1,grau+1

write(2,\*)"alpha",(i),"=",g(i)!,"numero de iterações",n,"erro =",e

end do

deallocate (A,x,y)

end program min\_quad

# interpp.f90

!gfortran -I/usr/local/include/fgsl interpp.f90 -lfgsl -lm ; ./a.out

!esse programa é um template da pasta "examples" da bibli. fgsl\_int

!possui modificações para adaptar ao problema

!Institudo Militar de Engenharia - IME

!Programa utilizado na disciplina de introdução a computação científica

!Aluno: Tobias José Degli Esposte Rosa

program interpp

use fgsl

implicit none

integer(fgsl\_size\_t), parameter :: n = 17

integer(fgsl\_int) :: i, status,nnn

real::nn

real(fgsl\_double) :: xi, yi

real(fgsl\_double) :: x(17) = (/1.59D0, 4.46D0, 7.32D0, 10.29D0, 13.31D0, 16.28D0,19.33D0, 22.40D0, &

&25.29D0, 28.17D0, 31.21D0, 34.32D0, 37.22D0, 40.20D0, 43.25D0, 46.25D0, 49.20D0/), &

y(17) = (/92.75D0, 79.16D0, 76.56D0, 67.39D0, 65.92D0, 61.00D0,50.76D0, &

&49.86D0, 43.83D0, 40.74D0, 38.44D0, 34.87D0, 30.39D0, 28.70D0, 25.88D0, 22.72D0, 21.48D0 /)

! Note: first = last for periodic data

type(fgsl\_interp\_accel) :: acc

type(fgsl\_spline) :: spline

nnn=50

nn=50

spline = fgsl\_spline\_alloc(fgsl\_interp\_cspline\_periodic, n)

write(6, '(''#m=0,S=5'')')

do i=1,n

write(6, '(2(F10.5,1X))') x(i), y(i)

end do

write(6, '(''#m=1,S=0'')')

status = fgsl\_spline\_init(spline, x, y)

do i=0, nnn

xi = 2+38\*(i/nn)

yi = fgsl\_spline\_eval(spline, xi, acc)

write(6, '(2(F10.5,1X))') xi, yi

end do

call fgsl\_spline\_free (spline)

call fgsl\_interp\_accel\_free (acc)

end program interpp