Rheokinetics (44 points)

In [1]:
```
#import packages here
import numpy as np
import matplotlib.pyplot as plt
from math import e
```

In the last notebook, you got familiar with the kinetics related to the curing of thermoset polymers. This time, the focus will be on flow-related topics of such polymers. This is applicable to several manufacturing techniques you heard of in the lectures, like liquid composite moulding. This area of study is called rheology, and it plays a fundamental role in the manufacturing of polymers, serving as a crucial scientific discipline that governs the viscosity, flow and deformation behavior of polymer materials. Deep knowledge of rheology is indispensable to work in polymer composite manufacturing. This field enables control of material behavior and improved quality of produced parts.

As you now know, the curing of a thermoset consists of chemical reactions between individual monomers, or polymer chains and monomers, or polymer chains and other polymer chains. This phenomenon, known as cross-linking, results in a progressive increase in the molecular weight of thermosets, leading to a corresponding rise in viscosity.

The viscosity of a fluid is a measure of its resistance to deformation at a given rate. Viscosity can be conceptualized as quantifying the internal frictional force that arises between adjacent layers of fluid that are in relative motion.

Question 1 (2 points)

As explained above, viscosity changes as the degree of cure α changes. Explain, on a physical and chemical level, why this occurs. Also briefly discuss why this variable is important to understand for liquid composite moulding manufacturing.

Answer: During the curing, covalent bonds form between the molecules. These bonds link the monomers together to form mulecular chains, and also form cross-links, linking the molecular chains together. These bonds and crosslinks make it more difficult for the molecules to slide past each other, thus increasing the viscosity. This understanding is important in liquid composite moulding to ensure the impregnation occurs properly. Too viscous and the resin does not cover the entire material before the cure time has been reached, and too low viscosity means the capillary forces will dominate over the viscous, and the process is no longer properly balanced.

Several models exists to predict the flow behaviour of thermoset materials. The Castro-Macosko model is a widely used and well-established model to simulate the changes in viscosity of thermoset materials. The model describes the viscosity as a function of temperature T and degree of cure α. Shear rate γ will not be considered in this version of the model. The model is displayed in equation 1.

$$\eta(T,\alpha) = \eta_0(T)(\frac{\alpha_g}{\alpha_g - \alpha})^{c1+c2\alpha}$$

For which $\eta_0(T) = A\exp\frac{T_b}{T}$ and $T_b, c1, c2, \alpha_g$ and A are material properties.

The parameter α_g identifies the degree of cure at which gelation occurs, at which point the viscosity of material increases significantly.

The following model is given to you for the considered dicyanate: 1,1-bis(4-cyanatophenol)ethane, known as AroCy L-10, supplied by Rhone-Poulenc Inc.

$$\eta(T,\alpha) = 3.32\cdot 10^{-8}\exp\frac{5160.39}{T}(\frac{0.64}{0.64-\alpha})^{2.32+1.4\alpha}$$

The parameters are reported below: $T_b = 5160.39K$ $c_1 = 2.32$ $c_2 = 1.4$ $\alpha_g = 0.64$ $A = 3.32\cdot 10^{-8} Pa\cdot s$

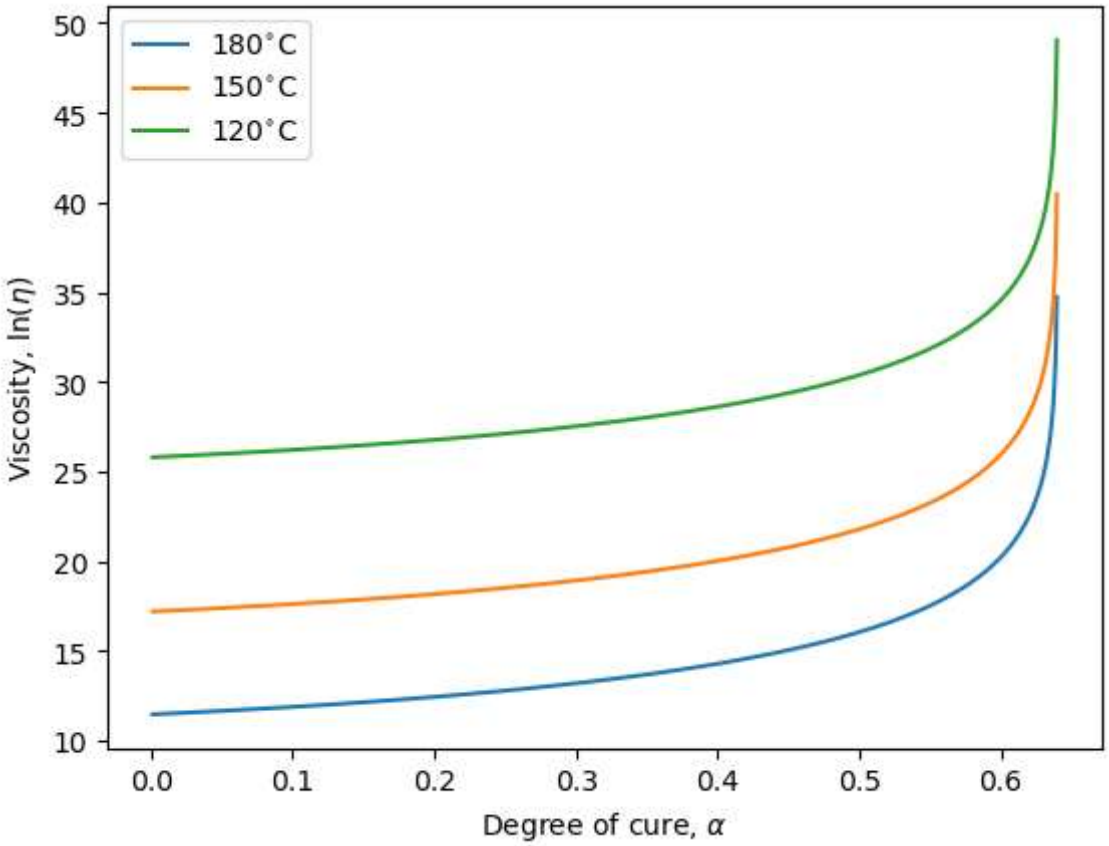
Source: Chen, Y.-T. and Macosko, C.W. (1996), Kinetics and rheology characterization during curing of dicyanates. J. Appl. Polym. Sci., 62: 567-576. https://doi-org.tudelft.idm.oclc.org/10.1002/(SICI)1097-4628(19961017)62:3567::AID-APP143.0.CO;2-W

Question 2 (3 points)

Question 2.1: Plot for this resin system the viscosity versus the degree of cure for several isothermal cure temperatures. Report the chosen temperatures and choose a suitable scale for the axes.

In [2]:
```
Tb = 5160.39
c1 = 2.32
c2 = 1.4
alpha_g = 0.64
A = 3.32e-8

def viscosity(alpha, T):
    return A * np.exp(Tb/T) * (alpha_g/(alpha_g-alpha))**(c1+c2*alpha)

alpha = np.linspace(0.001,0.999,2000)

T1 = 180
T2 = 150
T3 = 120
vis_1 = viscosity(alpha,T1)
vis_2 = viscosity(alpha,T2)
vis_3 = viscosity(alpha,T3)

plt.plot(alpha, np.log(vis_1), label='180$^{\circ}$C')
plt.plot(alpha, np.log(vis_2), label='150$^{\circ}$C')
plt.plot(alpha, np.log(vis_3), label='120$^{\circ}$C')
plt.legend()
# plt.yscale('ln')
plt.xlabel(r"Degree of cure, $\alpha$")
plt.ylabel(r"Viscosity, ln($\eta$)")
plt.show()
```

```
C:\Users\coenh\AppData\Local\Temp\ipykernel_14304\622293621.py:8: RuntimeWarning:
invalid value encountered in power
  return A * np.exp(Tb/T) * (alpha_g/(alpha_g-alpha))**(c1+c2*alpha)
```



Question 2.2: Which information regarding the processability of this resin can be retrieved from the graph? You can include considerations that would be relevant for LCM processes such as choice of different cure temperatures and how viscosity plays a role in the process and discuss important observations made from the graph.

Answer: This graph shows the minimum cure temperature required to obtain a cure while staying below a given viscosity. For example, if the natural logarithm of the viscosity needs to be below 25 to fully infuse the part before it cures, then an isothermal cure temperature of 120 degrees celcuis will not be suitable. From this graph, we can't see what the maximum temperature is, as this will depend on the time it takes to reach a particular degree of cure. Additionally, it can be noted that the degree of cure at gellation does not change with differing temperature. Hence, it is more likely that a lower temperature will have more time until the degree of cure of gellation is reached.

Question 3 (4 points)

The goal of this question is to analyze the correlation between viscosity, temperature and heating rate for non-isothermal curing. In the image below, a typical viscosity vs. temperature plot is displayed for values obtained experimentally and through the Castro-Macosko model. A U-shape viscosity curve pattern can be observed when the temperature is increased. The viscosity drops at first, but ater reaching a minimum, viscosity soon increases significantly.

Explain the behaviour of viscosity while curing, specifically:

1. What is the reason behind the initial lowering of resin viscosity?

2. What information does the minimum of the curve provide?

3. Why does this trend change significantly, after reaching a certain temperature?

4. How does the heating rate effect the shape of the curve and why?

Write your answer in the textbox below the image and please provide complete and detailed explanations. You can refer to the Castro-Macosko equation as part of your answer.

Answer:

1. The initial lowering of the resin viscosity is due to the fact that an increase in temperature lowers the viscosity. This can be explained physically as the modelcules have more energy, and hence they more around more easily. This was also seen in the plotting of the Castro-Macosko model above.
2. The minimum of the curve describes the minimum viscosity, after this the viscosity shoots up, and the gellation has occured. Hence, this minumum indicates the degree of cure at gellation. It makes sens that this occurs at a lower temperature with a slower increase in temperature, as the system will, have had a longer time to react.
3. This trend changes significatly due the the theory described above. At gellation, the cross-linking of the molecules starts, hence significantly increasing the viscosity of the resin. In the Castro-Macosko model, when alpha approaches alpha gellation, the second term rapidy increases, yielding the peak in viscosity we see in both graphs.
4. A slower heating rate means the resin has been at a temperature for longer, and hence means the degree of cure is higher than that of the same resin which has had half the reaction time. A higher degree of cure at the same temperature means the resin will reach the gellation point at a lower temperature than the faster heating rate counterpart.

Dynamic Mechanical Analysis, otherwise known as DMA, is an experimental technique employed to investigate the response of materials to cyclic deformations under controlled conditions (stress, temperature, frequency, and various other parameters).

DMA works by applying a sinusoidal deformation to a sample of known geometry. The sample can be subjected to a controlled stress or a controlled strain. For a known stress, the sample will then deform a certain amount (or vice versa). The extent of deformation exhibited by the specimen is indicative of its stiffness.

DMA characterizes the material in terms of stiffness and damping, which are quantified as modulus and tangent delta (tan δ). Given the sinusoidal excitation, modulus can be further divided into in-phase (storage modulus, G') and out-of-phase (loss modulus, G'') components. The storage modulus, G', delineates the material's elastic behavior. $tan(\delta)$, calculated as the ratio of the loss modulus to the storage modulus, represents damping and serves as a measure of a material's capacity to dissipate energy.

Question 4 (3 points)

The image below shows a shear-stress shear-strain relationship for various materials over time. As mentioned above, the shear-strain is applied in a sinusoidal manner. This leads to shear-stresses in various different ways.

Comment on the shapes and phase-shifts of the graphs of the Hooke solid, Newton fluid, and polymer. Explain why these are exactly the results you would expect from a DMA analysis on these 3 materials.

Answer: The Hookes solid has zero phase shift, and it stores all the energy given to it. In class this was shown as the ball not deforming, but bounding back to the original height. Here the shear strain is at maximum when the largest shear stress is applied. In contrast, the liquid has pi/2 phase shift, it loses all the energy given to it. If the liquid is dropped, it will deform completely and form a puddle instead of bouncing back up. In this case, the shear strain is zero while the shear stress is maximal. Correspondingly, when the shear stress is zero, the strain is at maximum. The polymer is a mix between the two, and depending on the voscosity, degree of cure, etc. it acts more like a Hookes solid, or a Newton liquid.

From tests like these, information about the storage modulus G' and loss modulus G'' can be derived. This is achieved via a known complex relationship between shear-stress and shear-strain.

$$G* = \frac{\tau_{max}}{\gamma_{max}} = G' + iG''$$

This can then be used to relate to the phase shift δ in the following manner.

$$tan(\delta) = \frac{G''}{G'}$$

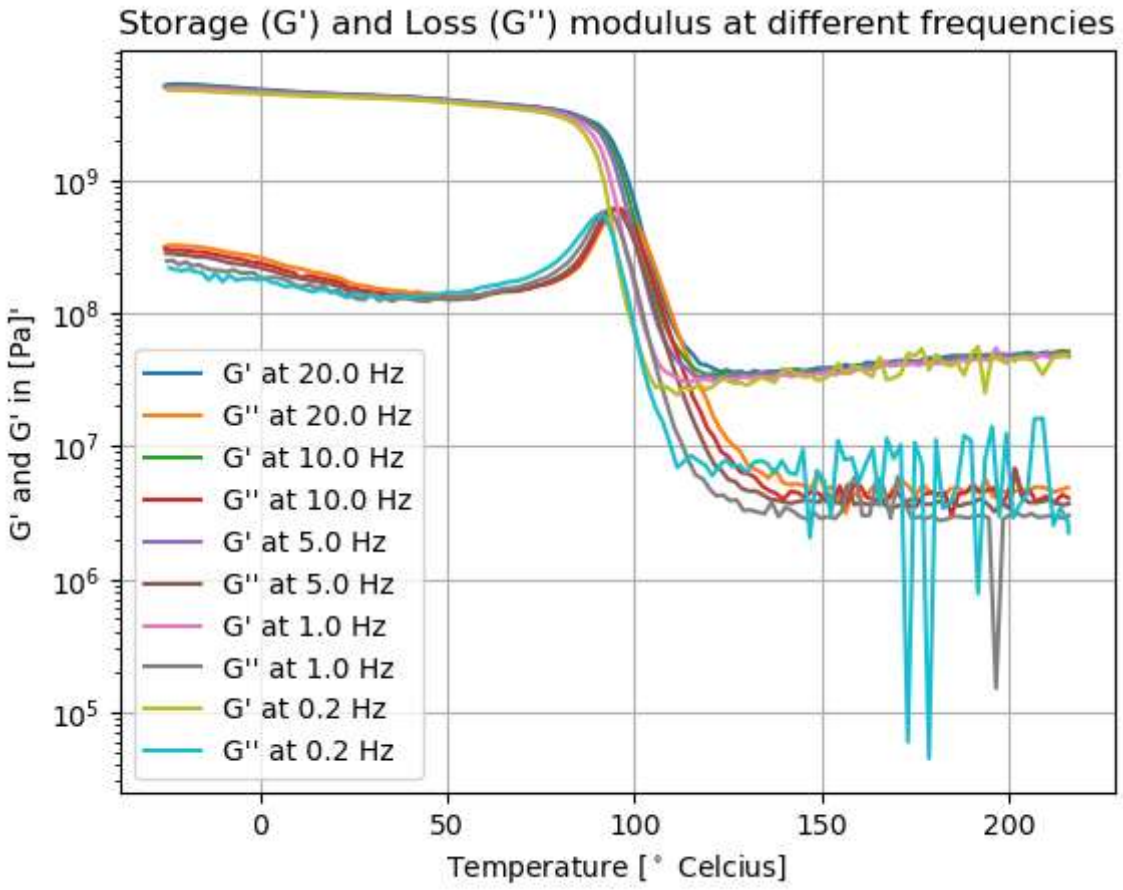
Question 5 (2 points)

A spreadsheet containing DMA measurement data has been provided to you with this notebook. In this dataset, storage and loss moduli are noted for different frequencies at different temperatures. The sample is Airstone 780E, at 95% cure (only use sample #1).

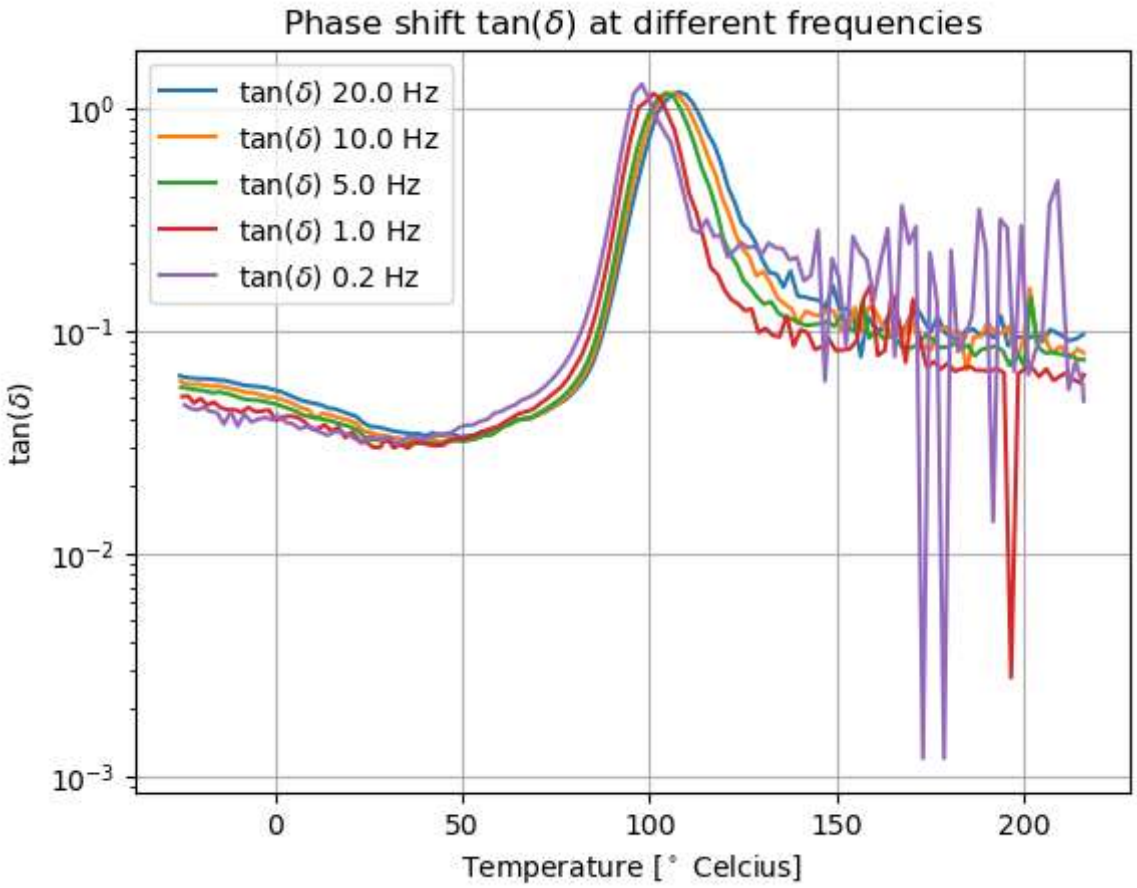
From this data, plot two graphs showing G' and G'' vs. temperature (for each frequency) and $tan(\delta)$ vs. temperature (for each frequency). Make sure to choose a suitable scale for the axes.

In [3]:
```
file = np.genfromtxt('dmaedit.txt',skip_header=2)

Freqs       = file[:,0]
Temp        = file[:,1]
Gprime      = file[:,2]
Gpp         = file[:,3]
tanDelta    = file[:,4]

Temp20 = Temp[np.where(Freqs==20)]
Temp10 = Temp[np.where(Freqs==10)]
Temp5  = Temp[np.where(Freqs==5)]
Temp1  = Temp[np.where(Freqs==1)]
Temp02 = Temp[np.where(Freqs==0.200000003)]
```

```
Temptot = np.array([Temp20[:-1],Temp10[:-1],Temp5[:-1],Temp1,Temp02])
Temptot = Temptot[:,:130]

Gprime20 = Gprime[np.where(Freqs==20)]
Gprime10 = Gprime[np.where(Freqs==10)]
Gprime5 = Gprime[np.where(Freqs==5)]
Gprime1 = Gprime[np.where(Freqs==1)]
Gprime02 = Gprime[np.where(Freqs==0.200000003)]

Gprimetot = np.array([Gprime20[:-1],Gprime10[:-1],Gprime5[:-1],Gprime1,Gprime02])
Gprimetot = Gprimetot[:,:130]

Gpp20 = Gpp[np.where(Freqs==20)]
Gpp10 = Gpp[np.where(Freqs==10)]
Gpp5 = Gpp[np.where(Freqs==5)]
Gpp1 = Gpp[np.where(Freqs==1)]
Gpp02 = Gpp[np.where(Freqs==0.200000003)]

Gpptot = np.array([Gpp20[:-1],Gpp10[:-1],Gpp5[:-1],Gpp1,Gpp02])
Gpptot = Gpptot[:,:130]

tanDelta20 = tanDelta[np.where(Freqs==20)]
tanDelta10 = tanDelta[np.where(Freqs==10)]
tanDelta5 = tanDelta[np.where(Freqs==5)]
tanDelta1 = tanDelta[np.where(Freqs==1)]
tanDelta02 = tanDelta[np.where(Freqs==0.200000003)]

tanDeltatot = np.array([tanDelta20[:-1],tanDelta10[:-1],tanDelta5[:-1],tanDelta1,ta
tanDeltatot = tanDeltatot[:,:130]

freqsteps = np.array([20,10,5,1,0.2])

f5 = plt.figure()
for i in range(len(freqsteps)):
    plt.plot(Temptot[i,:],Gprimetot[i,:],label=r"G' at "+str(freqsteps[i])+" Hz")
    plt.plot(Temptot[i,:],Gpptot[i,:],label=r"G'' at "+str(freqsteps[i])+" Hz")

plt.grid()
plt.legend()
plt.xlabel(r"Temperature [$^\circ$ Celcius]")
plt.ylabel(r"G' and G' in [Pa]'")
plt.yscale('log')
plt.title("Storage (G') and Loss (G'') modulus at different frequencies")
plt.show()
```



In [4]:
```
f6 = plt.figure()
for i in range(len(freqsteps)):
    plt.plot(Temptot[i,:],tanDeltatot[i,:],label=r"$\tan(\delta)$ "+str(freqsteps[i

plt.grid()
plt.legend()
plt.xlabel(r"Temperature [$^\circ$ Celcius]")
plt.ylabel(r"$\tan(\delta)$")
plt.yscale('log')
plt.title(r"Phase shift $\tan(\delta)$ at different frequencies")
plt.show()
```



Question 6 (2 points)

Consider the plots you generated in the previous question, showing the variations of G' and G'' as functions of temperature for Airstone 780E. Now, please address the following points regarding the relationship between moduli and temperature:

- Explain why there is a decrease in the storage modulus G'.

- Elaborate on why the loss modulus G'' shows a peak

Please provide detailed answers below

Answer: As the temperature rises, the material approaches its glass transition temperature. At this stage, the material's expanding volume reaches a point where chain flow becomes possible. Consequently, the material begins to exhibit viscous liquid's characteristics and its stiffness decreases because the entangled chains are less resistant to deformation. This understanding is represented in the graph by the decreasing in the storage modulus, G'. Also the increased molecular mobility result in more energy dissipation within the material because of the raise of internal friction, which is reflected by the peak in the loss modulus, G''.

Question 7 (3 points)

What can you derive from your results of question 5? What effect does the frequency have of storage modulus G'? Explain how this happens and why it is important to take this into consideration.

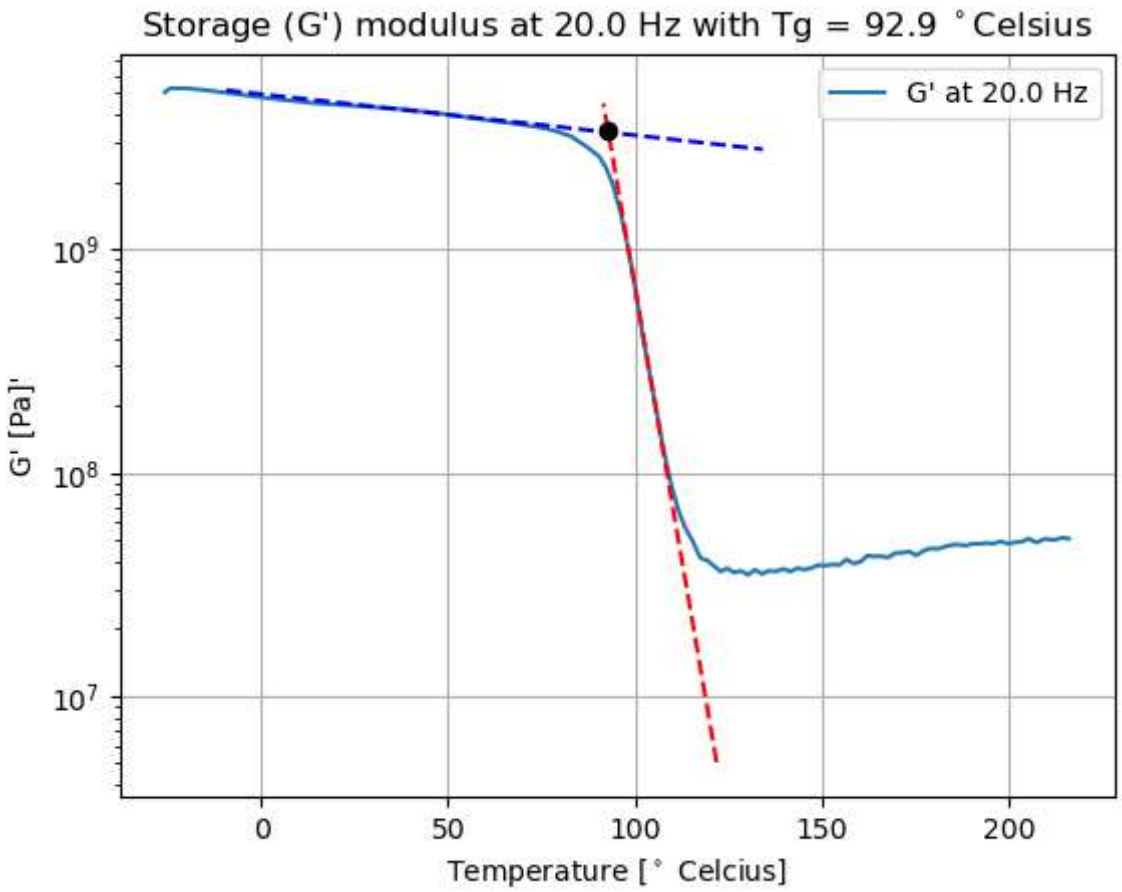
Answer: As observed in the graph of G' vs. temperature for each frequency, the storage modulus tends to increase with increasing frequency. As the frequency raises, the time intervals between the applied deformations get shorter. As a result, the material may not have enough time to fully relax and dissipate energy as heat, and the elastic response will dominate. This is important as it affects the damping properties of the material, and in some applications materials with specific viscoelastic properties at certain frequencies are required.

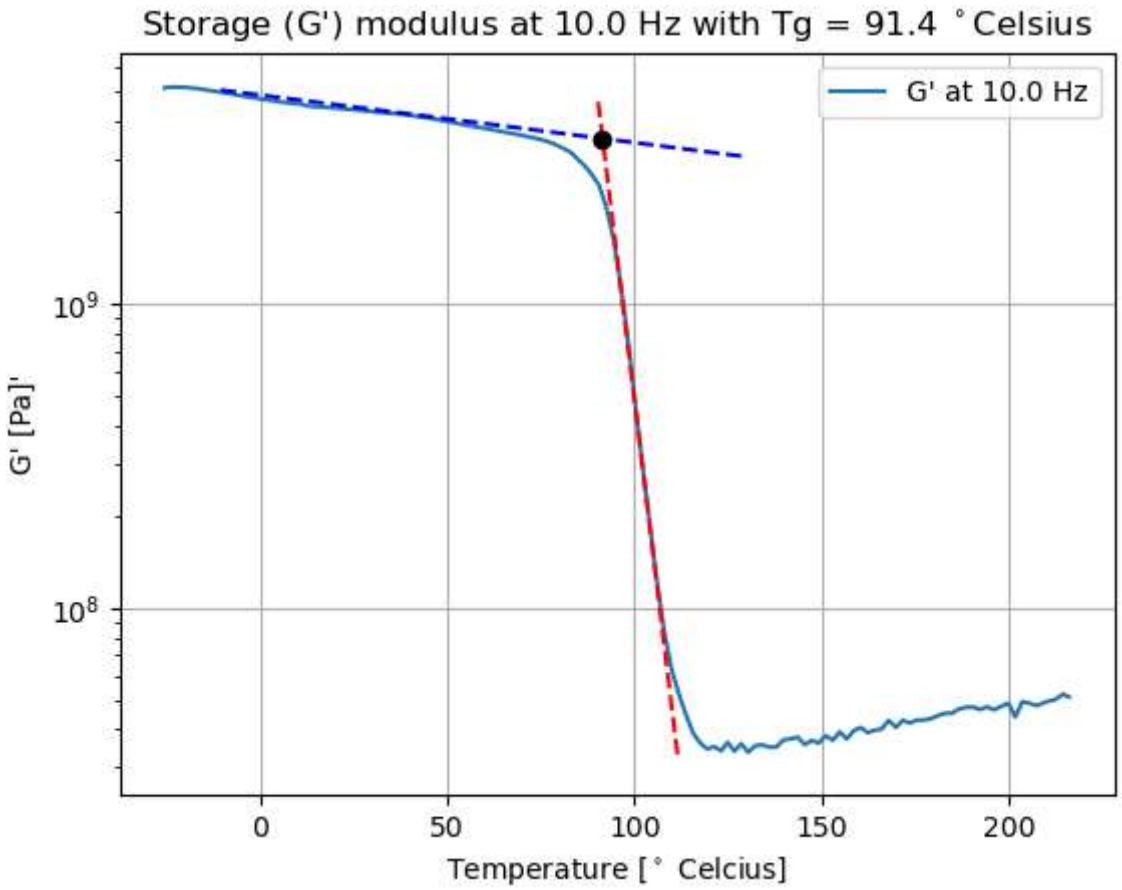
Question 8 (8 points)

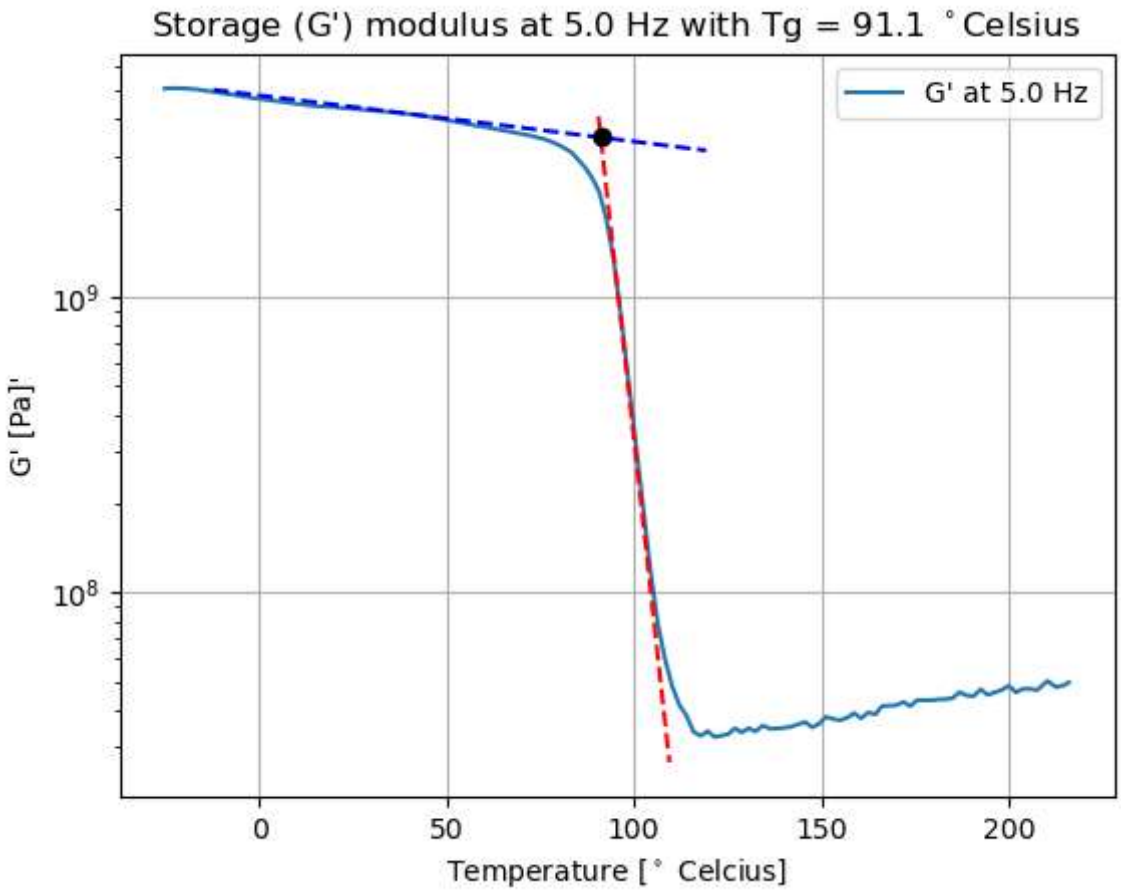
Question 8.1: From the above-used dataset, plot storage modulus vs. temperature for every frequency. Then, find T_g for each curve by using the following method: Tg can be determined from the intersection of two lines that are drawn in two regions; one in the brittle glassy state and the other in the transition region. The temperature at which these 2 lines intersect can be considered T_g. Report the found values of T_g in the title(s). Also, demonstrate this method in the said plot: plot the intersecting lines on top of the data. If you feel it's necessary, zoom in on the points of intersect.

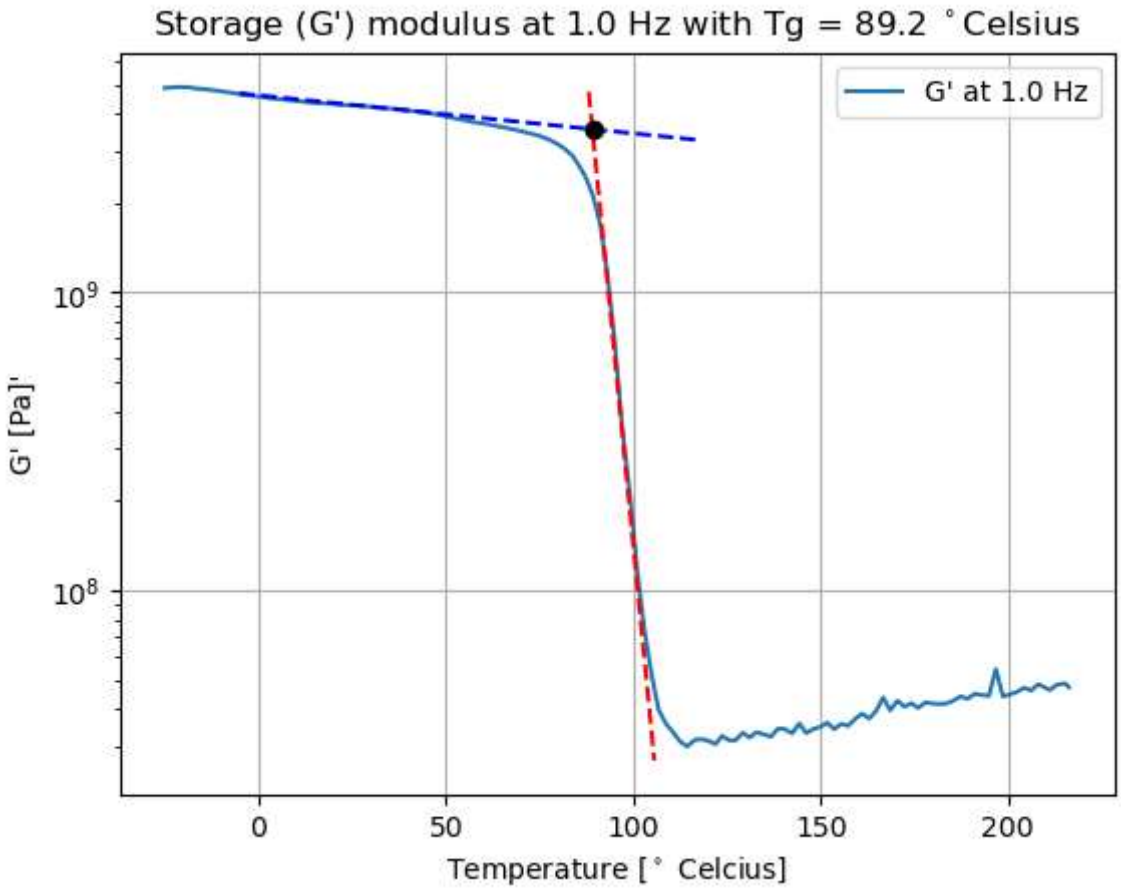
NOTE: make an individual plot for each frequency dataset, therefore five graphs should be plotted separately

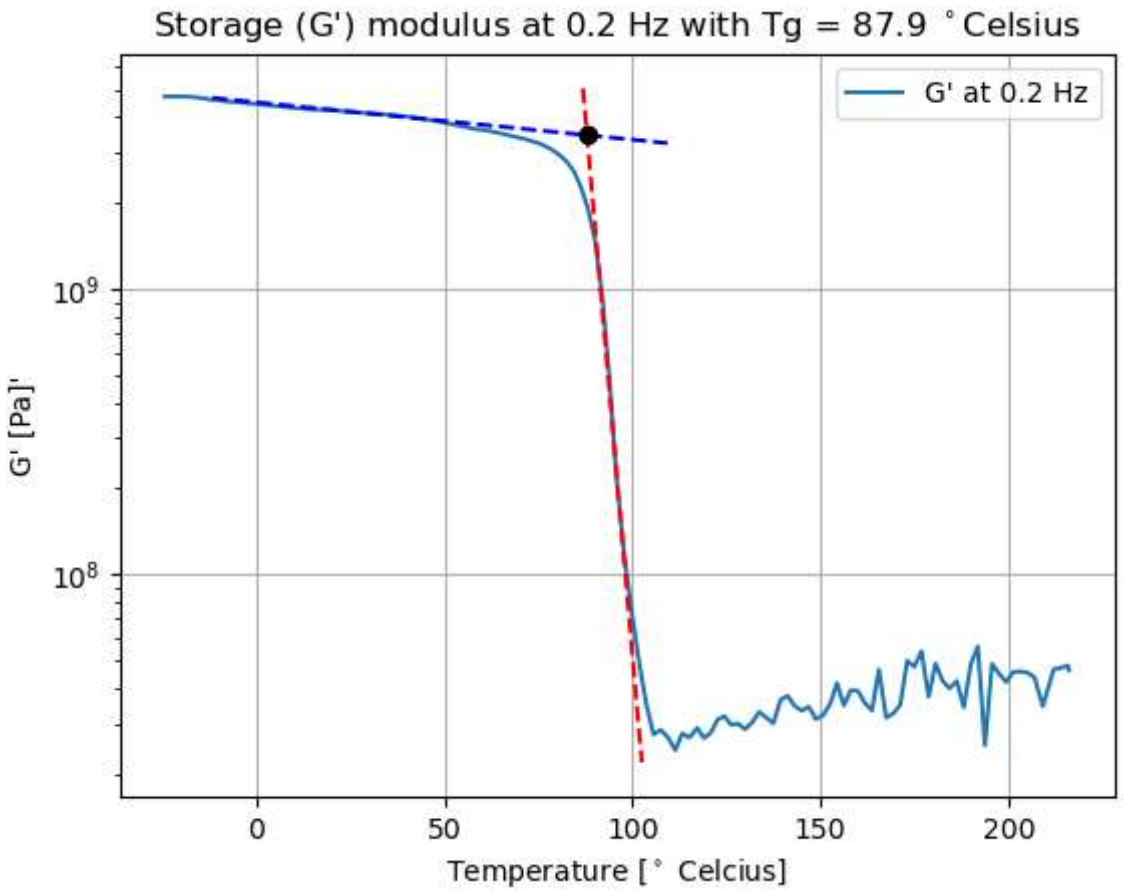
In [5]:
```
Brittlelinesx = np.array([[-9.259,134.156],[-10.548,128.542],[-12.2905,119.274],[-5
Brittlelinesy = np.array([[5.1474e9,2.8018e9],[5.0524e9,3.0632e9],[5.0118e9,3.126e9

Translinesx = np.array([[1.21811e2,9.1563e1],[90.279,111.686],[90.5028,109.46],[88.
Translinesy = np.array([[5.0524e6,4.4904e9],[4.62195e9,3.15558e7],[4.0738e9,2.66827

Lowerlinesx = np.array([[106.276,223.972],[103.831,222.278],[102.442,222.991],[100.
Lowerlinesy = np.array([[3.16228e7,5.65621e7],[3.12295e7,5.23845e7],[3.03876e7,5.13

Tgmethod1 = np.array([[92.9,3.37522e9],[91.4,3.4936e9],[91.1,3.46737e9],[89.2,3.541
Tgmethod2 = np.array([[103.3,3.19927e8],[101.4,3.36646e8],[100.1,3.31854e8],[97.2,3


for i in range(len(freqsteps)):
    plt.figure()
    plt.plot(Temptot[i,:],Gprimetot[i,:],label=r"G' at "+str(freqsteps[i])+" Hz")
    plt.plot(Brittlelinesx[i],Brittlelinesy[i],'b--')
    plt.plot(Translinesx[i],Translinesy[i],'r--')
    plt.plot(Tgmethod1[i,0],Tgmethod1[i,1],'ko')
    plt.grid()
    plt.legend()
    plt.xlabel(r"Temperature [$^\circ$ Celcius]")
    plt.ylabel(r"G' [Pa]'")
    plt.yscale('log')
    plt.title("Storage (G') modulus at "+str(freqsteps[i])+" Hz with Tg = "+str(Tgm
    plt.show()
```


Storage (G') modulus at 20.0 Hz with Tg = 92.9 ˚Celsius
G' at 20.0 Hz
10^9
10^8
10^7
G' [Pa]'
0
50
100
150
200
Temperature [˚ Celcius]


Storage (G') modulus at 10.0 Hz with Tg = 91.4 ˚Celsius
G' at 10.0 Hz
10^9
10^8
G' [Pa]'
0
50
100
150
200
Temperature [˚ Celcius]


Storage (G') modulus at 5.0 Hz with Tg = 91.1 ˚Celsius
G' at 5.0 Hz
10^9
10^8
G' [Pa]'
0
50
100
150
200
Temperature [˚ Celcius]

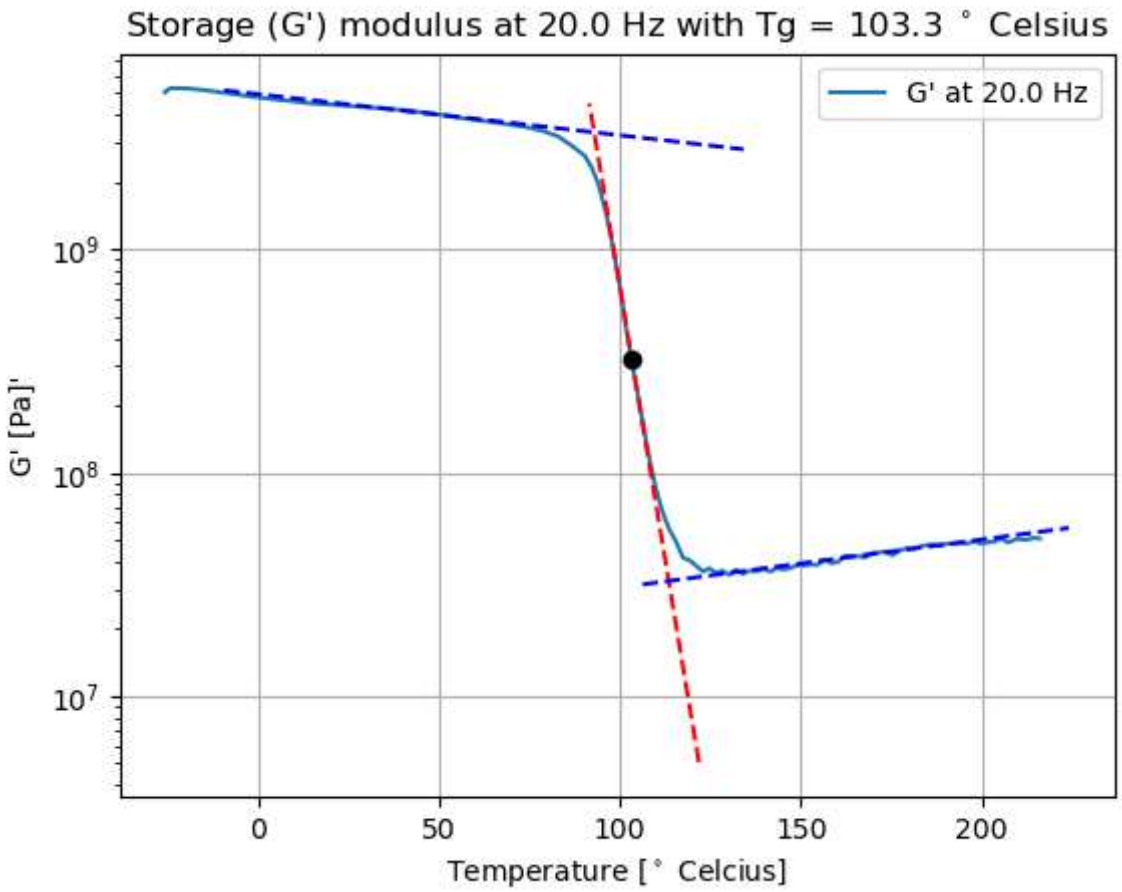

Storage (G') modulus at 1.0 Hz with Tg = 89.2 ˚Celsius
G' at 1.0 Hz
10^9
10^8
G' [Pa]'
0
50
100
150
200
Temperature [˚ Celcius]

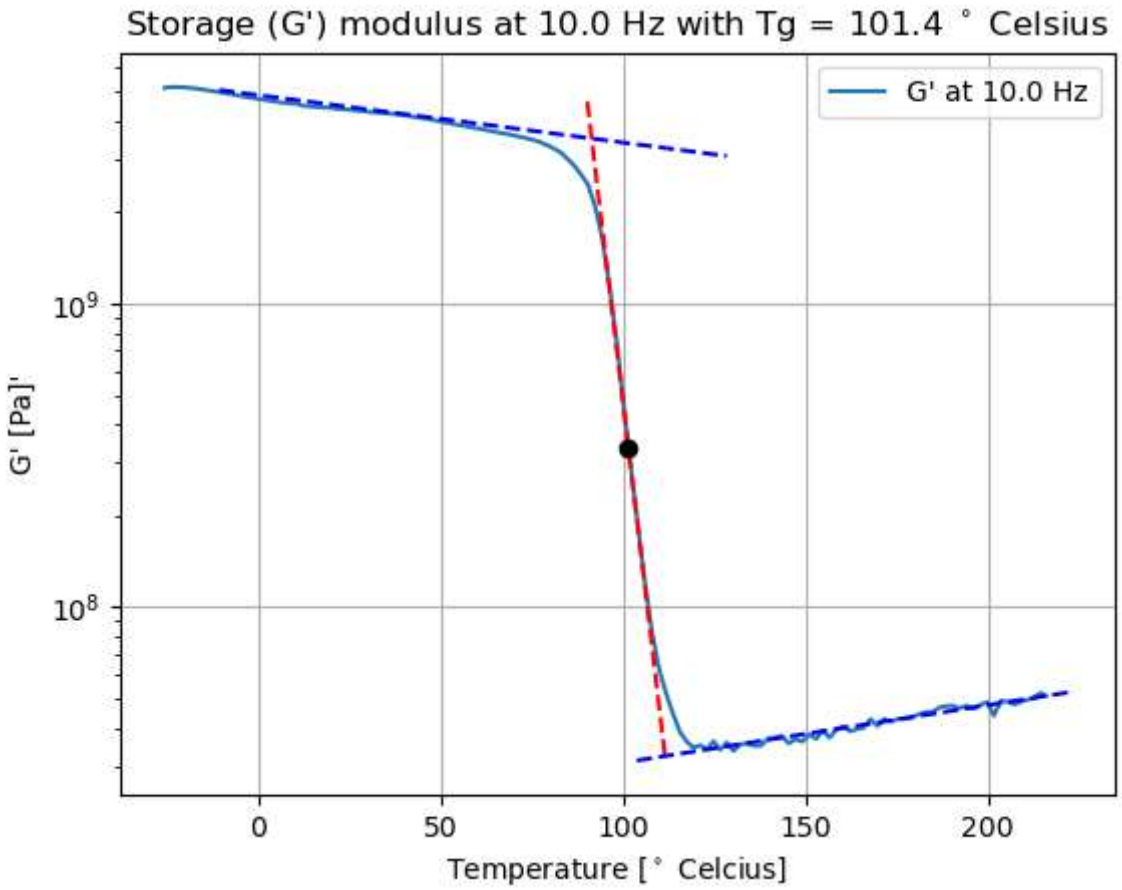


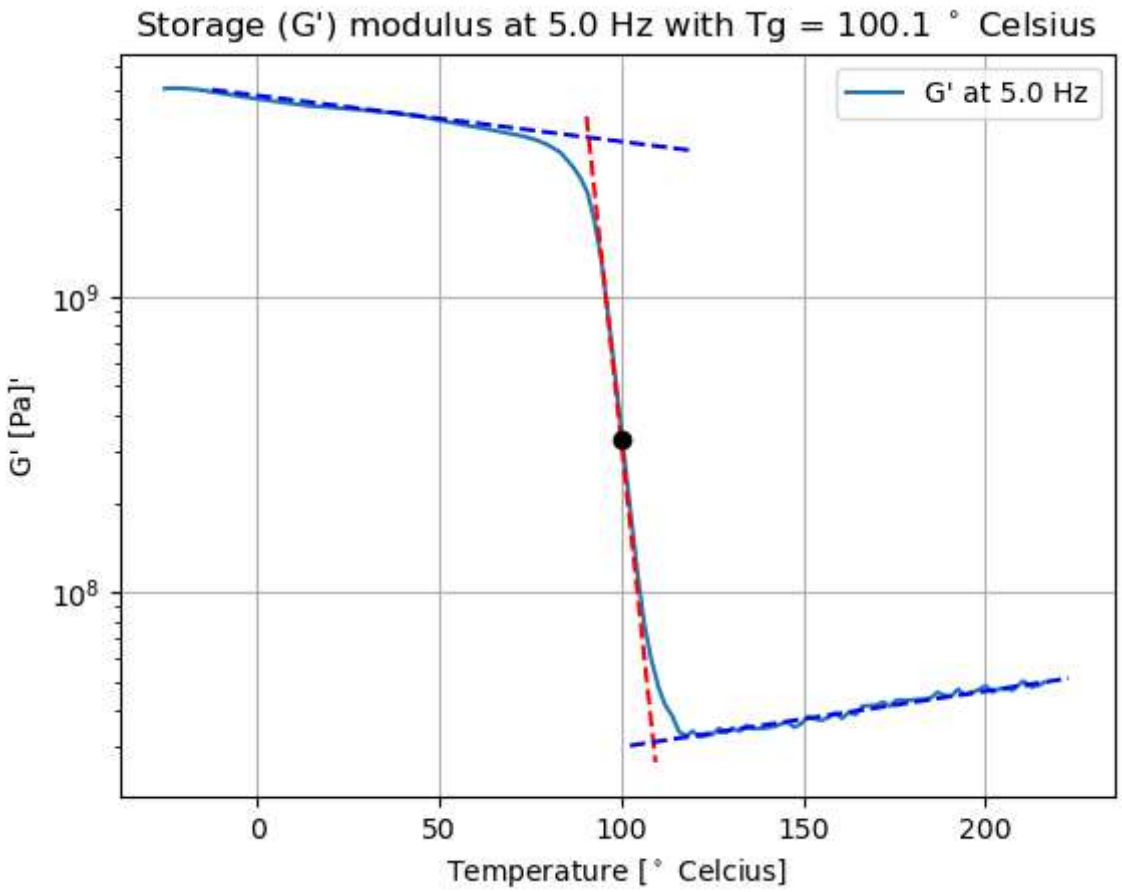
Question 8.2: In Question 8.1, you determined the T_g for each frequency by determining the onset of the storage modulus curve. Now, find values for T_g (for each frequency) using the inflection point of the storage modulus curve. The inflection point is defined as the midpoint between the onset and offset of the curve. Like you did before, plot the intersecting lines and the point of interest on top of the data.

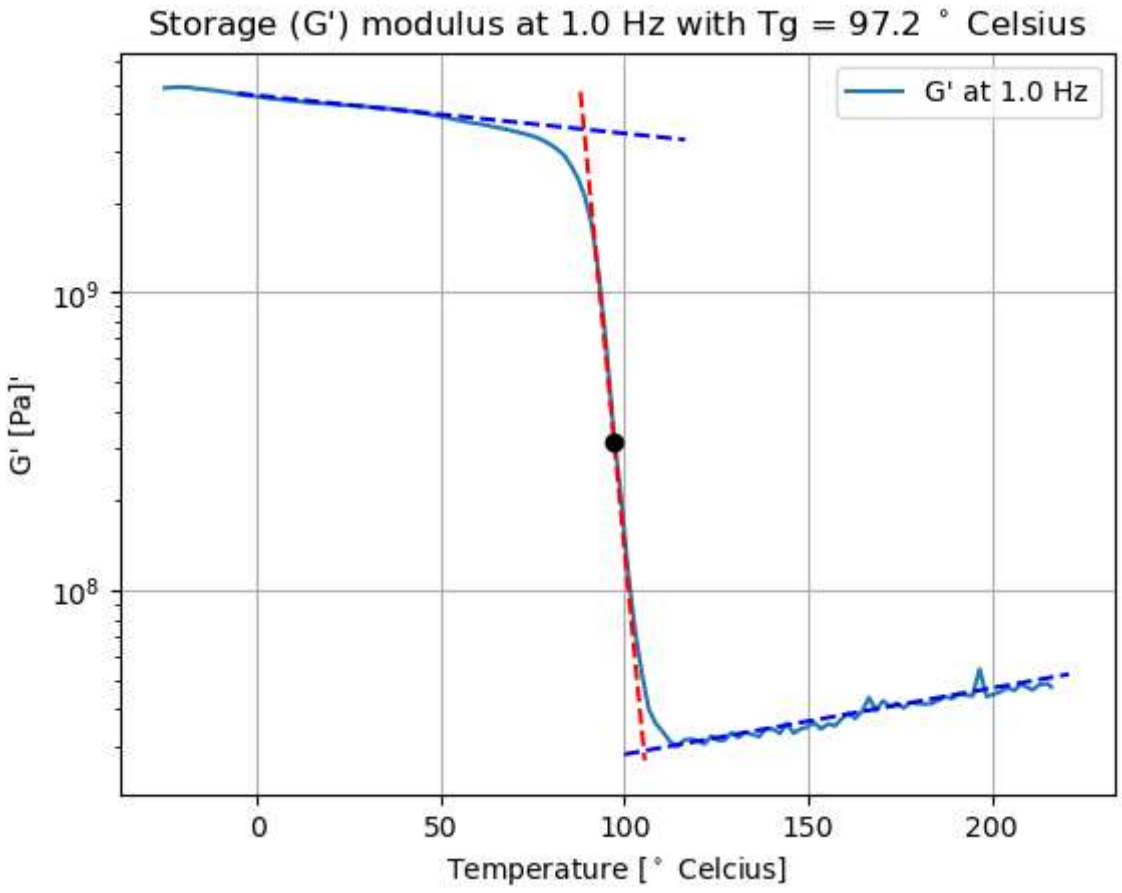
NOTE: make an individual plot for each frequency dataset, therefore five graphs should be plotted separately

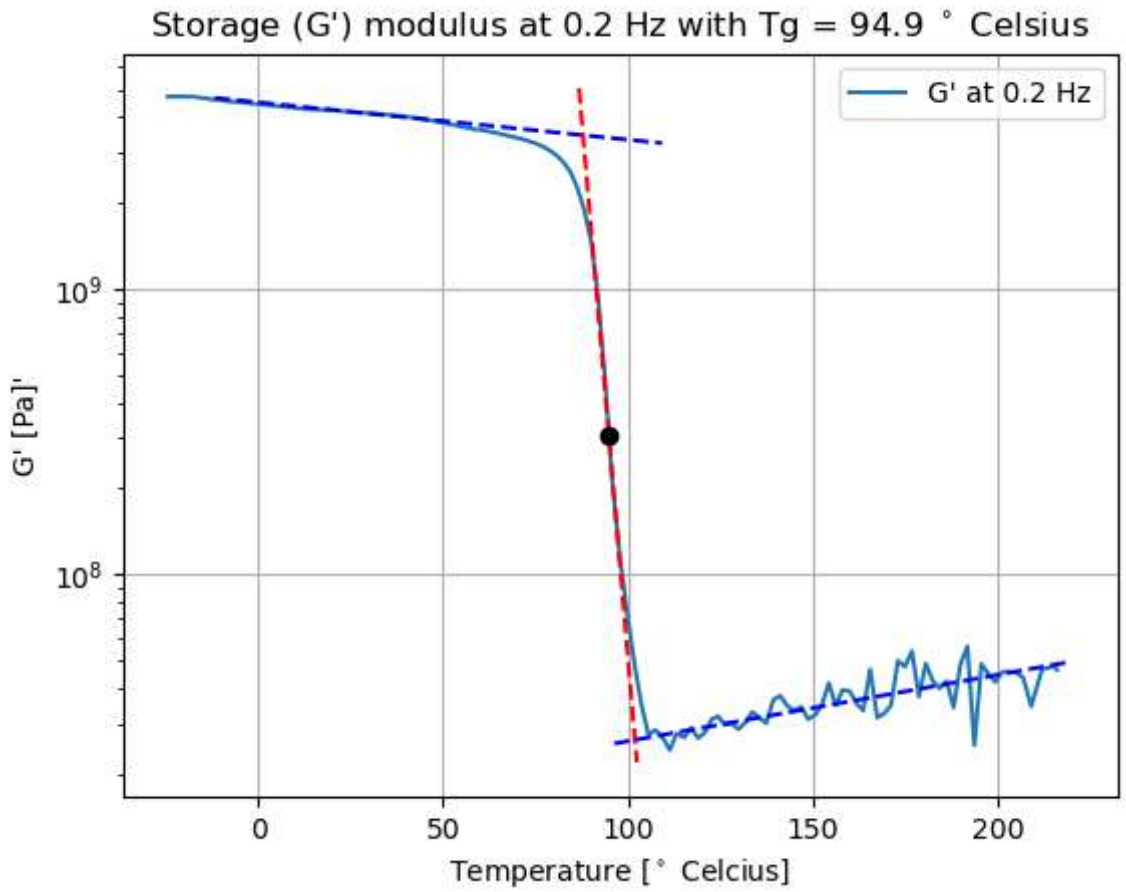
In [6]:
```
for i in range(len(freqsteps)):
    plt.figure()
    plt.plot(Temptot[i,:],Gprimetot[i,:],label=r"G' at "+str(freqsteps[i])+" Hz")
    plt.plot(Brittlelinesx[i],Brittlelinesy[i],'b--')
    plt.plot(Translinesx[i],Translinesy[i],'r--')
    plt.plot(Lowerlinesx[i],Lowerlinesy[i],'b--')
    plt.plot(Tgmethod2[i,0],Tgmethod2[i,1],'ko')
    plt.grid()
    plt.legend()
    plt.xlabel(r"Temperature [$^\circ$ Celcius]")
    plt.ylabel(r"G' [Pa]'")
    plt.yscale('log')
    plt.title("Storage (G') modulus at "+str(freqsteps[i])+" Hz with Tg = "+str(Tgm
    plt.show()
```


Storage (G') modulus at 20.0 Hz with Tg = 103.3 ° Celsius
G' at 20.0 Hz
G' [Pa]'
10^9
10^8
10^7
0
50
100
150
200
Temperature [° Celcius]


Storage (G') modulus at 10.0 Hz with Tg = 101.4 ° Celsius
G' at 10.0 Hz
G' [Pa]'
10^9
10^8
0
50
100
150
200
Temperature [° Celcius]


Storage (G') modulus at 5.0 Hz with Tg = 100.1 ° Celsius
G' at 5.0 Hz
10^9
10^8
G' [Pa]'
0
50
100
150
200
Temperature [° Celcius]


Storage (G') modulus at 1.0 Hz with Tg = 97.2 ° Celsius
G' at 1.0 Hz
10^9
10^8
G' [Pa]'
0
50
100
150
200
Temperature [° Celcius]



Question 9 (6 points)

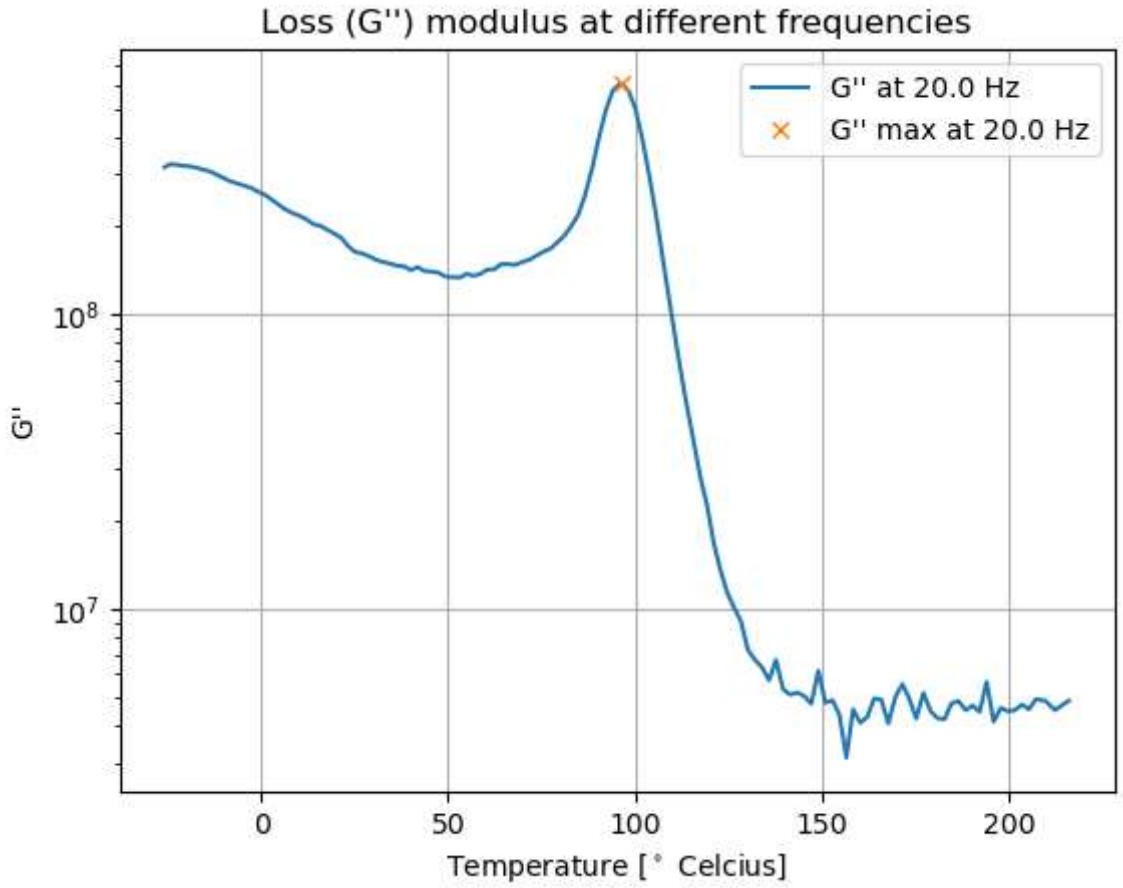
Other methods of finding T_g also exist. One of these methods works by taking the peak of the loss modulus G''. The corresponding temperature is then said to be T_g. Another method works the same as the previously mentioned method, but uses the $tan(\delta)$ function.

Question 9.1: For each frequency dataset create 2 figures below and apply both methods separately. Again, report the found values of T_g and show a visualisation of the method.

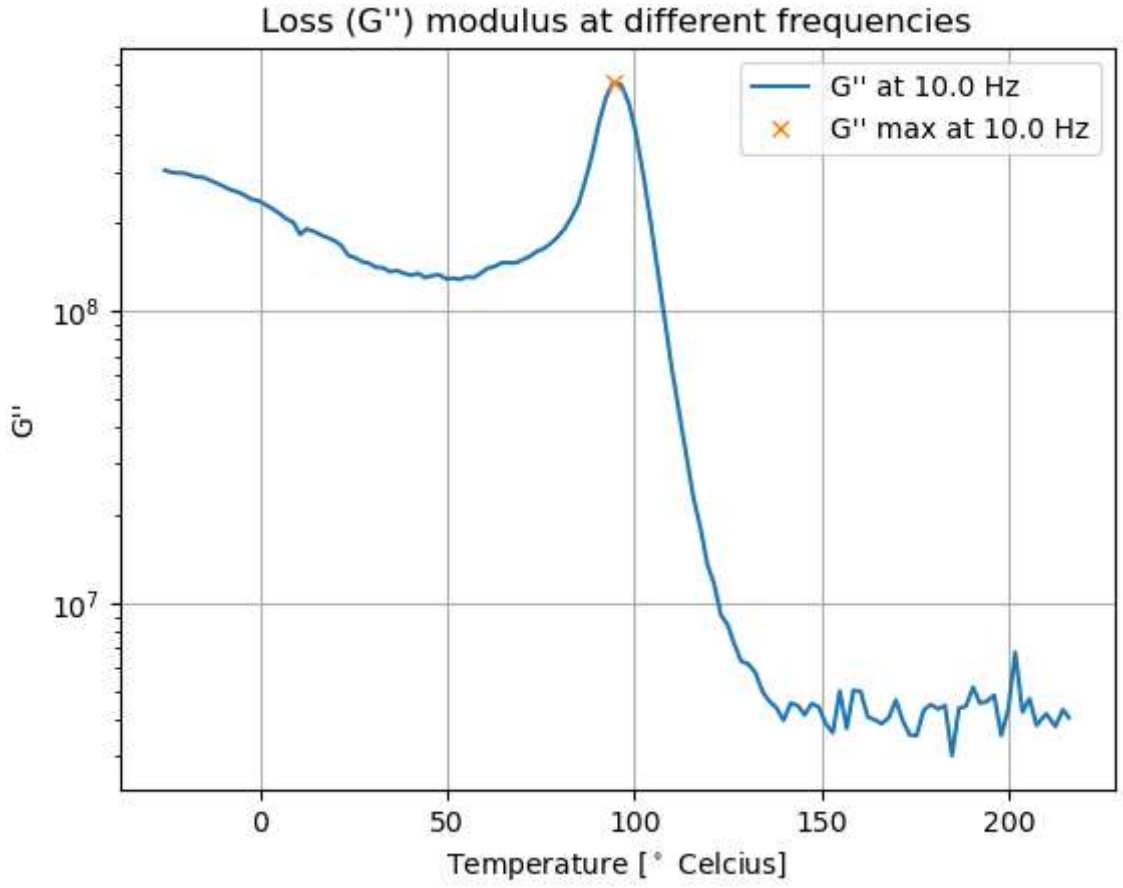
NOTE: make an individual plot for each frequency dataset, therefore ten graphs should be plotted separately

In [7]:
```
Tgmethod3 = np.zeros(len(freqsteps))
for i in range(len(freqsteps)):
    plt.figure()
    plt.plot(Temptot[i,:],Gpptot[i,:],label=r"G'' at "+str(freqsteps[i])+" Hz")
    max_gpp = np.max(Gpptot[i,:])
    idx = np.where(Gpptot == max_gpp)
    print("The found value for Tg at ",freqsteps[i],' Hz is ', Temptot[i,idx][1][0]
    Tgmethod3[i] = Temptot[i,idx][1][0]
    plt.plot(Temptot[i,idx][1][0],max_gpp, 'x',label=r"G'' max at "+str(freqsteps[i
    plt.grid()
    plt.legend()
    plt.xlabel(r"Temperature [$^\circ$ Celcius]")
    plt.ylabel(r"G''")
    plt.yscale('log')
    plt.title(r"Loss (G'') modulus at different frequencies")
    plt.show()
```

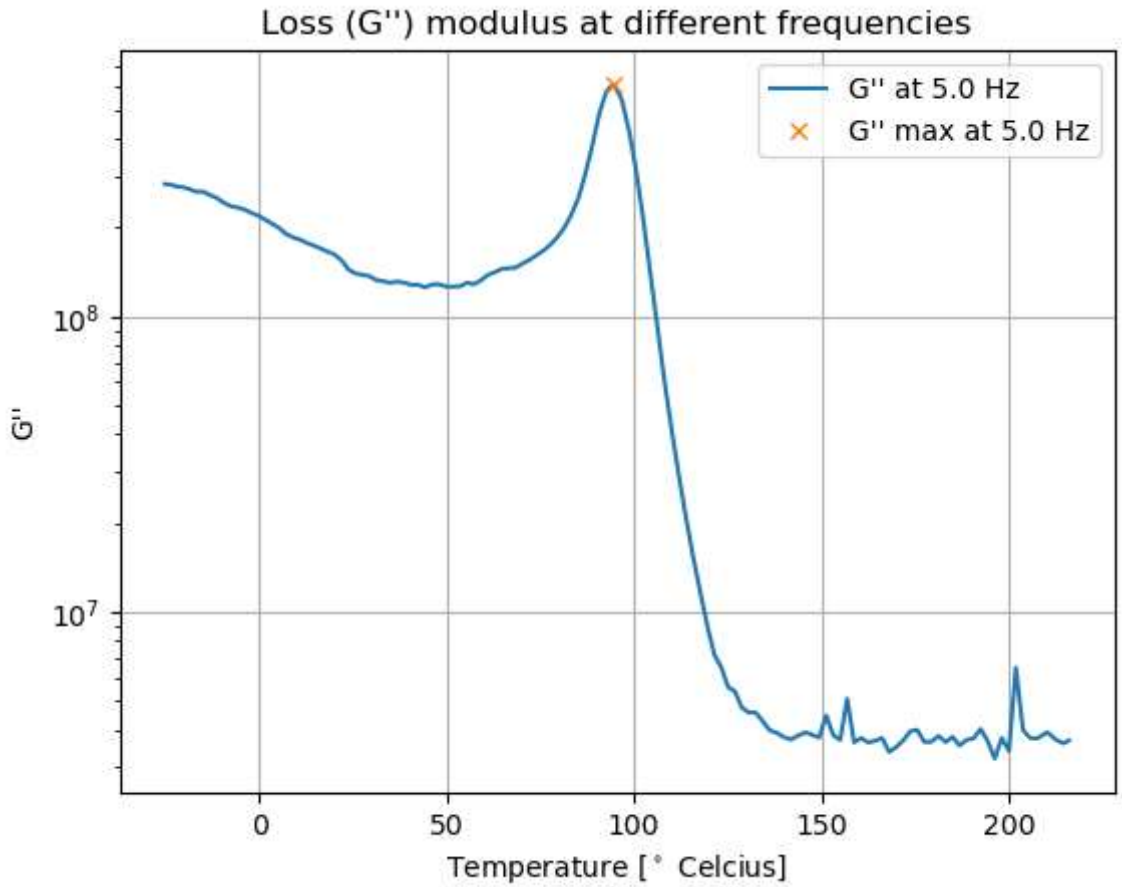
```
The found value for Tg at  20.0  Hz is  96.21512604 degrees Celcius
```



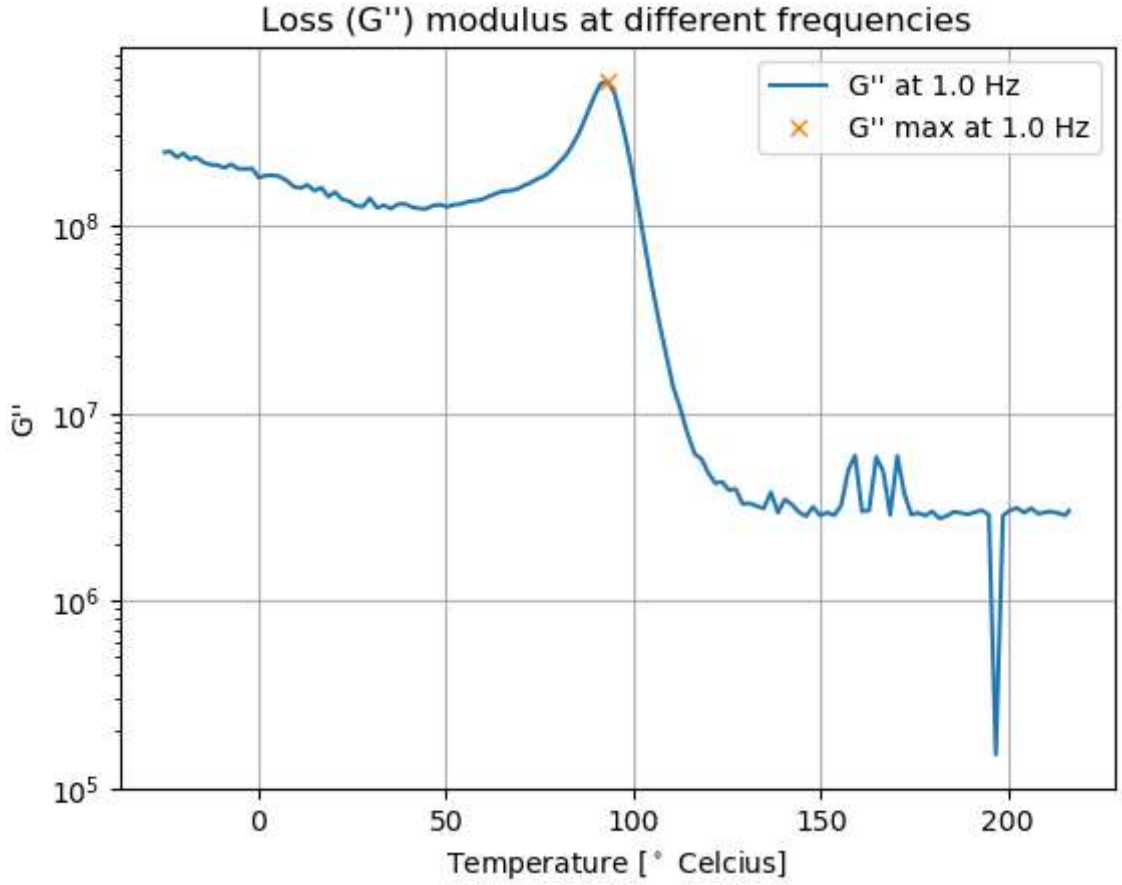

```
The found value for Tg at  10.0  Hz is  94.31813049 degrees Celcius
```



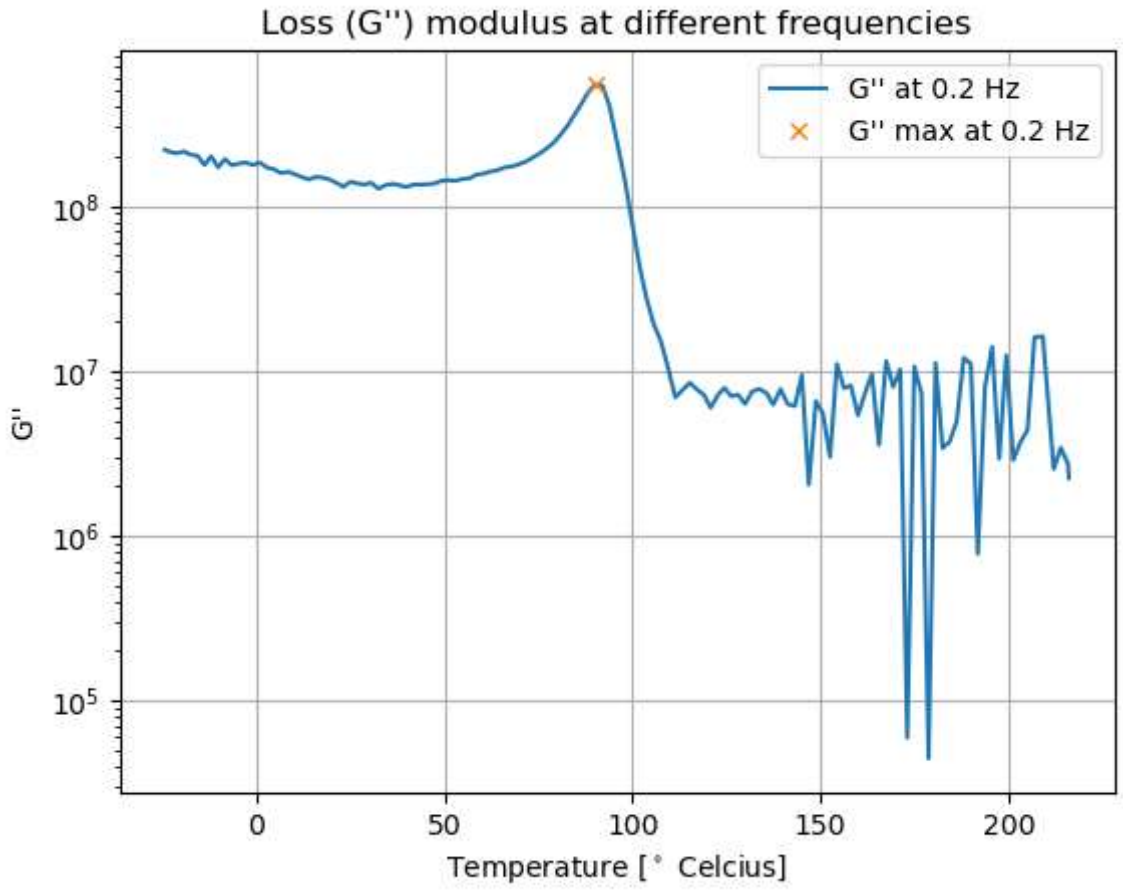

```
The found value for Tg at  5.0  Hz is  94.55767059 degrees Celcius
```




```
The found value for Tg at  1.0  Hz is  93.02075195 degrees Celcius
```

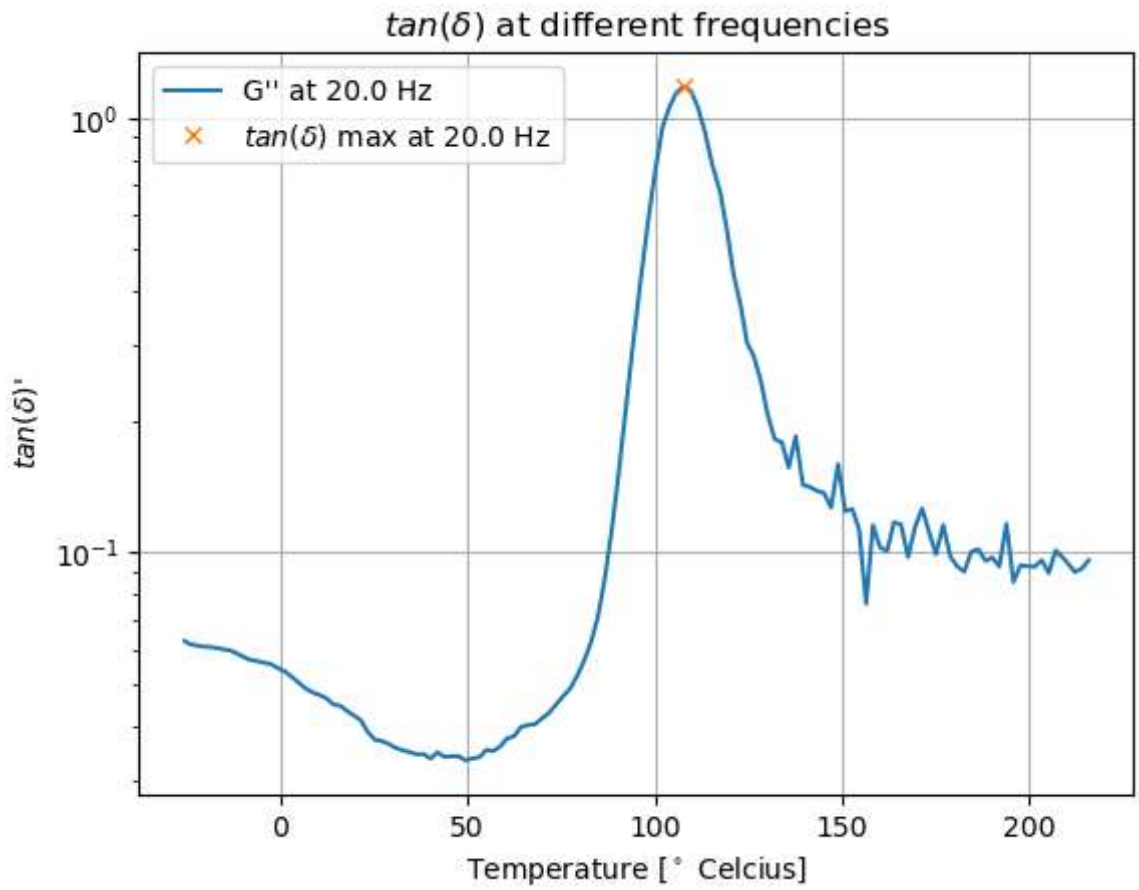



```
The found value for Tg at  0.2  Hz is  90.10639954 degrees Celcius
```

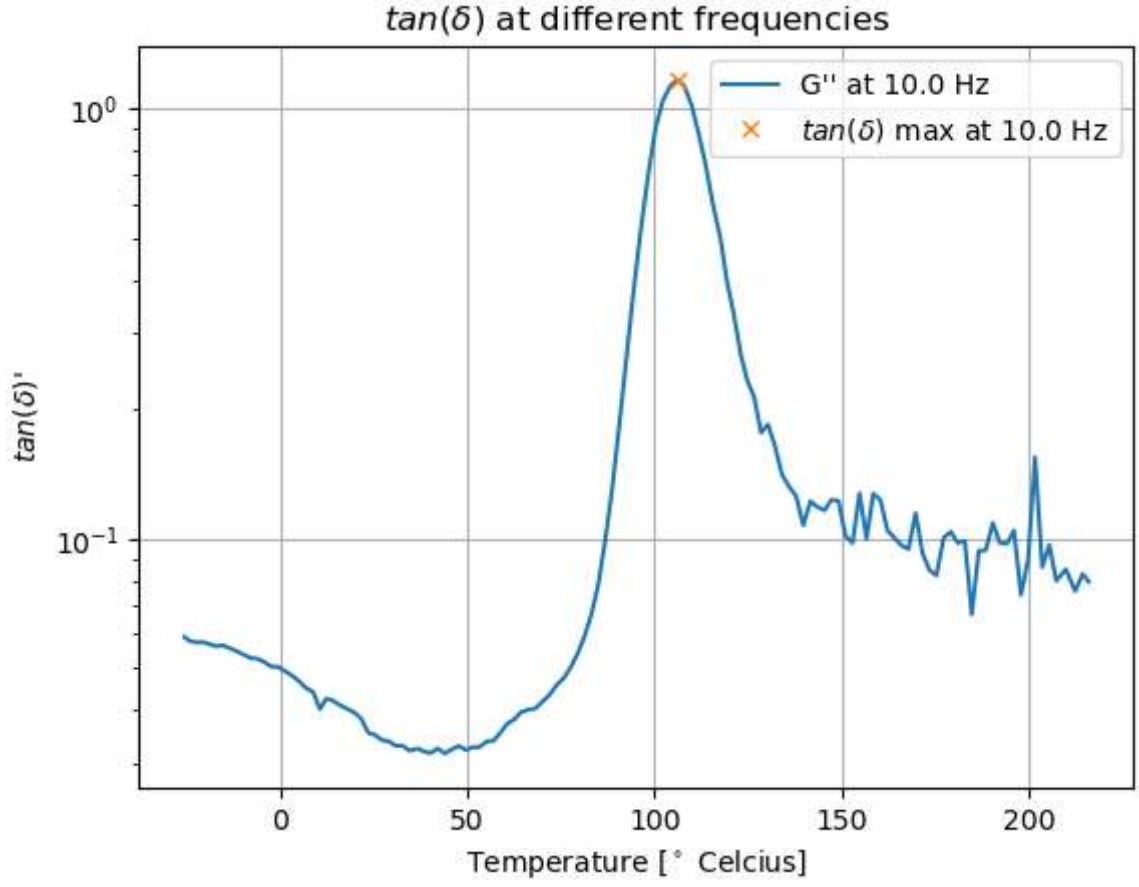


In [8]:
```
Tgmethod4 = np.zeros(len(freqsteps))
for i in range(len(freqsteps)):
    plt.figure()
    plt.plot(Temptot[i,:],tanDeltatot[i,:],label=r"G'' at "+str(freqsteps[i])+" Hz'
    max_td = np.max(tanDeltatot[i,:])
    idx = np.where(tanDeltatot == max_td)
    print("The found value for Tg at ",freqsteps[i],' Hz is ', Temptot[i,idx][1][0]
    Tgmethod4[i] = Temptot[i,idx][1][0]
    plt.plot(Temptot[i,idx][1][0],max_td, 'x',label=r"$tan(\delta)$ max at "+str(fr
    plt.grid()
    plt.legend()
    plt.xlabel(r"Temperature [$^\circ$ Celcius]")
    plt.ylabel(r"$tan(\delta)$'")
    plt.yscale('log')
    plt.title(r"$tan(\delta)$ at different frequencies")
    plt.show()
```

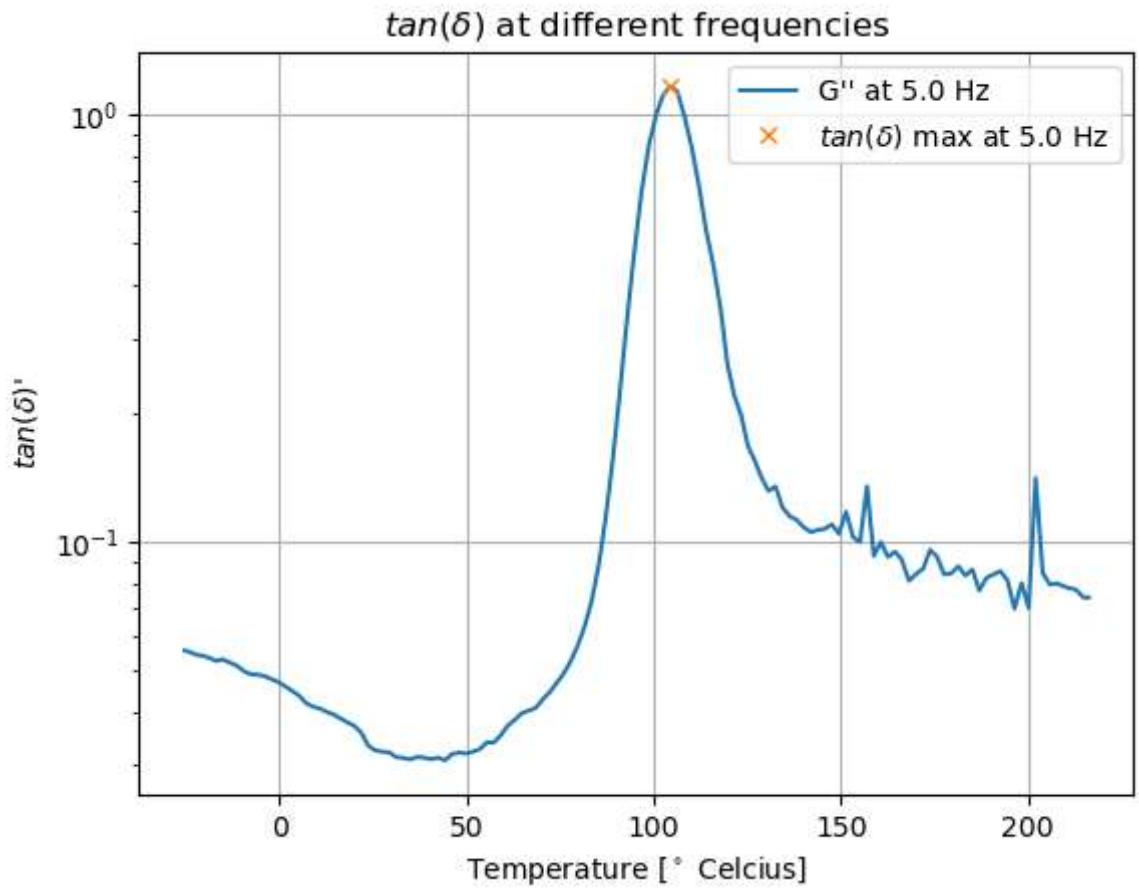
```
The found value for Tg at  20.0  Hz is  107.7480621 degrees Celcius
```



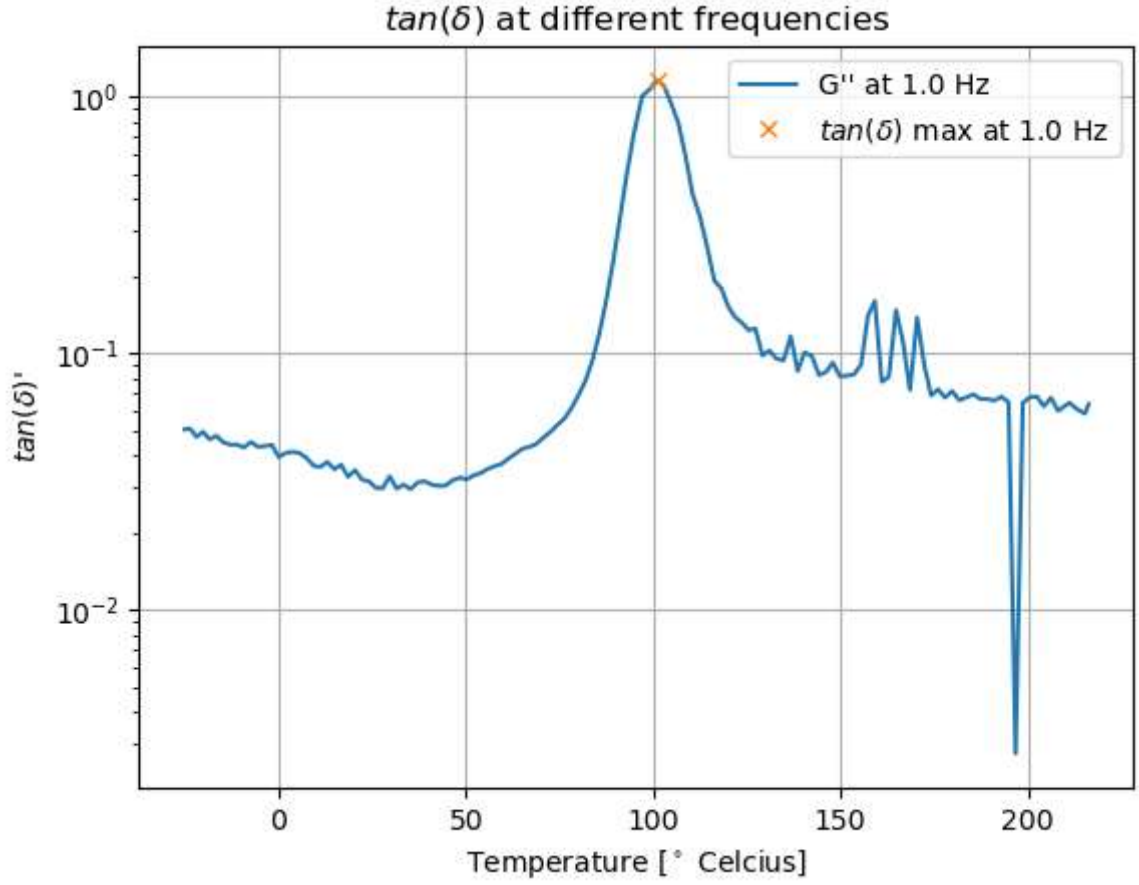

```
The found value for Tg at  10.0  Hz is  106.0559387 degrees Celcius
```



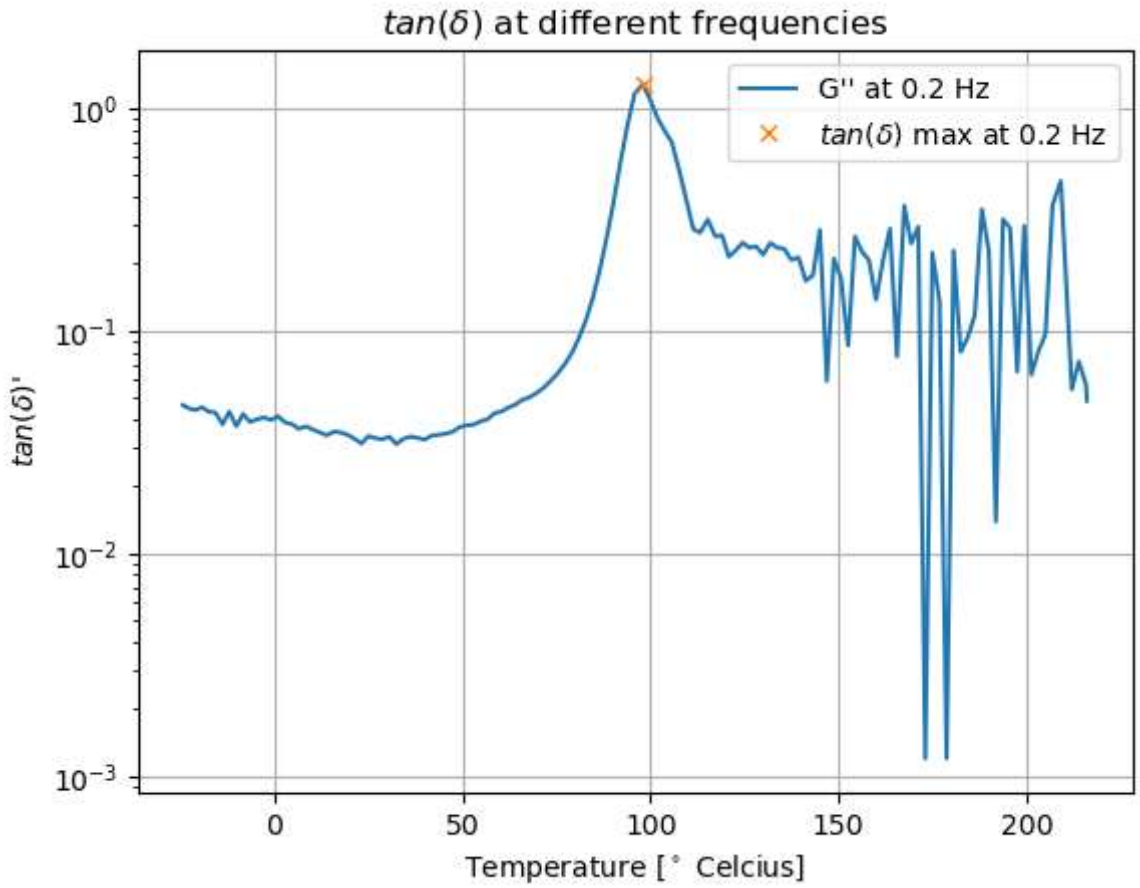

```
The found value for Tg at  5.0  Hz is  104.4466324 degrees Celcius
```




```
The found value for Tg at  1.0  Hz is  101.0025482 degrees Celcius
```



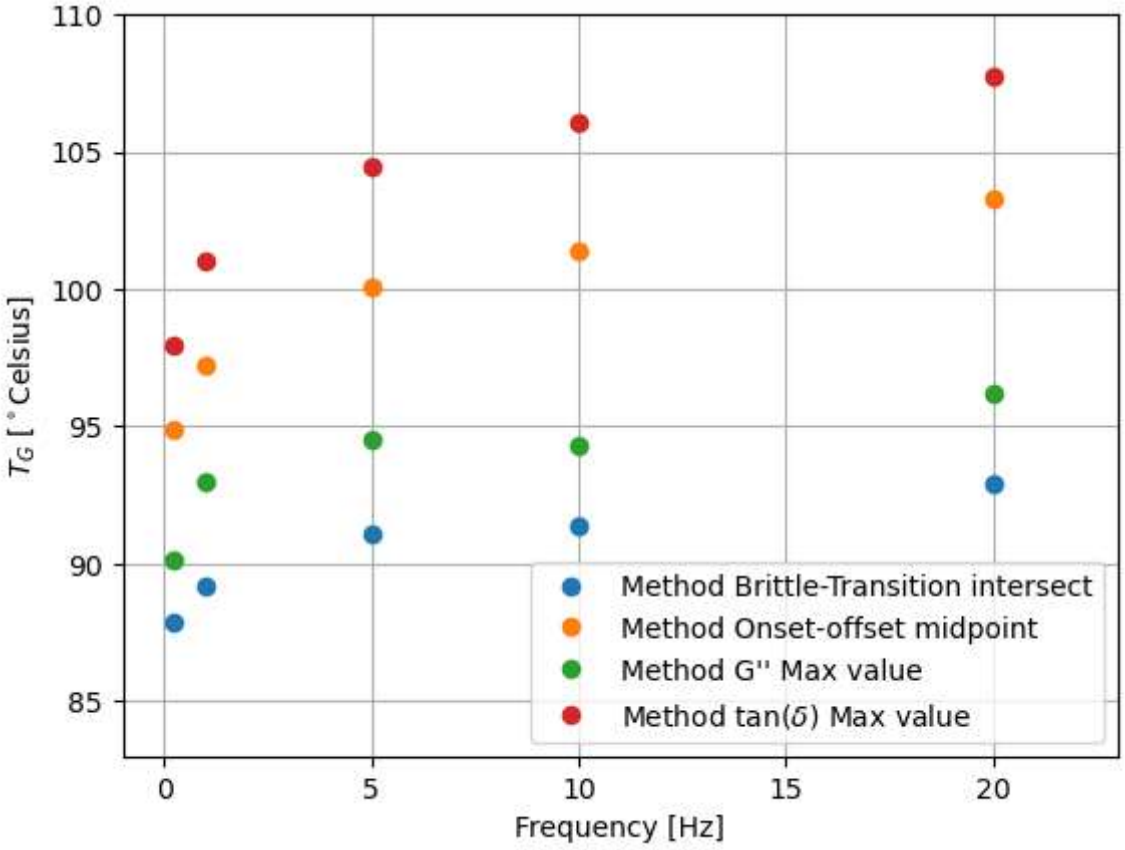

```
The found value for Tg at  0.2  Hz is  97.92134857 degrees Celcius
```



Question 9.2: Summarize the obtained results by generating a single graph plotting the T_g at different frequencies for the following methodologies (as calculated in questions 8.1-8.2 and 9.1): meaning $tan(\delta)$, loss modulus, storage modulus (onset), storage modulus (inflection point).

NOTE: the output should be a single graph, showing four temperature values corresponding to the four methods, with the tested frequencies reported on the x-axis

In [9]:
```
f92 = plt.figure()
plt.plot(freqsteps,Tgmethod1[:,0],'o',label='Method Brittle-Transition intersect')
plt.plot(freqsteps,Tgmethod2[:,0],'o',label='Method Onset-offset midpoint')
plt.plot(freqsteps,Tgmethod3,'o',label=r"Method G'' Max value")
plt.plot(freqsteps,Tgmethod4,'o',label=r"Method $\tan(\delta)$ Max value")
plt.legend(loc=4)
plt.xlabel(r"Frequency [Hz]")
plt.ylabel(r"$T_G$ [$^\circ$Celsius]")
plt.xlim([-1,23])
plt.ylim([83,110])
plt.grid()
plt.show()
```



Now that several important aspects of viscosity of reacting polymers have been discussed, we will take a look at how this material property and other material and process parameters play a role in polymer flow. . In this case we will consider RTM. This will be done with the help of Darcy's law. The most widely used equation for describing flow through RTM molds is Darcy's equation for flow through porous media, displayed in the equation below.

$$Q = \frac{K_{ij}A}{\eta L}\Delta P$$

For which K_{ij} is the permeability tensor of the preform and L is the distance travelled by the resin. An average value of K can be found from the Kozeny-Carman equations for resin flow in a unidirectional fibre network. For further information, please visit Chapter 4.4 "Resin Flow" from Processing of Polymer Matrix Composites by P.K. Mallick.

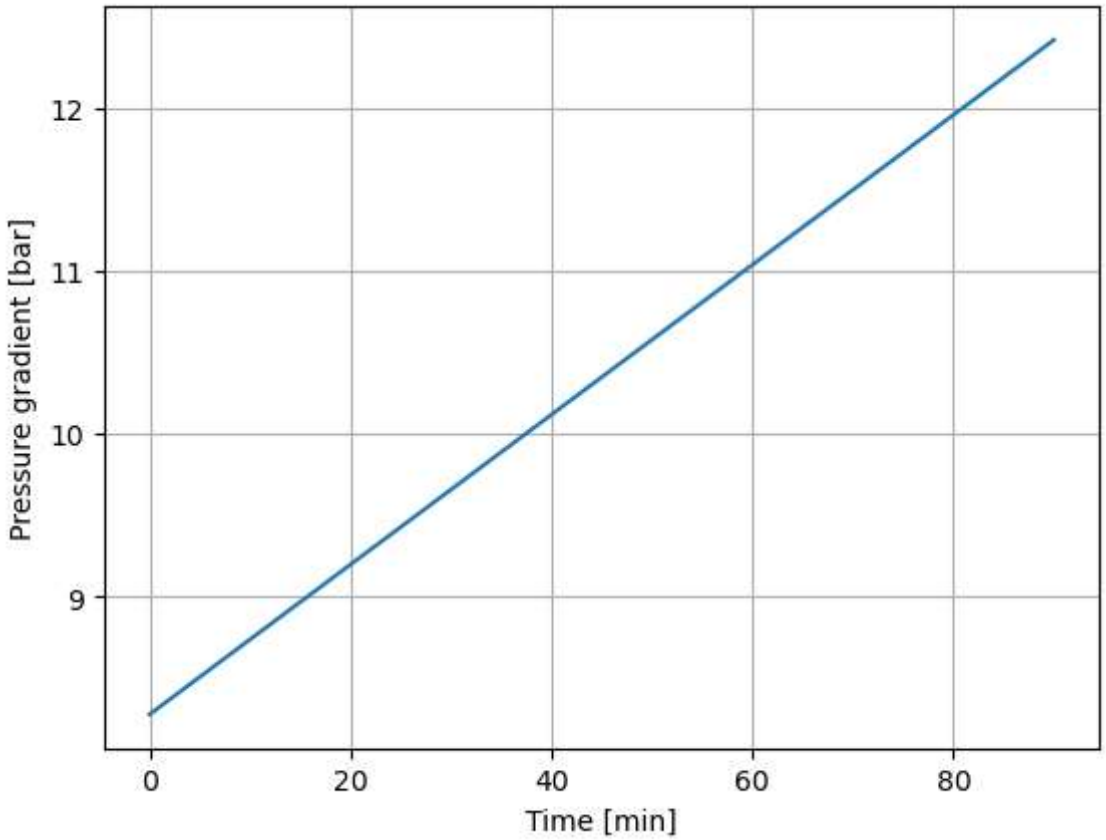
Question 10 (7 points)

Question 10.1: Consider the panel in the picture below. The panel consists of a preform and still needs to be injected with epoxy resin. The fibres are T700 carbon fibres, the fibre volume content can be set to 40% (already at time of injection). The lay-up is [20%@90° / 30%@+-45° / 50%@0°] symmetric and balanced, with the 0 degree direction aligned in the length direction of the panel. The injection and mould temperatures are 120°C. The gelation time of the resin is set to 90 minutes. The viscosity of the resin can be assumed to linearly increase from 0.03 to 0.045 Pa.s over the course of 90minutes, at a temperature of 120°C.

It is required to perform injection at a constant flow rate. To achieve this, please determine the required pressure gradient [Bar] over time [min] and report your results in a graph. Additionally, report the undertaken steps to find a proper value of K (provide sources as well).

In [10]:
```
# Values from Mallik for a square fibre arrangement
C1 = 57
C2 = 0.4
vf_max = 0.785

# Values from question
df = 7e-6  # fibre diameter [m] for T700 carbon fibre (from https://www.toraycma.co
vf = 0.4  # fibre volume fraction

# Equation from Mallik for parallel and perpendicular fibre arrangement
K11 = 2*df**2*(1-vf**3)/(C1*vf**2)
K22 = C2*(np.sqrt(vf_max/vf)-1)**(5/2)*df**2/4

theta1 = 0
theta2 = np.pi/4
theta3 = np.pi/2
# Equation from Mallik for premeability in x-direction. Weighted average of the thr
Kxx = 0.5*((K11+K22)/2+(K11-K22)/2*np.cos(2*theta1))+0.3*((K11+K22)/2+(K11-K22)/2*r
print("Found value for Kxx:",Kxx)

Tinfusion = 90*60
Lpanel = 1

QoverA = Lpanel/Tinfusion
k = Kxx
visc = np.arange(0.03,0.045,(0.045-0.03)/Tinfusion)
dp = np.zeros(len(visc))
time = np.arange(0,Tinfusion)

for i in range(Tinfusion):
    dp[i] = QoverA*visc[i]/k*Lpanel


f1 = plt.figure()
plt.plot(time/60,dp*10**-5)
plt.xlabel('Time [min]')
plt.ylabel(r"Pressure gradient [bar]")
plt.grid()
plt.show()
```

```
Found value for Kxx: 6.712147136705635e-12
```



Answer: The method given in Chapter 4.4 ''Resin Flow'' of Processing of Polymer Matrix Composites by P.K. Mallick was used. This method has semi-emperical equations for permeability in the parallel and perpendicular directions. A number of paramaters are assumed, and these were chosen based on a square fibre arrangement, and found in Mallick. Next, the permeability in the x-direction for each of the three layup angles is calculated, and a weighted average is found based on the fibre layup. Again, the equation comes from Mallick. The fibre diameter was found from https://www.toraycma.com/wp-content/uploads/T700S-Technical-Data-Sheet-1.pdf.pdf, and based on the T7000 fibre.

Question 10.2: First, comment on the required pressure gradients you obtained in the previous question.

Now, consider another infusion of the same preform. The gelation time of the resin is set to 90 minutes at 120°C.

If you were to optimize the manufacturing parameters for this technique, what specific modifications or improvements would you propose? Consider infusion direction, number of resin inlets, any possible steps in the process and which pressure should be applied. Propose two possible approaches and compare them.

Finally, compare your optimized strategies with the results obtained in question 10.1 by visualizing plots for the required pressure gradient [Bar] over time [min]. Provide your reflection about the outcomes in the cells below.

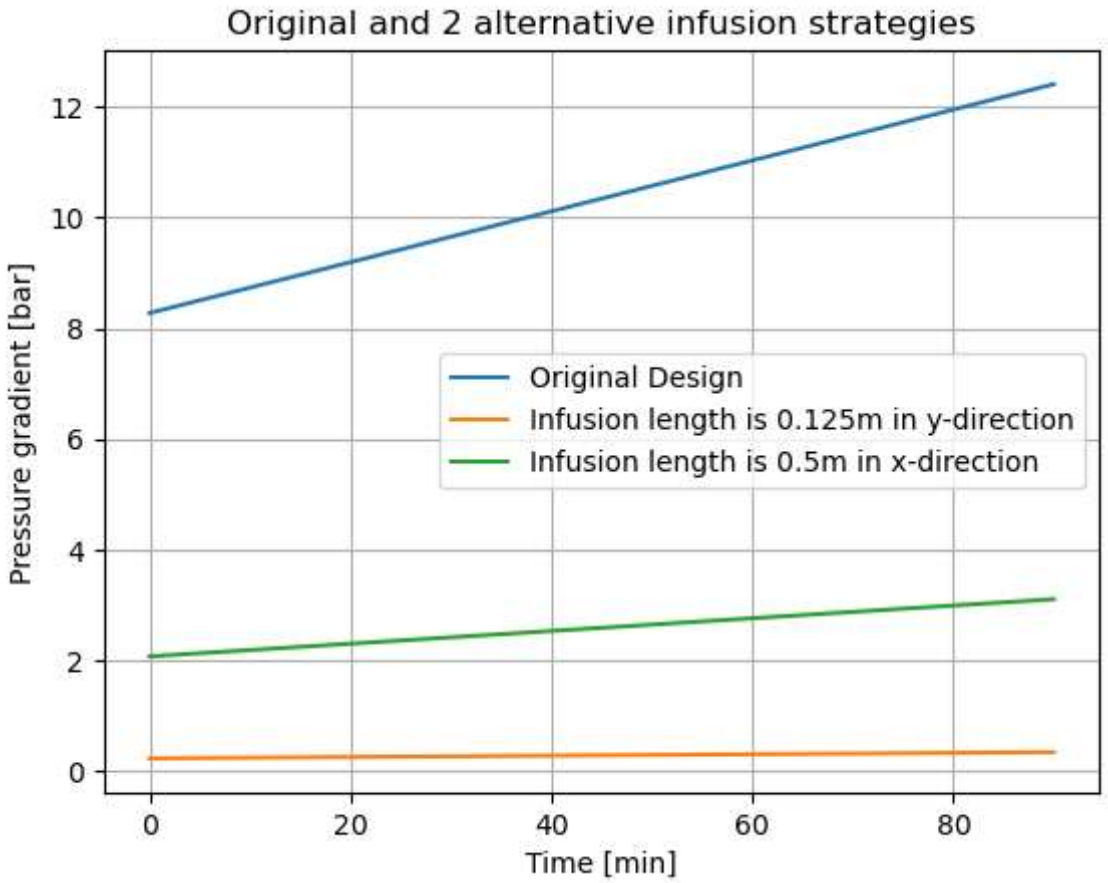
NOTE: If your optimized manufacturing process includes changes in infusion direction, make sure to consider proper assumptions and values for the calculated permeability of the preform.

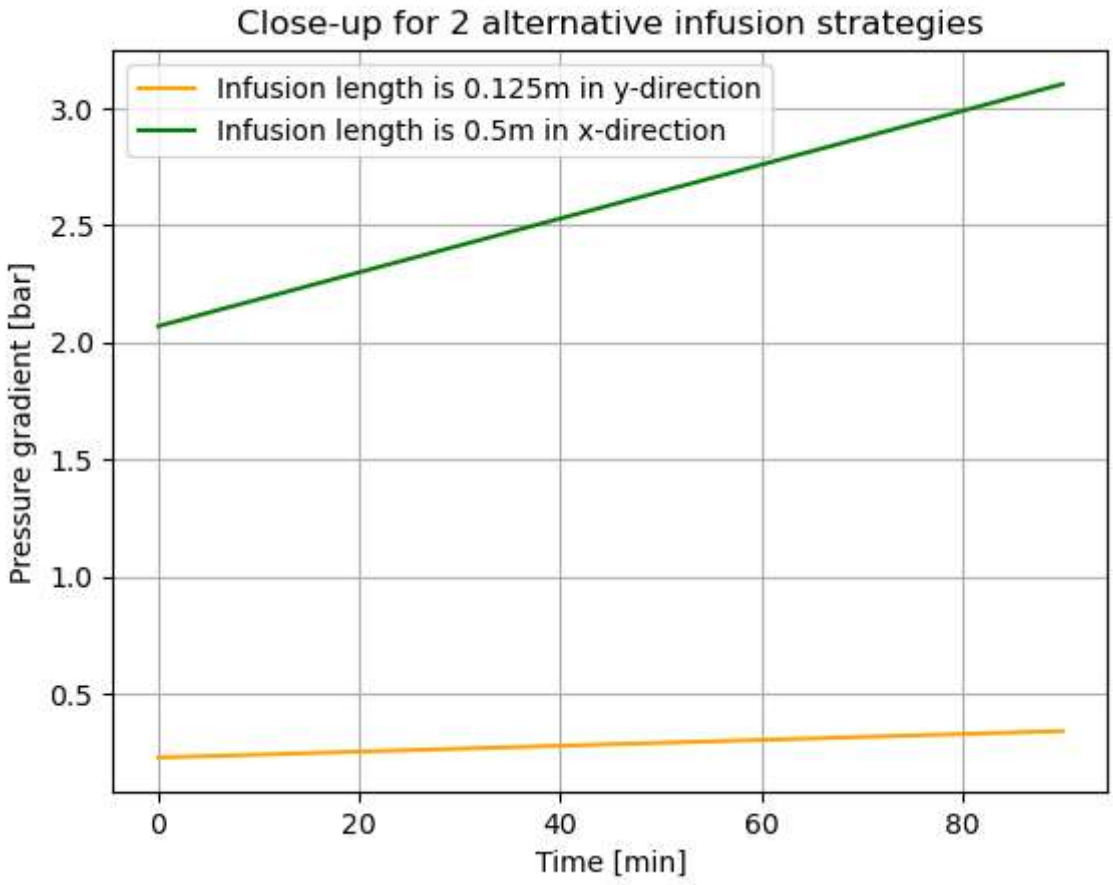
Reflection: The pressure gradient needs to be very high, especially at the end of the infusion. This can be doable with two solid moulds, but would require the moulds to be high quality, and thus expensive.

Optimization strategy: The infusion pressure is relatively high, as the full length of 1 meter needs to be infused before the 90mins have passed. One solution to reduce this is to create a single horizontal infusion line from the center of the panel. This would reduce the infusion length to 0.125m, in y-direction. Another option is to infuse in a single vertical line, creating an infusion length of 0.5m. The pressure gradient required for both strategies should be lower than the original, as both are using a shorter infusion length. The first option does change the permeability of the preform, as the direction of infusion is rotated by 90 degrees.

In [11]:
```
Lpanel2 = 0.5
visc = np.arange(0.03,0.045,(0.045-0.03)/Tinfusion)
QoverA2 = Lpanel2/(Tinfusion)
dp2 = np.zeros(len(visc))
for i in range((Tinfusion)):
    dp2[i] = QoverA2*visc[i]/k*Lpanel2

Kyy = 0.2*((K11+K22)/2+(K11-K22)/2*np.cos(2*theta1))+0.3*((K11+K22)/2+(K11-K22)/2*r
print("Found value for Kyy:",Kyy)
k = Kyy
Lpanel1 = 0.125
QoverA1 = Lpanel1/Tinfusion
dp1 = np.zeros(len(visc))
for i in range(Tinfusion):
    dp1[i] = QoverA1*visc[i]/k*Lpanel1

f1 = plt.figure()
plt.plot(time/60,dp*10**-5,label='Original Design')
plt.plot(time/60,dp1*10**-5,label='Infusion length is 0.125m in y-direction')
plt.plot(time/60,dp2*10**-5,label='Infusion length is 0.5m in x-direction')
plt.xlabel('Time [min]')
plt.ylabel(r"Pressure gradient [bar]")
plt.title('Original and 2 alternative infusion strategies')
plt.legend()
plt.grid()
plt.show()

f2 = plt.figure()
#plt.plot(time/60,dp*10**-5,label='Original Design')
plt.plot(time/60,dp1*10**-5,color='orange',label='Infusion length is 0.125m in y-di
plt.plot(time/60,dp2*10**-5,color='green',label='Infusion length is 0.5m in x-direc
plt.xlabel('Time [min]')
plt.ylabel(r"Pressure gradient [bar]")
plt.legend()
plt.title('Close-up for 2 alternative infusion strategies')
plt.grid()
plt.show()
```

```
Found value for Kyy: 3.844363479445805e-12
```





Reflection: As was expected, both strategies have a significantly smaller pressure gradient needed throughout the infusion time compared to the initial infusion strategy. The permeability of the preform in the y-direction is lower than the permeability in x-direction, roughly by a factor 2. This difference stems from the non-uniform volume fractions for fibres in the 0 and 90 degree orientation (50% vs 20%). The contribution of the 45 degree plies to

the permeability is the same for both infusion directions, as this orientation doesn't change when infusing in either the x or y direction.

Eventhough the permeability in y-direction is lower, the difference in flow length more than make up for this, and this infusion strategy of infusion along the y-direction results in the lowest pressure gradient required. It is less than 0.5 bar pressure difference, so one could consider just using a vacuum pump and ambient atmospheric pressure to obtain the required pressure difference to infuse the part within the 90 minutes. One can also look at reducing the infusion time, which could lead to chosing a different resin system with a shorter gel time, but potential better final properties.

Question 11 (4 points)

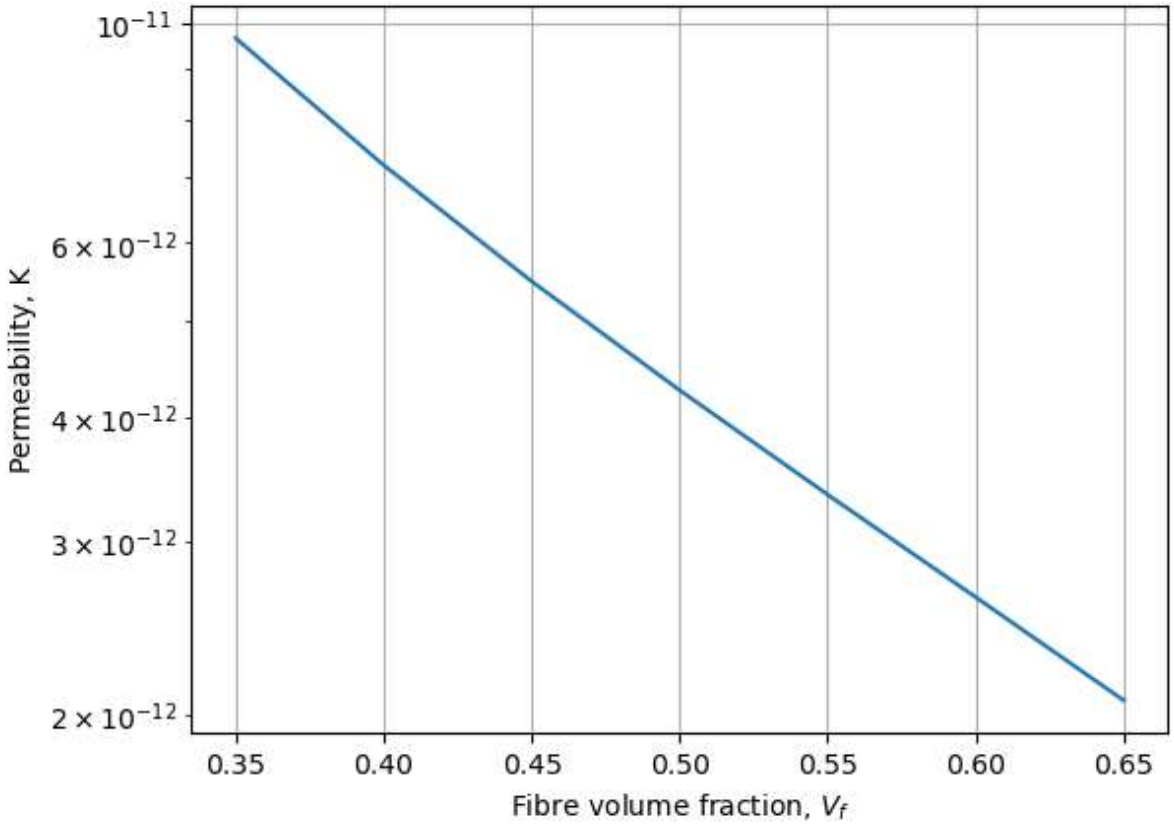
As you know, fibre volume content V_f plays an important role in the permeability of a preform and therefore influences the required injection pressure. Now, assume a constant viscosity of 0.03 Pa.s and a distance of 1 meter for the resin to flow. Determine the required pressure to inject over this distance in 40 minutes, for different fibre volume fractions from 0.35 to 0.65 with steps of 0.05. Note: the laminate is the same as described in the previous question (Q10).

- Create a graph showing K (on a logarithmic scale) vs. V_f (x-axis)

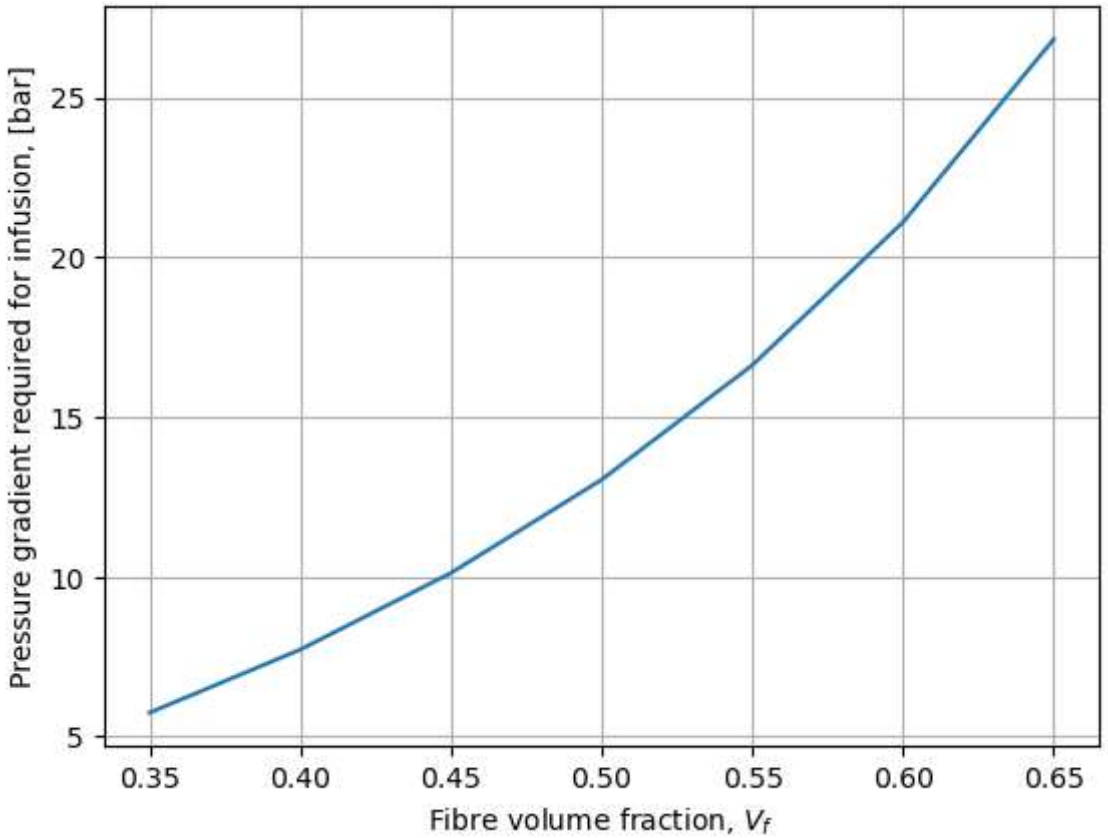
- Next, report your results by plotting the required ΔP for infusion per fibre volume fraction V_f

- Finally, comment on your results and critically analyse the obtained values

In [12]:
```
vf = np.arange(0.35, 0.65, 0.05)  # fibre volume fraction
visc = 0.03  # [Pa*s]
Lpanel = 1  # [m]
t_infusion = 40*60 # [s]

# Equation from Mallik for parallel and perpendicular fibre arrangement
K11 = 2*df**2*(1-vf**3)/(C1*vf**2)
K22 = C2*(np.sqrt(vf_max/vf)-1)**(5/2)*df**2/4

theta1 = 0
theta2 = np.pi/2
theta3 = np.pi
# Equation from Mallik for premeability in x-direction. Weighted average of the thr
Kxx = 0.5*((K11+K22)/2+(K11-K22)/2*np.cos(2*theta1))+0.3*((K11+K22)/2+(K11-K22)/2*r

plt.plot(vf,Kxx)
plt.xlabel('Fibre volume fraction, $V_f$')
plt.ylabel(r"Permeability, K")
plt.yscale('log')
plt.grid()
plt.show()
```



In [13]:
```
QoverA = Lpanel/Tinfusion
dp = QoverA*visc/Kxx*Lpanel
plt.plot(vf,dp*1e-5)
plt.xlabel('Fibre volume fraction, $V_f$')
plt.ylabel(r"Pressure gradient required for infusion, [bar]")
plt.grid()
plt.show()
```



Comment: It makes sense that as the fibre volume fraction increases (tere are more fibres in the same volume), the permeability decreases. This is because there is more resistance to the flow arounf the larger number of fibres.

Additionally, it also makes sense that with the fibre volume fraction increasing, the pressure required also incrceases. A more dense fibre (with lower permeability) will require more force to push a resin through.

Bonus question (3 points)

In this notebook you focused on how DMA measurements can be used to evaluate the mechanical and physical behaviour of polymers while curing and their glass transition temperatures. In this section, DMA is used to evaluate thermal cycling effects on CFRP.

In the coming decades carbon fiber-reinforced polymer materials will be widely employed in manufacturing parts suitable for space applications, due to their combination of optimal mechanical properties and low density. When exposed to the space environment, operational conditions cause the composites to undergo cyclic temperature changes, between -196°C and 180°C.

Through experimental simulations, the effect of thermal cycles has been studied.

Which chemical and physical effects of thermal cycling on composites would you expect? Focus specifically on the variations you would expect to see through DMA testing.

Answer: DMA testing gives an insight into the ophysical properties of polymers; specifically the storage and loss modulus, as well as the glass transition temperature. These are important properties, as they define what the material is capable of, and give insight into the chemical structure of the ploymer. Thermal cycling can have many effects on materials. For polymwers specifically it is important to note whether the thermal cycling exceeds certain boundaries, ie. the glass transition temperature (Tg). This is the temperature at which the amorphus part of the polymer starts to become more fluid, as the weaker bonds are breaking.
If the temperature goes above this Tg during the thermal cycling, the properties of the material change. The material temporarily loses part of its storage modulus, as the loss modulus increases. This is not good for a spacecraft where constant material properties are required.

In some cases, the material will revert to its original state once colled back below the glass transition temperature. However, while in the heated state, a number of things can happen to prevent the return to the original properties. Firstly, if the temperature is above the glass transition temperature, but below the melting point, it may be ideal for the formation of crystals. Increasing the cyrstalinity prolongs the horizontal section of the storage modulus, meaning the material properties remain longer at higher temperatures. The opposite can happen if the temperature increases to above or near the melting point; at this temperature all bonds within crystals also break. Leaving a material with no crystalinity. Rapid cooling from this temperature means the crystals have no chance to form, leaving a material with

lower physical properties than the original. The storage modulus curve will have become steeper at a lower temperature.