ar structure and reaction theory. *Eur. Phys. J. Plus* **133**, 385 (2018).

49. Meissner, C. Nuclear Fusion through a Computational Microscope. *Science & Technology Review* (2014).

50. Shirokov, A. M. *et al.* Prediction for a four-neutron resonance. *Phys. Rev. Lett.* **117**, 182502 (2016).

51. Brown, D. A. *et al.* ENDF/B-VIII.0: The 8th Major Release of the Nuclear Reaction Data Library with CIELO-project Cross Sections, New Standards and Thermal Scattering Data. *Nucl. Data Sheets* **148**, 1–142 (2018).

52. Garçon, M. & Van Orden, J. W. The Deuteron: Structure and Form Factors. in *Advanc- es in Nuclear Physics* (eds. Negele, J. W. & Vogt, E. W.) 293–378 (Springer US, 2001).

53. Schwinger, J. Nuclear Energy in an Atomic Lattice - Causal Order. *Progr. Theoret. Phys.* **85**, 711–712 (1991).

54. Teller, E. *NSF/EPRI workshop on anomalous effects in deuterated materials*. (16-18 Oct 1989).

55. Šibalić, N., Pritchard, J. D., Adams, C. S. & Weatherill, K. J. ARC: An open-source library for calculating properties of alkali Rydberg atoms. *Comput. Phys. Commun.* **220**, 319–331 (2017).

56. Barredo, D., Lienhard, V., de Léséleuc, S., Lahaye, T. & Browaeys, A. Synthetic three-di- mensional atomic structures assembled atom by atom. *Nature* **561**, 79–82 (2018).

57. Briggs, J. S. & Eisfeld, A. Equivalence of quantum and classical coherence in elec- tronic energy transfer. *Physical Review E* **83**, 051911 (2011).

58. Van Amerongen, H. & Van Grondelle, R. *Photosynthetic excitons*. (World Scientific, 2000).

59. Alder, K. & Steffen, R. M. Electromagnetic moments of excited nuclear states. *Annu. Rev. Nucl. Sci.* **14**, 403–482 (1964).

60. Wong, I., Grigoriu, A., Roslund, J., Ho, T.-S. & Rabitz, H. Laser-driven direct quantum control of nuclear excitations. *Phys. Rev. A* **84**, 053429 (2011).

61. von der Wense, L. *et al.* The theory of direct laser excitation of nuclear transitions. *Eur. Phys. J. A* **56**, 176 (07/2020).

62. Pálffy, A., Evers, J. & Keitel, C. H. Electric-dipole-forbidden nuclear transitions driven by super-intense laser fields. *Phys. Rev. C Nucl. Phys.* **77**, 044602 (2008).

63. Gunst, J., Wu, Y., Kumar, N., Keitel, C. H. & Pálffy, A. Direct and secondary nuclear excitation with x-ray free-electron lasers. *Phys. Plasmas* **22**, 112706 (2015).

64. Hagelstein, P. L. & Chaudhary, I. U. Phonon models for anomalies in condensed matter nuclear science. *Curr. Sci.* 507–513 (2015).

65. Metzler, F. Experiments to Investigate Phonon-Nuclear Interactions. (Massachu- setts Institute of Technology, 2019).

66. Bocklage, L. *et al.* Coherent control of collective nuclear quantum states via tran- sient magnons. *Science Advances* **7**, eabc3991 (2021).

67. Hagelstein, P. L. Phonon-mediated Nuclear Excitation Transfer. *J. Cond. Mat. Nucl. Sci* **27**, 97–142 (2018).

68. Mitra, A. N. & Narasimham, V. L. Role of spin-orbit force in nuclear interactions. *Phys. Rev. C Nucl. Phys.*

14, 407–428 (1960).

69. Breit, G. Approximately relativistic equations for nuclear particles. *Physical Review* **51**, 248 (1937).

70. Elliott, J. P. & Skyrme, T. H. R. Centre-of-mass effects in the nuclear shell-model. *Proc. R. Soc. Lond. A Math. Phys. Sci.* **232**, 561–566 (1955).

71. Kurasawa, H. & Suzuki, T. Relativistic vs. Non-relativistic Nuclear Models. *arXiv:nu-cl-th/0201035* (2002).

72. Machleidt, R. Nuclear Forces. *Scholarpedia J.* **9**, 30710 (2014).

73. Goeppert Mayer, M. Nuclear Configurations in the Spin-Orbit Coupling Model. I. Em-pirical Evidence. *Phys. Rev.* **78**, 16–21 (1950).

74. Li, X., Wu, Z. & Liu, J. Rashba spin-orbit coupling in graphene monolayer coated byperiodic magnetic stripes. *Sci. Rep.* 7, (2017).

75. Negele, J. W. Relativistic nuclear models: reason or treason? *Comments Nucl. Part. Phys.* **14**, 303–319 (1985).

76. Hagelstein, P. L. Calculation of the Boosted Spin – orbit Contribution to the Phonon – Nuclear Coupling Matrix Element for 181 Ta. *J. Cond. Mat. Nucl. Sci* **29**, 392–400 (2018).

77. DeLosh, R. G. & Grant, W. J. C. Electronic Energy Transfer via Virtual-Phonon Pro-cesses. *Phys. Rev. B Condens. Matter* **1**, 1754–1765 (1970).

78. Andrews, D. L. *Resonance energy transfer: theoretical foundations and developing applications.* (SPIE Press: Washington, 2009).

79. Hore, P. J. & Mouritsen, H. The Radical-Pair Mechanism of Magnetoreception. *Annu. Rev. Biophys.* **45**, 299–344 (2016).

80. Heeg, K. P. *et al.* Coherent X-ray–optical control of nuclear excitons. *Nature* **590**, 401–404 (2021).

81. Schwinger, J. Nuclear energy in an atomic lattice. *Z. Phys. D: At. Mol. Clusters* **15**, 221–225 (1990).

82. Preparata, G. *QED Coherence in matter.* (World Scientific, 1995).

83. Hagelstein, P. L. Unified phonon-coupled su(n) models for anomalies in metal deu-terides. in *Condensed Matter Nuclear Science* 837–869 (WORLD SCIENTIFIC, 2005).

84. Hagelstein, P. L. Models coupling based on phonon-nuclear. *Cold Fusion: Advances*

in Condensed Matter Nuclear Science 283 (2020).

85. Schwinger, J. Cold Fusion Theory–A Brief History of Mine. *Fusion Technol.* **26**, XIII (1994).

86. Chumakov, A. I. *et al.* Superradiance of an ensemble of nuclei excited by a free elec-tron laser. *Nat. Phys.* **14**, 261–264 (2018).

87. Haber, J. *et al.* Rabi oscillations of X-ray radiation between two nuclear ensembles. *Nat. Photonics* **11**, 720–725 (2017).

88. Dicke, R. H. Theory of superradiance. *Physical Review* **93**, 99–110 (1954).

89. Kaur, M., Kaur, P., Sahoo, B. K. & Arora, B. Dynamics of resonant energy transfer in one-dimensional chain of Rydberg atoms. *Eur. Phys. J. D* **72**, 150 (2018).

90. Lim, J., Tame, M., Yee, K. H., Lee, J.-S. & Lee, J. Phonon-induced dynamic resonance energy transfer. *New J. Phys.* **16**, 053018 (2014).

91. Oliveira, H. M. & Melo, L. V. Huygens synchronization of two clocks. *Sci. Rep.* **5**, 11548 (2015).

92. Hagelstein, P. L. & Chaudhary, I. U. Central and tensor contributions to the phonon-ex-change matrix element for the D2/4He transition. *J. Cond. Mat. Nucl. Sci* 15 (2013).

93. de la Mora, M. B., Amelines-Sarria, O., Monroy, B. M., Hernández-Pérez, C. D. & Lugo, J. E. Materials for downconversion in solar cells: Perspectives and challenges. *Sol. Energy Mater. Sol. Cells* **165**, 59–71 (2017).

94. Hagelstein, P. L. A unified model for anomalies in metal deuterides. in *Conference Proceedings - Italian Physical Society* vol. 70 363–368 (Editrice Compositori; 1999, 2000).

95. Nave R. HyperPhysics. Georgia State University, Department of Physics and Astron-omy (2000).

From the text above follows the usefulness and the importance of modeling materials systems in such ways where chemical and nuclear properties of constituting nuclei are both included in the modeling process in order to arrive at systems that are best suited for a range of desired applications. In other words, in such a process, modeling and design techniques that traditionally consider chemical properties of atoms and modeling techniques that traditionally consider nuclear properties of atoms (related to nuclear excited states) are combined.
Such applications can include the generation of heat or electricity or radiation such as photons from nuclear reactions – and specifically from nuclear reactions where reaction rates are enhanced beyond presently conventionally accepted rates; or where reaction products are changed beyond presently conventionally accepted products.
It can also involve optimizing across a number of different desired output parameters or combinations thereof, including but not limited to: maximization of output energy, minimization of cost, reduction of hazardousness, minimization of weight.
Specifically, the modeling and design process and resulting software and tools comprise:

- chemical data including but not limited to: behavior in a lattice, lattice structure, electron density
- nuclear data including but not limited to: nuclear excited states, nuclear reaction resonances and cross section data, photonuclear cross sections

The modeling and design process and resulting software and tools can include:

- modeling the proximity and the electron screening configurations possible in different lattice configurations, including finding a desired combination between proximity and screening
- modeling resulting nuclear reactions and reaction products (and if applicable excitation transfer chains) given different material and lattice configurations
- modeling coupling strengths between nuclei in different lattice configurations including phonon-nuclear coupling strengths (that enable nuclear excitation transfer), as affected by phonon modes and phonon energies in the lattice, possible superradiant amplification, phonon coherence domains and lifetimes, and the nuclear transitions of involved (coupled) nuclei.

Claims

1. Claimed is a process for modeling or designing lattice configurations, which involves chemical and nuclear data of constituting atoms and which results in a system that involves a lattice whose configuration is informed by such models or designs.
2. The claim of 1 where nuclear excitation transfer processes are modeled or designed, and their key parameters are determined (as described in the text above).
3. The claim of 2 where the resulting system's desired output and performance parameters of are optimized according to the designer's preferences.
4. The claim of 1 where enhancement of nuclear reaction rates beyond the presently published and widely accepted rates are achieved in the system.
5. The claim of 4 where the nuclear reactions comprise one of nuclear fusion, nuclear fission, or nuclear decay.
6. The claim of 1 where in the resulting system a change of conventionally expected nuclear reaction products is achieved.

What is described and claimed is a system…

…where two or more fusion rate enhancement mechanisms are combined i.e. simultaneously applied…

…to reach an enhancement of nuclear reaction rates of 40 orders of magnitude or more.
…to reach an enhancement of fusion rates of 40 orders of magnitude or more.
…to reach an enhancement of D+D fusion rates of 40 orders of magnitude or more.
…to reach an enhancement of fusion rates of 50 orders of magnitude or more.
…to reach an enhancement of D+D fusion rates of 50 orders of magnitude or more.

In order to combine such mechanisms a lattice needs to be designed and created…

…that allows for high screening (preferably >100 eV) and close proximity of nuclei (preferably <100 pm) as well a high-energy phonon modes (preferably >1 THz), large phonon coherence domains (preferably >5 nm coherence length), and resonant or close-to-resonant nuclear states between nuclear reactants (donor subsystems) and receiver nuclei (receive subsystems) within coupled systems of nuclei (nuclear ensembles).

The mentioned fusion rate enhancement mechanisms can comprise

- electron screening
- increased proximity through molecule formation or isomerization
- nuclear reaction rate enhancement through coherent energy transfer (equivalent to accelerated deexcitation)

The above-mentioned coherent energy transfer can be mediated by phonons or photons via phonon-nuclear interactions or photon-nuclear interactions.

What is described and claimed is the design of materials…

…where the modeling of nonradiative energy transfer pathways between nuclei in a solid-state environment informs the materials composition of such an environment, with respect to binary, ternary or multi-component composition whereas such materials are then created accordingly by doping, alloying or other known methods of creating multi-component materials of desired specifications.

…where such modeling also takes into account the maximization of proximity to the extent that such maximization does not lessen the overall performance of the system (e.g. due to weakening the effects of other mechanisms considered).

…where such modeling also takes into account the maximization of screening to the extent that such maximization does not lessen the overall performance of the system (e.g. due to weakening the effects of other mechanisms considered).

…where such modeling also takes into account the maximization of phonon energy to the extent that such maximization does not lessen the overall performance of the system (e.g. due to weakening the effects of other mechanisms considered).

…where such modeling also takes into account the maximization of the phonon coherence domain to the extent that such maximization does not lessen the overall performance of the system (e.g. due to weakening the effects of other mechanisms considered).

…whereas such modeling also takes into account the potential reduction in overall output performance due to the clogging up of reaction sites with reaction products and rewards structures that minimize such clogging up of reaction sites (e.g. nanoparticles) to the extent that such minimization does not lessen the overall performance of the system (e.g. due to weakening the effects of other mechanisms considered).

For further details as to modeling principles, system dynamics, and different modes of implementation, see the attached codes where the process of nuclear transition rate enhancement and fusion rate enhancement as well as changes in nuclear reaction products through secondary reactions are illustrated.

What is described and claimed is a system and method for the design of energy transfer and conversion pathways in solid-state materials, as is beneficial for energy and information processing and conversion applications.

The focus is on the de-excitation and excitation times of (nuclear) excited state transitions -- including metastable excited states as described below -- and on modifications to their de-excitation and excitation times due to coherent couplings to nearby subsystems.

Such modifications -- and the extent/magnitude to which they manifest -- depend on multiple variables and internal relations in solid-state material systems. Consequently, an Edisonian approach does not lead to optimal outcomes if, for instance, a particular transition is to be accelerated in an optimal way (e.g. maximizing certain desired output metrics) within given constraints.

The system and method described here allows to design such systems in a systematic way by -- for the first time -- combining atomic level lattice modeling with the modeling of nuclear states via the interactions and dynamics that take place across those levels.

Nuclear transitions here describe any change from one nuclear state to another state in a nuclear system (whereas a nuclear system is an aggregation of nucleons). Such state changes are typically associated with a rearrangement of nucleons. Individual states can be either short-lived or long-lived (unstable, metastable or highly metastable). Note that Julian Schwinger described the nuclear fusion reaction D2 --> He-4 as a nuclear transition from the metastable D2 state (a clustered four-nucleon configuration) to the more stable He-4 state (a more compact and more stable four-nucleon configuration). The same logic applies to other rearrangements of nucleons. Therefore, the term nuclear transition is to be seen in this broader light.

The system and method comprises one computer or several connected computers **a** that can carry out processing as well as store information (here simply referred to as "the computer").

The memory of the computer holds codes for crystal structure prediction and phonon mode calculation (such as Quantum Espresso or similar) as well as codes for the calculation or look-up of nuclear excited state levels (such as NuShellX or similar) as well as nuclear data on excited states, recorded cross sections (including photonuclear cross sections), and decay pathways/channels (such as available in the form of ENDF files and similar files from well-known sources of nuclear data) as well as codes to calculate phonon-nuclear coupling matrix elements between nuclear states mediated by shared oscillators (such as boosted spin-orbit coupling calculations or calculations of central and tensor contributions to the phonon-nuclear coupling matrix element). Alternatively, if coherent photons are used to create and change couplings between nuclei, codes to calculate or look up photon-nuclear coupling matrix elements are used.

This system and method is then used to relate a set of input variables, that can be modified, to output variables of the to-be-designed-system (TBDS) and determine the optimal choice of input variable values for reaching the parameter subspace of desired output variable values in the overall parameter space that results from relating input variables to output variables.

The global input variables of the TBDS include:

- The composition of the resulting solid-state material (the different nuclear species used in the lattice or amorphous structure and their ratio R (vector), including vacancies as a possible "alloying component" e.g. in ternary combinations such as PdDVac)
- boundary conditions (temperature T, pressure P, electromagnetic field E)
- the type of dynamic stimulation applied to the system (excited phonon modes or excited photon modes of frequency ν_m and energy E_m that interact with the nuclei in the lattice).

The global output variables of the TBDS include:

- The nuclear transition rates L_x of all excited states in the system, including meta-stable excited states. (Such transitions rates can be of interest both for energy production purposes e.g. as nuclear reaction rates or, in different contexts, for information processing purposes i.e. as qubit transitions).

The intermediate states of the TBDS include:

- the equilibrium lattice structure (or amorphous structure) of the system, including the distances/proximities between all nuclei in the structure;
- the phonon modes of the system and their density of states;
- the coupling matrix elements (phonon-nuclear and photon-nuclear, as applicable) that describe interactions between nuclear states and phonons, nuclear states and photons, as well as nuclear states with other nuclear states via phonons or photons.

The process is as follows:

1. Start with a particular composition of nuclear species and their ratio as well as with particular boundary conditions (e.g. Pd with 0.5% B and D [with Pd:D = 1] at ambient temperature and pressure; or Pd with Vac and D at 700 C and 1 GPa pressure). Compositions where preliminary evidence suggests the manifestation of accelerated nuclear state transitions are particularly suitable starting points, although not exclusively so. Corresponding structures can then be a starting point for further refinement and optimization (e.g. of performance).
2. Calculate the expected crystal structure.
3. At this point the subsequent addition of further materials can be considered e.g. adding helium (He-3 or He-4) to the system to fill available interstitial sites (including by displacing other interstitials).
4. Calculate -- or look up from nuclear data datasources -- the expected nuclear state transition rates when the lattice is not stimulated (can vary as a function of proximity and electron density/screening which needs to be taken into account based on well-known methods).
5. Calculate the available phonon modes.
6. Then start with a particular type of stimulation (e.g. a ultrashort laser pulses that excite a range of phonon modes in the resulting solid-state lattice).
7. Calculate which phonon modes will get excited (frequency) and how strongly (energy) i.e. determine quantitatively the effect of the applied stimulation on the lattice (e.g. which phonon or photon modes, i.e. at which frequencies, are excited and how much energy ends up in those modes).
8. Calculate the phonon-nuclear and photon-nuclear coupling matrix elements between the nuclei in the

lattice -- based on the nuclear states of nuclei participating in the oscillator modes and on the frequencies and amplitudes of the oscillator modes. Here it is possible to limit oneself to those nuclei of particular interest. Quantum dynamical calculations will take into account effects from quantum coherence such as an amplification of coupling matrix elements from superradiance.

9. Determine the dominant energy transfer pathways in the overall system/lattice/structure based on the obtained strength of the couplings between nuclei (the nuclei of interest).
10. Determine the resulting de-excitation and excitation times for all nuclear transitions of interest at the time of coherence breaking (which occurs either naturally or is induced e.g. through disintegration of nuclei). Note that in the case of strong couplings and dominant excitation transfer pathways, induced de-excitations of some nuclei can coincide with excitations of other nuclei – which in turn can trigger secondary reactions such as disintegration. The range of secondary reactions can be determined by looking up from nuclear data, or by calculating from first principle based on the relevant literature, the behavior of thus excited nuclei upon receipt of the transferred quantum of energy. Note that, depending on the lattice configuration and the parameters above, dominant pathways may involve upconversion or downconversion of energy in the system (including energy released from nuclear reactions). For instance, D2 → He-4 de-excitation could drive 2 x Pd → 2 x Pd* excitation with subsequent disintegration or further transfer according to the same principles. Similarly, in another system, a dominant pathway could involve multiple D2 → He-4 de-excitations with simultaneous excitation of Pd → Pd* beyond 48 MeV which would ultimately result in symmetric or close-to-symmetric Pd fission products. For details regarding upconversion and downconversion scenarios, consult the attached simulations.
11. From these state transition rates follow the nuclear reaction rates and what are the dominant nuclear reactions (primary, secondary, tertiary, etc.). From the nuclear reactions that are dominant follow the dominant nuclear reaction products (and their properties such as their type/charge and kinetic energy) which can be further used as seen fit (e.g. charged particles for electricity production or neutrons for heat production).

All the above steps can be implemented and tested separately with corresponding input variables, output variables, and intermediate states. Eventually, the steps can be chained such that the model steps result in an overall model of the TBDS with global input variables and global output variables.

Once the system is in place and integrated this way (implemented in the computer **a**), the model delivers de-excitation and excitation times for all nuclear transitions of interest as a function of the chosen composition of the lattice, the boundary conditions, and the stimulation type and characteristics. Then, the input variables can be varied and the effect on output variables observed.

This process can be further automated as an iterative process to vary input variables systematically and automatically and observe a resulting "landscape" for each output variable of interest. Minima and maxima of this landscape are points of interest that inform optimal outcomes i.e. an optimal set of input variable values for certain desired outcomes. Multiple output variables can be coupled e.g. if certain tradeoffs are preferred such as between overall energy gain and the energy range of charged particles for instance.

Further input and output variables can be added such as the cost of the nuclear species chosen as input variables

for the structure composition, the cost of obtaining the boundary conditions (e.g. the cost of providing very high pressures), and the cost of the stimulation mechanism (e.g. the cost of ultrafast laser stimulation vs other mechanisms of coherent photon or phonon production). Additional output variables can include the type of particles that get emitted from the reactions that result from the nuclear transitions that are predominant based on the predominant energy transfer pathways as well as their ultimate economic value (e.g. in terms of the released energy that can ultimately be harnessed and utilized this way).

Moreover, constraints can be imposed on input variables e.g. to include as candidate nuclear species only materials that are readily available or that do not exhibit certain hazards. Similarly, output variables can be constrained e.g. by only including predominant nuclear transitions that lead to charged particle emission and not neutron emission.

This way, TBDS can be designed in such a way that optimal outcomes are achieved (or near optimal outcomes), along the output variables considered to be most relevant, from selecting and adjusting a set of candidate input parameters to choose from for the input variables.

For instance, the described system and process can be used to identify the most suitable isotopes, among a group of isotopes, to be used in a system for efficient energy production. Additionally, different geometric arrangements of such isotopes to be used and their effects on the overall system dynamics and system outcomes, e.g. energy production rates, can be compared and evaluated. Geometric arrangements that may want to be considered, tested, and evaluated for their performance with the system and process described above include different nanostructures such as nanoparticles, nanosheets, thin-film configurations, nanopillars, nanoneedles, nanowires, nanotubes etc.

jupyter nbviewer

Models to demonstrate decay rate enhancement via couplings

In this tutorial I will create "classical analog" toy models to illustrate how decay rates/nuclear state transition rates can be enhanced through couplings to other subsystems and resulting excitation transfer.

The power of classig analog models as well as their direct transferability and applicability to quantum models (including nuclear models) has been shown by Briggs & Eisfeld 2012. Therefore insights gained and derived from the models presented here are directly applicable to the design of systems of quantum-coherent nuclear engineering i.e. systems designed to trigger and optimize certain nuclear reactions with desirable outputs as a result of the system composition and trigger (such as coherent phonons or coherent photons).

Briggs, J. S., & Eisfeld, A. (2012). Coherent quantum states from classical oscillator amplitudes. Physical Review A, 85(5), 052111.

1. Python helper functions

In [1]:
```
# Loading some common python libraries

import numpy as np
from numpy import linalg
import matplotlib.pyplot as plt
from matplotlib import animation, rc
from IPython.display import Image
from IPython.core.display import HTML
import sympy as sp
#from google.colab.output._publish import javascript
mathjaxurl = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?config=def
sp.init_printing(use_latex='mathjax')
import matplotlib.patches as patches
import pandas as pd
```

In [2]:
```
# OS and SYS will be used to ensure the 'ImageMagick' directory is found by matplotlib.
import os, sys
# Adjust this as needed, my ffmpeg.exe was in "C:\ImageMagick\ffmpeg.exe"
# NOTE: including the "ffmpeg.exe" solved the [Win Permission Denied] Error mentioned e
ff_path = os.path.join('C:/', 'ImageMagick', 'ffmpeg.exe')
plt.rcParams['animation.ffmpeg_path'] = ff_path
if ff_path not in sys.path: sys.path.append(ff_path)

FFwriter = animation.FFMpegWriter(fps=15)
```

In [3]:
```
def create_fig1(): # create 1 empty subplots with 5 empty lines in each
    global lines1A

    fig1 = plt.figure() #figsize=(8,2)
```

```
    ax1A = fig1.add_subplot(111, xlim=(-5, 5), ylim=(-3, 3))
    ax1A.grid(color='lightgrey',alpha=1)
    ax1A.set_xticks(np.arange(-5,5,1))
    ax1A.set_aspect('equal', 'box')

    lines1A = []
    for i in np.arange(0,5,1):
        lobj = ax1A.plot([], [], 'o-',markersize=12, markevery=[0])[0] # , lw=0, color=
        lines1A.append(lobj)

    plt.close()
    return fig1
```

In [4]:
```
def create_fig2(xmax=100,ymin=-1.2): # create 2 empty subplots with 5 empty lines in ea
    global lines2A,lines2B

    fig2 = plt.figure(figsize=(6,8))

    ax2A = fig2.add_subplot(211, xlim=(-5, 5), ylim=(-3, 3))
    ax2A.grid(color='lightgrey',alpha=1)
    ax2A.set_xticks(np.arange(-5,5,1))
    ax2A.set_aspect('equal', 'box')

    lines2A = []
    for i in np.arange(0,5,1):
        lobj = ax2A.plot([], [], 'o-',markersize=12, markevery=[0])[0] # , lw=0, color=
        lines2A.append(lobj)

    ax2B = fig2.add_subplot(212, xlim=(0, xmax), ylim=(ymin,1.2))

    lines2B = []
    for i in np.arange(0,5,1):
        lobj = ax2B.plot([], [], '-')[0]
        lines2B.append(lobj)

    plt.close()
    return fig2
#create_fig2()
```

In [5]:
```
def animate1(i,data_graph1): # animation function that is called once for every frame (
    global lines1A

    for lineno,_ in enumerate(data_graph1): # strep through the list and the data for e

        lines1A[lineno].set_xdata(data_graph1[lineno][i][0])
        lines1A[lineno].set_ydata(data_graph1[lineno][i][1])
```

In [6]:
```
def animate2(i,data_graph1,data_graph2): # animation function that is called once for e

    for lineno,_ in enumerate(data_graph1): # strep through the list and the data for e
        lines2A[lineno].set_xdata(data_graph1[lineno][i][0])
        lines2A[lineno].set_ydata(data_graph1[lineno][i][1])

    for lineno,_ in enumerate(data_graph2): # strep through the list and the data for e
        lines2B[lineno].set_xdata(data_graph2[lineno][i][0])
        lines2B[lineno].set_ydata(data_graph2[lineno][i][1])
```

In [7]:
```
def animate1special(i,data_graph1): # animation function that is called once for every
    global lines1A,offset1,offset2,offset3,thisfig
```

```
    if i == int(offset1/5):
        lines1A[3].set_lw(1.5) # make coupling lines thicker
        lines1A[4].set_lw(1.5)
        thisaxis = thisfig.axes[0]
        redpatch = patches.Rectangle(xy=(-5, -3),width=10,height=5,facecolor="red",alph
        thisaxis.add_patch(redpatch)

    if i == int(offset2/5): # make coupling lines invisible
        lines1A[3].set_lw(0)
        lines1A[4].set_lw(0)

    if i == int(offset3/5): # in case the red patch gets removed again
        lines1A[3].set_lw(0.2)
        lines1A[4].set_lw(0.2)
        thisaxis = thisfig.axes[0]
        thisaxis.patches[len(thisaxis.patches)-1].remove()

    for lineno,_ in enumerate(data_graph1): # strep through the list and the data for e

        lines1A[lineno].set_xdata(data_graph1[lineno][i][0])
        lines1A[lineno].set_ydata(data_graph1[lineno][i][1])
```

In [8]:

```
def animate2special(i,data_graph1,data_graph2): # animation function that is called onc
    global lines2A,lines2B,offset1,offset2,offset3,thisfig

    if i == int(offset1/5):
        lines2A[3].set_lw(1.5) # make coupling lines thicker
        lines2A[4].set_lw(1.5)
        thisaxis = thisfig.axes[0]
        redpatch = patches.Rectangle(xy=(-5, -3),width=10,height=5,facecolor="red",alph
        thisaxis.add_patch(redpatch)

    if i == int(offset2/5): # make coupling lines invisible
        lines2A[3].set_lw(0)
        lines2A[4].set_lw(0)

    if i == int(offset3/5): # in case the red patch gets removed again
        lines2A[3].set_lw(0.2)
        lines2A[4].set_lw(0.2)
        thisaxis = thisfig.axes[0]
        thisaxis.patches[len(thisaxis.patches)-1].remove()

    for lineno,_ in enumerate(data_graph1): # strep through the list and the data for e
        lines2A[lineno].set_xdata(data_graph1[lineno][i][0])
        lines2A[lineno].set_ydata(data_graph1[lineno][i][1])

    for lineno,_ in enumerate(data_graph2): # strep through the list and the data for e
        lines2B[lineno].set_xdata(data_graph2[lineno][i][0])
        lines2B[lineno].set_ydata(data_graph2[lineno][i][1])
```

In [9]:

```
def add_pendulum_patches(thisfigure):

    thisaxis = thisfigure.axes[0]

    thisaxis.add_patch(patches.Rectangle(xy=(-5, 2),width=10,height=1,facecolor="gainsb
```

In [10]:

```
def add_polar_coordinates(df):
    alltheta1 = df.x1
    df['sin1'] = np.sin(alltheta1)*L1
```

```
    df['cos1']  = np.cos(alltheta1)*L1
    alltheta2 = df.x2
    df['sin2'] = np.sin(alltheta2)*L2
    df['cos2'] = np.cos(alltheta2)*L2
    alltheta3 = df.x3
    df['sin3'] = np.sin(alltheta3)*L3
    df['cos3'] = np.cos(alltheta3)*L3
    return df
```

In [11]:
```
def add_energies(df):
    # not used right now
    return df
```

2. REVIEW: a damped coupled system

Here is a brief review of the last part of the previous notebook:

For three coupled oscillators we can get the equations of motion as follows. In this system, we assume that each oscillator has its own spring constant k1, k2, and k3 and in addition, we have two equal coupling constants k_c between the oscillators.

Starting with the force equations:

$$F_1 = m\ddot{x}_1 = -k_1 x_1 - k_c(x_1 - x_2) - c\dot{x}_1$$

$$F_2 = m\ddot{x}_2 = -k_2 x_2 - k_c(x_2 - x_1) - k_c(x_2 - x_3) - c\dot{x}_2$$

$$F_3 = m\ddot{x}_2 = -k_3 x_3 - k_c(x_3 - x_2) - c\dot{x}_3$$

$$\ddot{x} + c\dot{x} + \frac{k}{m}x = 0$$

Organizing in terms of x1 and x2 gives us:

$$\dot{\vec{x}}_1 = \dot{v}_1 = x_1\frac{-k_1 - k_c}{m_1} + x_2\frac{k_c}{m_1} - cv_1$$

$$\dot{\vec{x}}_2 = \dot{v}_2 = x_1\frac{k_c}{m_2} + x_2\frac{-k_2 - 2k_c}{m_2} + x_3\frac{k_c}{m_2} - cv_2$$

$$\dot{\vec{x}}_3 = \dot{v}_3 = x_2\frac{k_c}{m_3} + x_3\frac{-k_3 - k_c}{m_3} - cv_3$$

From this, we can create the time evolution function based on the Euler-Cromer method:

In [12]:
```
def get_next_state_3CD(present_state,dt):

    global m1, m2, m3, k1, k1, k3, kcA, kcB

    x1 = present_state[0]
    v1 = present_state[1]
    x2 = present_state[2]
    v2 = present_state[3]
    x3 = present_state[4]
    v3 = present_state[5]
```

```
    dx1 = v1*dt
    nextx1 = x1 + dx1

    dx2 = v2*dt
    nextx2 = x2 + dx2

    dx3 = v3*dt
    nextx3 = x3 + dx3

    dv1 = (nextx1 * -(k1+kcA)/m1 + nextx2 * kcA/m1 - v1*c)*dt
    nextv1 = v1 + dv1

    dv2 = (nextx1 * kcA/m2 + nextx2 * -(k2+kcA*kcB)/m2 + nextx3 * kcB/m2 - v2*c)*dt
    nextv2 = v2 + dv2

    dv3 = (nextx2 * kcB/m3 + nextx3 * -(k3+kcB)/m3 - v3*c)*dt
    nextv3 = v3 + dv3

    next_state = np.array([nextx1,nextv1,nextx2,nextv2,nextx3,nextv3])
    return next_state
```

4. Creating the illustrations

4.1. Setting general parameters

In [13]:
```
offset1 = offset2 = offset3 = 1000 # start defining these as far out
```

In [14]:
```
t_end = 250 # time for simulations
allt = np.arange(0,t_end,0.1) # time vector size 2000
```

In [15]:
```
dt = 0.1
```

4.2. Uncoupled damped oscillator

This section illustrates the decay of the excited state of an isolated quantum system where the initial state is weakly coupled to a large number of final states (Golden Rule case).

This includes for instance metastable nuclear excited states whose "natural" decay (through spontaneous emission) follows this exponential probability distribution. The same model applies to spontaneous fusion from tunneling e.g. incoherent D2 --> He-4 as shown in Koonin and Nauenberg 1989.

In [166...
```
k1 = k2 = k3 = 1
m1 = 1
m2 = 2
m3 = 1

L1 = 2
L2 = 2 # this is only for Looks
L3 = 2

c = 0.005 # DAMPING PARAMETER
kcA = 0 # COUPLING A
kcB = 0 # COUPLING B
```

```
x1i = 0 #initial displacement m1
v1i = 0 #initial velocity m1
x2i = -1 #initial displacement m2
v2i = 0 #initial velocity m2
x3i = 0 #initial displacement m3
v3i = 0 #initial velocity m3
```

In [167…
```
thisstate = np.array([x1i,v1i,x2i,v2i,x3i,v3i])
thisstate
```

Out[167…
```
array([ 0,  0, -1,  0,  0,  0])
```

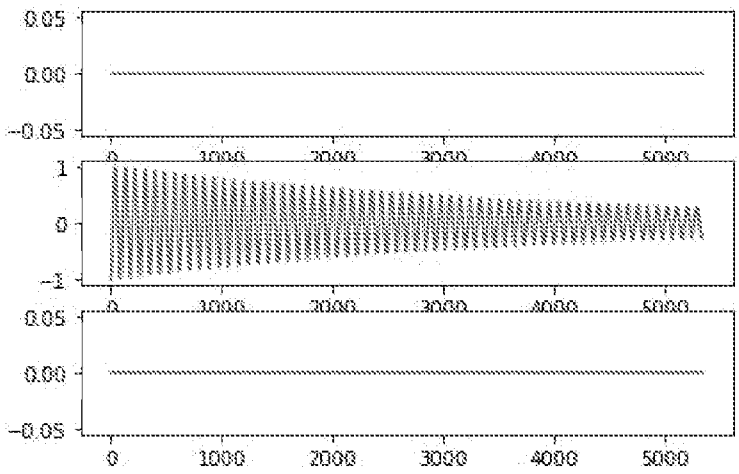
In [168…
```
allstates = []
for index,t in enumerate(allt): # step through every time step

    allstates.append(thisstate)
    nextstate = get_next_state_3CD(thisstate, dt)
    thisstate = nextstate

df1 = pd.DataFrame(np.row_stack(allstates), columns = ['x1', 'v1', 'x2', 'v2', 'x3', 'v
df1.insert(0, "time", allt, True)
df1 = add_polar_coordinates(df1)

plt.subplot(311)
plt.plot(df1.x1)
plt.subplot(312)
plt.plot(df1.x2)
plt.subplot(313)
plt.plot(df1.x3)
```

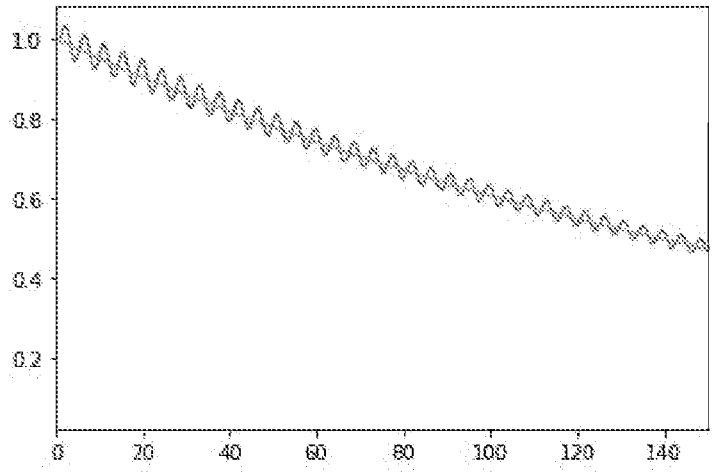
Out[168…
```
[<matplotlib.lines.Line2D at 0x29b11db16d0>]
```



In [179…
```
E_potential2 = 2*0.5*k2*df1.x2**2
E_kinetic2 = 2*0.5*m2*df1.v2**2
E_total2 = E_potential2 + E_kinetic2
exp2 = 1*np.exp(-c**2*0.5*allt/m1)

#plt.plot(allt,E_potential)
#plt.plot(allt,E_kinetic)
plt.plot(allt,E_total2)
plt.plot(allt,exp2)
plt.xlim(0,150)
```

```
graph = pd.DataFrame({'allt': allt, 'E_total2': E_total2, 'exp2': exp2})
graph.to_csv("graph1.csv", index=False)
```



In [31]:

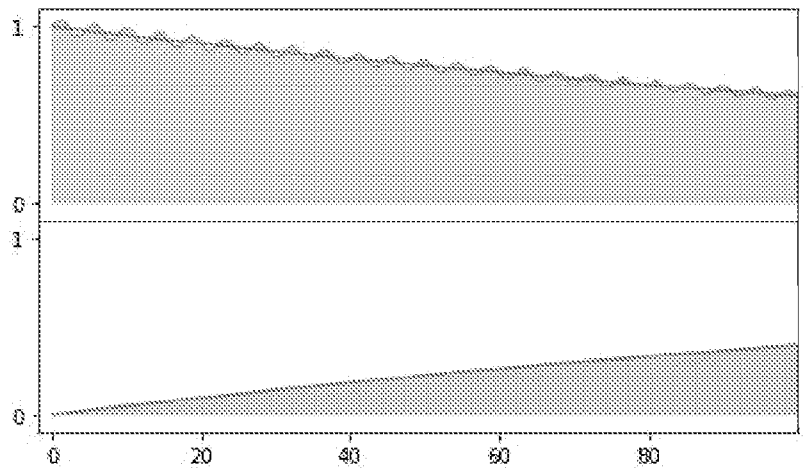
```
f = plt.figure(figsize=(7,4))

ax1 = plt.subplot(211)
ax2 = plt.subplot(212, sharex = ax1)

#ax1.plot(allt,E_total[:2500],color="C{}".format(1))
#ax1.fill_between(allt, E_total[:2500],color="C{}".format(1), alpha=0.5)
#ax1.plot(allt,exp,color="grey")
ax1.plot(allt,E_total2[:2500],color="C{}".format(1))
ax1.fill_between(allt, E_total2[:2500],color="C{}".format(1), alpha=0.5)
ax1.plot(allt,exp2,color="grey")
ax1.set_ylim([-.1, 1.1])
ax1.set_xlim([-2,100])
ax1.set_yticks([0,1])

#ax2.plot(allt,1-exp,color="grey")
#ax2.fill_between(allt,1-exp,color="grey", alpha=0.5)
ax2.plot(allt,1-exp2,color="grey")
ax2.fill_between(allt,1-exp2,color="grey", alpha=0.5)
ax2.set_ylim([-.1, 1.1])
ax2.set_xticks([0,20,40,60,80])
ax2.set_yticks([0,1])

plt.subplots_adjust(wspace=0, hspace=0)
```



4.3.2. Worse isolated oscillator

Now we cosider a case where the isolated quantum system decays faster compared to the above case due to stronger couplings to the final states (stronger damping).

This example already hints at how an acceleration of decay is connected with an increase in coupling strength to other states.

In [3]:
```
k1 = k2 = k3 = 1
m1 = 1
m2 = 2
m3 = 1

L1 = 2
L2 = 2 # this is only for looks
L3 = 2

c = 0.1 #0.02 #0.005 # DAMPING PARAMETER
kcA = 0 # COUPLING A
kcB = 0 # COUPLING B

x1i = 0 #initial displacement m1
v1i = 0 #initial velocity m1
x2i = -1 #initial displacement m2
v2i = 0 #initial velocity m2
x3i = 0 #initial displacement m3
v3i = 0 #initial velocity m3
```

In [33]:
```
thisstate = np.array([x1i,v1i,x2i,v2i,x3i,v3i])
thisstate
```

Out[33]:
```
array([ 0,  0, -1,  0,  0,  0])
```

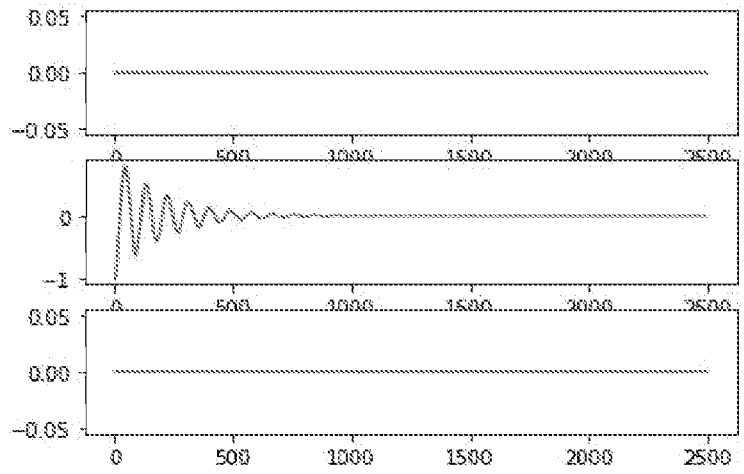
In [34]:
```
allstates = []
for index,t in enumerate(allt): # step through every time step

    allstates.append(thisstate)
    nextstate = get_next_state_3CD(thisstate, dt)
    thisstate = nextstate

df1 = pd.DataFrame(np.row_stack(allstates), columns = ['x1', 'v1', 'x2', 'v2', 'x3', 'v
df1.insert(0, "time", allt, True)
df1 = add_polar_coordinates(df1)

plt.subplot(311)
plt.plot(df1.x1)
plt.subplot(312)
plt.plot(df1.x2)
plt.subplot(313)
plt.plot(df1.x3)
```

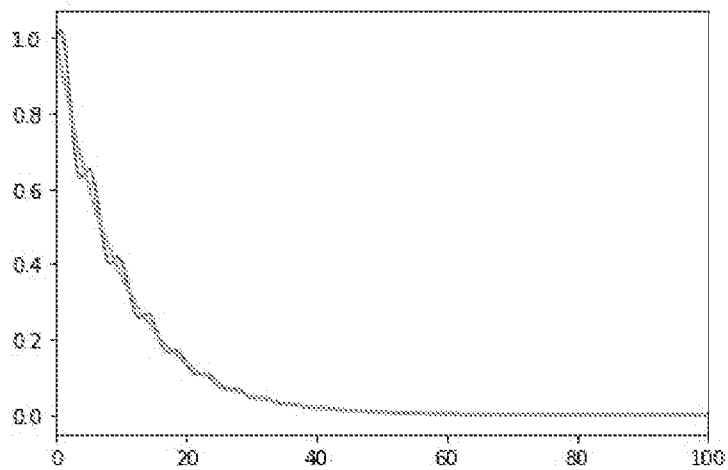
Out[34]:
```
[<matplotlib.lines.Line2D at 0x29b77d47af0>]
```



In [35]:
```
E_potential2 = 2*0.5*k2*df1.x2**2
E_kinetic2 = 2*0.5*m2*df1.v2**2
E_total2 = E_potential2 + E_kinetic2
exp2 = 1*np.exp(-c*2*0.5*allt/m1)

#plt.plot(allt,E_potential)
#plt.plot(allt,E_kinetic)
plt.plot(allt,E_total2)
plt.plot(allt,exp2)
plt.xlim(0,100)
```

Out[35]: (0.0, 100.0)



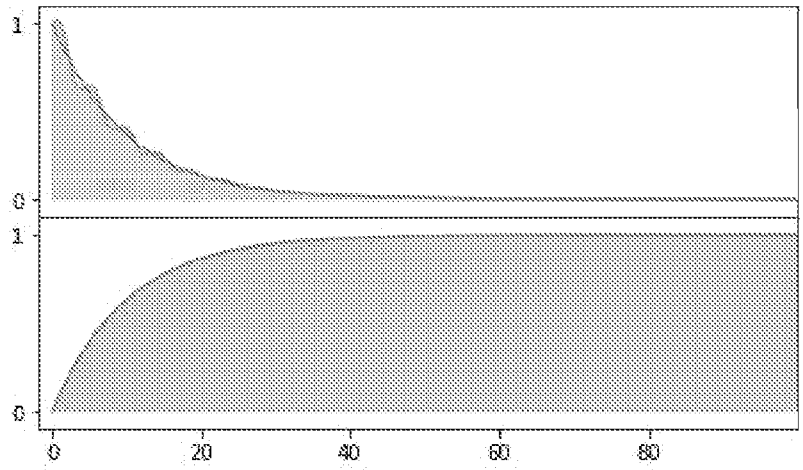
In [36]:
```
f = plt.figure(figsize=(7,4))

ax1 = plt.subplot(211)
ax2 = plt.subplot(212, sharex = ax1)

#ax1.plot(allt,E_total[:2500],color="C{}".format(1))
#ax1.fill_between(allt, E_total[:2500],color="C{}".format(1), alpha=0.5)
#ax1.plot(allt,exp,color="grey")
ax1.plot(allt,E_total2[:2500],color="C{}".format(1))
ax1.fill_between(allt, E_total2[:2500],color="C{}".format(1), alpha=0.5)
ax1.plot(allt,exp2,color="grey")
ax1.set_ylim([-.1, 1.1])
ax1.set_xlim([-2,100])
ax1.set_yticks([0,1])
```

```
#ax2.plot(allt,1-exp,color="grey")
#ax2.fill_between(allt,1-exp,color="grey", alpha=0.5)
ax2.plot(allt,1-exp2,color="grey")
ax2.fill_between(allt,1-exp2,color="grey", alpha=0.5)
ax2.set_ylim([-.1, 1.1])
ax2.set_xticks([0,20,40,60,80])
ax2.set_yticks([0,1])

plt.subplots_adjust(wspace=0, hspace=0)
```



4.4. Coupled pair of damped oscillators

4.4.1. Strong coupling from the beginning

In this example, there is a strong coupling (compared to the damping factor i.e. the incoherent decay parameter) given to another discrete state (the second oscillator). This accelerates the de-excitation of the initially excited system.

This example illustrates the dynamics behavior of Rabi oscillations e.g. between strongly coupled D2 molecules and He-4 nuclei that periodically excite and de-excite.

In [4]:
```
k1 = k2 = k3 = 1
m1 = 1
m2 = 1
m3 = 1

L1 = 2
L2 = 2 # this is only for looks
L3 = 2

c = 0.005 # DAMPING PARAMETER
kcA = 0.1 # COUPLING A
kcB = 0 # COUPLING B

x1i = 0 #initial displacement m1
v1i = 0 #initial velocity m1
x2i = -1 #initial displacement m2
v2i = 0 #initial velocity m2
x3i = 0 #initial displacement m3
v3i = 0 #initial velocity m3
```

In [176…
```
thisstate = np.array([x1i,v1i,x2i,v2i,x3i,v3i])
thisstate
```

Out[176… array([0, 0, -1, 0, 0, 0])

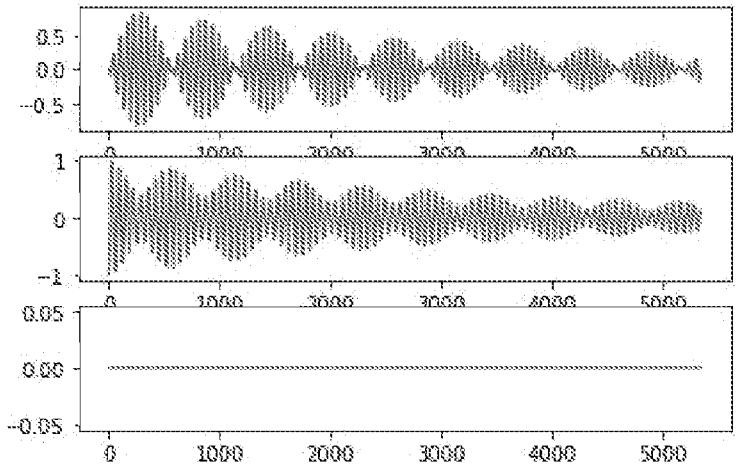
In [177…
```
allstates = []
for index,t in enumerate(allt): # step through every time step

    allstates.append(thisstate)
    nextstate = get_next_state_3CD(thisstate, dt)
    thisstate = nextstate

df1 = pd.DataFrame(np.row_stack(allstates), columns = ['x1', 'v1', 'x2', 'v2', 'x3', 'v
df1.insert(0, "time", allt, True)
df1 = add_polar_coordinates(df1)

plt.subplot(311)
plt.plot(df1.x1)
plt.subplot(312)
plt.plot(df1.x2)
plt.subplot(313)
plt.plot(df1.x3)
```

Out[177… [<matplotlib.lines.Line2D at 0x29b11e54970>]



In [178…
```
df = df1

# create position vectors for the first animated graph
data1,data2,data3,data4,data5 = ([],[],[],[],[])
for index,t in enumerate(allt): # step through every time step

    if index % 5 == 0: # take only every nth element
        data1.append([[-3*df.sin1[index],-3],[2-df.cos1[index],2]]) # first number: x o
        data2.append([[0*df.sin2[index],0],[2-df.cos2[index],2]]) # first number: x and
        data3.append([[3*df.sin3[index],3],[2-df.cos3[index],2]]) # first number: x and
        data4.append([[0*df.sin2[index],-3*df.sin1[index]],[2-df.cos2[index],2-df.cos1[
        data5.append([[3*df.sin3[index],0*df.sin2[index]],[2-df.cos3[index],2-df.cos2[i
```

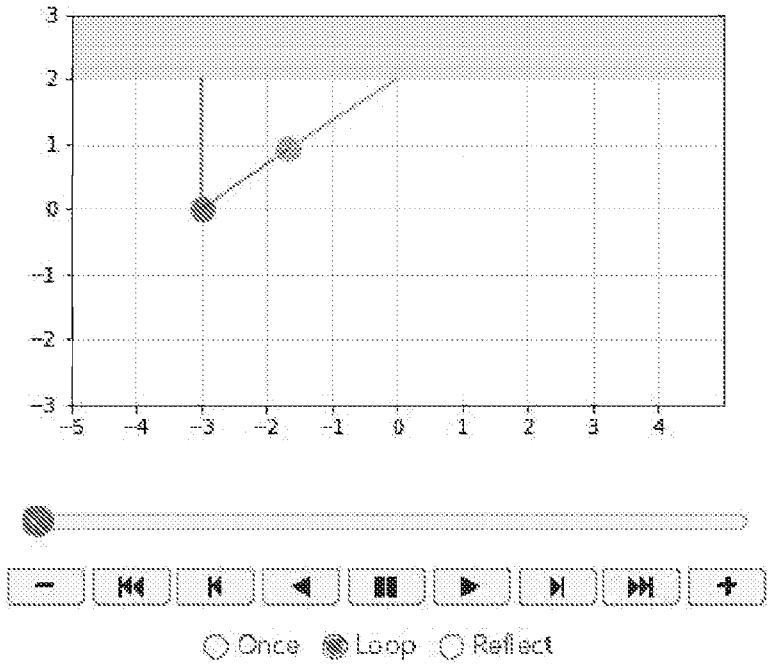
In [179…
```
# call the animation function
subplot1_data = [[np.asarray(data1),np.asarray(data2),np.asarray(data3),np.asarray(data

thisfig = create_fig1()
add_pendulum_patches(thisfig)
```

```
lines1A[3].set_marker('') # first spring
lines1A[4].set_marker('') # second spring
lines1A[3].set_color('brown') # first spring
lines1A[4].set_color('brown') # second spring
lines1A[3].set_lw(1) # first spring
lines1A[4].set_lw(0) # second spring
lines1A[2].set_marker('') # third pendulum
lines1A[2].set_lw(0) # third pendulum

ani = animation.FuncAnimation(thisfig,animate1,frames=200,interval=100,fargs=(subplot1_
rc('animation', html='jshtml')
ani
```

Out[179]:



In [180]:

```
# create vector of position function up to respective point for the second animated gra
data6,data7,data8 = ([],[],[])
for index,t in enumerate(allt): # step through every time step

    if index % 5 == 0: # take only every nth element
        position1_sofar = df1.x1[:index+1]
        position2_sofar = df1.x2[:index+1]
        position3_sofar = df1.x3[:index+1]
        time_sofar = allt[:index+1]
        data6.append([time_sofar,position1_sofar])
        data7.append([time_sofar,position2_sofar])
        data8.append([time_sofar,position3_sofar])
```

In [181]:

```
# call the animation function
subplot1_data = [np.asarray(data1),np.asarray(data2),np.asarray(data3),np.asarray(data4
subplot2_data = [np.asarray(data6),np.asarray(data7),np.asarray(data8)]

thisfig = create_fig2()
add_pendulum_patches(thisfig)

lines2A[3].set_marker('') # first spring
lines2A[4].set_marker('') # second spring
lines2A[3].set_color('brown') # first spring
```

```
lines2A[4].set_color('brown') # second spring
lines2A[3].set_lw(1) # first spring
lines2A[4].set_lw(0) # second spring
lines2A[2].set_marker('') # third pendulum
lines2A[2].set_lw(0) # third pendulum

lines2B[2].set_lw(0) # third pendulum

ani = animation.FuncAnimation(thisfig,animate2,frames=200,interval=100,fargs=(subplot1_
rc('animation', html='jshtml')
ani.save('double_pos.gif', writer='imagemagick', fps=15) #ani.save('double_pos.mp4', wr
ani
```

```
C:\Users\X1\anaconda3\lib\site-packages\numpy\core\_asarray.py:83: VisibleDeprecationWar
ning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
s-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to
do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

Out[101...