Exercise 1. (Finite elements)
Use the Galerkin finite element method with continuous piecewise linear basis functions to solve the problem

$$-\frac{d}{dx}\left((1+x^2)\frac{du}{dx}\right) = f(x), \quad 0 \le x \le 1 \tag{1}$$

$$u(0) = 0, u(1) = 0 \tag{2}$$

(a) Derive the matrix equation that you will need to solve for this problem.

(b) Write a code to solve this set of equation. You can test your code on a problem where you know the solution by choosing a function $u(x)$ that satisfies the boundary conditions and determining what $f(x)$ must be in order for $u(x)$ to satisfy the differential equation. Try $u(x) = x(1-x)$. Then $f(x) = 2(3x^2 - x + 1)$.

(c) Try several different values for the mesh size h. Based on your results, what would you say is the order of accuracy of the Galerkin method with continuous piecewise linear basis functions?

(d) Now try a non-uniform mesh spacing, say, $x_i = (i/(m+1))^2$, $i = 0, 1, m+1$. Do you see the same order of accuracy, if h is defined as the maximum mesh spacing $\max_i(x_{i+1} - x_i)$.

(e) Suppose the boundary conditions were $u(0) = a$ and $u(1) = b$. Show how you would represent the approximate solution $\hat{u}(x)$ as a linear combination of hat functions and how the matrix equation in part (a) would change.

Solution 1. (a) We begin by writing the weak form of the equation

$$-\int_0^1 \frac{d}{dx}\Big((1+x^2)u'(x)\Big)\varphi(x)dx = \int_0^1 f(x)\varphi(x)dx$$

Integrating the left side by parts

$$-\int_0^1 \frac{d}{dx}\Big((1+x^2)u'(x)\Big)\varphi(x)dx = \int_0^1 (1+x^2)u'(x)\varphi'(x)dx - \Big[(1+x^2)u'(x)\varphi(x)\Big]_0^1.$$

This gives us

$$\int_0^1 (1+x^2)u'(x)\varphi'(x)dx - \Big[(1+x^2)u'(x)\varphi(x)\Big]_0^1 = \int_0^1 f(x)\varphi(x)dx.$$

Writing φ as a sum of continuous piece-wise linear basis functions $\varphi(x) = \sum_{i=1}^{n-1} d_j\varphi_i(x)$ where φ_i is given by equation (5) of the finite element notes, we have that

$$\Big[(1+x^2)u'(x)\varphi_i(x)\Big]_0^1 = 0$$

since $\varphi_i(x) = 0$ for all $i = 1, \dots, n-1$. Therefore, using linearity and writing u in terms of the basis φ_j we write

$$\sum_{j=1}^{n-1} c_j \int_0^1 (1+x^2)\varphi_j'(x)\varphi_i'(x)dx = \int_0^1 f(x)\varphi_i(x)dx,$$

for any $i = 1, \dots, n-1$. We can represent this as

$$\mathbf{Ac} = \mathbf{f},$$

where $\mathbf{c}$ is the vector of the coefficients to the φ_i expansion of u and $\mathbf{f}$ contains has entries equal to the right hand side of the above equation and the entries of A are given by

$$\int_0^1 (1+x^2)\varphi_j'(x)\varphi_i'(x)dx$$

We'll now simplify the entires of A. Starting with the diagonal entries a_{ii},

$$\begin{aligned} a_{ii} &= \int_0^1 (1+x^2)\varphi_i'(x)^2 dx \\ &= \left(\frac{1}{x_i - x_{i-1}}\right)^2 \int_{x_{i-1}}^{x_i} 1 + x^2 dx + \left(\frac{1}{x_{i+1} - x_i}\right)^2 \int_{x_i}^{x_{i+1}} 1 + x^2 dx \\ &= \frac{1}{(x_i - x_{i-1})^2}\left(\frac{x_i^3 - x_{i-1}^3}{3} + x_i - x_{i-1}\right) + \frac{1}{(x_{i+1} - x_i)^2}\left(\frac{x_{i+1}^3 - x_i^3}{3} + x_{i+1} - x_i\right) \end{aligned}$$

Since the φ_i and φ_j only overlap when $j = i \pm 1$ or $i = j$, we know that this matrix should be tridiagonal. Similarly, a look at the equation for a_{ij} shows it is symmetric. Therefore, we can compute the remaining non-zero elements as follows:

$$\begin{aligned} a_{i,i+1} = a_{i+1,i} &= \int_0^1 (1+x)^2\varphi_i'(x)\varphi_{i+1}'(x)dx \\ &= \frac{-1}{(x_{i+1} - x_i)^2}\int_{x_i}^{x_{i+1}} (1+x^2)dx \\ &= \frac{-1}{(x_{i+1} - x_i)^2}\left(\frac{x_{i+1}^3 - x_i^3}{3} + x_{i+1} - x_i\right). \end{aligned}$$

With these equations, we can now solve them using code.
(b) See appendix for plots and code.
(c) I would say the accuracy is order $O(h^2)$. As seen in the code appendix, halving h (doubling M) leads to the error being multiplied by 1/4.
(d) As shown in the code appendix, the order appears to be the same as in part (c).
(e) In order to accommodate for non-zero boundary conditions, we would need to add two basis functions φ_0 and φ_n which are just the same as all other φ_i but one-sided so that they

stay in the desired interval $[0, 1]$. That is,

$$\varphi_0(x) = \frac{x_1 - x}{x_1 - x_0}, x \in [x_0, x_1]$$
$$\varphi_n(x) = \frac{x - x_{n-1}}{x_n - x_{n-1}}, x \in [x_{n-1}, x_n].$$

We would then write that

$$\hat{u}(x) = \sum_{j=0}^{n} c_j \varphi_j(x).$$

Which in our matrix gives extra equations for these conditions

$$a_{0,j} = a_{j,0} = \int_0^1 (1 + x^2)\varphi_j'(x)\varphi_0'(x)dx - \Big[(1 + x^2)\varphi_j'(x)\varphi_0(x)\Big]_0^1 = \int_0^1 f(x)\varphi_0(x)dx$$
$$a_{n,j} = a_{j,n} = \int_0^1 (1 + x^2)\varphi_j'(x)\varphi_n'(x)dx - \Big[(1 + x^2)\varphi_j'(x)\varphi_n(x)\Big]_0^1 = \int_0^1 f(x)\varphi_n(x)dx$$

HW-3-plus-code-Figgins

February 2, 2021

0.1 Exercise 1: Code Appendix

[1]:
```
using LinearAlgebra, Plots
```

[2]:
```
function make_FEM_A(x)
    M = length(x)

    offDiag = zeros(M-2)
    onDiag = zeros(M-2)

    # Loop over all grid points except boundary
    for i in 2:(M-1)
        offDiag[i-1] = -1/(x[i+1] - x[i])^2 * ( (x[i+1]^3 - x[i]^3)/3 + x[i+1]␣
↪- x[i])
    end

    # Loop over all grid points except boundary
    for i in 2:(M-1)
       onDiag[i-1] = 1/(x[i+1] - x[i])^2 * ( (x[i+1]^3 - x[i]^3)/3 + x[i+1] -␣
↪x[i]) + 1/(x[i] - x[i-1])^2 * ( (x[i]^3 - x[i-1]^3)/3 + x[i] - x[i-1])
    end

    A = Tridiagonal(offDiag[1:M-3], onDiag, offDiag[1:(M-3)])

    #return offDiag, onDiag
    return A
end
```

[2]:
```
make_FEM_A (generic function with 1 method)
```

[3]:
```
M = 30
h = 1/M

x = [i*h for i in 0:M]
```

[3]:
```
31-element Array{Float64,1}:
 0.0
 0.03333333333333333
```

```
0.06666666666666667
0.1
0.13333333333333333
0.16666666666666666
0.2
0.23333333333333334
0.26666666666666666
0.3
0.3333333333333333
0.36666666666666664
0.4
⋮
0.6333333333333333
0.6666666666666666
0.7
0.7333333333333333
0.7666666666666666
0.8
0.8333333333333334
0.8666666666666667
0.9
0.9333333333333333
0.9666666666666667
1.0
```

[4]:

```
A = make_FEM_A(x)
```

[4]:

```
29×29 Tridiagonal{Float64,Array{Float64,1}}:
  60.0889  -30.0778                       …
 -30.0778   60.2889  -30.2111
           -30.2111   60.6222  -30.4111
                     -30.4111   61.0889
                               -30.6778
                                         ⋱

                                         …




                                         …
```

```
…  -53.4111
   108.622   -55.2111
   -55.2111  112.289   -57.0778
             -57.0778  116.089
```

[5]:
```
function make_F(f, x)
    M = length(x)
    F = zeros(M-2)

    for i in 2:(M-1)
        # Approximating integral with Trapezoid Rule
        F[i-1] = (f(x[i])*(x[i+1] - x[i]) + f(x[i])*(x[i] - x[i-1]))/2
    end
    return F
end
```

[5]:
```
make_F (generic function with 1 method)
```

[6]:
```
true_f(x) = 2*(3*x^2 - x + 1)
true_u(x) = x*(1 - x)
```

[6]:
```
true_u (generic function with 1 method)
```

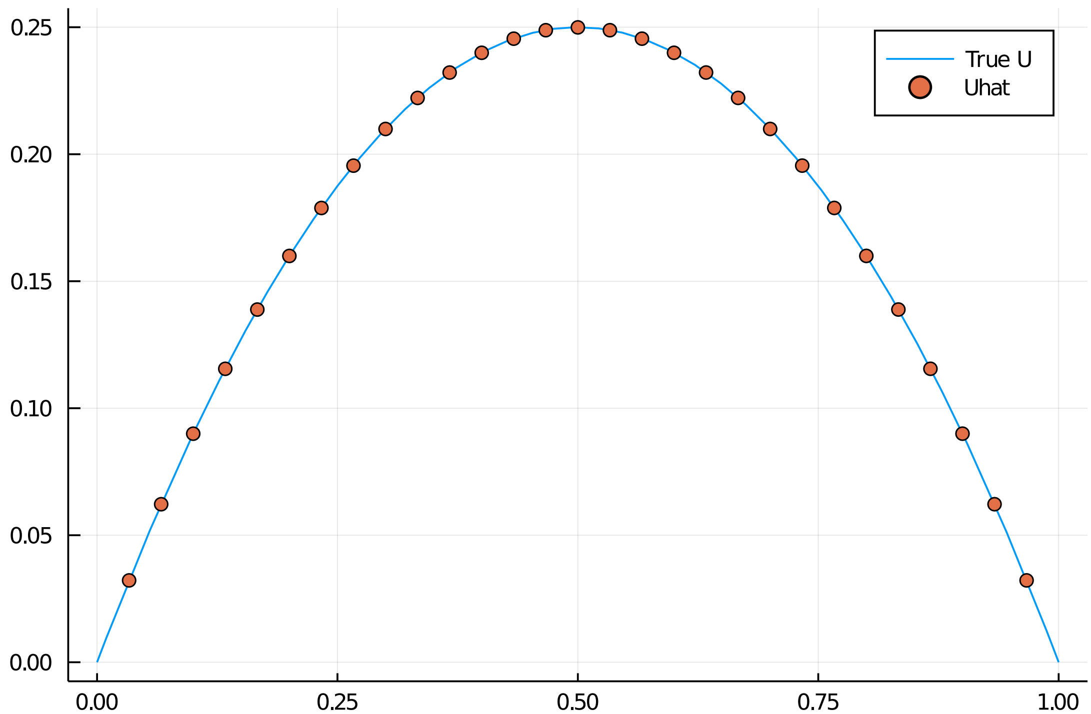
[7]:
```
F = make_F(true_f, x)
C = A \ F
```

[7]:
```
29-element Array{Float64,1}:
 0.032211740809253056
 0.06220210354504637
 0.08997110388553643
 0.11551875164797433
 0.13884505078806578
 0.15994999952536154
 0.17883359058506879
 0.19549581154212434
 0.20993664524997221
 0.2221560703343856
 0.23215406173187558
 0.23993059125263508
 0.2454856281493823
 ⋮
 0.23993723603165953
 0.2321625947453146
 0.22216621669943917
```

```
0.2099480677284199
0.19550811468848658
0.17884632562136
0.15996266987791094
0.13885711820623636
0.11552964280864293
0.08998021737194493
0.06220881707526405
0.03221541857921432
```

[8]:
```
plot(x -> true_u(x), 0, 1,
    label = "True U")
scatter!(x[2:(length(x) - 1)], C,
    label = "Uhat")
```

[8]:



0.2 Part (c) Test Accuracy as function of h

[10]:
```
function get_inf_accuracy(true_u, u_hat, x)
    discret_u = [true_u(xi) for xi in x[2:(length(x)-1)]]

    return maximum(abs.(discret_u .- u_hat))
end
```

[10]: get_inf_accuracy (generic function with 1 method)

[11]:
```
M = 200
x = [i / M for i in 0:M]

A = make_FEM_A(x)
F = make_F(true_f, x)
C = A \ F

get_inf_accuracy(true_u, C, x)
```

[11]: 1.5737179526187361e-6

[12]:
```
### For various M, generate error
M_values = [5, 10, 20, 40, 80, 160, 320]

uniform_accuracy = []
for M_size in M_values
    x = [i / M_size for i in 0:M_size]

    A = make_FEM_A(x)
    F = make_F(true_f, x)
    C = A \ F

    push!(uniform_accuracy, get_inf_accuracy(true_u, C, x))
end

uniform_accuracy
```

[12]:
```
7-element Array{Any,1}:
 0.002503650522406753
 0.0006249615050749058
 0.00015733694607772408
 3.933102762151974e-5
 9.835245210587651e-6
 2.4588865398522675e-6
 6.147364057795812e-7
```

Accuracy appears to become 1/4th as h halves. Order is $O(h^2)$.

0.3 Part (d). Non-uniform grid

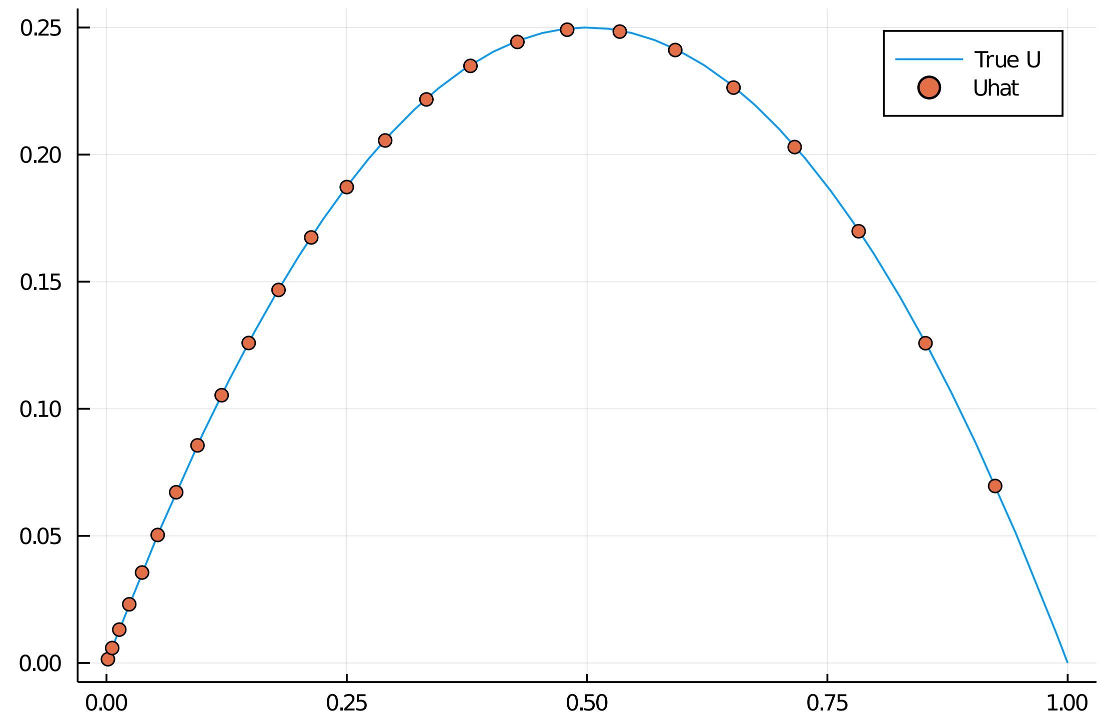
[13]:
```
M = 25
x_nonunif = [(i/(M+1))^2 for i in 0:M+1]

A = make_FEM_A(x_nonunif)
F = make_F(true_f, x_nonunif)
```

```julia
C = A \ F;
```

[14]:
```julia
plot(x -> true_u(x), 0, 1,
    label = "True U")
scatter!(x_nonunif[2:(length(x_nonunif) - 1)], C,
    label = "Uhat")
```

[14]:



[15]:
```julia
### For various M, generate error
M = [5, 7, 10, 14, 20, 28, 40, 56]

non_uniform_accuracy = []
non_uniform_h = []

for M in M_values
    x = [(i/(M+1))^2 for i in 0:M+1]

    A = make_FEM_A(x)
    F = make_F(true_f, x)
    C = A \ F

    push!(non_uniform_h, maximum(diff(x)))
    push!(non_uniform_accuracy, get_inf_accuracy(true_u, C, x))
end
```

[16]:
```
non_uniform_h
```

[16]:
```
7-element Array{Any,1}:
 0.30555555555555547
 0.17355371900826455
 0.09297052154195018
 0.04818560380725767
 0.02453894223441555
 0.012383781489911705
 0.006220824720256979
```

[17]:
```
non_uniform_accuracy
```

[17]:
```
7-element Array{Any,1}:
 0.007747431796788495
 0.002471203614839018
 0.0006799635712734509
 0.00017902020615093162
 4.594047342171281e-5
 1.1629164467424902e-5
 2.9255791754445593e-6
```

As $h = \max_i x_{i+1} - x_i$ halves, accuracy (according to infinity norm) shrinks by 1/4. Order is once again $O(h^2)$.