Computing for Mathematical Physics 2022/23

Homework6

Mark for homework6: /38

(to be competed by your marker)

Feedback from marker:

(to be competed by your marker)

Give your answers in the code cells marked (* Enter your solution here *)

Double click the vertical braces on the RHS of each of the question headings to open and view them.

- 1. `FindRoot`, `NSolve`, `FindMinimum` and `FindMaximum`. [17 marks]
  - In the following we aim to compute the x values of all maxima and minima of the function

    - eq. 1 : $f(x) = \text{BesselJ}[0, x] + \text{Sin}[3\,x]$,

    in the range $x_{min} < x < x_{max}$, with $x_{min} = 0$ and $x_{max} = \pi$.

  - a) Set `xMin = 0` and `xMax = π`, and define a function `f[x_]` as per the definition above (eq. 1).
    [0 marks]

In[1]:=
```
xMin = 0;
xMax = π;
f[x_] := BesselJ[0, x] + Sin[3 x]
```

- b) Plot the full extent of $f(x)$, for $x_{min} \leq x \leq x_{max}$. Add a `Frame` to the plot, labelling the axes with `FrameLabel`, and add grid lines. Use `LabelStyle` to set the font colour, weight and family to black, bold and Courier respectively.
  [4 marks]

In[4]:=
```
Plot[f[x], {x, xMin, xMax}, (*function and range*)
 Frame → True, FrameLabel → {X, Y},
 GridLines → Automatic, (*frame specifications*)
 LabelStyle → Directive[Black, Bold, FontFamily → "Courier"]
 (*label specifications*)]
```

- c) Compute the derivative, df/dx, and define a function `fPrime[x_]` returning the value of df/dx for a given input `x`. Display the numerical value of `fPrime[2]` to 4 s.f. only.
  [1 mark]

In[5]:=
```
dx = D[f[x], x];
fPrime[a_] := dx /. x → a;(*straighforward derivative*)
```

In[7]:=
```
N[fPrime[2], 4]
```

- d) Use `fPrime[x]` to plot the full extent of df/dx for $x_{min} < x < x_{max}$ with the same styling specified in b).
  [2 marks]

In[8]:=
```
(*Copy part b and replace function*)
Plot[fPrime[x], {x, xMin, xMax}, Frame → True,
 FrameLabel → {X, Y}, GridLines → Automatic,
 LabelStyle → Directive[Black, Bold, FontFamily → "Courier"]]
```

- e) Use `FindRoot` together with `fPrime[x]` to compute the x-values of all maxima and minima of f(x) in $x_{min} < x < x_{max}$.
  [4 marks]

In[9]:=
```
(*find roots using an appropriate
 starting value that we can deduce from the graph*)
FindRoot[fPrime[x], {x, 0.5}]
FindRoot[fPrime[x], {x, 1.6}]
FindRoot[fPrime[x], {x, 2.6}]
```

- f) Use `FindMaximum` and `FindMinimum` with `f[x]` to determine the x-values of all maxima and minima of f(x) in $x_{min} < x < x_{max}$.
  [4 marks]

In[12]:=
```
FindMaximum[f[x], {x, 0.5}]
FindMinimum[f[x], {x, 1.6}]
FindMaximum[f[x], {x, 2.6}]
```

- g) Use `NSolve` to determine the x-values of all maxima and minima of f(x) in the same latter range.
  [2 marks]

In[37]:=
```
NSolve[(fPrime[x] == 0) && xMin < x < xMax, x]
```

- 2. `FindRoot`. [6 marks]
  - [If this question is proving difficult please move on to the next one and return here later.]
  - The code in the cell directly below this one makes a plot with the following two elements.

    i) A green/yellow 3D surface, defined as function of *x* and *y*, with *z* coordinates given by the function

    - `surface[x_, y_] := BesselJ[0, ((x - 1/6)^2 + y^2)];`

    ii) A red wavy line, also in 3D, described by the following three parametric equations depending on the parameter *x* [which also happens to directly correspond to the x-coordinate here]:

    - [line x-coordinate] `xFn[x_] := x`
    - [line y-coordinate] `yFn[x_] :=` $\frac{1}{6}$ `Sin[6 x]`
    - [line z-coordinate] `zFn[x_] :=` $\frac{1}{6}$ `Cos[6 x]` + $\frac{1}{6}x^2 + \frac{1}{2}$

    The *region of interest* is defined as, $-3 \le x \le 3$, $-3 \le y \le 3$.

    The output of the code is also included below for convenience.

    As you can see, the red wavy line intersects the green/yellow surface at just two points in the region of interest.

    a) Use `FindRoot` to determine the 3D coordinates of the two points where the red wavy line intersects the green/yellow surface, in the region of interest. Store these two points as `firstPoint` and `secondPoint`.

    b) Make two `Graphics3D[...]` objects using `Sphere[...]`, to display two 3D spheres with radius $\frac{1}{5}$ centred on each of the two intersection points determined in part a), and insert these into the `Show[...]` command in the code given below.

[6 marks]

In[102]:=

```
(* height/z-coordinate of surface as a function of x & y *)
surface[x_, y_] := BesselJ[0, ((x - 1/6)^2 + y^2)];

(* Parametric equation describing wavy line *)
xFn[x_] := x
yFn[x_] := 1/6 Sin[6 x]
zFn[x_] := 1/6 Cos[6 x] + 1/6 x^2 + 1/2

(* Region of interest *)
{xMin, xMax} = {-3, 3};
{yMin, yMax} = {-3, 3};

(* Making a combined 3D plot of the surface and line *)
Show[

 (* Plot of green/yellow 3D surface *)
 Plot3D[
  (* Equation defining height of surface *)
  (* as a function of x and y *)
  surface[x, y],

  (* Dependent variables and their ranges *)
  {x, xMin, xMax},
  {y, yMin, yMax},

  (* Presentation *)
  ColorFunction → "AvocadoColors",
  PlotRange → {All, All, {-1/2, 3/2}}
 ],

 (* Plot of wavy red line in 3D *)
 ParametricPlot3D[

  (* Functions giving the line's x, y, and z *)
  (* coordinates in terms of x *)
  {xFn[x], yFn[x], zFn[x]},

  (* Parameter x *)
```

```
  {x, xMin, xMax},

  (* Presentation *)
  PlotStyle → {Red, Thickness[0.01]}
 ],
 Graphics3D[Sphere[sphere1Coordinates, 0.2]],
 Graphics3D[Sphere[sphere2Coordinates, 0.2]],
 (* Presentation *)
 PlotLabel → "Determining complex intersections with FindRoot",
 AxesLabel → {"x", "y", "z"},
 LabelStyle → {Black, Bold, 14},
 ImageSize → Large
]
```

(* Enter code in a new cell immediately below this one. *)
The Bessel function equals the parametric function. The Bessel function has inputs of x and y. So if we can make the subject of the parametric equation z, we can express it in terms of x and y.
But there’s a trick we can do. The line intersects the Bessel function at two points. If we look at the graphic we can see that the x and z values are the same at only two points for the two graphs, the points which are the same for the 2 graphs in 3 dimensions. This isn’t true for x-y values or z-y values. This means we can make a parametric equation for z in terms of only x. It also means we can we can put rearrange y in terms of x and input it into the Bessel function without having to worry about other points of intersection that might be missing if we don’t consider the y-axis.
Lets use the variable t for our parametric equations:
x(t) = t
y(t) = $\frac{1}{6}$ Sin[6 t]
z(t) = $\frac{1}{6}$ Cos[6 t] + $\frac{1}{6}$ t^2 + $\frac{1}{2}$
Now we can express z in terms of x, and y in terms of x
y= $\frac{1}{6}$ Sin[6 x]
z =$\frac{1}{6}$ Cos[6 x] + $\frac{1}{6}$ x^2 + $\frac{1}{2}$
Now we can solve for the two points of intersection by equating the Bessel function to the parametric equation.

In[8]:=

```
firstPoint = FindRoot[
  surface[x, 1/6 Sin[6 x]] == 1/6 Cos[6 x] + 1/6 x^2 + 1/2, (*equate two equations*)
  {x, -1} (*specify starting point which we can induce on the graph*)
 ]
secondPoint = FindRoot[
  surface[x, 1/6 Sin[6 x]] == 1/6 Cos[6 x] + 1/6 x^2 + 1/2,
  {x, 1}
 ]

sphere1Coordinates = {x, 1/6 Sin[6 x], 1/6 Cos[6 x] + 1/6 x^2 + 1/2} /. firstPoint
(*create spheres using coordinates*)
sphere2Coordinates = {x, 1/6 Sin[6 x], 1/6 Cos[6 x] + 1/6 x^2 + 1/2} /. secondPoint
```

- 3. Solving differential equations with `DSolve` and `NDSolve`. [15 marks]
  - Consider the following differential equation
    - eq. 2 $\frac{dy}{dt} = \frac{y^2}{1+t^2}$

    with the boundary condition $y(0) = 1$.

  - a) Solve eq. 2, above, with its associated boundary condition, analytically, using `DSolve`, and store the solution as `q3Analytic`. [3 marks]

In[130]:=

```
q3Analytical = DSolve[{y'[t] == y[t]^2 / (1 + t^2), y[0] == 1}, y[t], t]
```

  - b) Check the validity of the `q3Analytic` solution, by substituting the corresponding expressions for $y(t)$ and $y'(t)$ back into the original differential equation, eq. 2, using `/.` (`ReplaceAll`), and checking whether it reduces to `True` or not. [4 marks]

In[134]:=

```
y'[t] == y[t]^2/(1 + t^2) /. q3Analytical /. D[q3Analytical, t]
```

  - c) Repeat part a), but now using `NDSolve` in place of `DSolve`, to obtain a numerical solution over the range $0 \le t \le 1.5$. Store this solution as `q3Numerical`. [2 marks]

In[136]:=

```
q3Numerical = NDSolve[{y'[t] == y[t]^2/(1+t^2), y[0] == 1}, y[t], {t, 0, 1.5}]
```

- d) Plot the full extent of the `q3Analytic` $y(t)$ solution, over the range $0 \le t \le 1.5$. [2 marks]

In[140]:=

```
Plot[y[t] /. q3Analytical, {t, 0, 1.5}]
```

- e) Plot the full extent of

$$\frac{y(t) \text{ from q3Numerical}}{y(t) \text{ from q3Analytic}} - 1$$

over the range $0 \le t \le 1.5$. [2 marks]

In[148]:=

```
Plot[(y[t] /. q3Numerical) / (y[t] /. q3Analytical) - 1, {t, 0, 1.5}]
```

- f) Copy your code from parts c) and e) into a new cell. Increase the `WorkingPrecision` in `NDSolve` to `25`, and plot again

$$\frac{y(t) \text{ from q3Numerical}}{y(t) \text{ from q3Analytic}} - 1$$

Explain, clearly, in one sentence, in a text cell, the effect of this option on the numerical solution from `NDSolve`. [2 marks]

In[154]:=

```
q3Numerical = NDSolve[{y'[t] == y[t]^2/(1+t^2), y[0] == 1},
  y[t], {t, 0, 1.5}, WorkingPrecision → 25]
Plot[(y[t] /. q3Numerical) / (y[t] /. q3Analytical) - 1, {t, 0, 1.5}]
```

Increasing the working precision causes the difference between the numerical solution and the analytical solution to be smaller. This makes the the numerical solution more accurate and similar to the analytical solution.

- **Total marks available: 38**

- **Solutions are due *by* 1200 noon on Thursday March 2nd *here*: allow time for uploading on moodle.**
- A 10% mark deduction will be made (4 marks) if the template isn't used.
- Name your solution notebook file in the format **WK6_HMWK_<Initials>_<Family Name>.nb**, e.g. **WK6_HMWK_K_Hamilton.nb**

- Make a *backup copy* of your solutions.
- Delete all cell evaluation output by selecting **Cell → Delete All Output** from the drop-down menus at the top of the screen, then save and upload *that* file to Moodle.
- The first thing your marker will do when they receive your notebook is to evaluate all of it, to regenerate the output, by clicking **Evaluation → Evaluate Notebook** from the drop-down menus at the top of the screen. *It is your responsibility to check that carrying out this process, with a fresh kernel, will produce the output you intend it to, before you upload your work.*

K. Hamilton
15:18 21 Feb 2023